Lehrstuhl für Raumfahrttechnik
Technische Universität München

Model-based Framework for the Adaptive Development of Engineering Systems

Viktor Lévárdy

Vollständiger Abdruck der von der Fakultät für Maschinenwesen
der Technischen Universität München
zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs

genehmigten Dissertation.

Vorsitzender:	Univ.-Prof. Dr. rer. nat. Ulrich Walter
Prüfer der Dissertation:	1. Univ.-Prof. Dr.-Ing. Eduard Igenbergs, i.R.
	2. Univ.-Prof. Dr.-Ing. Michael Zäh
	3. O. Univ.-Prof. Dr. sc. techn. Reinhard Haberfellner, (Technische Universität Graz, Österreich)

Die Dissertation wurde am 22. Februar 2006 bei der Technischen Universität München eingereicht und durch die Fakultät für Maschinenwesen am 27.Oktober 2006 angenommen.

ACKNOWLEDGMENTS

From my first day as a research assistant at the Institute of Astronautics of the TU München until the final publication of this thesis has been an almost five year journey in academics and research, which simply would not have been possible without the kind help of many people and organizations. I owe many thanks to all those individuals who supported my doctoral work and made me feel at home in Germany.

First to my advisor Prof. Eduard Igenbergs, who taught me that systems engineering is a philosophy, a way of thinking about engineering problem-solving and thus it affects every aspect of our daily work as engineers or project managers. He perfectly fulfilled the role of the *Doktorvater*, he stood by me in difficult situations, and encouraged me to reach for more and more.

I also would like to thank Prof. Ulrich Walter for providing me this great opportunity of working at the Institute of Astronautics. Furthermore, I would like to thank Prof. Michael Zäh (TU München) and Prof. Reinhard Haberfellner (TU Graz), who served on my dissertation committee, for their comments, which considerably increased the quality of this thesis.

I would like to give special thanks to Prof. Tyson Browning (Texas Christian University), who provided inspiration and generous support during our joint research work. I will never be able to appropriately reference the ideas contributed by Tyson. Further, I am thankful to Prof. Stefan Thomke (Harvard Business School) for "frontloading" my dissertation project with his excellent publications and thus directing me to a fascinating research path. I also appreciate the cooperation wirh Prof. Ali Yassine (University of Illinois at Urbana-Champaign).

The SysTest research project, funded by the EC, was a great environment for experimenting with systems engineering methods. I appreciate foremost the kind support of Carlo Leardi (TetraPak Carton Ambient) during the development and validation of the methods in this thesis. I would also like to thank all the researchers involved in SysTest for the excellent cooperation: Dr. Avner Engel, Izhak Bogomolni, Shalom Shachar (Israel Aircraft Industry), Guido Scarafiotti (Centro Ricerche Fiat), Dr. Joachim Wegener, Frank Lammermann, Andreas Kraemer (DaimlerChrysler AG Research & Technology), Inigo Mendikoa, Mikel Sorli (Labein), Cecilia Haskins (NORSEC), Eric Honour (INCOSE), Dr. Nicolas De Abajo (Arcelor), Hugues Granier (Hispano Suiza), and Roberto Borsari (TetraPak Carton Ambient).

I am thankful to every colleague at the Institute of Astronautics, but especially to Dr. Markus Hoppe for the 42 months of joint research work. It was a great experience to work with Markus. I would also like to thank Dr. Andreas Vollerthun for teaching me the basics of systems engineering and disciplined project work during the first year in SysTest. I am also grateful for the friendship of Dr. Stefan Wenzel and Stephan Finkel and the kind support of Dr. Armin Schulz and Dr. Martin Wilke. Many thanks to every other colleague: Dr. Ernst Fricke, Dr. Herbert Negele, Michael Schiffner, Kristian Pauly, Jürgen Letschnik, Christian Ofer, Robert Senger, Markus Brandstätter, Tom Dirlich, Matthias Raif, and every student supporting us in SysTest.

Finally, I am exceedingly grateful for the continuous support and endless love of my wonderful parents. I dedicate this dissertation to them, because their share in my success is immeasurable. I also would like to thank to my sister and my brother, and all my crazy friends in Hungary, who always welcome me with open arms, support me and help me switch off from my daily problems.

TABLE OF CONTENTS

A. INTRODUCTION

A.1. RESEARCH CONTEXT

The development of engineering systems is a long, complex endeavor between the definition of a market opportunity based on the actual and predicted customer's needs, and the beginning of the production [Browning 2003]. System development (SD) is a search for something unknown, and the result of SD is a description of a thing to be made, including instructions about how to make it [Baldwin & Clark 2000]. Thus, SD is a process of gradually building up a body of information, until it eventually provides a complete formula for manufacturing a new system [Smith & Reinertsen 1998]. In this process, persons, technologies and tools, resources, existing company practices and knowledge, *etc.,* are utilized in a systematic manner to achieve the SD system objectives and generate value to the society (*Figure A.1*).

One difficulty of today's SD lies in the dynamics of its environment. Between the exploration of a market opportunity and the manufacturing of the first piece of product is a long period and during the course of the project, the SD environment changes. As *Figure A.1* depicts, many external and internal factors influence the operation of the SD system. Even if project planning usually considers the uncertainty incorporated in the external and internal SD factors, the predictions are often imprecise. Furthermore, many unpredictable events happen during the project, which affects the value of the final SD outcome. To avoid the consequences of these unanticipated events, *changes* are made in the SD system during the project to increase the value of its outputs [*e.g.,* Clark & Fujimoto 1991, Fricke *et al.* 2000]. While modifications are only possible in the scope of the available, planned resources of the project, changes in the SD are a major source of programmatic (cost and schedule) risk [*e.g.,* Browning 1999b].

Authors in the field of recent systems engineering and SD literature argue that traditional SD philosophies (*e.g.,* the waterfall SD model) and conventional project planning methods (*e.g.,* Program Evaluation and Review Technique (PERT), Critical Path Method (CPM), Gantt chart techniques, *etc.*) are not effective in extremely dynamic SD contexts, because they do not address the *high uncertainty and ambiguity* that characterize today's SD projects [*e.g.,* Smith & Reinertsen 1998, Haeckel 1999, Pall 2000, Highsmith 2000, Dove 2001, Thomke 2003]. Hence, for companies working in highly innovative and uncertain industry environments, the application of traditional SD methods is a risky decision [*e.g.,* Takeuchi & Nonaka 1986, Clark & Fujimoto 1991, Eisenhardt & Tabrizi 1995]. So the quality of the final product of the SD project defined by the fulfillment of the four key SD objectives in *Figure A.2* might be jeopardized by inappropriate conventional SD philosophies and planning methods. This could result in decreasing market

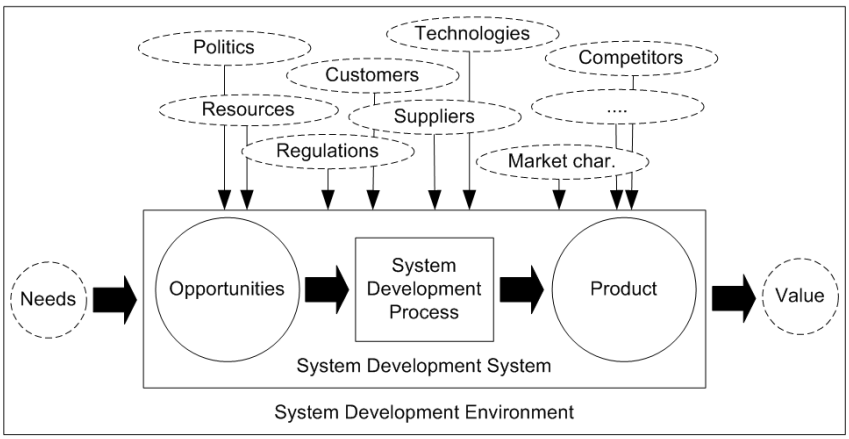


Figure A.1 System development system

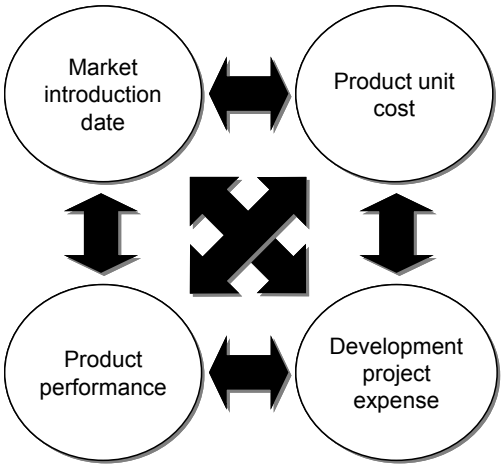


Figure A.2 Four key SD objectives (adapted from [Smith** & **Reinertsen 1998])

success and reduced profitability, which can lead to decreasing market share in the long term.

The main challenge of SD organizations under highly dynamic circumstances is to enable *efficient changes* in the SD system, which *increase the final value* of the product. That is, new SD system architecture models are required with the ability to accommodate changes without substantial negative impacts on the key project objectives (*Figure A.2*). Furthermore, novel SD philosophies are required that foster the exploration and capturing of design opportunities in a dynamic environment and thus contribute to the delivery of high value products.

A system (also an SD system) comprises a high number of system elements that are related to each other [Igenbergs 2000]. Thus, a change made in one element of the system to increase its value affects other related system elements, which might also require changes (*i.e.,* a change in one element *propagates* through the system). Therefore, the cost of a system change depends on the scope of change and thus it has two main aspects: the *direct cost of change* and the *indirect cost* caused by the propagation of the change to other elements.

Whether the scope and thus the total cost of a change in a system is high or low depends on the type of system architecture. If a system architecture has the *emergent* characteristic of low modification cost, the system architecture is called *flexible* [*e.g.,* Ulrich 1995, Thomke 1997], *adaptable* [*e.g.,* Rajan *et al.* 2004], *reconfigurable* [Son *et al.* 2000, Dove 2001, Nishinaga *et al.* 2003, Siddiqi *et al.* 2005], or *changeable* [Schulz & Fricke 1999, Fricke *et al.* 2000, Fricke & Schulz 2005]. **Flexibility, adaptability, reconfigurability, and changeability are similar terms for describing the structural or detail complexity of a system and thus the capability of the system structure to accommodate changes easily.**

However, besides structural complexity, changes in an SD system have major effects on the behavioral or dynamic complexity of the system. Many complex systems adapt or shift in response to changes in their context or to changes in their underlying components in the pursuit of *better fitness* [Holland 1995, 1999]. That is, changes contribute to the operational and behavioral characteristics of the system and thus affect the fitness (*i.e.,* value) of the SD process outputs. **SD systems that are capable of sensing changes in their dynamic environment, and of responding to them quickly by adapting their architecture to the changed conditions, are called *adaptive* [*e.g.,* Haeckel 1999, Highsmith 2000, Pall 2000] or *agile* systems [*e.g.,* Dove 2001, Haberfellner & De Weck 2005].** These systems are the major focus of this thesis[1].

[1] It is difficult to clearly distinguish between the terms *flexibility*, *agility*, *adaptability*, and *adaptiveness*. Even authors who are native speakers define these terms differently. Thus, the author of this thesis does not attempt to

While adaptiveness is an emergent system characteristic, the planning, operation, and management of adaptive SD systems require a *holistic system view* and the *application of systems engineering methods and principles*. Furthermore, it is necessary to develop and apply novel project planning and control methods that support the creation of an SD environment, where opportunities are sought and changes increasing the overall system value are supported, and not prohibited.

The major objective of this thesis is to provide systems engineering methods for effective planning and management of adaptive SD systems. While systems engineering is a model-based engineering language, the basis of adaptive systems engineering is also modeling. With the help of modeling and simulation, the structure, architecture, and behavior of the SD as a complex adaptive system can be understood, and adequate responses to the unpredictable changes in the dynamic system environment can be ensured.

The understanding of the system behavior in a dynamic environment is a basic step of adaptive SD project planning. It fosters the definition and appropriate sizing of activities that must be conducted to reduce risk and seize opportunities required to achieve market success. Furthermore, good planning provides the decision makers with alternative ways to fulfill project goals even in unexpected situations.

Thus, this thesis underlines that **enhanced, systematic project planning and control** are fundamental elements of adaptive SD system management; because these activities assure that the SD system is capable of *sensing* shifts in its internal and external environment through an adequately planned and operated feedback system. Furthermore, effective project planning is essential for the implementation of flexibility in the critical parts of the system designs (*e.g.,* goal, product, process, organization, technology systems), which increases the capability of the SD system to efficiently *respond* to the changes sensed.

Consequently, this thesis proposes systems engineering methods to successfully deal with the dynamic SD system environment. **First**, the described, existing system engineering methods help model and estimate uncertainty during planning. **Second**, further existing and novel methods proposed in this thesis support the design of flexible SD systems that are capable of accommodating the identified uncertainties. **Third**, the thesis proposes an adaptive SD framework, a model-based philosophy, to effectively implement and manage adaptiveness in SD systems. As project planning and control lies in the heart of adaptive SD systems, two systems engineering methods are developed to support these two project management functions. Hence, **fourth**, a model-based adaptive project planning method is introduced that delivers a *flexible SD project plan* including every activity option required to respond to anticipated and unexpected situations in the adaptive SD project. **Fifth**, a decision-making framework for adaptive project control is developed in the thesis that facilitates deliberate decisions on how the flexible SD system should be adapted to respond to the sensed changes. **Sixth**, the proposed new systems engineering methods were validated in industry environment and feedback on their feasibility is gathered from industry experts. In the next part of the *Introduction*, the requirements on adaptive SD systems are summarized.

A.2. REQUIREMENTS ON ADAPTIVE SD SYSTEMS

Systems engineering is the treatment of engineering design and development as a decision-making process [Hazelrigg 1996]. In this process, the agents of the SD system (*i.e.,* the developers) make decisions that form the behavior and improve the value of the output of the

---

put a firm stake in the ground for the precise meaning of these terms, but will use the simple distinction presented here.

SD. The decisions the agents of the SD make, are driven by the system objectives, and constrained by the available SD system capability, and allocated resources. Furthermore, during decision-making, the developers act as elements of the SD system. That is, the culture, the shared vision, and the strategic objectives of the SD system guide their decisions, and they seek the solutions that mean value for the SD system [Senge 1990]. Hence, the holistic philosophy of adaptive SD described in this thesis only works if it is implemented and understood at each hierarchy level of the SD organization. Otherwise, system adaptations get expensive, time-consuming, ineffective, and do not increase the value of the system.

SD as an adaptive system has the capability of sensing changes in its inputs and responding quickly to changes by reorganizing and changing its outputs [Haeckel 1999]. *Reorganizing* in this context means that the agents of the SD system collect feedback from the system environment, sense and interpret the characteristics, directions, and effects of the changes, and respond to them by selecting and reconnecting the change-relevant system elements that contribute to the delivery of the highest value outputs in the new SD system state.

While conventional SD philosophies offer only a limited capability towards system changes, revolutionary methodologies are required which allow and drive system adaptations that maximize the overall value of the project according to the stakeholders' needs, and the enterprise vision and mission. A thorough review of literature on novel SD systems provided the following requirements for SD adaptiveness:

- **The ultimate goal of system adaptation is to change the SD system to move to a state with higher overall stakeholder value** [*e.g.,* Baldwin & Clark 2000, Browning & Honour 2005]. Thus, core functions of an adaptive SD system are *opportunity management* and *innovation management* [Dove 2001]. These functions foster effective learning and support the discovery and capturing of design opportunities that contribute to the development of superior products.

- **Adaptive SD system architectures must be highly flexible** [*e.g.,* Upton 1994, Ulrich 1995, Sanches & Mahoney 1996, Thomke 1997, Baldwin & Clark 2000, Dove 2001, MacCormack & Verganti 2003]. That is, changes in the system architecture have to be easy, cheap, and controllable. Ideally, adaptive systems consist of reusable, fairly independent elements that are reconfigurable in a scalable framework [Dove 2001]. Modular systems are highly change-tolerant and flexible, because they are composed of independent modules and few, well-defined interfaces. So changes remain usually "hidden" in the modules enabling independent module-level SD work and frequent changes [Baldwin & Clark 2000]. In addition, Schilling [2000] argues that the primary action of increasing modularity in a system is to enable heterogeneous inputs to recombine into a variety of heterogeneous configurations. Thus, *flexible, adaptable systems must be basically modular.*

- **The procedure of system adaptation must follow a systematic framework** (based on [Eisenhardt & Tabrizi 1995, Iansiti & MacCormack 1997, MacCormack *et al.* 2001]). This framework must guide the decision-makers through the necessary steps of system adaptation, highlight the information requirements, and thus provide a firm basis for the evaluation of change options and selection of the best one for the project. The role of the adaptive SD system framework is to guide and foster effective SD work, not to constrain opportunities in the project.

- **System adaptation must be a collaborative decision-making process** [Agile Manifesto Website, Senge 1990, Dove 2001]. Thus, when it comes to decisions, all the experts from relevant design teams, competencies, and departments (including Marketing, Sales, and Technology Development if required) must be present to foster the effectiveness and

efficiency of decision-making. The competencies required in the system adaptation process shall be determined on the basis of the actual SD system state.

- **Decisions on system adaptation must be made on the basis of detailed information about the actual state of the SD system** (*i.e.,* characteristics of changes) [Dove 2001]. Hence, in adaptive SD, project measurement and control are key systems engineering functions. Both endogenous (*e.g.,* discovery of opportunities, higher risks than anticipated, lower process efficiency and effectiveness, *etc.*) and exogenous SD factors (*e.g.,* shifting customer needs, competitors' new products, emerging technologies, new government regulations, *etc.*) have to be continuously monitored, and changes have to be sensed and communicated.

- **In the decision-making process on system adaptation, it is inevitable for the actors to be aware of and understand the objectives, structure, and behavior of the SD system** (based on [Sterman 2000]). On the one hand, system modularity supports this requirement by reducing structural system complexity and thus allowing a manageable problem scope when it comes to making decisions. On the other hand, system adaptation requires an existing and continuously updated company-wide knowledge base and lessons learned from previous projects. All the information necessary for the decisions must be contained in this knowledge base; otherwise, the decisions will be sub-optimal leading to lower enterprise profits.

- **Effective decision-making on system adaptation requires thorough planning** (based on [Eisenhardt & Tabrizi 1995, Iansiti & MacCormack 1997, MacCormack *et al.* 2001]). This improved planning effort has to include the identification and description of change options that show how the system can respond to changes in its inputs. That is, project planning cannot be restricted to the definition of one feasible SD process, but it has to consider all activity options that might be necessary to reduce the anticipated and unanticipated risks, and find and capture design opportunities during the SD project. Thus, the main task of project planning in an adaptive SD is to define a flexible SD process that can be easily adapted to changed SD conditions.

- **System adaptations demand appropriate management reserves and resources otherwise they are major risk drivers in the project** (based on [Browning 1999b]). As system adaptations are foreseen and unforeseen changes in the SD system, the resources allocated to the project have to be adequate to cover all required costs of these changes.

Based on these requirements, the next part describes the objectives of the thesis.

A.3. THESIS OBJECTIVES AND DELIVERABLES

In short, this thesis attempts to show that a new way of thinking about SD—*i.e.,* a new, holistic, model-based, adaptive SD philosophy—is the key to long-term success. While the models we use form the way, we think and not the other way round; one goal of this thesis is to propose a simple framework for adaptive SD systems that supports ***controlled learning under dynamic circumstances***. Adaptive SD systems organized around this framework handle environmental changes effectively, which results in better products and increased stakeholder value.

As the adaptive SD framework creates a *new mental model* for project managers and developers, existing methods are not feasible to implement adaptiveness successfully in the projects. Hence, this thesis explores and describes the main aspects of SD in a dynamic environment, and provides systems engineering methods for the planning and control of *sense and response SD systems*

according to the requirements found. A simplified model of the adaptive SD system is depicted in *Figure A.3*. Adaptive SD, as a system, receives inputs from its environment, which are then transferred into outputs inside the system. The adaptive SD system model in *Figure A.3* differs from conventional SD system models, because it is designed to have the capability of sensing the changes in its environment, and responding to them in the form of improved products, as outputs.

Furthermore, the outputs of the adaptive SD system generate new needs towards new products and technologies. That is, the SD system changes its context through its behavior [Schilling 2000, Sterman 2000]. The shifting market needs create new opportunities for the adaptive SD system that can be utilized in terms of enhanced new versions or increments of the flexible products. Hence, an adaptive SD system utilizes its capability of effectively sensing changes and responding to them to generate competitive advantage from this characteristic. Adaptive SD systems are *masters of changes* and profit from them in the form of the development of superior market products.

This thesis recommends that two fundamental characteristics make systems migrate towards adaptiveness and agility: (1) the growing *complexity* of SD systems and their context; and (2) the increasing *uncertainty* and *ambiguity* regarding the internal and external needs of the SD. These two interrelated facets of SD are the major challenges of today's SD projects.

The enormously growing system complexity increases the difficulty of the SD as a problem to be solved. If the complexity of conventional systems is no longer manageable, new types of system architectures are required that facilitate the effective decomposition of the SD problem into individual sub-problems with controllable scope and complexity. Adaptable products consisting of individual modules can be considered as *systems of systems*, where independent modules are developed and fabricated by individual organizations that are integral parts of the *system of systems enterprise*. This thesis proposes that **modular products with high flexibility and adaptability are basic building blocks of such adaptive SD systems.**

Furthermore, modularity must be a *structural* characteristic of the system, where modules

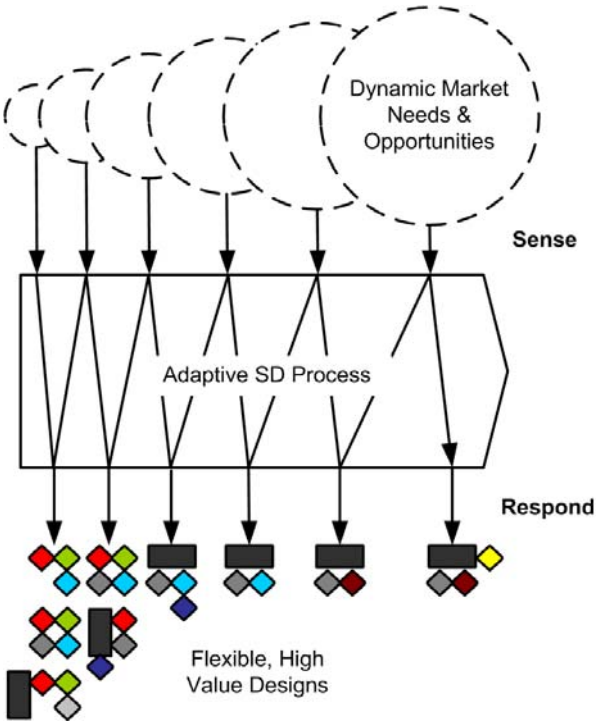


Figure A.3 Adaptive SD process

involve closely related system components with many *intra*modular and just a few clearly defined *inter*modular element interactions. Hence, modularization in an adaptive SD system aims to define a system structure containing independent modules that can be developed and changed independently and that can be easily reconnected during the system adaptation. This thesis includes methods for structural modularization that support this objective.

Uncertainty, the other aspect that pushes SD systems towards adaptiveness, emerges when the dynamic complexity of the SD system and its context gets so high that the future behavior of the overall system is no longer predictable. That is, the *dynamics of the external* (*e.g.,* market needs, technologies, suppliers, competitors, *etc.*) *and internal SD characteristics* (*e.g.,* product performance, technology maturity, resources, *etc.*) *that drive the system lifecycle value are not foreseeable,* and the exact characteristics of the system to be developed cannot be determined at the outset of the project.

This thesis proposes that the best instrument against uncertainty is to design *response ability* in the SD system that allows quick reactions to shifts in the SD context and thus long-term maximization of the overall SD system value. Furthermore, SD has to be organized for collaborative and individual learning, as **understanding and learning is the main driver of adaptiveness**. Thus, the adaptive SD system framework developed in this thesis is ***process-centric.*** Adaptive SD systems are organized around the iterative SD process characterized by **permanent experimentation, frequent prototype releases, and continuous validation with the customer**.

The main deliverable of this thesis is a systems engineering framework for the planning and control of adaptive SD projects (*Figure A.4*). In this framework, the SD enterprise is considered as a system in permanent interaction with its environment—*i.e.,* the SD system is an element of the market with a close relation to other market elements (*e.g.,* customers, competitors, government, suppliers, *etc.*). Thus, the actions of other market elements affect the behavior and also the output of the SD enterprise system. Furthermore, the SD system is internally comprised of system elements as well. These system elements interact on the basis of the system structure, and produce the deliverables of the SD system.

The goal of systems engineering is to plan and control the work in the SD system to support the *right development of the right product*. In a highly dynamic SD context, this requires continuous interaction with the SD stakeholders to follow and interpret the trends of the quickly evolving needs. Systematic project planning and control are basic means to build a bridge between internal and external stakeholders, and synchronize SD needs and performance to achieve maximal stakeholders' satisfaction.

Consequently, the two main products of the thesis support the effective planning and control of a flexible SD process, which facilitates the fulfillment of the requirements on adaptive SD defined in the previous section. Adequate planning is important, because SD plans guide the developers on their way from the identification of a design opportunity to the start of the production. Adaptive SD requires process plans with increased flexibility that show multiple ways

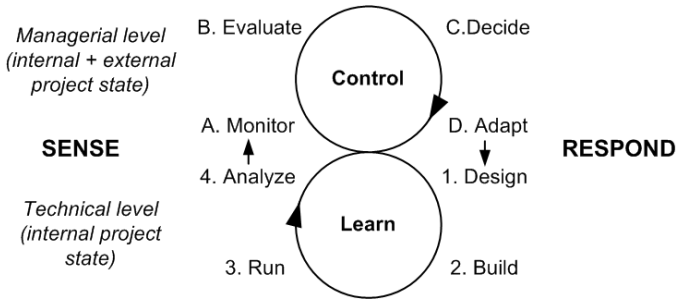


Figure A.4 Adaptive SD framework

to fulfill the requirements. These alternative ways are *real options* for the developers, activities that can be chosen in certain SD situations, but do not have to be carried out if not required. Thus, SD plans in an adaptive SD system are not mere control mechanisms, but instruments that support deliberate decision-making by depicting the decision options for certain SD states.

The implementation of adaptiveness in SD systems is supported by a novel planning technique; the ***Adaptive System Development Process (ASDP),*** a process modeling method that simulates the behavior of the adaptive SD systems, which is the first main deliverable of the thesis. This process modeling method implemented in a software tool models and simulates decision-making in adaptive SD, and always selects the best activity for the project from a flexible process space. Decision-making on process adaptation in ASDP is based on the actual project state described by the project budget and effort spent, and the product performance achieved until the actual point in the project. A system of process attributes is applied to track process performance during simulation, and calculate risk and opportunity values to obtain comparable management measures showing the overall maturity of the design and the performance of the project from different technical and business units.

ASDP revolutionizes process modeling by simulating the behavior of a flexible process space with the main objective of stakeholder value maximization. Thus, ASDP delivers a more realistic SD project schedule and a set of process options for the decisions on system adaptation during SD. Furthermore, using the results of the simulation in ASDP, adequate management reserves can be planned that incorporate resources for foreseen and unforeseen adaptations as well.

The second systems engineering method that supports the implementation of adaptiveness in SD systems is a ***decision-making framework for the control of adaptive SD projects***. This systematic project control procedure guides the SD team towards higher system value by providing the decision-makers with ways to resolve the actual SD state, process options to improve the actual state, evaluation methods to determine the effect of each process option on the overall system value, and decision rules to select the best alternative towards maximal system value.

Both systems engineering methods were validated in industrial environment. The pilot projects at TetraPak Carton Ambient in Modena, Italy, showed that the methods are not only applicable in the industry, but they contribute to significant increases in process effectiveness and efficiency.

A.4. THESIS STRUCTURE

After the research context, the requirements, main objectives, and deliverables of the thesis have been described, in this final section of the *Introduction*, the structure of the thesis is described. The structure represents the author's systematic research approach to solve the problem defined in the previous sections, and deliver the products that fulfill the requirements. As the writing of this dissertation was a real, adaptive SD project (*i.e.,* an iterative journey to the unknown) the objectives, scope, and thus the structure of this thesis emerged and changed until the last days of the process (and would obviously change if more resources were available).

The product of the dissertation writing process (*i.e.,* the PhD thesis) is a system in itself made up of individual components. These components are organized into three functionally different modules depicted in *Figure A.5*. The function of the first main module of the system is to introduce the concept of adaptive SD systems, and provide a concrete theoretic fundament for the methods introduced and validated in the later modules. The five subsystems of the first module are organized in a top-down manner, discussing the aspects of adaptive systems engineering in growing detail.

Subsystem B in the first theoretical module presents basic systems engineering principles and models, which are the basic building blocks of the description of adaptive SD systems. As the author of this thesis is a PhD student of the *Munich systems engineering school* at the Institute of Astronautics of the Technische Universität München, the thesis builds on the systems engineering concepts developed by Prof. Eduard Igenbergs and his former doctoral students. However, this first chapter also shows how the concepts of Munich systems engineering fit in the world of systems engineering research. Additionally, the last part of the first chapter defines the meaning of systems engineering from the author's point of view.

Subsystem C of module one reviews SD lifecycle models and philosophies presenting the way from the waterfall model to Agile SD. The philosophy an organization applies for SD has a basic influence on the behavior and output of the SD system. This section shows that though conventional SD models cannot host adaptive SD principles; there is a lack of adequate, mature philosophies for the adaptive development of complex engineering systems. Nevertheless, this section closes with a list of fundamental characteristics for adaptive SD systems.

The third thesis component, *Chapter D,* deals with the inputs of the SD system, and discovers the effects of the dynamic system context on the system objectives. Because the system objectives show the developers the right direction, they must be always kept up-to-date with the changing system environment. Furthermore, this section introduces the term *value*, a measure that shows the fitness of the SD system performance from the stakeholders' point of view.

Chapter E focuses on the output of the adaptive SD system and discusses the required architectural characteristics of the products of adaptive SD. This section shows that modularity is a main driver of flexibility and thus a fundamental requirement for the products of adaptive SD systems. Even though many other design aspects contribute to SD system adaptiveness, there is no effective adaptive SD without modularity and flexibility.

The last subsystem of the first theoretic module of the thesis, *Chapter F,* analyses the

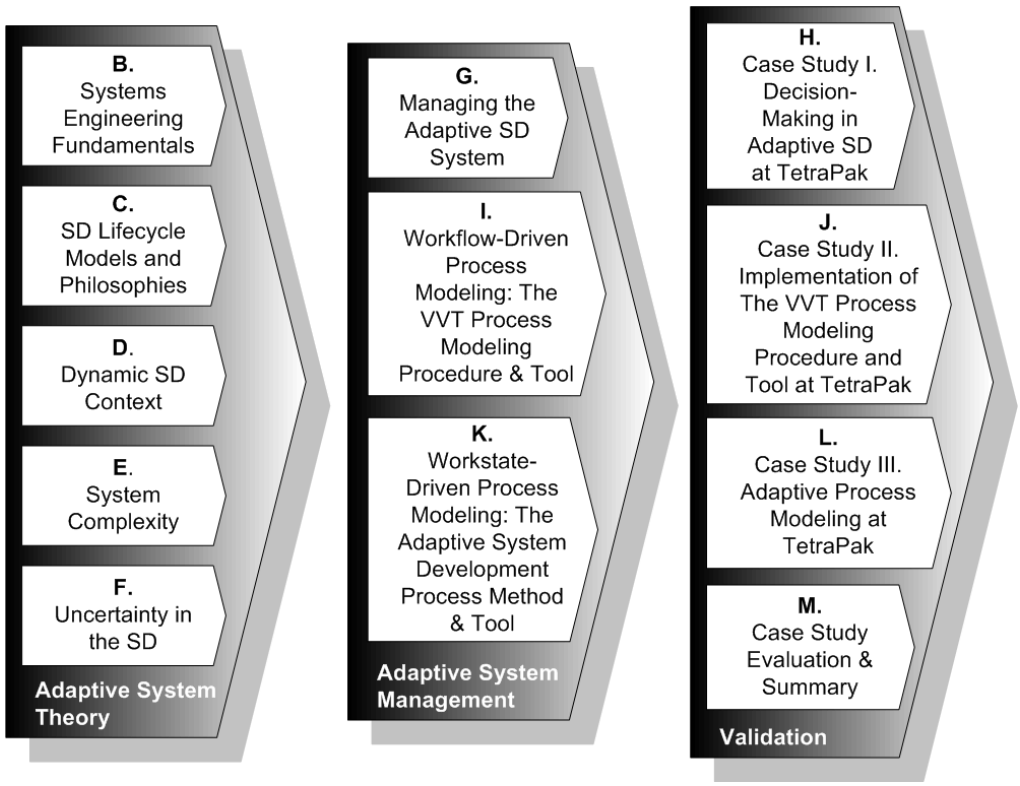


Figure A.5 PhD thesis structure

phenomenon *uncertainty* and the process-related aspects of adaptive SD. This section discovers that design iteration and systematic learning are fundamental means to resolve uncertainty. Thus, SD project structures have to be basically iterative, built up by a hierarchic system of experimentation cycles between design reviews and milestones. Such a project structure ensures rapid experimentation (*i.e.,* the cyclic process of system design, verification, and evaluation) and prompt feedback from the customer on the prototypes delivered by the experiments. The role of the frequent reviews and milestones is to provide a place for informed decisions on process adaptation using internal and external feedbacks on the SD process deliverables.

The second and third modules of the thesis include the developed adaptive SD methodology and its validation. Even though the three chapters in the second module of the thesis are functionally closely related, they do not follow one after the other in the dissertation. The reason is that each chapter of the *Adaptive System Management* module is followed by a chapter of the third, *Validation* module to show how the developed methods were implemented and validated at *TetraPak Carton Ambient*, in the food packaging system development industry. This rule is shown by the chapter numbering in *Figure A.5*.

The first subsystem of module two, *Chapter G* describes the *adaptive SD framework* that defines the basic structure of adaptive SD projects. This *double loop of learning and control* is a risk- and opportunity-based, work*state*-driven systems engineering management framework that allows the generation of an emergent SD strategy. The adaptive SD strategy and process evolve together with the design and the knowledge in the project enabling the right development of the right products. The iterative process of learning is one main element of this SD framework that follows Thomke's four-step *experimentation cycle* model [Thomke 2003]. Experiments are the main places for learning in the SD and provide evolving representations of the system design in a systematic manner.

The second basic element of the adaptive SD framework is the *control loop*. The goal of the control loop is to *monitor* process performance by collecting actual information inside and outside the project, *evaluate* this information, and determine the actual risk and opportunity status regarding the main requirements and constraints of the SD. This information provides an excellent basis for *decision-making*, because the actual risks and opportunities represent the probable profit losses and gains the management can anticipate without changing the project plans. In the final step of the control loop, the management can decide to *adapt* the SD process to better deal with the actual SD needs, and to reduce risks and capture opportunities.

After the introduction of the adaptive SD framework, *Chapter G* focuses on the presentation of the control loop. This procedure integrates project monitoring, risk management, and project control to process internal and external project information describing the actual state of the SD to support deliberate decisions on project adaptation.

In the subsequent part of the thesis, in *Chapter H* the implementation and validation of the *procedure of adaptive project control* and the *decision framework for system adaptation* is demonstrated in a real industry environment through a case study. The case study at TetraPak Carton Ambient in Italy demonstrates how parameter-based project monitoring and risk- and opportunity-driven decision-making can be implemented in the food packaging industry SD processes and integrated with conventional SD methodologies.

Chapter I is the first of four chapters on process modeling, an effective tool for project planning. These four chapters present the way from traditional network techniques, over work*flow*-driven process modeling—representing the state-of-the-art in the industry—to work*state*-driven methods, which might be the next generation of project planning tools. Two parameter-based stochastic process modeling methods are described and validated in these four

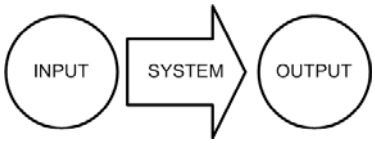
chapters to illustrate the difference between conventional, work*flow*-driven, and the novel, adaptive, work*state*-driven approaches for project planning. The work*flow*-driven VVT Process Modeling (VVTPM) procedure and tool is presented in *Chapter I* and validated in an industry environment in *Chapter J*.

Chapter K deals with work*state*-driven process modeling and presents the *Adaptive System Development Process (ASDP)* method and tool. The ASDP method implements the *double loop of learning and control* in a process modeling framework, and simulates the SD process as an intelligent system that evolves toward maximal stakeholder value. ASDP is validated in *Chapter L* using SD processes from TetraPak Carton Ambient. This chapter also includes a comparison of the results of the VVTPM and ASDP tools to highlight the differences between work*flow* and work*state*-driven process modeling.

The final chapter of the thesis includes the evaluation of the case studies conducted at TetraPak Carton Ambient from an industrial point of view. This chapter contains the results of a questionnaire aiming to explore the *feasibility* of the methods developed in the thesis and implemented in industry environment. This positive feedback from the industry is particularly important for the final evaluation of this thesis, because it shows that the thesis findings are valuable and feasible for industry application.

B. SYSTEMS ENGINEERING FUNDAMENTALS

B.1. CHAPTER ABSTRACT



This first chapter gives an overview of the history of systems sciences, proposes basic definitions, and reviews system models that are of great importance for the later chapters of this thesis. The definitions, models, and modeling methods reviewed in this first chapter represent a way of thinking about systems, and particularly SD systems that separates the systems engineer from the product developer. Further, the systems engineering theory described in this chapter involves the system elements that provide the fundament for the development of the adaptive SD systems theory introduced in the second part of the thesis.

B.2. THE ROOTS OF SYSTEMS SCIENCES

Systems theory is the philosophy of understanding and solving complex problems. Its historical roots can be traced back to ancient Greece, where Plato and his student Aristotle documented the first holistic thoughts around 350 BC [Negele 1998]. Besides the development of holistic methods to deal with physical and biological phenomena, Aristotle formulated the famous basic principle of holistic thinking in his book Metaphysica: *the whole is more than the sum of its parts*.

This sentence shows that even the first holistic thinkers recognized that systems cannot be understood simply by understanding the parts, but the interactions between the parts and the consequences of these interactions are equally significant. Hence, systems must be looked at in their entirety, recognizing that many phenomena are more than the sum of their parts. Systems theory calls the system level phenomena that cannot be observed at the analysis of the parts *emergent properties* of the system.

Emergence refers to the macro-level patterns arising in systems of interacting agents. Emergent phenomena cannot be deduced from knowledge of behavior of individual parts and are not reducible to the parts alone. Emergent complexity is driven by a few simple patterns that combine to create infinite variety [Wikipedia Website].

Furthermore, systems theory and systems sciences argue that it does not matter how complex or diverse the world that we experience is; different kinds of organizational patterns can always be found in it, and these patterns can be described by generic concepts and principles independent of the specific domain. The specific organizational patterns that system analysts seek are the characteristic elements of the system and the interactions between them. Thus, the *systems approach* distinguishes itself from the more traditional *analytic approach* by emphasizing the interactions and connectedness of the different components of a system [Heylighen *et al.* 1999].

The principles that describe the characteristics and behavior of different complex systems form an interdisciplinary science (*i.e.,* systems theory) adapted for a universal application with a common language and area of concepts. This approach is seen as a means of not only overcoming the fragmentation of knowledge and the isolation of the specialist, but also finding new solutions to problems created by the earlier "solution of problems".

Systems sciences evolved during the 1940s, 1950s and 1960s in the USA as the result of research conducted by various development teams in different scientific disciplines. However,

three main research fields are considered as the major contributors to today's systems sciences (*e.g.,* systems engineering, systems dynamics, systems thinking, *etc.*):

- General systems theory
- Cybernetics and systems dynamics
- Systemology and operations research

The *general systems theory* was proposed in the 1940s by the Hungarian biologist Ludwig von Bertalanffy [1968], and furthered by Ross Ashby [1956]. Von Bertalanffy was both reacting against reductionism and attempting to revive the unity of science. He emphasized that real systems are open to, and interact with, their environments, and that they can qualitatively acquire new properties through emergence, resulting in continual evolution.

Rather than reducing an entity (*e.g.,* the human body) to the properties of its parts or elements (*e.g.,* organs or cells), systems theory focuses on the arrangement of, and relations between the parts, which combine them into a whole (*cf.,* holism). This particular organization determines a system which is independent of the concrete substance of the elements (*e.g.,* particles, cells, transistors, people, *etc.*). Thus, the same concepts and principles of organization underlie the different disciplines (physics, biology, technology, sociology, *etc.*), providing a basis for their unification. Systems concepts include system-environment boundary, input, output, process, state, hierarchy, goal-directedness, and information [Wikipedia Website].

Cybernetics is a theory of the communication and control of regulatory feedback. The word cybernetics was first used by the mathematician Wiener, who adapted it from the Greek word *kybernetes* meaning "steersman" to invoke the rich interaction of goals, predictions, actions, feedback, and response in systems of all kinds [Wiener 1948]. As Ashby wrote in [1956]

> *cybernetics offers a method for the scientific treatment of a system in which complexity is outstanding and too important to be ignored.*

Cybernetics has a threefold contribution to the evolution of systems sciences: (1) it stresses information flow as a distinct system component differentiating between the activating power and the information signal; (2) it recognizes that similarities in the action of control mechanisms involve fundamentally identical principles; and (3) it gives the basic principles of feedback control a mathematical treatment [Blanchard & Fabrycky 1990].

In fact, cybernetics and general systems theory study essentially the same problem, that of organization independent of the substrate in which it is embodied. Whereas systems theory focuses more on the structure of systems and their models, cybernetics deals with how systems behave, *i.e.,* how they control their actions, how they communicate with other systems or with their own components. Since structure and function of a system cannot be understood separately, the concepts of cybernetics and systems theory complement each other in a discipline that is more than the sum of the two components.

The concept of feedback is also central in *system dynamics*, a general discipline that was developed by the cybernetics research group at Massachusetts Institute of Technology (MIT). The fundamental idea behind system dynamics is that all systems, no matter how complex, consist of networks of positive and negative feedbacks, and all dynamics arise from the interaction of these loops with one another [Sterman 2000]. Thus, the goal of system dynamics is to understand the basic principles of feedback and with the use of these to find management policies and organizational structures that lead to greater success [Forrester 1961]. Furthermore,

feedback is also the main driver of learning. Thus, effective decision-making and learning requires expanding the boundaries of managerial and engineering mental models, and developing tools that foster systems thinking, *i.e.,* the understanding of how the structure of complex systems creates their behavior [Sterman 2000].

Operations Research (also called the *management sciences*), another forerunner of modern systems engineering, was developed prior to and during World War II in the USA and Great Britain with the pragmatic goal of improving military operations through the use of mathematics. The founders of the field of operations research came from diverse backgrounds, including physics, mathematics, engineering, and economics. These researchers teamed up to use mathematics to solve complex problems like logistics, precision bombing, or radar development and implementation. After the war, business had learned the practicality of this discipline that made operations research possible to expand to commercial areas.

By the 1970s the use of computers for mathematic computation and modeling presented new opportunities for growth in operations research. The field of operations research today is integrated into many disciplines, including the military, government policy, medicine, transportation, computer sciences, and business. Its defining characteristics are the applications of mathematics, physics, and systems thinking to solve problems.

After the short description of traditional systems sciences, basic systems engineering definitions and approaches are presented in the next section.

B.3. SYSTEMS ENGINEERING IN LITERATURE

Systems engineering literature includes various definitions and procedures that explain the basic essence of this discipline. While the models and procedures in this chapter describe the fundamental philosophy of systems engineering, which is a unique way of thinking about the products and processes of engineering, it is important to present them in this early section of this thesis and thus provide a sound basis for the theoretical findings in the later sections.

As the previous section showed, systems engineering is the application of systems theory to the development of engineering systems. As Blanchard & Fabrycky [1990] suggest:

> *…systems engineering is a process employed in the evolution of systems from the point when a need is identified through production and/or construction and ultimate deployment of that system for consumer use.*

Hazelrigg [1996] defines systems engineering as *the treatment of engineering design as a decision-making process*. Furthermore, he compares systems engineering to the three-step process of strategic planning. As *Figure B.1* depicts, systems engineering starts with the identification of a need through the honest assessment of the actual situation described by the extant technology. The second step of systems engineering is the definition of the system objectives, *i.e.,* a statement where the stakeholders of the system want to be in some time in the future. The third element of the system in *Figure B.1* is the systems engineering process that shows the way from the current situation to the satisfaction of the future needs.

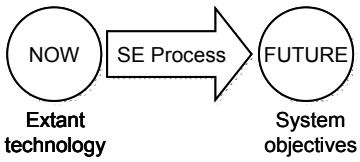


Figure B.1 Systems engineering process

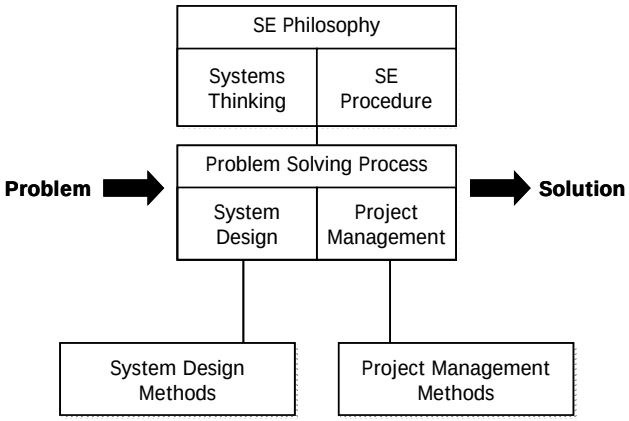


Figure B.2 Components of the Systems Engineering Methodology [adapted from Haberfellner et al. 2002]

Haberfellner *et al.* [2002] consider systems engineering a methodology for problem solving, where the *problem* is defined as the difference or gap between the actual system state and the targets or the system objectives describing the future system state (see also [Sterman 2000] for a similar definition). Furthermore, systems engineering provides the model-based methods and tools that help find the ways of solution that bridge the gap between the actual and the target state of the system. As *Figure B.2* depicts, the *problem solving or systems engineering process* is the central element of the systems engineering methodology. It consists of two separate components: (1) system design, the main constructive task for problem solving; and (2) project management, the task of organizing and coordinating the problem solving work. On the one hand, the *systems engineering philosophy,* involving *systems thinking* and the *systems engineering procedure* in the top in *Figure B.1*, provides guidelines to the problem solving process. On the other hand, *system design* and *project management* supports the problem solving process with traditional techniques and methods [Haberfellner *et al.* 2002].

Igenbergs [2000], as a traditional system theorist, considers systems engineering an independent discipline that provides model-based methods for problem-solving in general. Hence, the systems engineering procedure used by Igenbergs depicted in *Figure B.3* is a generic approach for the definition and analysis of a model to solve a complex problem. The same systems engineering procedure, a basic mental model for systems engineering problem solving in the German-speaking systems engineering literature, can also be found in [Haberfellner *et al.* 2002]

The iterative decision-making procedure in *Figure B.3* starts with the evaluation of the state-of-the-art and identification and description of the problem. In *step 2*, the system objectives and

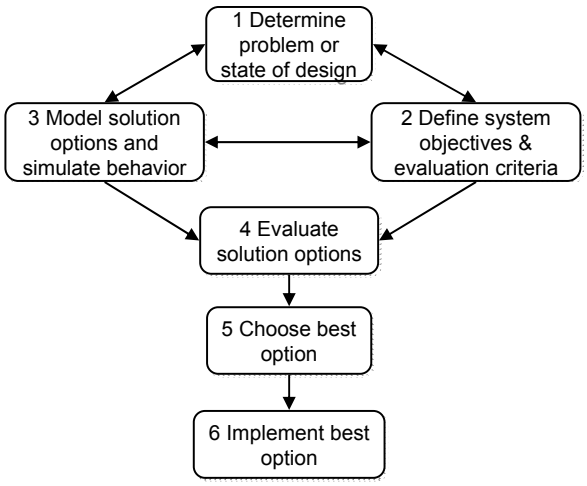


Figure B.3 Systems engineering procedure (adapted from [e.g., Igenbergs 2000])

quantifiable evaluation criteria are defined to translate the problem into engineering language applicable at the evaluation of the solution options. The options modeled in *step 3* are alternative ways (systems engineering process options) to solve the problem defined in *step 1* (*e.g.,* to improve the inadequate aspects of the system design). These solution options are then evaluated against the criteria in *step 4*. The best option is selected in *step 5* and implemented in *step 6*. This generic iterative decision procedure is usually used to solve diverse problems at various system levels at different times and system maturity levels in the SD.

Another distinguished theory for the selection of the best option during decision-making was proposed in [Hazelrigg 1998]. The proposed decision analysis process is similar to the basic systems engineering procedure in German-speaking literature as described previously. During this process, the decision makers (1) clarify the problem or situation, (2) identify options for the solution, (3) determine the expectations on each option, (4) express values for each option, and (5) build a rank order of the alternatives to choose the best option.

The kinds and number of options vary at each decision in the SD. While a solution to a simple problem can result in a yes/no answer to a design question, other more complicated problems require the assessment of multiple design aspects and thus the solution has to cover all these aspects, *i.e.,* it has to include an answer to all design questions that comprise the problem.

While systems engineering can be generally considered as a discipline supporting engineering problem solving, technical standards provide a more detailed description of the role of systems engineering in the development process of complex engineering systems. In the following sections, first of all, the definitions for systems engineering and the term "system" in technical standards are listed, and then the author's personal understanding of this topic is described.

B.4. SYSTEMS ENGINEERING IN TECHNICAL STANDARDS

Technical standards agree that systems engineering deals both with the *system being developed* (the product system) and the *system that does the developing* (the producing system) [NASA 1995]. Furthermore, these standards usually divide systems engineering into two significant disciplines: the *technical knowledge domain* in which the systems engineer operates, and *systems engineering management* [DoD 2001a]. The next part of the thesis shows how technical standards and guidebooks define systems engineering (for more definitions see the section *Glossary of Terms*):

- *A logical sequence of activities and decisions that transforms an operational need into a description of system performance parameters and a preferred system configuration* [MIL-STD-499A]

- *An interdisciplinary, collaborative approach that derives, evolves, and verifies a life-cycle balanced system solution which satisfies customer expectations and meets public acceptability* [IEEE P1220].

- *An interdisciplinary engineering management process that evolves and verifies an integrated, life-cycle balanced set of system solutions that satisfy customer needs* [DoD 2001a].

- *A robust approach to the design, creation, and operation of systems. In simple terms, the approach consists of identification and quantification of system goals, creation of alternative system design concepts, performance of design trades, selection and implementation of the best design, verification that the design is properly built and integrated, and post-implementation assessment of how well the system meets (or met) the goals. The approach is usually applied repeatedly and recursively,*

with several increases in the resolution of the system baselines [NASA 1995].

Now that the term systems engineering has been thoroughly defined, the main object of this discipline, the *system* is introduced.

B.5. COMMON DEFINITIONS OF SYSTEMS

There are many different definitions for the term *system* in systems engineering literature and standards. Organizations like INCOSE, NASA, and US Deptartment of Defense (DoD) define a system as follows:

- *A system is an interacting combination of elements to accomplish a defined objective. These include hardware, software, firmware, people, information, techniques, facilities, services, and other support elements* [INCOSE 2002].

- *A system is a set of interrelated components, which interact with one another in an organized fashion toward a common purpose. The components of a system may be quite diverse, consisting of persons, organizations, procedures, software, equipment, and facilities* [NASA 1995].

- *A system is a composite of equipment, subsystems, skills, and techniques capable of performing or supporting an operational role* [MIL-STD-499A]

General definitions in systems engineering literature are also quite similar to these. For example, Terry Bahill proposes a quite simple definition: *a system is any process that converts inputs to outputs* [see INCOSE 1998a]. Von Bertalanffy [1968] suggests a longer definition:

The notion of a system may be seen as simply a more self-conscious and generic term for the dynamic interrelatedness of components.

A system defined by Igenbergs [2000] is an object with the following four characteristics:

- *it consists of elements*

- *the elements have attributes*

- *the interaction between elements is described by relations*

- *an element can be a system*

This last definition accounts for the structural, functional, and hierarchical characteristics of the system. Furthermore, this definition includes the basic rules for systems modeling, because it defines the fundamental modeling elements: the system components, their attributes (properties and functions), their interactions, and the hierarchy in the system.

Finally, Moses [2004] provides a definition of *engineering systems* analogouos to the previous general system definitions:

engineering systems are systems designed by humans having some purpose and are composed of interacting parts.

B.5.1. ZOPH Model

A more specific definition for the *SD system* was proposed by Negele [1998] based on earlier work by Patzak [1982]. As the standards listed before, Negele also divides systems involved in the SD into three categories: the objectives of the SD (*i.e.,* the *goal system*), the systems being developed (*i.e.,* the product or the *product system*) and the systems that do the development. The developing systems are then further classified into the work to be done (*i.e.,* the *process system*) and the persons and groups of persons that accomplish this work (*i.e.,* the *organization system*). The name of this system model is *ZOPH model*. ZOPH is an abbreviation built up from the first letters of the German terms for the four system types in the model, (*i.e.,* **Z**ielsystem – goal system; **O**bjektsystem – product system; **P**rozesssystem – process system; and **H**andlungssystem – organization or agent system). Additionally, the four systems of the ZOPH model are interacting with the *system environment* through the *system boundary*.

Another similar model for the SD system was proposed by Browning *et al.* in [2006]. This model includes one additional element: the *tool system*, which comprises the tools and technologies required to develop and fabricate a product that fulfills the system objectives. In the ZOPH model, tools and technologies are included in the organization (or agent) system. However, it is sensible to define a separate system for technology, particularly if technology development is divided and handled separately from SD in the organization.

This thesis proposes the separation of technology and system development as a key requirement for adaptive SD. Therefore, the SD system is modeled here as a system that is made up of the five interacting components with the process system in its core (*Figure B.4*). This modified ZOPH system (or ZOPH+T to distinguish it from Negele's original ZOPH model) is a grand model of the SD enterprise that develops diverse systems and products in a multi-project environment. All the different projects represented by the multiple *process system* arrows in the middle of *Figure B.4* are run by the persons involved in the *organization system* of the enterprise, and use technologies (methods, tools, equipment, *etc.*) included in the *technology system*. The technology system comprises the existing and emerging technologies from the technology development section of the enterprise. Technology development employs parallel processes to the SD processes that deliver the technologies which are applied later in the SD. These processes are also included in the parallel arrows of the process system in the middle.

The SD processes deliver various systems (enabling and end products of the SD) included in the *product system*. All four subsystems operate according to the organizational and strategic goals of the SD enterprise represented by the *goal system*. The goal system, as all other subsystems as well, is hierarchic including enterprise- and project-level goals. The strategic goals and vision of the enterprise are in the top hierarchy level of the goal system providing a sound basis for every decision concerning all other subsystems in the enterprise. Furthermore, all these five subsystems depend on and interact with the external environment of the SD enterprise (*i.e.,* the market, suppliers, government, *etc.*).

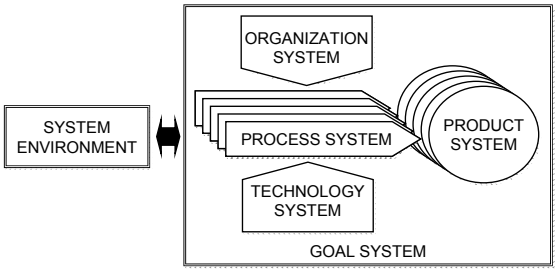


Figure B.4 ZOPH+T system development system model (modified from [Browning et al. 2006, Negele 1998])

The ZOPH+T model represents an interrelated, dynamic system, where a change in one component might change all other components of the system. The detailed modeling of both the system components and their interactions is therefore the key for effective system management. The next section describes a generic approach for the modeling of the elements of each ZOPH+T subsystem.

B.5.2. IPO Notation

A generic approach for the modeling of systems and their components is the *IPO* approach [*e.g.,* Negele 1998] describing the main characteristics of a system element operating in an (internal or external) system environment according to the above definitions. IPO, an abbreviation of input-(process, product, person or purpose)-output, is a generic, object oriented modeling technique applicable to any kinds of systems.

A system element which uses the object oriented IPO notation is depicted in *Figure B.5*. System elements are characterized by their attributes (inputs, outputs, properties, and functions), and connected to other elements through element relations. These element attributes form the system-level characteristics of a system, the key parameters that together express the overall value of a system. Thus, modeling helps to quantify system performance and quality, and thus provides key information for the quantification of the value of the system for its stakeholders.

The generic IPO notation includes the following main modeling elements that can be used to model any kinds of systems:

- *Inputs and outputs:* interfaces between the elements and their environment. Other system-internal or -external elements influence the system element by providing the element with the outputs they produced, *i.e.,* the results of their behavior.

- *Properties:* The states, effects, and behavior of the elements are described through their properties. Based on the actual values of the properties, the characteristics of an element can be determined.

- *Functions:* Functions describe the dependencies between the inputs, properties, and outputs of system elements. The results of the functions, (*i.e.,* the element outputs) depend on how the inputs influence the properties of the element. Thus, functions describe how the element transforms inputs into outputs.

- *Relations:* Relations are the interrelations or dependencies among system elements. Relations between system elements form a network of causes and effects that describe how the system operates [Sterman 2000]. Hence, the network of relations shows the underlying logic of a system, *i.e.,* the characteristics of the causal network of elements in a system.

Modeling (*e.g.,* using the IPO notation) fosters *parameter-based SD*. That is, the application of models and modeling approaches requires the qualitative, metrics-based definition of the system objectives, and thus decision and success criteria for the project. These quantitative criteria are derived from the system requirements and support the control of the implementation of the

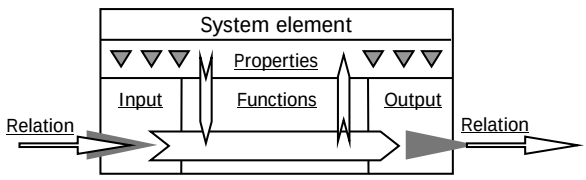


Figure B.5 Generic IPO system element (adapted from [Negele 1998])

system requirements in the design.

In parameter-based SD, the same hierarchic structures of design parameters are used to define objectives for the development work, verification, and validation of the design, the management, and quantification of risks incorporated in the SD, and the control of the development project and valuation of the achievements of the SD. Furthermore, modeling and parameter-based SD generates reproducible development information and drives the understanding and reuse of development knowledge among projects.

The ZOPH+T model and the IPO approach are fundamental building blocks of the adaptive SD theory described later in this thesis. Subsequent chapters will show how they can be tailored and applied to fulfill the purpose of adaptive development.

B.6. SYSTEMS ENGINEERING IN THIS THESIS

Systems engineering in this thesis is considered as the discipline that applies a model-based approach to understand, describe, decompose and integrate the different dimensions of SD in one adaptive system that permanently interacts with its environment and adapts to it in a controlled manner. That is, systems engineering helps handle the complex internal and external relations among the elements of the SD system and its environment. Furthermore, systems engineering collects and analyzes feedback on the behavior of the SD system in its operational environment, and provides quantitative information on its actual performance with regard to the system objectives.

The major task of systems engineering is to *define a process* that guides the developers through the complex journey of SD. SD activities are the basic building blocks of this process and have to be organized in an orderly fashion to follow the evolution of the system design and support this evolution by producing the necessary information at every stage of the SD. In addition, this process has to foster the timely integration of development products delivered by the various groups conducting the SD. The exchange of deliverables of SD tasks between engineering teams forms an *information flow* in the process. This information flow (*i.e.,* interactions among activities

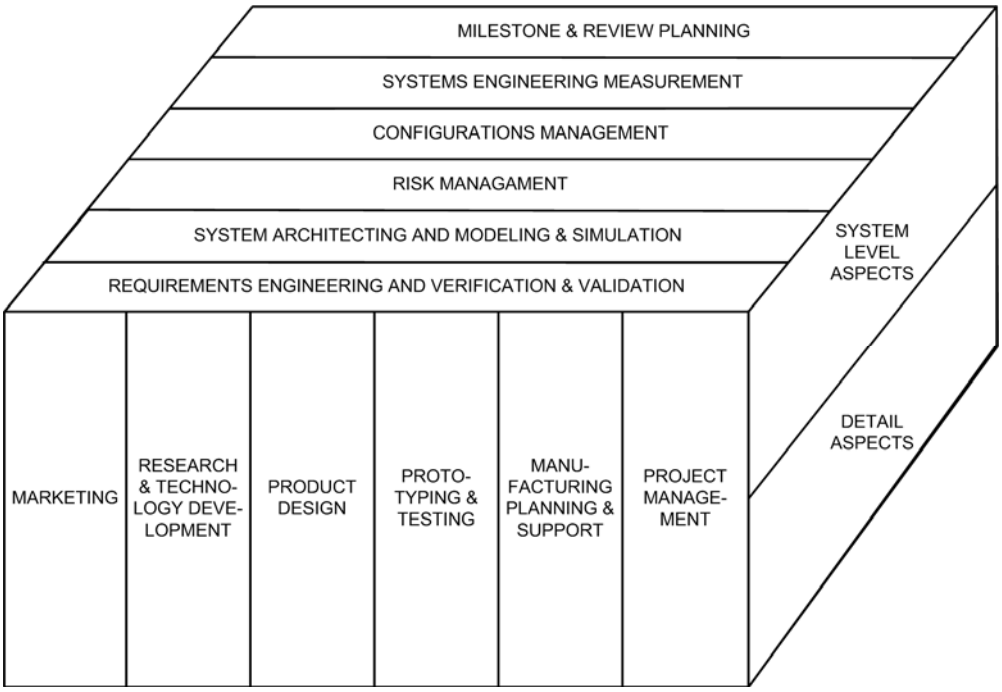


Figure B.6 Dimensions and functions of a system development system

and thus the SD staff) creates the underlying logic and thus the structure of every SD project.

The different dimensions of SD and systems engineering are depicted in *Figure B.6*. The front of the cube represents the most important tasks of SD that define the final characteristics of the developed system. Systems engineering employs modeling and analysis to create system-level representations of the design that foster the effective fulfillment of these tasks by showing a transparent, realistic picture of the difficulties and opportunities of the SD.

The SD as a hierarchic system involves two basic levels: *system-level* and *detail-level work*. The goal of system-level development is to ***define and maintain an evolving vision*** for the developers, a reference that includes the most important functions and aspects of the design, and shows the right direction for the project. These system-level descriptions of both the product and the process are frameworks that are continuously filled with more and more detailed and complex information delivered by the various disciplines of the SD.

The system-level design and the SD process plan show the strong and weak points of the SD system and draw the attention of the developers to the critical design aspects that might cause problems during their detail-level work and later during the integration of the developed system components. Furthermore, the system-level design includes the exact specification of the most important characteristics of the main modules and interfaces that build up the system. These *system-level design rules* drive and bound the development at the detail-level and guarantee that the small puzzle pieces of detail-level design (*i.e.,* the components and modules of the system) fit together during integration and testing.

The top of the cube in *Figure B.6* depicts the main functions of systems engineering that foster the understanding, description, optimization, and control of the complex process of SD. Systems engineering work starts early in the SD with modeling and analyzing the main aspects of system-level development, and it continues until the end of the SD. Modeling is applied during this process, because models are reusable knowledge bases that can be continuously updated and detailed as new information from the project emerges. Hence, the characteristics of the system models provide a realistic picture of the state of the SD: the problems already encountered in the past and anticipated to arise in the future.

The major functions of systems engineering generate a network of development knowledge in a systematic manner early in the project. The deliverables of systems engineering are highly dependent on each other and together they form a complete set of system-level SD information. These disciplines are as follows:

- **System architecting:** *The system architecture is the arrangement of elements and subsystems and the allocation of functions to meet system requirements. System architecting is the task that delivers the system architecture* [INCOSE 1998a].

  Modeling and simulation: *These tasks provide virtual duplication of products and processes, and represent those products or processes in readily available and operationally valid environments. Use of models and simulations can reduce the cost and risk of life cycle activities* [DoD 2001a].

  Model-based system architecting supports the understanding and thus the early generation of knowledge concerning each of the five subsystems of the ZOPH+T model. Furthermore, modeling and simulation support the work in all dimensions of the SD enterprise in *Figure B.6*. Modeling increases transparency regarding the challenges of the SD and reduces the complexity of the problems to be solved.

- **Requirements engineering:** *A requirement is an essential condition that a system has to satisfy* [ISO 2382-20]. Requirements engineering comprises the process of elicitation, analysis, specification, validation/verification, and management of requirements.

  Requirements engineering deals with the definition of the goal system in the ZOPH+T SD enterprise. Requirements are defined during the system design stage of the SD, and provide the specification of the design at every hierarchy level. Thus, *requirements are one representation of the design* that shows the characteristics the final system has to fulfill. Requirements are defined, detailed, and validated gradually according to a successively refined framework of engineering design.

- **Validation:** Ensuring that the right system is being built in the SD project, *i.e.,* writing specifications and checking performance to make sure that the system does what it is supposed to do.

  Verification: Ensuring that the system is built right in the SD project, *i.e.,* ensuring that the system correctly implements the specifications.

  Verification and validation (V&V) represents the intersection of systems engineering and testing. Hence, the purpose of testing is to verify technical performance, operational effectiveness, and suitability, and provide essential information to support the decision-making [DoD 2001a]. The key benefit of V&V is that it reduces uncertainty in the SD by generating useful information about the functionality and manufacturability of the product design. V&V activities are typically followed by corrective changes or rework.

- **Risk management**: *In the context of industrial systems engineering, risk management is the recognition, assessment, and control of uncertainties that may result in schedule delays, cost overruns, performance problems, adverse environmental impacts, or other undesired consequences* [INCOSE 2002].

  Hence, risk management helps understand what can go wrong in the SD, how critical are the risks associated with these possible failure modes, and how to handle and control these identified risks. Furthermore, risks represent the requirements and design aspects that deserve particular attention during SD, the weak points of the system that might jeopardize the whole project if not mitigated effectively. That is, risk management classifies the elements of all five subsystems of the ZOPH+T model by their criticality, proposes actions to reduce these risks, and controls the success of risk mitigation actions by tracking the changing risk status of each subsystem throughout the development project.

- **Configuration management**: *A discipline applying technical and administrative direction and surveillance to identify and document the functional and physical characteristics of configuration items; control changes to configuration items and their related documentation; and record and report change processing and implementation status.* [MIL-STD-480B]

  Configuration management defines the major states of the SD, *i.e.,* the required maturity of the main deliverables of the most important stages of the SD process. Configuration management describes how the product system of the ZOPH+T model dynamically evolves during the project, and how the characteristics of the design elements change during the

project. Configuration management is one important tool to handle the dynamic complexity of the SD system.

- **Systems engineering measurement:** *The process of assigning numerical values to process, product, or project attributes according to defined criteria. This process can be based on estimation or direct measurement. Estimation results in planned or expected measures. Direct measurement results in actual measures* [INCOSE 1998b].

  Measurement is a control function of systems engineering that collects and analyzes the actual status of the major requirements (both on the technical and managerial levels) and provides a clear picture of the risk status in the project. The results of systems engineering measurement help "tune" the SD process by putting development efforts in the critical areas (*i.e.,* high-risk areas).

- **Milestone and review planning:** *The systems engineer measures design progress and maturity by assessing its development at key event-driven points in the development schedule. The design is compared to pre-established exit criteria for the particular event to determine if the appropriate level of maturity has been achieved. These key events are generally known as Technical Reviews and Audits* [DoD 2001a].

  Milestones and reviews are the major system-level decision points in the project, where the design team evaluates the major deliverables (*i.e.,* configuration items) of the SD and the collected feedback from the internal and external project environment, and decides on the feasibility and value of the achievements of the development. Reviews and milestones are the points where systems engineering integrates its deliverables and provides a system-level overview on the accomplishments of the project compared to the requirements. Thus, milestones and reviews are the key points to measure the performance of the SD. Furthermore, the complex web of these decision points gives its structure to the project, because each review and milestone is a starting and final point of a major development stage or sub-process.

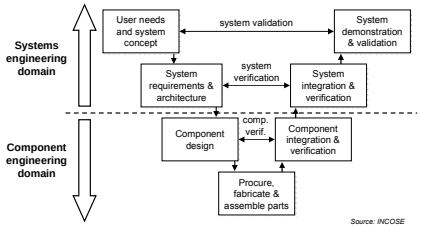
It is indisputable that these systems engineering functions are vital for the final project success. However, different SD projects with different characteristics require different systems engineering efforts. The determination of the required effort in each function of systems engineering is out of the research scope of this thesis. *Thus, in this thesis the assumption is made that the SD enterprise employs a sound mix of the systems engineering functions in order to maximize the overall project value and thus the profitability of the SD endeavor.*

B.7. CHAPTER SUMMARY

Adaptiveness is an emergent system characteristic and thus, a holistic view and the application of systems engineering methods are inevitable at the design and analysis of adaptive engineering systems. Model-based methods, like the ZOPH+T or IPO approaches provide an effective means to capture the most important characteristics of systems and their elements, and optimize their architecture and behavior with respect to the purpose of the system. SD enterprises are considered as adaptive systems in this thesis, and systems engineering is the model-based philosophy that supports the effective design, creation, operation, and control of these SD enterprise systems.

C. SYSTEM DEVELOPMENT LIFECYCLE MODELS AND PHILOSOPHIES

C.1. CHAPTER ABSTRACT



Lifecycle models represent the philosophy of a company regarding the way it develops its products. Hence, the applied SD lifecycle models fundamentally affect the effectiveness and efficiency of the SD work and the results of this work in the given SD environment. In this part of the thesis, lifecycle models are reviewed regarding their applicability in dynamic SD environments, where dynamics means the predictable and often unpredictable behavior of internal and external project factors. To be able to successfully deal with the dynamic SD context, the first step in this thesis is to identify appropriate SD lifecycle models and thus philosophies that account for uncertain and changing SD characteristics. The sequence of lifecycle models described in this chapter represents the trend of thinking about SD projects from the conventional linear or sequential working style to the iterative, evolutionary, or agile models of modern SD projects.

C.2. SYSTEM DEVELOPMENT LIFECYCLE

SD projects are usually divided into project phases with clear objectives and deliverables to provide this process with a logical structure, better management control, and appropriate links to the ongoing operations of the performing organization. Collectively, the project phases are known as the project life cycle [PMI 1996]. Ulrich & Eppinger [2004] separate SD into five major stages or phases: (1) Concept Development, (2) System-Level Design, (3) Detail Design, (4) Testing and Refinement, and (5) Production Ramp-Up.

The phases in the generic model of Ulrich & Eppinger represent the main system-level steps of the SD project that are necessary to transform the customer's needs into market products. Usually each industry segment has different lifecycle models comprising phases with different names. However, the basic logic behind each model is analogous to the one presented here.

The need for such a guidance and control tool, like the lifecycle models, arose as the size and complexity of the developed systems reached such a high level that project managers were unable to effectively manage the projects. *While the decomposition of a complex problem into sub-problems enhances the ability to solve this problem* [*e.g.,* Alexander 1964, von Hippel 1990], the definition of phases helps project managers to plan and control the project in a better way.

Phases mean major work packages for the project team with clear goals for a manageable period. As major SD projects can last several years or even a decade (*e.g.,* development of an aircraft or spacecraft), it is useful to define smaller units of work for a reasonable period for the people working in the development. The highest level of such working units (*i.e.,* the phases) usually deliver main working products, *i.e.,* major steps towards the complete solution of the problem in form of representations of the system, which conclude an important part of the SD. These deliverables are evaluated at the end of each phase to decide on the *maturity of the design* and the *feasibility of the project plans* defined at the outset of the project. The evolution of systems engineering and design theory during the last decades affected the SD lifecycle models as well. In the following part of the thesis, renowned lifecycle models and the SD philosophies are described.

C.3. WATERFALL LIFECYCLE MODEL

The first and probably most famous lifecycle model for SD is the *waterfall lifecycle model* proposed in [Royce 1970] and depicted in *Figure C.1*. The SD philosophy of the waterfall lifecycle follows the traditional engineering thinking, *i.e.,* the system as a whole is developed top-down as a sequence of development stages. *Figure C.1* shows the main stages and deliverables of the waterfall lifecycle. The dashed lines in *Figure C.1* depict feedback relations between consecutive lifecycle phases that were not part of the original model. That is, the original waterfall lifecycle was a strictly sequential process, where the deliverables (*e.g.,* system requirements or specifications, system design, *etc.*) of the single phases were frozen after the milestone decisions and were not changed or reworked any more in the project. This is a main deficiency of the waterfall model that forces engineers to finalize development products (*e.g.,* system specifications) early in the project without having a chance to prove their real validity and feasibility.

To change this characteristic and increase the flexibility of the SD process, the feedback relations shown in *Figure C.1*, (*i.e.,* the possibility of rework on deliverables from the previous phase) were introduced in the model later. This improved version of the waterfall lifecycle including the possibility for iteration is a more realistic representation of the SD process, where changes in the design are often caused by new information in the development process, *e.g.,* failures discovered during system integration or testing.

Another characteristic of the traditional waterfall lifecycle model is that activities within a phase are performed strictly sequentially, and verification and validation (V&V) activities are usually performed at the end of each phase. Furthermore, the main emphasis is on system-level V&V in the *system verification* phase at the end of the lifecycle. As a consequence, design failures are often found quite late in the SD project, which contributes to major rework and thus, costly budget and schedule overruns. If more than one phase has to be repeated as a result of the failures found during system verification, the project management might decide to terminate the project in spite of the high amount of development efforts already spent on the design. Boehm, the developer of the iterative, spiral lifecycle model wrote the following about the waterfall model in [1986]:

> *Some of its initial difficulties have been addressed by adding extensions to cover incremental development, parallel developments, program families, accommodation of evolutionary changes, formal software*

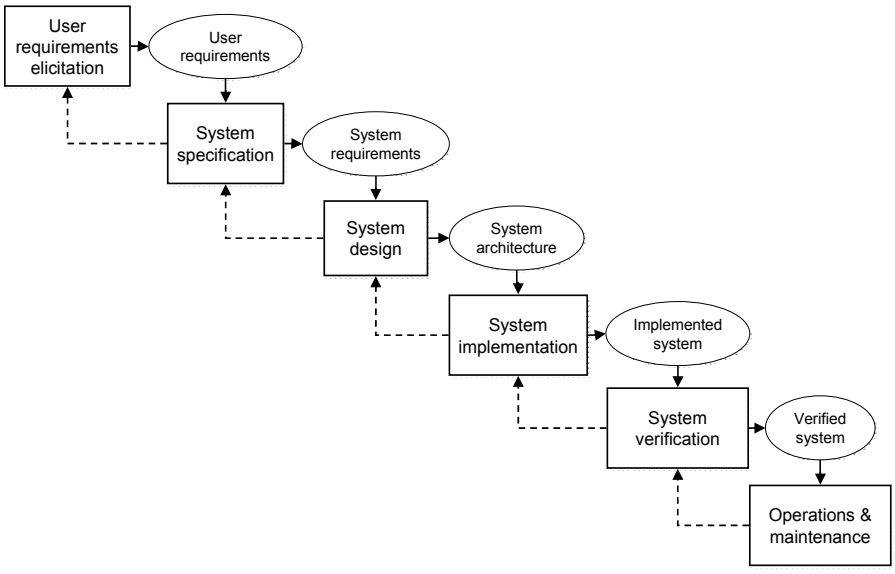


Figure C.1 Waterfall lifecycle model

development and verification, and stagewise validation and risk analysis. However, even with extensive revisions and refinements, the waterfall model's basic scheme has encountered some more fundamental difficulties, and these have led to the formulation of alternative process models.

The evolution of software engineering and the growing importance of software products brought changes in conventional SD. The specific character of software development (*i.e.,* the developing environment and the developed models, prototypes, and final product are all software products) requires an environment that fosters more effective and efficient SD work with regard to these special characteristics. In addition, the evolution of software tools for hardware and embedded systems enables the introduction of software-like SD methods and rapid, virtual system design that call for different SD philosophies, too.

C.4. V LIFECYCLE MODEL

The *V model*, an improvement of the waterfall model, was originally developed to regulate the software development process within the German federal administration. It describes the activities and results that have to be produced during software development. The current version of the V model is the V model XT [V-Model XT Website] which was finalized in February 2005.

Though the V model was originally designed for software projects, it was proven to be suitable for SD projects including the development of both hardware and software components. Hence, the V model is nowadays a widely applied lifecycle model in SD [INCOSE Website]. The shape of the V model depicted in *Figure C.2* characterizes a philosophical change in SD. The left tail of the V represents the *system specification stream*, where the system requirements and the system and subsystem or component designs are specified. The designed components are then fabricated in the bottom part. Component fabrication is followed by the *testing stream* in the right tail of the V, where the gradually evolving and growing system is verified against the specifications defined in the right tail of the V.

A main benefit of the V model is that it separates the disciplines of system and component engineering. This way, *top-down* and *bottom-up* development approaches are integrated in the V model. That is, the system is specified top-down and then the subsystems are designed bottom-up. Additionally, the definition of distinct steps for the design at different hierarchy levels that appears first in the V model fosters the efforts of decomposing the system into independent subsystems and components. These subsystems can be then designed and fabricated *in parallel* according to the system specifications defined in the previous phase. When it comes to the development of highly complex systems, the independent, concurrent development of

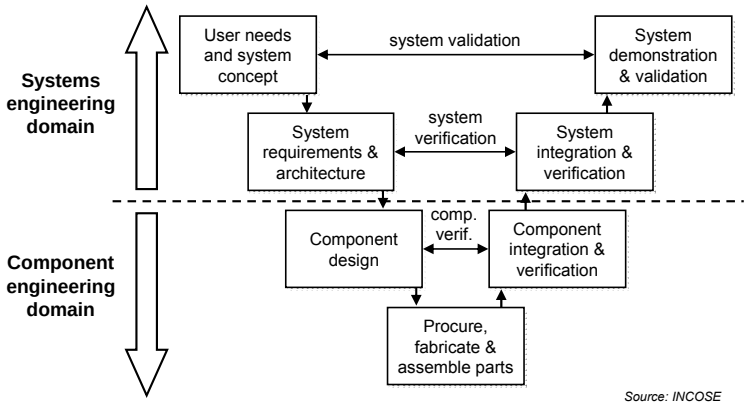


Figure C.2 V lifecycle model

subsystems is a great possibility to accelerate the design process, and it also supports a better involvement of suppliers in the SD [Eisenhardt & Tabrizi 1995].

Another benefit of the V model is that it breaks down system experimentation and V&V into three separate stages. These three main stages, shown in the right tail of the V in *Figure C.2*, form three iteration loops in the development of the system with increasing scope and complexity. The first design loop is on the component or subsystem level. In case of a *modular design,* the subsystem verification loops can be performed in parallel, independently of each other [Baldwin & Clark 2000]. That is, the three phases of *component design, fabrication, and verification* in the bottom of the V consist of numerous parallel Vs, as many as many subsystems build the system.

During the development of complex systems like aircraft or automobiles, the development of certain subsystems is outsourced to subcontractors. These suppliers conduct the complete design, development and testing of the subsystem, and deliver the system developer the finished design. Thus, in these cases the development of the subsystem can be considered as an independent SD project, where the customer is the contractor system developer company.

The second loop of system design involves system-level design verification. In this loop, the integrated design is verified against the system specifications delivered in the second lifecycle phase in the left tail of the V. Unambiguous and robust subsystem and interface specifications, and a thorough subsystem-level verification facilitate "smooth" system-level verification. The third and last design iteration loop in the V model is the *system validation loop*, also called system qualification. The outcome of this usually very long, expensive, and comprehensive system test process has the objective to prove that the developed system satisfies the customer's needs as well as industry and government regulations. Major design failures found in this phase are extremely costly to correct, and they jeopardize the success of the whole project.

The main benefit of the V model is that it introduces iterations and hierarchy in the SD. It is also a quite useful tool for the integration of the system suppliers in the development project. However, the V model is not applicable for evolutionary SD, where the lifecycle consists of several small iteration cycles to foster frequent prototyping and the collection of timely feedback from the customer. These aspects drive the definition of iterative and incremental lifecycle models described in the next part.

C.5. INCREMENTAL AND EVOLUTIONARY LIFECYCLE MODELS

After the introduction of the two traditional, sequential lifecycle models of SD, this part of the thesis discusses four lifecycle models for incremental and evolutionary SD: (1) Boehm's spiral lifecycle model, (2) Evolutionary lifecycle model, (3) Incremental lifecycle model, and (4) Agile.

These iterative and evolutionary development approaches were developed due to the obvious disadvantages of the sequential waterfall model for software and system development to deal with the high complexity and dynamics of the SD environment. According to the CHAOS study of the Standish Group in 1998, top failures of 23,000 analyzed projects were associated with waterfall development practices. One of the report's key conclusions was to adopt incremental and evolutionary SD:

> *Research also indicates that smaller timeframes, with an early and frequent delivery of system components, will increase the success rate. Shorter timeframes result in an iterative process of design, prototype, development, test, and deployment of small elements.*

All four models described in this section are iterative in nature and were originally developed for software projects, but they are also suitable for system development projects. The four SD philosophies are discussed in one chapter, because the fundamental concepts behind them are similar. That is, the development lifecycle of a large, complex system is divided into smaller iteration cycles delivering systems or prototypes of systems with evolving performance and capability. These evolving prototypes, versions or increments allow an early and continuous validation of the system with the customer and the consideration of the customer's feedback in the next version.

With evolutionary and incremental SD projects, prototypes, products, or product versions are released more often with a lower rate of innovation between two versions than with traditional products. Hence, due to the frequent product release and the continuous collection of feedback from the customers, design options can be kept open and incorporated in a following version in case of respective positive feedback from the customers [Smith & Reinertsen 1998]. On the other hand, modules including disliked functionalities can be quickly upgraded or substituted by other, alternative modules to validate the product quickly in the market again.

The goal of iterative SD lifecycles is to support learning and effectively transfer knowledge between development projects. Evolutionary and incremental projects are considered as experiments [Smith & Reinertsen 1998, Thomke 2003] where rapid development cycles provide evolving product versions that can be tested in the market. Feedback on the released products is collected and incorporated in the next product release with improved functionality. Since the invested development effort is significantly lower than with traditional development projects, the risk of lower product quality is also rather low. Furthermore, the lower product quality can be quickly increased by improving the weak parts of the product and releasing a new, improved version at the end of the next iteration cycle.

Another characteristic of iterative SD lifecycles is that they allow *higher flexibility* in the project than traditional models. One reason for higher flexibility is that the management of such projects is *workstate-driven* [Highsmith 2000], not *workflow-driven*. That is, the main driver of project performance with iterative development projects is the maturity and actual performance of the design. This is a significant difference to traditional project management, where project performance is usually measured with the earned value of the project, *i.e.,* the programmatic project performance. Hence, the iterative SD approaches described next strive to maximize the desired performance and thus the value of the delivered products. As the value of the product is the central element of iterative SD, not the rigorous execution of the project schedule, the SD process must be flexible to allow the adaptation of the process architecture to the actual state of the design. During adaptation, the SD effort in the design-critical phases is increased and activities and even phases that do not contribute to the actual development goals are downsized or even deleted.

C.5.1. Boehm's Spiral Model

The "evolving, risk-driven approach for software development" or the *spiral model* is a lifecycle model (*Figure C.3*) for information technology (IT) projects integrating features of the prototyping model and the waterfall model, in an effort to combine advantages of top-down and bottom-up concepts. The spiral model was defined by Boehm in [1986]. This model was not the first to discuss iteration, but it was the first model to explain why iteration matters [Wikipedia Website]. Additionally, this model provided an excellent management framework for the risk-driven, iterative development of software systems.

The spiral model builds on the existing waterfall model and early evolutionary development models (*e.g.,* prototyping model [McCracken & Jackson 1982, Gladden 1982]) to integrate them

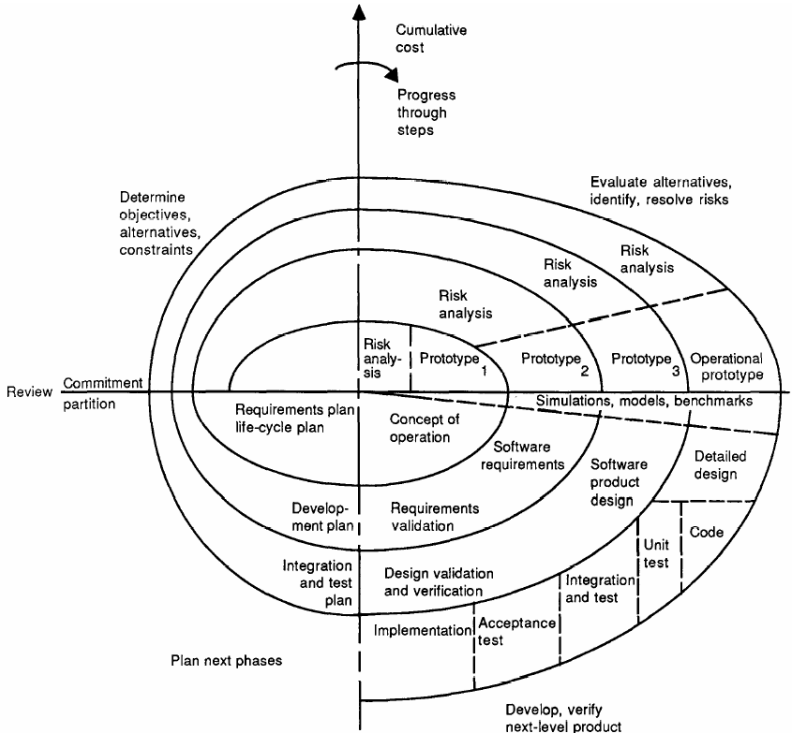


Figure C.3 Boehm's spiral model [adapter from Boehm 1986]

into an effective lifecycle model for IT systems. The spiral approach can be used as a traditional waterfall model in case of low risk and inflexible design structure, but it can also applied as an evolutionary development model in case of high risk and higher design flexibility. In the first case, the lifecycle will be long and without much iteration, while in the latter case, the evolutionary characteristic of the spiral model can be utilized, and many short iterative lifecycles can be performed. Boehm's spiral model provided the basics for the definition of modern evolutionary and incremental lifecycles described in the next parts.

C.5.2. Evolutionary Lifecycle Model

Evolutionary development is an iterative and incremental approach to software and system development. Instead of creating a comprehensive artifact, such as a requirements specification that is reviewed and accepted before creating a comprehensive design model, the critical development artifacts are evolved over time in an iterative manner. The evolutionary development model divides the development cycle into smaller projects, using the incremental waterfall model in which customers are able to get access to the product at the end of each cycle (*Figure C.4*). The products released at the end of each mini-project incorporate only parts of the full functionality and capability of the final product, but these products can be demonstrated to the customer, and feedback on the feasibility can be collected. The customers provide feedback on the product for the planning stage of the next cycle and the development team responds, often by changing the product, plans, or process. By breaking the project into smaller, more manageable pieces and by increasing the visibility of the management team in the project, project risks can be addressed and managed.

Evolutionary SD is based on the regular, on-going assessment of customer needs and customer feedback. Hence, in such a project, the customer needs drive the course of development work aiming to always maximize the overall project value. This way, the development system, including the development goals, product, process, and organization, can be continuously improved and adapted to the changing external and internal project environments.

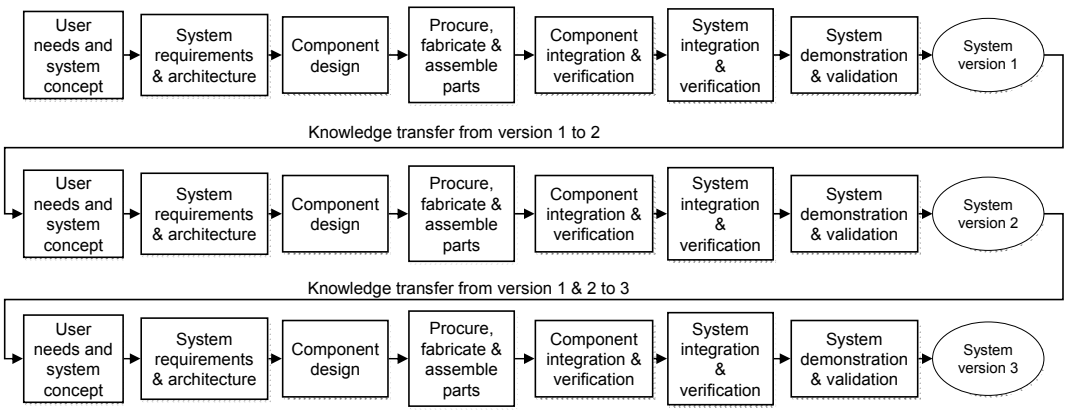


Figure C.4 Evolutionary lifecycle model

It must be noted though that the implementation of the evolutionary SD model demands a mature, flexible design structure and the good consideration and definition of the goals and objectives of the overall development project system (*i.e.,* the evolutionary sequence of mini development projects). Otherwise, if the design is inflexible and its adjustment to the customer's needs is costly and difficult, then the evolutionary development project transforms into an on-going rework effort aiming at implementing changes in a rigid system.

C.5.3. Incremental Lifecycle Model

The incremental development lifecycle shown in *Figure C.5* is an alternative to the evolutionary development lifecycle for highly flexible, complex systems (whereas the evolutionary approach is more suitable for software or systems with a lower complexity). The basic philosophical difference between the two approaches is how the system-level requirements and the system design concept are developed.

In evolutionary SD, the requirements and the system design evolve in parallel with the subsystem design. Thus, this approach is not constrained by long-term plans or considerations, the goals, products and thus the tasks of the project change in every cycle (*i.e.,* mini waterfalls). On the contrary, with incremental SD the project team defines a *grand program plan* or *product road map* [Smith & Reinertsen 1998] in the beginning of the project that includes many prototypes and product versions called *increments,* which will be released in the course of the *SD program.* The flexible program plan is then adjusted after each major change during or after the incremental mini projects to adapt the plans to the actual internal and external stakeholder feedback.

As *Figure C.5* depicts, the incremental model starts with the elaboration of the user's needs, to be followed by a thorough specification and design of the system. In these two phases, the system-level specifications and a grand system architecture including the modules and interfaces are defined. The flexibility of the grand system architecture is vital, because with this lifecycle model, the system is developed as an evolving sequence of increments. That is, in each mini-project of the *incremental SD program* the architecture of the design is improved by completing the previous version with new main functions. These new functions are usually implemented in independent design modules that augment the existing set of modules or replace one or more existing modules.

Hence, incremental SD allows the *independent evolution* of subsystems and modules of the design in fast development cycles. When a new technology is ready for implementation in a subsystem, the company does not have to wait for the whole, rather long waterfall SD cycle, which can last years, but they can integrate the new module in the existing design and launch a new product version or offer the customer a product extension module or a software update

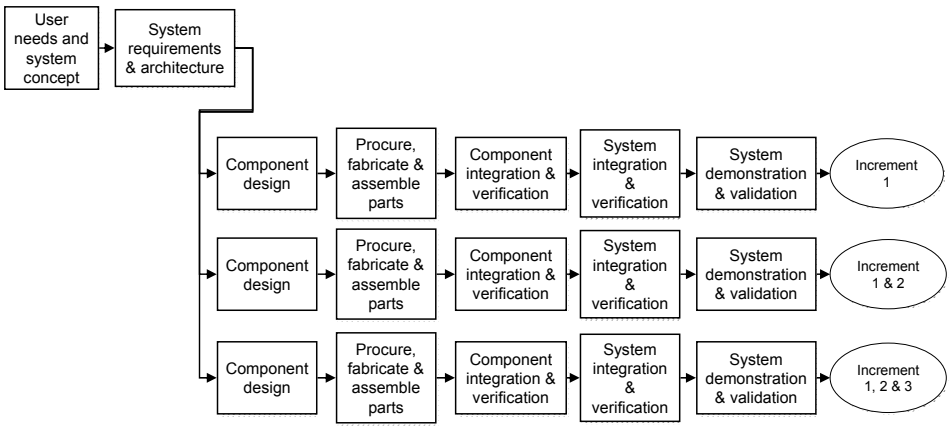


Figure C.5 Incremental lifecycle model

patch. This way, new technologies and design solutions can be inexpensively validated in the marketplace and the high risk of *big-bang* product introduction can be minimized by smaller technology commitments spread throughout the program [Smith & Reinertsen 1998]. Additionally, the continuous marketing work provides important information on the actual market trends and thus, supports the adjustment of the project goals to the customer's needs. This helps optimize the scheduling of technology introduction in the different product versions with regard to the profitability of the product road map.

C.5.4. Agile

The concept of the *Agile Enterprise* originated in 1991, based on a realization that the pace of change in business environment was accelerating and already outpacing the abilities of many established organizations [Dove 2001]. Accordingly, Agility is defined as ***the ability of an organization to thrive in a continuously changing, unpredictable business environment*** [Rigby *et al.* 2000]. Furthermore, an Agile organization has the capability and capacity to gain competitive advantage by intelligently, rapidly and proactively seizing opportunities and reacting to threats [Meredith & Francis 2000]. In fast changing environments, Agility means that the organization can respond to (anticipated or unexpected) changes in proper ways and due time. Additionally, Agile organizations can exploit changes and take advantage of changes as opportunities [Sharifi & Zhang 1999]. To sum up, the philosophy of Agile says that changes are inevitable in the organizational environment and organizations that are capable of responding to and profiting from these changes will be successful in the long term.

Agile is a philosophy stemming from the organizational theory, which was then transferred to other areas, for example manufacturing and software SD. In manufacturing, Agile is used to improve existing practices for flexible manufacturing [Wadhwa & Rao 2003]. Flexibility was introduced in manufacturing to have the capability of producing goods and services to meet a high variance of *anticipated* individual customer's needs with near mass production efficiency [Tseng & Jiao 1996]. A form of flexible manufacturing is *mass customization* [Toffler 1971, Davis 1987].

While flexibility accounts for a predictable variance in the customer's needs (*i.e.,* an expected variety of customized products), it has limited effectiveness in the presence of unpredictable changes. Hence, a new philosophy was needed that considers both foreseen and unforeseen uncertainty and thus supports quick system transformation to any kind of changes in the external environment (*i.e.,* market and technology).

Agile manufacturing systems are flexible manufacturing systems capable of quick adaptation to unforeseen changes in their inputs (*e.g.,* changes in customer's needs, product design,

manufacturing technology, manufacturing resources, *etc.*). In the heart of these Agile systems is an *adaptive production process* including all possible elements of the flexible manufacturing system that can be recombined to allow various modes of production with different product variants as outputs [Cisek *et al.* 2002, Zäh *et al.* 2004].

Though Agile practices are effective during the planning of the repetitive processes of flexible manufacturing systems, Agile and adaptive methodologies for SD systems are still rare. One exception is software development, where Agile has become an emerging, new development methodology. Because in software development every representation of the system is virtual, the quite expensive and long hardware implementation and testing is missing. Thus, every release of the software product including the models and prototypes is an increment of the final product and can be validated directly with the customer. After validation, the product can be adjusted according to the customer feedback, and the software can be verified and validated again. This iterative framework is presented in the next section.

C.5.5. Agile Software Development

In the last part of this section, novel evolutionary and adaptive software development methodologies are described. These adaptive methodologies are collectively called *Agile* software development methodologies. Though Agile methodologies are software-specific, some basic aspects can be also applied for adaptive system development. Hence, the goal of this part of the thesis is to review Agile methodologies and identify development aspects applicable to adaptive system development. The principles of Agile are documented in the *Agile Manifesto* [Agile Manifesto Website]:

1. *Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.*

2. *Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.*

3. *Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.*

4. *Business people and developers must work together daily throughout the project.*

5. *Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.*

6. *The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.*

7. *Working software is the primary measure of progress.*

8. *Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.*

9. *Continuous attention to technical excellence and good design enhances agility.*

10. *Simplicity—the art of maximizing the amount of work not done—is essential.*

11. *The best architectures, requirements, and designs emerge from self-organizing teams.*

12. *At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.*

Agile software development encompasses individual evolutionary and adaptive software development methodologies, like *Evo* [Gilb 2004], *Scrum* [Beedle *et al.* 2000], *Adaptive Software Development (ASD)* [Highsmith 2000], *Extreme Programming (XP)* [Beck 2000], *Crystal* [Cockburn 2005], *etc.* All these methodologies are based on the above-discussed evolutionary lifecycle model. However, Agile software development goes beyond iterative SD by emphasizing the benefits of "lightweight" project structures against the traditional "heavyweight" approaches [Riehle 2005]. *Lightweight* in this context means that these methodologies reject extensive planning and paper-based documentation or elaborate process handbooks and strive to concentrate on "real" development work instead (see the Agile Manifesto above).

As a consequence, these lightweight software development methodologies tend to be not dogmatic with regard to when and how to apply the techniques they provide. The philosophy here is that the developers have to decide on the applicability of the techniques in the actual SD environment and apply them only if appropriate. Thus, these methodologies are *meta-frameworks* consisting of a set of methodology building blocks applicable in certain SD situations (*i.e.,* software development methods). It is possible to tailor these meta-frameworks according to recipes or lessons learned written by practitioners based on their experience with the proposed methods in certain projects.

Though Agile software development methodologies are basically successful with small projects with small team size and complexity, they include valuable hints for complex SD projects as well. For example, the central goal of Agile is customer satisfaction similar to *Lean* [Womack & Jones 1996] or *Six Sigma* [Pande *et al.* 2000]. However, while Lean and Six Sigma are better applicable to stable, manufacturing processes, Agile philosophies focus on iterative SD. Thus, fast, iterative SD work is in the core of Agile, which produces frequent prototypes that can be validated with the customer (*Figure C.6*). As Agile rejects thorough project planning, projects are organized around product releases. A *release* is a piece of development, where the customer gets some new software. Releases can be from 2 weeks to 6 months, but are usually 3 months long. Releases comprise smaller working units called timeboxes. A *timebox* is 1 – 6 weeks long, but usually 3 – 4 weeks. The most important thing about a timebox is that the delivery date is fixed. Meetings follow usually timeboxes, where the project goals and plan are adjusted according to the achievements of the timebox.

The SD work between two releases is iterative, *e.g.,* following the adaptive development lifecycle by Highsmith [2000] depicted in *Figure C.7*. The goal of Agile and adaptive software

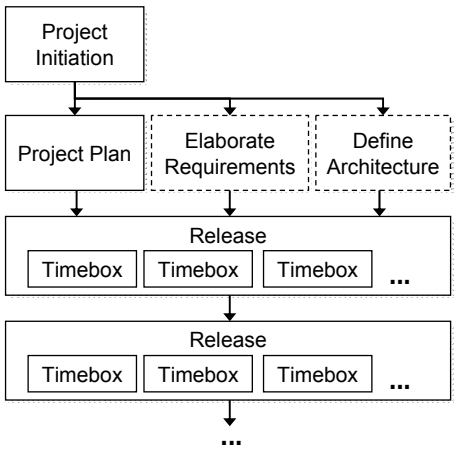


***Figure C.6 Agile software development lifecycle model (adapted from:* [Balagan Website])**

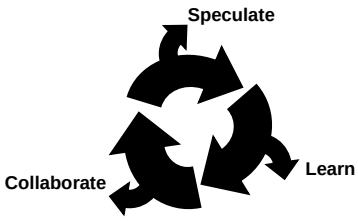


Figure C.7 Adaptive development lifecycle (adapted from [Highsmith 2000])

development is to include customers directly in the design cycles and collect their direct feedback on the achievements of the SD. This feedback is then used to adjust the SD goals and plans and the design itself to maximize project success. The more frequent the feedback arrives from the future users, the easier it is to capture their desires and implement them in the next release. **Thus, Agile SD is an *effective learning process*, where the work*state* of the project defined by the customer's feedback drives the SD work.**

In an adaptive SD culture, the definition of detailed project structure, long-term plans, detailed product specifications, or design architecture is only a waste of effort, because the characteristics of SD change so quickly that all these detailed specifications would have to be changed every week or month. Hence, the effort the detailed planning would require is rather invested in direct SD work, frequent prototype releases, and validation with the user. Thus, the net SD work in Agile is significantly higher than in traditional projects due to the unnecessary documentation. Furthermore, Agile is an environment that welcomes changes, since the simplicity of the product and the lack of documentation supports cost effective, frequent, and even late changes.

In spite of the fact that Agile cannot be directly applied for long SD projects with complex systems as outputs, adaptability that is at the heart of Agile shows novel ways for system development.

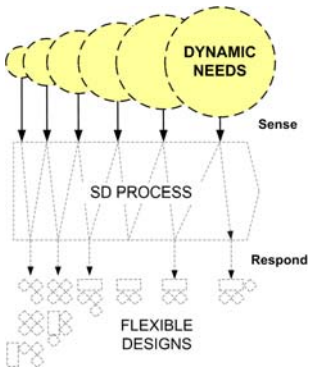
C.6. CHAPTER SUMMARY

Three main classes of SD lifecycle models were presented in this part to describe the evolution of SD philosophies. Which of the described models is the most suitable for a certain project depends on many factors, *e.g.,* organizational culture and flexibility, product type, architecture and complexity, characteristics of technology development and transfer, project team type, experience, and communication, *etc.* However, as the high dynamics and ambiguity of SD environments represent a growing problem for the SD enterprise, SD philosophies have to move towards adaptiveness and thus learn how to change effectively and efficiently, and achieve maximal profit from these changes. Basic principles of adaptive SD systems:

- System flexibility accommodates changes.
- Learning and experimentation are central elements of successful SD.
- A short, value-driven SD with frequent product releases and prototypes allows continuous validation with the customer and thus reduces the risk of changing the customer's needs.
- Adaptive SD is work*state*-driven, not plan-driven.
- System adaptation is a collaborative decision-making process

D. DYNAMIC SYSTEM DEVELOPMENT CONTEXT

D.1. CHAPTER ABSTRACT



The previous chapters introduced the fundamental theory of systems engineering and holistic thinking; and basic models about the way a system can be developed. Now, the following three chapters will discuss the main aspects of the SD system and its environment; and thereby underline the demand for SD systems capable of effectively sensing the shifting environmental characteristics and responding efficiently to them.

This first of these three chapters deals with the agents of the SD system who contribute to the SD results, and define the value of these results. Furthermore, this chapter shows how systems engineering can support the sensing and interpreting of these *stakeholders' voices* and translate them into engineering language. Customer centric systems engineering philosophies like *Lean,* which will be discussed in this chapter, propose to think about the SD as a *value-driven system* that attempts to maximize stakeholder value throughout its operation. Dynamic stakeholder value is what drives the transformations of an adaptive SD system towards maximal system success.

D.2. OBJECTIVES OF SYSTEM DEVELOPMENT

SD is a process of seeking an optimal solution to a complex problem called the engineering design and development of a new system. The precise characteristics of the design, *i.e.,* the main deliverable of the SD process, are unknown in the beginning of the project. As documented in [Baldwin & Clark 2000]:

> *At its core, an SD process is a search for something unknown. The ultimate form of the artifact is unspecified at the outset of the process. A search is mounted in hopes that a form (a particular set of parameters) may be discovered that jointly satisfies certain objectives (the system objectives) and takes account of certain constraints. The result of this search is a description of the thing to be made, including instructions about how to make it.*

In many cases, the customer only knows that she or he wants to have something smaller, better, faster, safer, more reliable, *etc.,* than the products representing the current state of the art, as soon, and as cheap as possible. In addition, these desires do not appear as products of a controlled process, they depend on the complexity (*i.e.,* structure and dynamics) of the marketplace. As proposed by Vollerthun in [2001, 2002], the complexity of the market depends on the characteristics and interactions of the five main elements that construct the *market system*: (1) market segmentation, (2) market size, (3) competitors, (4) price formation, and (5) customer behavior.

Besides the components and their interactions in the market system, the dynamics of the market have a main effect on the success of a product. *Market dynamics* describe how the characteristics of the market elements change over time because of the interactions among (1) the market elements, and (2) the market and its system environment. These system-internal and -external factors drive the overall value of the SD project and thus, the profitability of a product to be introduced in the market system.

In the market, customers purchase products, if these products satisfy their needs, *i.e.,* the products include functionalities that are of value for them. Hence, *value* means

> *a capability provided to a customer at the right time at an appropriate price, as defined in each case by the customer* [Womack & Jones 1996].

Customers are willing to pay more for products that they believe to be of more value, *i.e.,* the functions of the product fulfill their needs better. However, it is not always true that higher product performance or functionality means higher value, since the customers pay more only for higher *desired* product performance and functionality. Hence, the collection and analysis of customer needs is vital for the success of the project, because the fulfillment of the customer needs defines the value of a product and thus, the number of product pieces the company can sell.

Another factor that contributes to the value of a product in the market is the technological knowledge of the company and its environment. The available technologies define the capabilities of the SD system to fulfill the identified customer's needs in the market. While both technologies and market needs are dynamic, they can be represented as streams. An improved version of such a model by Allen [1997] depicting the SD process in its technology and market environment is shown in *Figure D.1*.

The original model includes only one arrow representing input information from the market and technology environment and one arrow for the output product that contributes to the changing technology and market needs. The reason is that in conventional SD organized according to the waterfall model, system objectives are defined early in the project to provide clear goals for the developers of the system. However, such an early definition and freeze of the system requirements constrain the design space at an early development stage, where the knowledge of the system and its environment is still rather low. As a result, important design options might be excluded due to inadequate information. In the dynamic and often unpredictable SD environment, the early elimination of design options reduces the probability of delivering superior products to the market that mean high value for the customers.

Another factor that affects SD is that its external environment (*i.e.,* the market and the available technologies) is *dynamic*. Thus, both the customer needs and available technologies might change in the course of a project as well. However, many engineers make the false assumption that Marketing provides mature and stable information on the actual and future customer's desires and that the availability, maturity, and suitability of technologies to fulfill the product requirements are discrete variables. This is not true indeed. Technologies and market characteristics are uncertain variables of the SD that require the continuous attention of the developers. Thus, due to the dynamics of both the market and technology characteristics, a permanent feedback relation is required between the SD system and its environment, which supports the sensing of changes and fosters adequate response from the SD. The dashed arrows in *Figure D.1* depict this *permanent relation of sense and response*, which lies in the heart of adaptive systems [Dove 2001].

The basic function of systems engineering is to build bridges between different SD

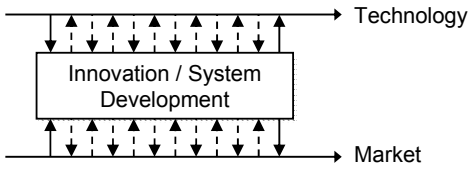


Figure D.1 A simple model of the innovation/SD process (modified from [Allen 1997])

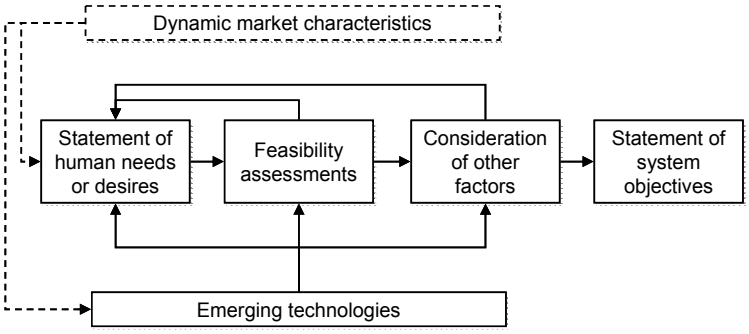


Figure D.2 Improved version of Hazelrigg's nonlinear model for determining system objectives

disciplines and departments of the SD enterprise fostering information exchange (*i.e.,* communication) between the various parties involved in SD project. Hence, a main task of systems engineering is to support SD by translating the customer needs into *system objectives* written in a language that is understandable for everyone in the SD organization. This language is part of every organizational culture and thus "spoken" by the members of the SD organization. As systems engineering is the treatment of engineering design as a decision-making process [Hazelrigg 1996], the definition of unambiguous and understandable system objectives facilitate the effective selection of design options by providing clear criteria for every decision during the complex process of the development of an engineering system.

The iterative process of determining system objectives is shown in *Figure D.2*. The original version of the model [Hazelrigg 1996] depicted with the solid lines in *Figure D.2* was defined for a *static* SD environment, where the system objectives are defined in the beginning of the project based on the actual customer needs and available technologies. To account for the dynamic SD environment, the original model was extended by an important external factor, the *dynamic market characteristics* (dashed lines in *Figure D.2*), which have a major impact on both the system objectives and the emerging technologies.

The system objectives in Hazelrigg's improved model depend on two main external and some further project-internal factors. During the definition of the system objectives, the feasibility of a system design is assessed that depends on (1) the characteristics of the market, (2) the available technologies, and (3) the capabilities of the SD enterprise (also including the suppliers) summarized as "other factors" in *Figure D.2*.

SD literature includes similar approaches for the definition of system objectives for Hazelrigg's model. Ulrich and Eppinger [2004] propose a *product planning process* for evaluating single SD projects and portfolios of projects based on their *mission statements* (a broader view of the system objectives including managerial aspects, too). In the model depicted in *Figure D.3*, internal and external factors that influence design characteristics are considered as *opportunities*. The identification of opportunities is part of the process of the customer's needs elicitation. Thus, design opportunities might come from different sources, *e.g.,* frustrations and complaints of customers with regard to current products, needs, and suggestions of lead users and other current customers, actual trends in lifestyles, demographics and technologies, competitive benchmarks, availability of emerging technologies from internal research. Such opportunities emerge continuously during the projects, thus the product planning process must be an ongoing

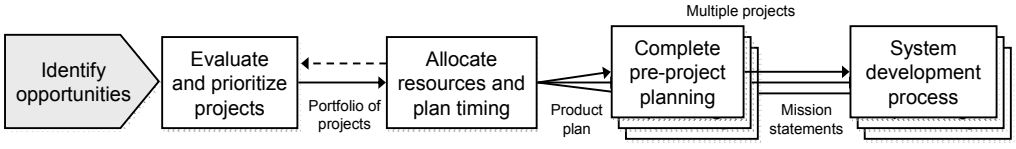


Figure D.3 The product planning process (modified from [Ulrich & Eppinger 2004])

effort to plan and update the SD project portfolio of the company.

The main difference in this approach compared with Hazelrigg's is that it also accounts for the analysis of multiple projects considered as future opportunities for the SD organization. That is, the product planning process raises the decision on the feasibility of a single SD project to the enterprise level to be able to consider organizational aspects such as competitive strategy, market segmentation, technical trajectories, product platform planning, new product opportunities, resource availability, staffing, project timing, *etc.* in the decision as well. Hence, the output of the decision process is not only the technical feasibility of a design idea, but the *overall value* of a project that is required to develop, manufacture, market, and sell a product with regard to the other elements of the SD system.

D.3. VALUE-DRIVEN SYSTEM DEVELOPMENT

Systems engineering provides a means to foster effective decision-making in the SD. The goal of these decisions is to steer the SD towards success. However, it is not always obvious, *who* are the persons and groups of persons, whose satisfaction is important for the SD, *what* satisfies these people, and *how* their highest satisfaction can be achieved. *Value engineering* supports decision-making by gathering important information on those people with a legitimate interest in the project outcome (*who*), and valuates their desires (*what*) to foster the selection of the optimal way to satisfy these people (*how*).

Decisions in the SD are made at different levels of the organization with the objective of maximizing project success. On the one hand, the success of the project is traditionally measured through *customer satisfaction*. That is, the more the developed system satisfies the customers, the more pieces they will buy of it. On the other hand, the suppliers of an SD organization also have an interest in the project success, because it might mean further contracts and thus increased profit for them. Furthermore, as the success of a project usually means rewards for the employees, the developers are directly interested in high quality project outcomes, too. However, the employees are also interested in good working conditions, high salaries, and realistic, reachable project goals. There are further parties involved in the SD, who all have expectations of the product and the development project as well. Together, these people are the *stakeholders* of the project, and systems engineering summarizes their expectations in documents like the stakeholders' needs, system objectives, system requirements and constraints, and system specifications. These documents involve information on the goals of the project in increasing detail using the organizational language spoken by the different groups of stakeholders involved in the development of a complex system. This information is a main source of decision-making in SD.

In order to support the effectiveness of the SD decisions, systems engineering derives evaluation criteria for every decision from the above-mentioned requirements documents. The notion of the definition of decision criteria independently from the varying complexity and outcomes of SD decisions is *to foster the precise definition of the stakeholders' expectations on each decision option in technical language*. This way, the characteristics of each decision option can be quantified using the same measurement system, *e.g.,* a set of hierarchic performance measures representing the key product requirements (*Figure D.4*). Then, a decision can be made based on the proportion of each option's individual performance versus the required technical performance stated in the evaluation criteria. That is, using the systems engineering decision procedure described in *Chapter B.3*, the decision makers evaluate the decision options based on the rate of the fulfillment of the product requirements.

The system of key parameters depicted in *Figure D.4* is an effective systems engineering tool to define quantitative decision criteria for each level of SD decisions. Measures on the higher

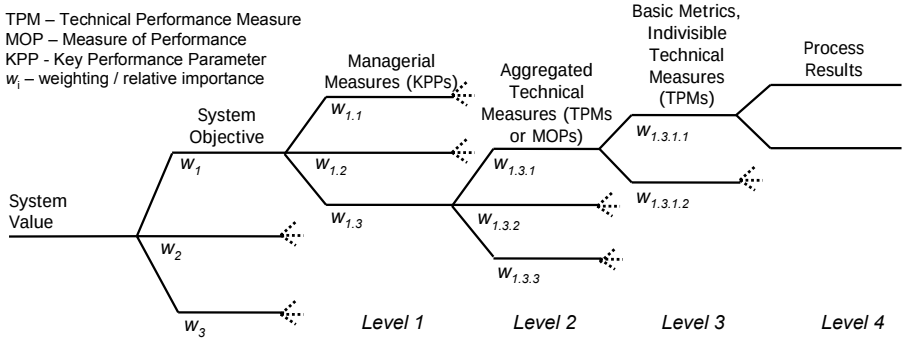


Figure D.4 System of key parameters as decision criteria

levels of the parameter tree are directly derived from the high-level system objectives. These measures represent the key performance and operational system characteristics, and are the most important project goals to fulfill. While these objectives cannot be directly used at lower level decisions, they are usually broken down into further system and subsystem level technical measures. Hence, using such a hierarchic "goal tree", it is possible to associate the low-level SD results with the highest level system objectives and make a decision that increases the overall system value.

However, many engineers stop at the technical level of the decision analysis process and try to use the mere technical information for the decisions. While decisions usually demand important tradeoffs between the achievements of various project goals, decisions are hard to make using only technical information. Thus, a common language is required, which can be used to translate different engineering measures into comparable units suitable for the purpose of the evaluation. A common language usually used for such translations is the *value* of each option expressed in monetary units. That is, after the engineer has determined the technical characteristics of each option, he or she has to determine the value of every option by answering the question: ***`how much profit will I make if I choose this option`*** [Hazelrigg 1998]? The option with the highest value and thus highest achievable profit is then selected for the project.

This question shows the fundamental objective of engineering decision-making, *i.e.,* to maximize the profit of the SD organization. Baldwin & Clark [2000] formulate the same principle in their engineering design axiom: ***`designers see and seek value in new designs`***. Thus, each decision in the SD process aims to increase value.

However, the value of a design, a system, or a whole project is usually difficult to quantify. Profitability is always the highest-level goal, but the aspects of profitability are complex, depending on many factors. Marketing research teaches to listen to the *voice of the customer* to be able to identify opportunities for products that are valuable for the different groups of customers. Marketing applies effective methods like *conjoint analysis*, *voice-of-the-customer analysis*, *perceptual mapping*, *intention scaling*, *portfolio optimization*, and *lifecycle forecasting* to determine actual and future customer needs that drive the value of a product in the market [Dahan & Hauser 2000]. Knowing the customer preferences and their meaning in technical language is a basic requirement for decision-making.

Value is also central in *Lean,* a systems engineering philosophy to handle complex SD projects [*e.g.,* Murman *et al.* 2002]. Lean, as a holistic philosophy, expands the scope of the determination of the system objectives to all the *stakeholders* of a system, *i.e.,* to all the people who are somehow affected by the development, manufacturing, operation, maintenance, and disposal of the system. Such stakeholders generally comprise the gamut of customer acquirers, end users, consumers, partners, supplier, unions, the corporation, the shareholders, and the society [Murman *et al.* 2002].

Hence, the overall success or value of the system is defined by the global satisfaction of the system stakeholders.

The stakeholder view of Lean includes an important system management aspect. It proposes that the success of a project is not a mere measure of the customer satisfaction and the profitability of the project this satisfaction brings, but that long-term system success depends on the satisfaction of everybody linked to the system. Thus, the objective of systems engineering management is to maintain a *long-term win-win situation* for each SD stakeholder and thus, system success by maximizing the overall lifecycle value of the delivered products.

Nevertheless, the value of a product is difficult to determine due to the diverse and sometimes even contradictory desires of the various stakeholders. Furthermore, value is not a static variable. It depends on the (structural and behavioral) complexity of the SD system and its environment. It changes during the SD lifecycle as the design itself, and its internal and external environment change. **Thus, the determination of the lifecycle value of a system must be an ongoing activity following the SD process and considering all aspects and subsystems of the dynamic ZOPH+T development system.**

The determination of the way the system lifecycle value changes during the period of the project or program is essential for the definition of realistic system objectives. Browning & Honour [2005] propose an iterative five-step approach to quantify the system lifecycle value during the whole lifecycle of a system: (1) identify the stakeholders, (2) identify the stakeholder preferences for key parameters, (3) anticipate and quantify the evolution of key parameters, (4) create a holistic measure of stakeholder value, and (5) measure stakeholder value over time: Lifecycle Value.

The key parameters (*i.e.,* the *managerial measures* in the "goal tree" in *Figure D.4*) that are the key drivers of the system lifecycle value for the stakeholders comprise the most important characteristics and requirements regarding the development and developed systems. The term *lifecycle value* implies that value is dynamic and valuation should not end at the end of the SD, but it has to track the shifting stakeholders' needs and discover new opportunities for existing and new products. Furthermore, the ultimate development goal should be to maximize the profitable length of the system lifecycle and thus, the lifecycle value of the systems. This can be achieved through designs and design platforms that account for changing stakeholder preferences and products that can be adapted to the changing environment.

The dynamic nature of system lifecycle value is an important driver of adaptive SD. That is, the goal of SD is not simply to deliver a system that satisfies the stakeholders at one point of

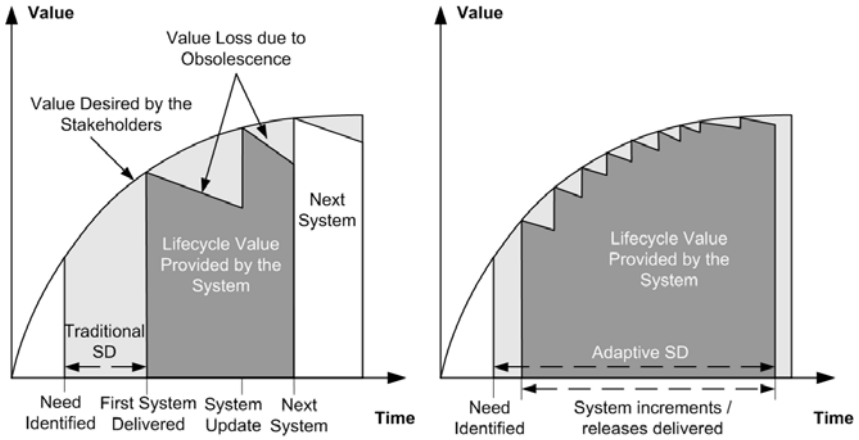


Figure D.5 Lifecycle value of traditional and incremental SD outputs (modified from [Browning & Honour 2005])

time in the future, but *to aim at a moving target and hit it many times* during the product lifecycle by releasing frequent system upgrades [Browning & Honour 2005]. This way, the lifecycle value of the system can be considerably increased.

Figure D.5 shows how system lifecycle value can be maximized in an adaptive SD system. Through the application of the models for capturing dynamic system objectives discussed in the last section, changes in the market needs and technology characteristics can be documented and the system objectives can be adjusted to set the scope of the required system adaptation. While in traditional SD few products incorporating a high rate of innovation are launched (*i.e.,* big-bang release), adaptive SD attempts to release many versions of the developed systems with evolving maturity and value. Thus, as *Figure D.5* depicts, adaptive SD projects can manage the dynamics of system lifecycle value, and develop and produce high value systems continuously.

The stakeholder centric philosophy of Lean is an important basis for adaptive and Agile SD. At each step in a Lean SD, the possible outcomes of the decisions are evaluated to quantify the effects on the overall system lifecycle value. This fosters the consideration of both positive and negative consequences of the SD steps and thus supports deliberate decision-making.

D.3.1. Lean System Value

The core principle of Lean is that each system element individually contributes to the overall system value and thus has to be evaluated separately (considering their effect on the emergent system characteristics, as well). That is, the elements in all five subsystems of the ZOPH+T development model can be evaluated, and their effects on the system value can be quantified. This way, requirements, design components, technology components, activities, and even persons that have a high contribution to the system value can be highlighted and those with low or no contribution can be removed from the system.

As the ZOPH+T subsystems are interrelated, a change in one subsystem causes changes in others, *e.g.,* if a goal or requirement is removed, the related function and physical component also has to be removed (*Figure D.6*). Furthermore, the technology planned to be used at the design of the certain component and the activity or sub-process implementing the design part might not be required any more either, which might then cause changes in the technology and process system, too. Hence, a Lean goal system contributes to the leanness of all other subsystems in ZOPH+T and vice versa.

Baldwin & Clark [2000] argue that there is a *fundamental isomorphism* between the structure of the product system and process system in the SD. NASA [1995] also states that the structure of

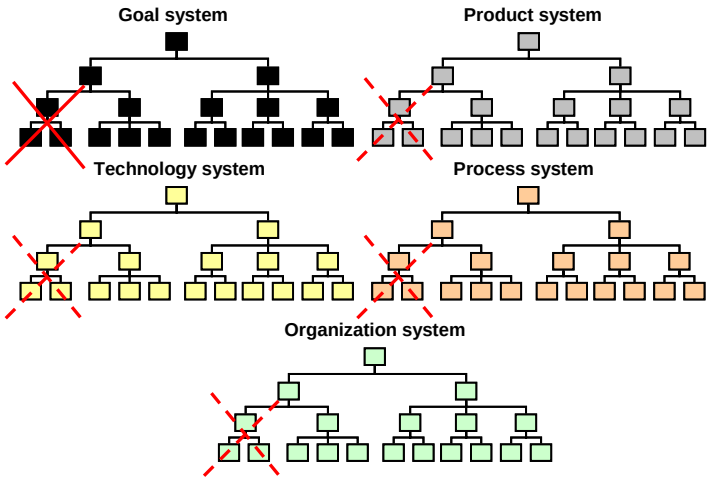


Figure D.6 Eliminating waste in an isomorphic ZOPH+T system

the product system resembles the structure of goal system (requirements). Furthermore, the organization of the project includes persons assigned to the activities in the SD following the process structure. Hence, if one block of elements is eliminated in one of the ZOPH+T subsystems, then all other subsystems have to be adjusted as well. *Figure D.6* shows this phenomenon, where the elimination of one chunk of goals affects the contributing elements in all other subsystems. This basic principle of systems engineering is often neglected while doing changes in certain ZOPH+T subsystems during an SD project causing chaos, rework, and costly correction actions in the project. Hence, the propagation of changes within and among the ZOPH+T subsystems is essential for efficient SD projects.

As proposed in Lean, there are three main classes of system elements based on their contribution to the overall system value [Womack & Jones 1996]. In the first class (*Type I*), there are system elements that *add value*. These have to be identified and categorized as core system components. Others in the second class (*Type II*) do not add value but are necessary to enable value production. These elements are called *necessary waste* in a system. System elements in the third class (*Type III*) do not contribute to the system success, *i.e.,* these are elements that do not add value and thus are unnecessary for the project. Elements in the third class are called *pure waste* and have to be eliminated from the system.

D.3.2. Lean System Development

Lean principles were first applied to production and business processes that are repetitive in nature. They yield high reductions for these processes in project cost and schedule if redundancies and unnecessary activities are eliminated, because they only cost time and money without increasing the success of the project.

However, SD has different characteristics. Iterations in the SD process (*i.e.,* the repetition of a set of activities to improve their results) that might only mean costly project changes, rework, schedule overruns, and thus process waste to the managers at the first glance, are the main places for innovation that drives system value in the SD. Thus, if Lean principles are strictly applied to the SD, it might lead to the elimination of valuable system elements. As offensive players in a football team are valuable because they score the goals, the defense and the goalkeeper have the same value for the team, since they are the players who prevent goals scored by the opponents. Thus, all players in each position are necessary to achieve system (*i.e.,* team) success.

Browning [2003] and Browning & Ramasesh [2005] propose to measure the value of SD activities based on their contribution to the *reduction of project risks, and discovering and seizing of design opportunities*. Both risk reduction and seizing opportunities increase the overall lifecycle value of the system. This view of activities makes it possible to determine the *real* lifecycle value of activities that would be waste according to the traditional Lean principles. This theory improves the traditional Lean philosophy by considering how process elements contribute to the maximization of stakeholder value in the actual and dynamic future SD environment. Thus, activity value is in this case not a mere individual, intrinsic attribute, but a parameter that is the function of the activity internal and external characteristics (*i.e.,* ZOPH+T system and its external environment).

The next part discusses the role of V&V, a typical "necessary waste" system element type according to traditional Lean with fundamental contribution to the final system value.

D.3.3. Validation and Verification – The "Vital Waste"

The determination of the lifecycle value of a system is a systems management task and thus it deals only with the desired and implemented most important high-level characteristics of the system and its main modules. At lower technical levels, systems engineering breaks down these

high-level project goals into detailed product requirements and specifications that together give the full desired performance and functionality of the system at every system level. V&V supports systems engineering by providing effective means for the evaluation of the actual against the desired system maturity and performance and thus the determination of the system value at the technical levels. As consistency is a key aspect of system decomposition (*i.e.,* the functionality and performance of the sub-systems together provide the full system functionality and performance after integration), a task of V&V is to ensure that the requirements at the lower technical levels are always consistent with the system-level goals.

V&V usually starts with *validation*, the process of evaluating the system objectives based on their relative values for the overall system at all managerial and technical system levels. Validation activities are coupled with design activities to analyze their deliverables (*e.g.,* prototypes of the design representing certain design aspects) and determine if *the right system is being built, i.e.,* if the requirements (derived from the system objectives) implemented in the design mean value for the customers and other stakeholders. Validation is effective if various stakeholders (*e.g.,* the customer, supplier, manufacturing staff, marketing staff, *etc.*) join the developers during validation to articulate their opinion about the actual design and so, contribute to the continuous improvement of it.

Validation deals with the determination of the *value* of the developed system. During validation, the requirements and the system specifications derived from the stakeholders' needs are evaluated to determine if the developers' view of the design matches the stakeholders' view of the same design. To avoid ambiguities, misinterpretations and bias between developers and stakeholders, validation must start early and done continuously in the SD.

Another form of system evaluation is *verification*. During verification, the design is assessed to determine the quality of the system, *i.e.,* how the design satisfies the stated system objectives and stakeholder requirements. Verification methods comprise *analysis*, *inspection and demonstration,* and *test* [DoD 2001a] with the common goal to determine the *quality* of the system, *i.e.,* if the design complies with the defined requirements assuming that the requirements are valid. Verification goes hand in hand with validation in the SD process, since these two systems engineering tasks ensure that the *right* product is built *right*, *i.e.,* valid requirements are implemented in the design in an appropriate way.

Both validation and verification reduce uncertainty and thus risk in the project by evaluating SD deliverables and comparing them to the stakeholders' needs and project plans. Furthermore, V&V is a fundamental part of experimentation cycles, where creative ideas and design options are implemented and tested to resolve design problems and reduce the uncertainty in the characteristics of the final SD outcomes. Later parts of the thesis deal with the role of V&V in adaptive SD and show how design and V&V activities can be effectively planned to support value creation in the project. V&V is the main source of information for the adaptation of the SD process to the changing environment and thus a key element of effective SD processes.

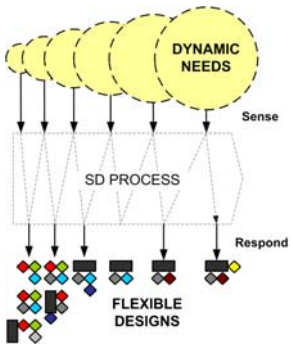
D.4. CHAPTER SUMMARY

This chapter found that the fundamental goal of SD enterprise systems is the *maximization of stakeholder value* throughout the whole system lifecycle. Each element comprising the isomorphic subsystems of the ZOPH+T SD enterprise system model has a contribution to the overall system lifecycle value. Thus, the main objective of adaptive systems engineering is the design and adaptation of the SD system to maximize value during its lifetime. Lean and Agile are systems engineering philosophies that accentuate the role of early and frequent prototyping to ensure that the *right system is built right*, *i.e.,* the system objectives represent the highest stakeholder value, and the SD process delivers a system according to these objectives.

E. SYSTEM COMPLEXITY – THE ROLE OF THE ARCHITECTURE IN THE SYSTEM DEVELOPMENT

E.1. CHAPTER ABSTRACT

The previous chapter reviewed the characteristics of the SD system environment and proposed theories and methods to assure high stakeholder value of the developed products. While the preceding chapter dealt with the inputs of the SD system, this chapter discusses the SD outputs. That is, following the systems engineering way of problem solving, after having discussed the inputs of the SD system in the last chapter, now the characteristics of feasible outputs are defined.



Systems in highly dynamic environments have to be capable of sensing the changes and responding to them by *restructuring* and thus increasing the overall system value. A main constraint to quick and cheap changes in a system is high *system complexity,* because highly complex systems cannot be changed easily. Hence, this chapter proposes methods that increase the flexibility and adaptability of a system by reducing its complexity. Further, the chapter finds that *structural modularity* is a main driver of system flexibility, because it allows the definition of independent subsystems that can be modified without considerable effects on the emerging system properties.

E.2. DEFINITION OF COMPLEXITY

A fundamental characteristic of SD is that the developers have to deal with the *complexity* of the system. It is important to discuss the problem of complexity, because as Rechtin & Maier indicate [1997]:

> *It is generally agreed that increasing complexity is at the heart of the most difficult problems facing today's systems of architecting and engineering. Systems are simply growing in complexity – the biggest cause of cost overruns.*

The general definition for the word "complex" is related to the complicatedness of something, *i.e.,* complex is *a whole made up of complicated or interrelated parts* [Merriam-Webster Online Website]. However, this definition is too general for the application in (systems) engineering. Fortunately, many researchers in different disciplines dealing with the description and analysis of systems documented a high number of definitions for this term. Naturally, these definitions are always related to the specific context and discipline of the researcher or author (for a large set of definitions of complexity from different disciplines see [*e.g.,* Negele 1998, Sussman 2003]).

In systems engineering and systems theory, definitions classify complexity into two main categories: *structural complexity* concerning the complicatedness of the order of the elements in the system and *behavioral complexity* regarding the behavior of the system over time [Sussman 2003]. Senge [1990] also defines two types of complexity in: *detail complexity* referring to the characteristics of the system elements and the interrelations between them (order or structure) and *dynamic complexity* caused by the processes of system change (behavior). German-speaking systems theory also provides definitions for both kinds of complexity [*e.g.,* Patzak 1982, Negele 1998, Igenbergs 2000, Wenzel 2003]:

Complexity is a characteristic describing the structure of the system that depends on the variety (diversity of elements) and the connectivity (diversity of relations) of the system. Furthermore, variability (changeability) has a main effect on system complexity.

Hence, the *structural or detail complexity* of a system is high if the variety of the system elements is high (*i.e.,* a high number of diverse elements compose the system) and the number and type of relations between system elements are high. For the latter aspect of structural complexity, sometimes the measure of relative connectivity or *connectivity density* is used meaning the actual number of relations in the system relative to the number of possible relations in the system. Connectivity density is however a rather theoretic measure that assumes that the maximum number of relations between two elements of the system is two, one in each direction between the elements. Thus, connectivity density is difficult to apply to real systems where usually numerous relations between system elements exist. According to Senge [1990],

behavioral or dynamic complexity exists if, (1) an action has different consequences on the system in the short and the long term, (2) an action has one set of consequences locally and a different set of consequences in another part of the system, or (3) obvious interventions produce non-obvious consequences in the system.

Dynamic complexity can arise in simple systems with low structural complexity if the system behavior ruled by the interactions among system elements are: (1) dynamic, (2) tightly coupled, (3) governed by feedback, (4) open, (5) nonlinear, (6) history-dependent, (7) self-organizing, (8) adaptive, (9) counterintuitive, (10) policy resistant, (11) characterized by tradeoffs, and (12) interdisciplinary, [Negele 1998, Sterman 2000].

A main driver of dynamic complexity in SD systems is long time delays between taking a decision and its effects on the system [Sterman 2000]. For example, long time delays reduce the effect of learning in the design iteration loops in the SD process. As learning is the main driver of innovation, long time delays contribute to lower effectiveness and efficiency in the SD that diminishes the value of the delivered product. This characteristic of dynamic complexity is a main aspect of structural process optimization [*e.g.,* Steward 1981a, 1981b, Browning 2001], experimentation planning [Thomke 2003], and V&V planning [Lévárdy *et al.* 2004a, Lévárdy & Browning 2005]

Hence, as Senge suggests, the real leverage in most management situations lies in understanding dynamic, not detail complexity of the SD project. However, most systems analysis tools focus on detail complexity and try to fight "complexity with complexity". That is, the more complex the system gets, the more complex the analysis methods will be as well. Nevertheless, as Senge concludes, this is the *antithesis of real systems thinking*. Thus, the clue to deal with increasing system complexity is not to try to improve existing analysis and design methods and tools by including plenty of new features in them, but to realize the opportunities of new theories and research directions that decrease complexity more effectively than before.

This thesis provides a novel way to consider development processes and provides an effective method to handle both detail and dynamic complexities of the SD system.

E.3. TECHNIQUES TO DEAL WITH SYSTEM COMPLEXITY

E.3.1. Modeling – Understanding System Complexity

As most definitions propose, complexity is not an absolute, intrinsic characteristic of the system, but it is an aggregated measure of different parameters that are drivers of system complexity in different disciplines [Schulz 2003]. Thus, the best way to measure system

complexity is to build a *model* of the system that accounts for the major characteristics that describe complexity, analyze the model, and seek a result that unambiguously shows the complexity of the system regarding the definition.

Modeling and simulation are effective tools to determine both structural and behavioral complexity of a system. The determination of complexity requires a three-step process [Kreichgauer 1995]:

1. *Define the measure(s) that quantifies system complexity.* The measure(s) must unambiguously reflect the modeler's understanding of complexity and provide a clear answer to the modeling question: "what is the complexity of the system?"

2. *Define a representation (i.e., model) of the system that is appropriate for the measurement of complexity.* That is, the model and the complexity measure(s) have to use the same modeling language. Otherwise, the model cannot provide a clear, unbiased answer to the modeler's question. *The complexity of the model, as a representation of the system then equals the complexity of the system regarding the defined measures*, *i.e.,* the modeling question.

3. *Determine the complexity of the model and thus, the system using the modeling components.* The result of modeling is as exact and valid as the model and its modeling language. Thus, the selection of an *appropriate modeling language* for both the measures and the model, and the capability of the modeling method to model the system have major effects on the quality, validity, and usability of the results.

Once a model of the system is defined, the degree of system complexity can be determined and models of different systems defined in the same modeling language can be compared [Negele 1998]. Furthermore, analysis on the models can be conducted to determine how the complexity of the actual model can be reduced in order to foster the work of the system developers.

A model is an abstraction of the reality [*e.g.,* Igenbergs 2000], *i.e.,* it accounts only for a certain aspect, a limited scope of the modeled system. Thus, a model is a representation of a system *with a definite purpose*. This purpose generates requirements on the modeling process, *e.g.,* modeling objective, required outcome, evaluation criteria, modeling method and procedure, required fidelity, economic constraints of the modeling process, analysis methods, *etc.* Consequently, the purpose of modeling has a main effect on the kinds and quality of the results obtained during modeling and simulation.

Another important aspect of modeling is the placement of the system boundaries in the model. The system boundaries limit the working range of the model and thus the scope of the analysis of the modeler. Where the boundaries of the model have to be drawn is defined by the framing or articulation of the problem that is to be solved with the model. The boundaries describe where the system operates, which variables are required in the model to provide an adequate solution to the modeling problem, and the relevant time horizon for the modeling [Sterman 2000]. Hence, the *effectiveness* of the modeling process demands that all the *relevant* system elements, variables, and cause and effect relations are included in the model, *but not more than that*, because otherwise the *efficiency* of modeling suffers. The right balance between the required modeling information and the fidelity of modeling outcomes drives the overall value of the modeling exercise.

If a system model is defined according to the four systems engineering modeling rules of Igenbergs, first, the system model will represent the *order* of elements and relations in the modeled system. Schulz [2003] defines order as the state of a system that characterizes the

system through a high number of varying system components connected to each other. The order of the system emerges naturally.

The order of the elements and relations in a system has a major effect on system complexity, because the order contributes to both the structural and behavioral characteristics. Thus, system complexity can be reduced by manipulating the order of the system elements [Haberfellner *et al.* 2002]. If the order of the system elements is modified for a certain purpose, the resulting new order will not be the natural order any more. This new artificial order is then called the system *structure* [Schulz 2003].

Manipulation of the order of the elements in the system model can be used to define different *levels of abstraction* that "hide" the complexity of the whole and allow the developers to focus on the details that are important and relevant to their SD tasks. As the development of complex systems requires the cooperation of hundreds or thousands of experts, abstraction helps define work scopes and clear objectives for these individuals and teams by reducing the problems to be solved to a manageable size. Dividing a larger task into smaller problems is one of the most fundamental problem solving paradigms, however dividing a large problem effectively requires either a great deal of insight or an insightful model [Eppinger 1991].

Two kinds of abstraction are described in the next sections that help structure the related system elements into groups and thus support the reduction of complexity during system architecting and modeling. *Hierarchy* in a system relates to the vertical relationship among system elements that is the result of system decomposition, while *modularity* refers to the architecture type where closely related elements of a system are grouped in clusters or chunks based on their horizontal interactions.

E.3.2. System Decomposition and Hierarchy

Alexander [1964] proposes that in architecting, the overall designs can be improved if they are made up of subsystems that can be adjusted relatively independently. Furthermore, he argues that the complexity of modern systems can be better handled if they consist of fairly independent subsystems. This way, the scope of problem solving can be reduced and its effectiveness increased.

Hence, it makes sense to decompose a system into vertical hierarchy levels if its complexity is too high and thus no longer manageable for the designers. Bahill defines hierarchy as `an ordered network of concepts or objects in which some are subordinate to others` [INCOSE 1998a]. Thus, hierarchy is a structural characteristic of a system representing an order of the system elements in which components are ranked into levels of subordination regarding their relations, attributes, and functions. System hierarchy orders the elements of a system into classes, where each class represents a level of abstraction of the system.

During SD, a large problem (*i.e.,* the system to be designed and developed) is *decomposed* into smaller, related problems (*i.e.,* subsystems of the system) that can be studied and solved (*i.e.,* designed) easier than the large problem as a whole. After the small problems are solved and the subsystem designs are finished they are *integrated* to form a complete solution for the system design.

Through decomposition, a complex system like an aircraft can be broken down into main subsystems in the first hierarchy level (*e.g.,* propulsion, controls, structure, support systems, *etc.*) and then the subsystems can be further decomposed into components in the second hierarchy level (*e.g.,* the propulsion consists of main components like the compressor, combustion area, turbine, *etc.*). The hierarchic decomposition of the system can be continued until the lowest level contains only atomic elements, that cannot be decomposed any further.

Figure E.1 shows two systems, one *(A)* without and another one *(B)* with hierarchy. The two systems in *Figure E.1* are depicted as tree diagrams. As the tree diagrams show, the two systems are different in both architecture and structure. The architecture of a system `is the arrangement of functional elements into physical chunks that become the building blocks for a product or family of products` [Ulrich & Eppinger 2004]. Hence, the architecture of a system shows how the system is built up by interacting subsystems and components whose individual functions and behaviors yield the performance of the original complex system [Yassine & Braha 2003]. Thus, the architecture of system *A* and *B* in *Figure E.1* is different, because their subsystems and the relations between the subsystems are different.

Furthermore, the two systems in *Figure E.1* are different in structure, as well. *The structure of a system is an abstraction of the architecture.* It shows which interactions or interdependencies exist between its subsystems without accounting for the functions or attributes that describe the individual characteristics of the subsystems. That is, in case the structure of the system is investigated, the properties and functions of the system elements are ignored (*i.e.,* they are considered as black boxes), only the underlying logic of the system represented by the relations among system elements is studied.

The hierarchy of a system is one view of the system structure. However, as *Figure E.1* shows, the system hierarchy, usually displayed as a tree diagram, only depicts how elements on one level depend on the elements on the adjacent levels, not the relations between the elements on one level. This is a natural effect of top-down system design, where the system is gradually decomposed and detailed until each aspect of system design is adequately defined.

During system decomposition, the goal is to understand the design problem and define an optimal system hierarchy, *i.e.,* the required levels of abstraction for the optimal solution of the problem. In case of a simple design with low complexity, the definition of the system at too many system levels creates unnecessary working effort, as the number of system elements and interactions to be specified grow immensely at every level (see *Figure E.1*). On the contrary, a design with high complexity requires numerous abstraction levels to uncover every important design aspect and define the characteristics of interaction between the components even in the lowest level.

Hierarchical diagrams are often applied in systems engineering to demonstrate hierarchic relations among elements in every subsystem of the ZOPH+T model. Due to the fundamental isomorphism among the structures of the ZOPH+T SD subsystems [Baldwin & Clark 2000,

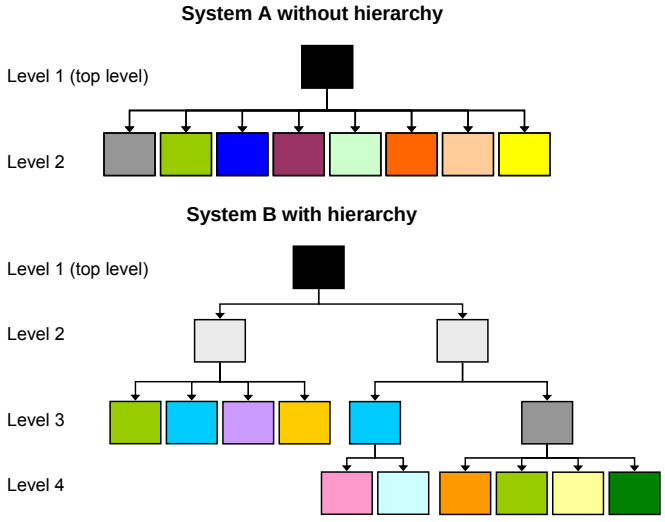


Figure E.1 Two systems with different hierarchies

NASA 1995], the hierarchic structure of each ZOPH+T subsystem follows the same pattern, that of the goal system. This is one of the basic purposes of system requirements: to represent a hierarchical description of the customer's desired product system as understood by the SD team. As these requirements are allocated, they become inexorably linked to the system architecture and *Product Breakdown Structure* (PBS), which consists of the hierarchy of the system, segments, elements, subsystems, *etc.* [NASA 1995].

The *Work Breakdown Structure* (WBS) is also a tree-like structure that contains the pieces of work necessary to complete the project. Each task in the WBS should be traceable to one or more of the system requirements, and contribute to the development of one or more of the product items in the PBS. Furthermore, each project should have an *Organizational Breakdown Structure* (OBS), which assigns personnel to perform the tasks, and a *Cost Breakdown Structure* (CBS), which depicts the cost of the development of the product items in the PBS including personnel, material, equipment, operating *etc.* cost.

Breakdown structures are important input sources of project scheduling, as they contain the core information about the project including the main goals and deliverables, major development products, every important step of the development, the persons that do the work, and the estimated costs for each main stage of the development.

It is important to discover and understand the fundamental similarity among the breakdown structures representing pieces or the entirety of the five ZOPH+T subsystems. As shown in a previous chapter the structures of the subsystems and thus the breakdown structures in the project are isomorphic. This characteristic is a main leverage in defining system hierarchy and modularization. However, many engineers and project managers still do not see the isomorphism among the main subsystems that limits the opportunities of optimizing the ZOPH+T SD system structure.

However, the application of hierarchical diagrams for SD also has some drawbacks. In case of high complexity, the decomposition of the system into multiple levels of hierarchy is a challenging task, since all requirements have to be accounted for at each level; the design has to be kept consistent between the levels; and interaction among the various branches of hierarchy have to be uncovered and understood. System decomposition into a clear hierarchical structure usually works best if the product can be broken into modules that are relatively independent. Such products or systems are called *modular* [Crawley *et al.* 2004]. These modules of the product can be then designed and developed concurrently, which significantly increases the efficiency of SD. However, if the there is a high interaction among the branches of the hierarchy, *i.e.,* the design is *integral*, it needs to be designed at the highest level of the hierarchy, or their design requires a lot of coordination of the modules at lower levels [Crawley *et al.* 2004]. The next part of the thesis discusses hiding of information and modularity in the system architecture.

E.3.3. Modularity – Hiding Information in the System

During system decomposition, the elements of the system design are refined at each hierarchy level, and the relations between system elements are determined to facilitate the later integration of the elements. Systems engineering and design literature suggest four main types of relations or interactions between system elements [*e.g.,* Pimmler & Eppinger 1994, Negele 1998]:

- *Spatial*: associations of physical space and alignment
- *Energy:* needs for energy transfer/exchange between two elements
- *Information:* needs for data or signal exchange between two elements

- *Material:* needs for material exchange between two elements

Pimmler & Eppinger [1994] further classify interactions into a five-point scale based on their relative importance for the functionality of the elements. The five categories of interactions or interdependencies between system elements are: (1) required, (2) desired, (3) indifferent, (4) undesired, and (5) detrimental.

Sharman & Yassine [2003] propose a modified version of the classification of interactions by Pimmler and Eppinger: a four-point scale based on the strengths of the interactions in a physical system. This four-point scale offers a general measure for the importance of interactions by defining the strength of a relation as the function of the required types of interactions between two system elements (*i.e.,* spatial, energy, information, and material). Thus, the strength of an interaction is

- *High,* if it includes the significant flow of three or more of the interaction types
- *Medium*, if it includes two of the above
- *Low*, if it includes one of the above
- *Zero*, if there is no significant relationship between two elements

The type and strength of the interactions between system elements are key inputs for the system analysis activity that usually follows decomposition in the system design process. During system analysis, the main tasks are to understand and document the interactions between the elements (*i.e.,* their integration) and analyze potential reintegration of the elements via *clustering* or integration analysis [Pimmler & Eppinger 1994, Browning 2001].

Integration analysis or clustering is a method to manipulate the structure of the system to recognize functionally related elements that are highly dependent on each other. These functionally related elements of a physical system are called *clusters* or *chunks* in literature [*e.g.,* Steward 1981a, Pimmler & Eppinger 1994, Browning 2001]. *The foremost objective of clustering is to maximize interactions between elements within clusters while minimizing interactions between clusters* [Rechtin 1991, Baldwin & Clark 2000]. Further goals of clustering can be to minimize the size of the clusters (*i.e.,* the number of elements involved in a cluster) or minimize the number of clusters [Browning 2001].

Pahl & Beitz [1999] directly link the definition of modules (*i.e.,* clusters) to functionality (*i.e.,* basic, auxiliary, special, and adaptive). Such a cluster is the *physical realization of a function* [Sharman & Yassine 2003]. The goal of clustering is slightly different in this case compared with the goal above, where clusters are defined based on the dependency of system elements on each other. Hence, the two ways of clustering can lead to alternative system structures that can then evaluated according to the complexity or other appropriate measures concerning the goal of the analysis. Finally, the feasible modular architecture can be selected for the product.

The result of clustering based on the interactions of the system elements is depicted in the second block diagram representing the architecture of system "*B*" in *Figure E.2*. During clustering, the design information (*e.g.,* elements of the design or design parameters representing the characteristics of the system and its modules) is partitioned into two categories: *visible information* and *hidden information* [Baldwin & Clark 2000]. That is, part of the interactions among system elements are outside the modules on the system level, and another part is limited to the module-level, inside the modules. If the rate of hidden information is high (*i.e.,* the interactions among system elements are concentrated on the module-level), then the design consists of a set

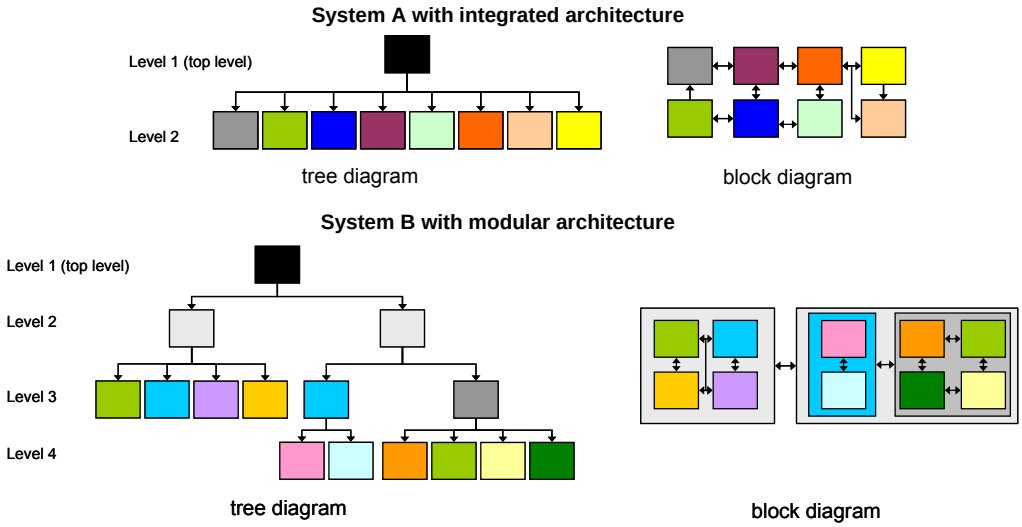


Figure E.2 Two systems with different architectures

of highly independent modules that interact with each other via interfaces. Hence, the above-defined goal of clustering can be interpreted as *maximizing hidden information in the design and thus the independence of the design modules*.

Visible information consists of interactions that serve to decouple hidden information from visible information. These major interactions between modules, called *interfaces*, are main building blocks of engineering systems. Interfaces are *the functional and physical characteristics required to exist at a common boundary or connection between persons, or between systems, or between persons and systems* [INCOSE 1998a]. Well-defined, robust interfaces are vital for modular systems, thus the interface specifications are among the most important system-level *design rules* for complex systems.

Hiding information is a type of abstraction, a technique for managing system complexity [Baldwin & Clark 2000]. Hidden information is not visible for other parts of the system, which reduces the connectivity and thus the structural complexity of the system. Furthermore, as the behavior of a system arises from its structure, information hiding and thus modularity reduces dynamic complexity as well. That is, the behavior of the hidden elements in the de-coupled modules has significantly lower effects on the other parts of the system than in a highly integrated system. This has a positive effect on the behavioral complexity of the system.

Level one and two of the tree chart depicting the hierarchy of system "*B*" in *Figure E.2* represent the visible information in this model. System "*B*" consists of two main modules or subsystems on "*level 2*" that are connected via an interface. This interface between the main modules of the system is the only visible interaction in the system. As "*level 3*" of the tree diagram shows, the first and second main modules are further broken down into four and two components respectively. These six components have only module-level interactions, *i.e.,* they are hidden in their modules. Thus, the *visibility* of the components in the first and second modules for the other parts of the system is limited.

To determine visibility for any element in a system, one needs to ask, *"What other elements would need to be redesigned if this element changed?"* [Sharman & Yassine 2003]. Therefore, if any of the system elements in system "*B*" on level three is changed, it has only consequences for the elements in the same module, but not for the elements in the other module. However, if a change occurs on level one or two, it has consequences for every element in the system. The same is true for the dynamic behavior of the hidden elements.

E.3.3.1. Benefits of Modularity

The previous section showed that product architectures with chunks or modules, which (1) implement one or a few functional elements in their entirety, and (2) comprise two or more components that strongly interact with each other within the cluster, and have few, well-defined relations to elements in other clusters are called *modular architectures*.

Modularization leads to major *design*, *manufacturing,* and *business benefits* for the development enterprise [Whitney 2003]. Furthermore, Baldwin & Clark [2000] suggest a fourth aspect, the *benefit of modularity in use*. The topic of this section is how modularization supports these four aspects of SD.

Modularity has numerous positive effects on the *design* of a system: (1) it reduces the scope of design changes due to the low visibility; (2) it increases product variety without adding extra complexity to the system, (3) it supports component standardization, which reduces manufacturing cost; and (4) it supports SD management by assigning concrete design tasks to relatively small design teams working on the detail design of chunks or modules [Ulrich & Eppinger 2004].

Baldwin & Clark [2000] argue that modularization has three main effects on the design and development of a product: (1) it makes complexity manageable by providing the developers with limited working scopes and clear, traceable and verifiable objectives; (2) it enables parallel work on the independent modules without ongoing coordination; and (3) it accommodates future uncertainty by enabling frequent module-level changes and improvements without affecting the system-level functionality of the product.

The first effect of modularity proposed by Baldwin & Clark concerns information hiding and visibility and was discussed in the previous section. The second aspect is a direct result of low visibility in the SD process. Development activities can be performed in parallel if there is no information dependency between them, *i.e.,* the deliverables of one activity are not required to complete the deliverables of the other ones. Activity dependency is particularly high if the system architecture is integrated, and thus each SD step in one piece of the system requires the consideration of the achievements of the development work in other parts of the system.

On the contrary, modular product designs enable independent development work on the main modules of the design and thus concurrency in the development process. Even though communication is still vital for efficient work, it can be done in a controlled way, since the few clearly specified interfaces between the system modules that define the most important aspects of communication are easier to handle than in a highly interrelated system. However, a basic rule for modularization in the SD is that the system-level design rules including the main system requirements, functions, modules, and interfaces cannot be changed once they have been specified. Without strict design rules, modularization can lead to chaos in the SD process.

The third benefit brought by modularization of the design according to Baldwin & Clark [2000] is that it allows "hidden work" on the modules and thus a later freeze of the module-level specifications. Thus, modularization fosters module-level experimentation and innovation that opens a myriad of new opportunities for SD. In case the system-level design rules are not changed (*i.e.,* the desired system functionality and the interfaces between the modules remain the same), the developers can experiment freely with various module design options by changing parts of the modules or implementing new technologies. This kind of *positive uncertainty* in the SD means that the module design options can be kept open much longer than with traditional integrated designs enabling later commitments towards the customer at higher design maturity and thus higher design knowledge. In addition, enhanced module-level experimentation and

testing foster early failure discovery and results in reduced system-level verification and qualification and as an effect, significant reductions in undesired late rework.

Ulrich [1995] discusses how *manufacturing* profits from modularization and finds that design modularity contributes to design flexibility that enables the flexible manufacturing of a high variety of products economically. Improved flexibility in manufacturing processes reduces the cost of changes between the manufacturing of different designs and thus fosters the implementation of *mass customization* in production. Hence, modularity is an important aspect of Design for Manufacturing (DFM) strategies [Ulrich & Eppinger 2004].

The *business* benefits of modularity include reduced costs through the reuse of standard components and a better definition of the opportunities for market dominance through interface capture [Moore 1999]. Additionally, modularity enables increased rates of technological or social innovation [Baldwin & Clark 2000] and increases product value through the ability to tailor product architecture and thus product characteristics to customer needs [Whitney 2003]; Finally, modularity enables better outsourcing, permitting companies to share risk or gain access to knowledge and capabilities not available in-house [Fine & Whitney 1999].

A long-term effect of modularization is that whole industries will operate as modular clusters. As the complexity of the overall system achieves such a high level that the SD is no longer manageable, systems can be broken down into individual modules with lower complexity that are developed independently. This way, system development can be transformed into the development of a system of systems, where a modular enterprise cooperated to achieve a common goal.

Baldwin & Clark [2000] show how the computer industry has grown to be a system of specialized clusters that deliver together the product: personal computer. The key driver of the emergence of industry clusters is the modular product architecture that enables parallel research, development, and production of independent product components with standard interfaces.

Modular products have advantages in their *use* as well. Modular products can be bought piecewise and assembled at home making it possible to define individual products from a set of standard components. Further benefits concern increased maintainability due to standard, plug-and-play modules, product adaptability to changing conditions, and increased lifecycle value through easy product upgradeability.

However, there are some disadvantages of modularity, negative aspects that can easily diminish the described benefits if not correctly understood and handled effectively. The next section describes these possible disadvantages.

E.3.3.2. Disadvantages of Modularity

Smith & Reinertsen [1998] argue that the cost of modularity might be increased project cost due to the higher required planning effort and reduced product performance. That is, the benefits of modularity defined above require a greater system-level working effort and detailed specification of the system-level design. So, both the system-level design parameters and functions have to be defined precisely. Additionally, Smith & Reinertsen [1998] claim that design flexibility, a usual benefit of modularity often comes at the expense of design performance, *i.e.,* the modular design has a lower performance than an integrated one with similar functionality [see also Höltta & De Weck 2005]. Nonetheless, this tradeoff is worth it if performance can be increased through frequent product upgrades enabled by the flexible architecture.

As modular designs accommodate uncertainty [Baldwin & Clark 2000], changes often occur during the design and development of modular products. Based on the predictability of the

future changes in the requirements and/or the module designs, the design team has to decide which parts of the system have to be robust and flexible. That is, to allow variability and changeability of the designs in the long term, parts of the design have to be built to resist changes (*i.e.,* robust parts) and others to enable them (*i.e.,* flexible parts).

Robust modules and interfaces are defined to be resistant to changes during SD. Robustness is an effective strategy in case slight, foreseeable changes in the stakeholders' needs are predicted. Flexibility in the design is deliberate on the other hand, if the requirements are imprecise or even unknown and thus difficult to predict in the long term. In this case, modules have to be designed to be flexible, *i.e.,* for easy changeability.

However, this demands a thorough understanding of the design and the planned development work. That is, the design team has to be aware of the main technical, technology and market risks that might threaten the stability and validity of the system objectives and thus the system-level requirements and design parameters. Therefore, the design team must account for possible changes in the development environment that could affect the system-level design and determine the required robustness and flexibility of the system architecture to withstand and accommodate changes.

Yet, robustness and flexibility are expensive. They are expensive due to the increased required planning effort and the high cost of robust and flexible interfaces and modules that have to be developed and implemented in the design. If there is no standardized solution for such interfaces or modules at a certain product, it might be risky to depend on immature components. Thus, a main task of research & technology development is to develop long-term solutions for modular product architectures for a product family or platform.

A benefit of modularity is improved module-level experimentation and thus, increased innovation. However, the strict system-level design rules might restrict technical opportunities that might be hindered by the design limits of the modules. With modular designs, where the rather high cost of changing the basic architecture exceeds the benefit achievable through the changes, the opportunities for fundamental innovations are limited.

Another drawback of modularity is that not all systems can be decomposed optimally into modules and interfaces [Whitney 2003] and if modularity is sub-optimal, the independent development work suffers from it [Sharman & Yassine 2003]. That is, modularity in theory is different from modularity in reality. Thus, modularization requires high design and development expertise and experience; otherwise, it can lead to high losses in the project.

Even if the modularity of the design is optimal, projects developing modular products require attention from project managers and systems engineers, because the developers have to learn how to deal with the increased independence and flexibility. Firstly, such projects must include an ongoing assessment of the *compatibility* of module-level designs to the system-level specifications. Furthermore, the *organization for communication* is vital due to the increased concurrency in the development work. And last but not least, the developers have to discover how the increased product and process *flexibility can be utilized effectively* and how to bring their ideas in the development work without being afraid of failing.

E.3.3.3. Methods to Create Modular System Architectures

The sections before described the advantages and disadvantages of modularity in system architectures. In this section, methods will be discussed that foster modularization. Then, one method, the *design structure method* (DSM) will be described in detail, which will be applied later in the thesis for different purposes.

There are various methods in systems engineering and product design literature that support the identification of modules in the system architecture. Holmqvist & Persson [2003] identify six basic methods in their study that are applicable for product modularization. Each of the six methods is structured around three main modularization steps: *decomposition*, *integration*, and *evaluation*. However, each method applies different techniques for modularization. Furthermore, Sharman & Yassine [2003] collect and compare similar methods for identifying modules in a design. The two studies identified the following methods for designing modular products:

1. Function Diagrams/Structures [Otto & Wood 2001]
2. Fractal Product Design [Kahmeyer *et al.* 1994]
3. Modular Product Development [Pahl & Beitz 1999]
4. Modular Function Deployment [Erixon 1998]
5. Axiomatic Design [Suh 1990]
6. Graph Grammars [Siddique & Rosen 1999]
7. Hatley/Pirbhai Method [Zakarian & Rushton 2001],
8. Modeling the Product Modularity with Interaction Graphs [Kusiak & Huang 1996]
9. Design Structure Matrix (DSM) [*e.g.,* Steward 1981a, Pimmler & Eppinger 1994]

The first seven methods are traditional design methods that provide modularized products as output. All seven methods describe a design procedure for the three steps: decomposition, integration, and evaluation that deliver the desired architectures. At each method, the *function structure* of the product is the basis for identifying the design modules, and the goal of modularization is to map one or as few functions as possible to physical product components in the physical architecture. However, as Holmqvist & Persson [2003] argue, *all seven methods work best if the requirements and function structures are intrinsically modular*. Furthermore, these methods are only effective with product designs and relatively useless with the design of other kinds of system architectures.

The last two methods are matrix-based methods that are based on earlier work by Warfield [1973] and Steward [1981a]. The strength of these design analysis methods using *adjacency matrices* is the identification of strongly related components in a graph representing the components of the system and their interactions. As traditional modularizing methods use function structures as starting points, *matrix methods identify related groups of system elements based on the dependency between them*. While the method by Kusiak & Huang [1996] uses binary matrices, Pimmler & Eppinger [1994] propose a system clustering method based on the importance of the interactions (see classification of interactions between system elements earlier in *Section E.3.3*).

In this thesis, DSM is applied for the description and manipulation of the *system structure* and modularization due to its simplicity and general applicability (*i.e.,* it can be applied to any kind of system structure, not just to products). Furthermore, the possibility to directly apply mathematical algorithms to rearrange and optimize the system matrix makes DSM appealing for system analysis. The next section deals with the DSM method. After a brief introduction, two basic types of DSM are introduced, and algorithms for DSM analysis are described.

E.3.3.3.1. Design Structure Matrix Method

The design structure matrix or dependency structure matrix (DSM) is a matrix-based modeling method for the representation of the order and structure of the elements in a system. Hence, the purpose of DSM is slightly different from traditional methods for system architecting (*e.g.,* IPO), where the main goal of modeling is to represent the system in its entirety and include all necessary attributes for the thorough understanding and description of the system.

DSM *is one view of the system*, which considers the elements to be black boxes and describes only the dependencies among elements in the system. So, DSM basically does not account for the element-internal system attributes like inputs-outputs, functions, and properties of the elements. Two views of a system using the IPO and the DSM notations are depicted in *Figure E.3* to demonstrate the difference between traditional and matrix-based modeling. Though the IPO view includes a high amount of information about the elements and their attributes, even a system with six elements and a few relations (see *Figure E.3*) takes some time to understand. On the contrary, the DSM view contains less information, but if someone is familiar with the notation (*i.e.,* he or she can "read" the DSM), even a more complex system than the one in *Figure E.3* is easy to handle. Hence, DSM emerged as system complexity reached a level where the number of elements and relations were so high that the transparency of the models disappeared.

In reality, complex systems like an automobile or an aircraft comprise tens of thousand of components in various hierarchy levels. This structural complexity is no longer manageable for a conventional system modeling method, which is intended to depict every element with its diverse functions, properties, and relations. Thus, the model loses its basic function, *i.e.,* to help the modeler understand the modeled system. The famous metaphor compares highly complex conventional system models to *spaghetti with meatballs*, where the meatballs are the elements and the spaghetti are the numerous relations among them. This metaphor can be easily understood if one considers what an IPO model with one hundred elements and one thousand relations would look like, as an IPO view of with six elements is so complicated like the one in *Figure E.3*.

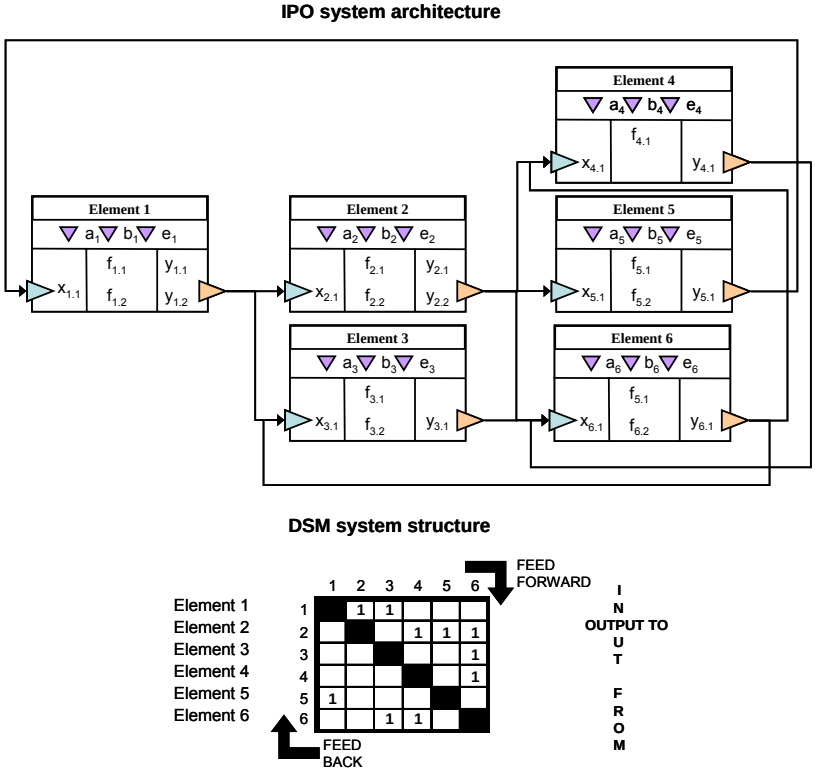


Figure E.3 IPO architecture and DSM structure of a system

As the behavior of systems arises from their structure [Sterman 2000], the modeling and optimization of the structure to foster system control is vital for the development. Furthermore, the greatest leverage in system architecting, process improvement and organizational planning is at the interfaces [Browning 2001]. Thus, the application of matrix representations of a system structure is not new in systems engineering, *e.g., N^2 charts* are commonly used to display subsystem and component interactions [*e.g.,* Grady 1994, DoD 2001a], the roofs of the houses in *Quality Function Deployment (QFD)* depict the interactions between the system elements in the lower parts of houses [*e.g.,* Hauser & Clausing 1988], and element-element matrices were also proposed to show organizational dependencies in companies [Walther 1994, Igenbergs 2000].

The DSM method was originally defined by Steward [1981a, 1981b] and then further developed by Steve Eppinger and his research group at MIT [*e.g.,* Pimmler & Eppinger 1994, Smith & Eppinger 1997a, 1997b, Browning 2001, Sharman & Yassine 2003]. Though different DSM techniques were developed to depict and analyze various types of systems, there are four basic DSM types used to model the four subsystems of the original ZOPH model[2] [Browning 1998a, Browning 2001]:

- *Component-based DSM:* for modeling different representations of products and their components based on component interrelationships (*e.g.,* function architecture, physical architecture, *etc.*)

- *Team-based DSM:* for modeling the organization structure based on the information flow between people and groups of people

- *Activity-based DSM:* for modeling the process and project schedule based on the information flow between activities

- *Parameter-based DSM:* for modeling the design parameter structure or goal system structure, and low-level relationships between design decisions and parameters, systems of equations, subroutine parameter exchange, *etc.*

Research work inside and outside of MIT was carried out to apply DSM for different purposes in the four main areas. Steward [1981a, 1981b] sees DSM as a basic system modeling and architecting tool, and uses it to model and analyze the system design process and systems of mathematic equations. Pimmler and Eppinger [1994] explored how DSMs can be applied with product architecting and how system modules can be identified based on component interactions. The breakthrough work of Pimmler & Eppinger in the area of matrix-based product architecting was followed by others, who used and improved the first DSM models for product modularization [*e.g.,* Kusiak 1999, Baldwin & Clark 2000, Rushton & Zakarian 2000, Sharman & Yassine 2003].

McCord & Eppinger [1993] applied DSM for the planning and optimization of cross-functional teams in SD. Browning [1998b, 1999c] continued this work and applied DSM to study organizational dependencies and support the selection of appropriate integration mechanisms. Wenzel [2003] used DSM for organizational planning and team definition based on the individual skills of the members of the organization.

[2] Note that the generic nature of DSM also allows the modeling of both the system environment (*e.g.,* network of competitors or customers) and the technology system (*e.g.,* dependencies of existing and/or emerging technologies or tools) introduced in the ZOPH+T model. However, the author is not aware of the application of DSM in these two areas.

The application of task-based DSM for project scheduling, iteration modeling, and the implementation of concurrent engineering are the most popular DSM-related research areas. The DSM approach for managing concurrent engineering proposed by Eppinger [1991] builds on Steward's original work and opens a new era in matrix-based process modeling. During the last decade, various approaches were proposed for DSM-based modeling of different aspects of the development process, *e.g.,* parallel, sequential and hybrid design iteration [Smith & Eppinger 1997a, 1997b, 1998], risk-based process simulation and the modeling of value streams in the SD process [Browning *et al.* 2002, Browning & Eppinger 2002], planning for concurrency [Denker *et al.* 2001, Yassine & Braha 2003, Yassine *et al.* 2003], *etc.*.

The fourth DSM application area with maybe the least invested research effort: the parameter-based DSM. The goal of the application of this kind of DSM is to understand the parameter relationships in the system design, and rearrange parameters and thus the design activities that affect them to reduce process schedule. While there are a few examples for the application of parameter DSMs in industrial practice, research applications and method improvements are rare in literature. Rask and Sunnersjö [1998] used a parameter-based DSM to describe the relationships between design variables of a robot arm and its housing. Black *et al.* [1990] applied a parameter-based DSM to automobile brake system design. Future research in this area is expected to concern multidisciplinary design optimization [Browning 2001].

Finally, Schulz [2003] combined DSM with IPO in his pioneering work for the decomposition and optimization of the *information architecture* of the complex process of automobile SD. The proposed model uses a genetic algorithm to find the ideal information structure for a certain SD system. Schulz's DSM approach can be considered as a fifth DSM category besides component-based, team-based, task-based, and parameter-based DSMs.

E.3.3.3.2. DSM Basics

DSM is an adjacency matrix, a matrix view of a graph depicting the elements of a graph and their relations. As shown in *Figure E.3*, DSMs are square matrices, *i.e.,* the labels of the rows and columns in the DSM are identical representing the system elements. The cells of the matrix depict how elements interact with other elements in the system. The diagonal in the DSM is usually painted black, indicating that these cells are unused, because *reflexive relationships* (*i.e.,* the output of an element is fed back to the input of the same element) are not allowed in DSMs. Along the row, it is possible to see which other elements interact with the one in the row, while the marks in the columns show the other elements the element in the column depends on. That is, while the rows represent output relations, the columns show which elements provide inputs to the examined one[3].

DSMs are mainly used for visualization and tracking of the element dependencies in complex structures and the identification of elements in close interaction called *chunks*. Regarding the function of the DSM, Browning [2001] distinguishes between static and time-based or dynamic DSMs.

E.3.3.3.3. Static DSMs

Static DSMs are used to display and analyze systems where the elements exist simultaneously, such as product and organizational structures. In these systems, many components interact at the same time, and the changes of the system structure in time are not important for the analysis. The DSM analysis techniques usually applied with static structures are the *clustering methods*. The

[3] Some authors use DSMs where rows represent input and columns output relations (exactly the opposite way as here). In these DSMs the feed forward relations are under the diagonal and feed back is over it. Otherwise the DSMs using both kinds of notation are identical.

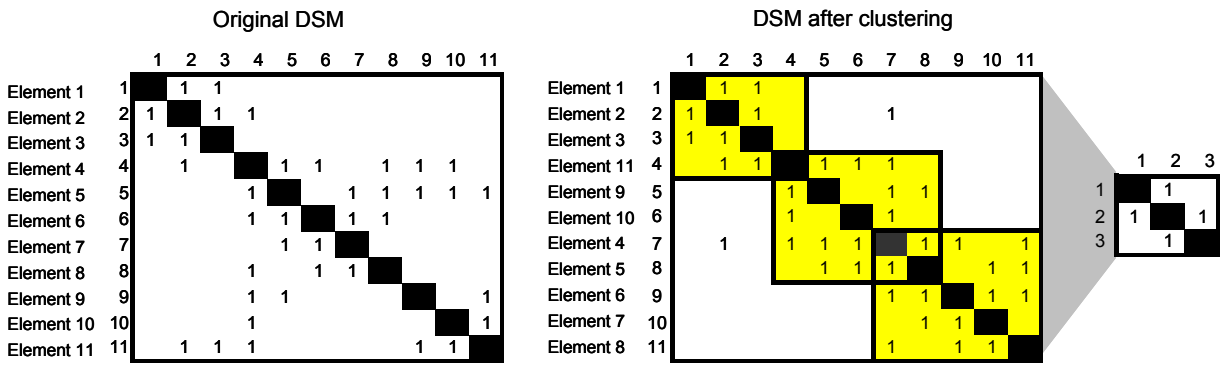


Figure E.4 DSM clustering results

goal of DSM clustering is to find subsets of DSM elements (*i.e.,* clusters or modules) that are mutually exclusive or minimally interacting. In other words, clusters contain most, if not all, of the internal interactions (*i.e.,* DSM marks), and the interactions or links between separate clusters are eliminated or minimized [Fernandez 1998]. Clusters resulting from the DSM analysis are then called modules, teams, or departments that collect functionally related and thus strongly interacting components.

The DSM view of a system structure is not just simple and appealing, but it is a system representation in the language of mathematics (*i.e.,* a matrix representation of a graph). Thus, a DSM model can be directly manipulated using existing and improved mathematical operators and algorithms. *Figure E.4* depicts a system structure before and after clustering. The resulting system structure after clustering shows three main, overlapping chunks that entail all elements of the system, and are connected to each other via a minimal number of interfaces. A comparison of the results of clustering in *Figure E.4* and the block diagram in *Figure E.2* representing a modular system architecture shows the power of DSM concerning modularization.

As the DSM in *Figure E.4* shows, the clusters have common elements belonging to two clusters[4]. Designers do not always desire overlapping elements, thus if a clustering algorithm shall seek overlapping elements, it also has to be part of the goal of clustering. Common clustering goals can be, *e.g.,* minimum number of extra-cluster interfaces, minimum number of elements in a cluster, maximum number of clusters, enable overlapping cluster recognition, enable bus recognition, enable three dimensional cluster recognition, *etc.* [Sharman & Yassine 2003].

Numerous researchers have used DSM to propose architectural improvements by simple manipulation of the order of rows and columns in the matrix [*e.g..,* Kehat & Shacham 1973, Hartigan 1975, McCord & Eppinger 1993, Pimmler & Eppinger 1994, Thebeau 2001]. Fernandez [1998] improved these first "manual" clustering algorithms in an attempt to automate DSM inspection and manipulation using *simulated annealing* techniques. Whitfield *et al.* [2002] developed similar DSM clustering techniques using *genetic algorithms*. However, as Sharman [2002] showed, both approaches are incapable of predicting the formation of "good" clustering arrangements for complex product architectures due to the oversimplification of the objective function utilized, and the frequent susceptibility of the search algorithm used to be trapped in local optimal solutions. Yu *et al.* [2003] improved existing clustering techniques by using a genetic algorithm and the minimum description length principle, and validated it by comparing the results of their DSM tool with manual clustering results of experts. As their results show, they succeeded to improve both the outcomes of existing algorithms and the experts' manual clustering results.

[4] With the DSM clustering in *Figure E.4*, the algorithm using genetic algorithms proposed in [Yu et. al. 2003] was applied. The author thanks Professor Ali A. Yassine at University of Illinois at Urbana Champaign for providing access to the web-based version of the DSM clustering tool proposed in [Yu et. al. 2003].

E.3.3.3.4. Time-Based DSMs

Time-based DSMs depict a sequence of system elements (*i.e.,* a *process*) in time. Thus, the interactions among elements in the matrix show how the output product of one element in the process activates another one through its input. Furthermore, time-based DSMs show the information needs of the activities—*i.e.,* which activity deliverables are required for the ideal operation of an activity. In an ideal SD process, information flow is optimal among the activities (*i.e.,* the right information is available at the right time and place), and thus process performance is maximal. However, complex SD processes are not ideal in reality. Activity deliverables do not always arrive on time and at the desired technical maturity. Hence, project managers usually have to make a tradeoff between the effectiveness and the efficiency of the SD process.

If project planning strives for process effectiveness (*i.e.,* the goal is to produce the highest performance outputs from given inputs), theoretically the activities would have to wait for all required inputs and start only after all related upstream activities are finished and delivered the desired output products. This strategy can be dangerous, if problems occur in the SD process that can lead to long iterations. In this case, activities depending on these deliverables have to wait until adequate solutions to the problems are found. This reduces process efficiency considerably and causes costly delays in the SD.

However, process effectiveness is not the only planning goal. As the time factor in SD is the source of major competitive advantage [Smith & Reinertsen 1998], companies try to reduce the duration of the SD process and thus time-to-market as much as possible. An effective method to reduce SD time is to perform activities and sub-processes concurrently and overlapping. In this case, activities do not wait for all required input products, but start with imperfect information and update their results when the missing input products arrive. Nevertheless, concurrent engineering and activity overlapping, if not managed well, can lead to serious oscillations in the process resulting in delays and schedule overruns in the project. DSM fosters planning for concurrency by enabling the identification of blocks of activities that can be performed in parallel without paying the high penalty of oscillation.

Element relations in time-based DSMs can be twofold based on their direction. Downstream relations between elements that are parallel with the process flow are over the diagonal and often called *feed forward* relations. Relations indicating that the product of an element affects an upstream activity (*i.e.,* an activity that was finished earlier in the process) are called *feedback* relations. The existence of feedback relations in a process implies a possible need for iteration on the deliverables of the SD that usually results in the repetition of the activities directly and indirectly affected by the feedback relation. As iterations are main causes for delays in the SD process, an objective of process planning is to *reduce the number and scope of iteration cycles to the required minimum.*

DSM *sequencing* or *partitioning algorithms* help optimize the sequence of activities in a process by manipulating the process structure. During partitioning, the goal is to get the DSM in an upper-triangular form to the highest possible extent with a minimum number of sub-diagonal marks pulled as closely to the diagonal as possible and grouped in blocks [Browning 2001]. These

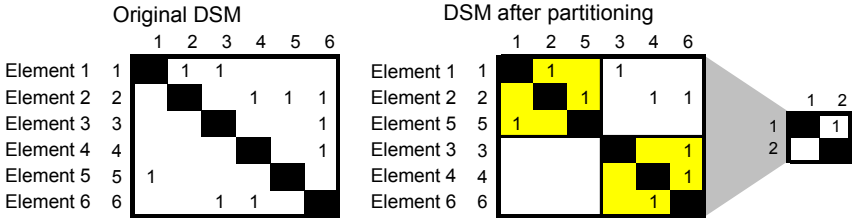


Figure E.5 Effects of DSM partitioning on the system structure

blocks of activities are the major iteration cycles in the SD process including only the activities directly involved in the design iterations. The decomposition of the SD process into a set of clearly defined blocks of highly dependent activities makes iteration cycles leaner and more efficient. Besides, it enables the parallel execution of independent activity blocks.

Figure E.5 depicts the effects of partitioning on the process structure. The integrated process of six activities and three feedback relations in the left DSM could be transformed into a process with two sequential activity blocks without any need for iteration between the two blocks. The two activity blocks can be considered as two main stages of the SD process with clear, measurable deliverables. Thus, as a result of DSM partitioning, a complex process with high likelihood of long rework cycles could be improved to a process with two activity blocks including short, effective iteration cycles.

Numerous researchers developed algorithms for partitioning [*e.g.,* Weil & Kettler 1971, Warfield 1973, Steward 1981b, Gebala & Eppinger 1991, Kusiak & Wang 1993, Tang *et al.* 2000]. Similarly to clustering algorithms, traditional partitioning algorithms are also different in their functions and outputs based on the goal of the analysis [Whitfield *et al.* 2003], for example:

- The *triangularization algorithms,* proposed by Kusiak & Wang [1993, 1995], attempt to reduce iterative blocks by minimizing the sum of the dependencies under the diagonal (*i.e.,* feedback relations) based on their weight. This algorithm does not consider the distance of the relation mark from the diagonal.

- Gebala & Eppinger [1991] developed a *partitioning* technique that similarly to the triangularization algorithms reduces the size and number of the iterative blocks by minimizing the sum of the dependencies above the leading diagonal. However, it goes further and multiplies the dependencies by their distance from the diagonal on the basis of their weight. The focus of Gebala & Eppinger partitioning is therefore to get as many dependencies either below the diagonal or, as close to it as possible.

- A third algorithm for partitioning was proposed by Scott [1998] to improve the Gebala & Eppinger algorithm. The improvement in Scott's algorithm is that he introduces an additional weighting factor for the dependencies based on their distance from the top right-hand corner of the matrix. Thus, the goal of partitioning here is not just to reduce the size and number of iterative blocks, but to move the dependencies into the right-hand corner.

Another group of partitioning techniques applies genetic algorithms (GAs) to achieve better results [*e.g.,* Rogers 1989, 1996, McCulley & Bloebaum 1996, Whitfield *et al.* 2003, Zhuang & Yassine 2004, Meier 2005]. GAs are particularly successful in case of high system complexity, where traditional partitioning and clustering algorithms are rather ineffective.

E.3.4. System Behavior in Dynamic Environments – Robustness and Flexibility

The value of a system can be determined through its behavior. That is, the way a system transforms *inputs into outputs* (*i.e.,* its performance) defines its value for the system environment. *System performance* is defined as `those operational and support characteristics of the system (or method) that allow it to effectively and efficiently perform its assigned mission over time` [DoD 2001b]. That is, performance depends on two characteristics (*Figure E.6*): (1) *efficiency* implying `the amount of output produced relative to the amount of resources (time and money) that go into the system` [Wikipedia Website]; and (2) *effectiveness* meaning `the extent to which the goals of the`

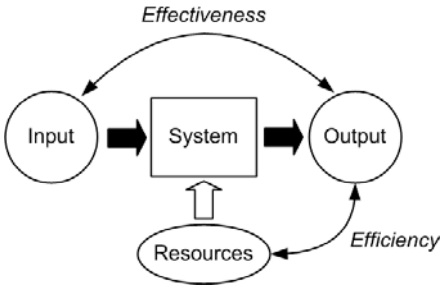


Figure E.6 Meaning of effectiveness and efficiency

system are attained, or the degree to which a system can be elected to achieve a set of specific mission requirements [DoD 2001b].

Obviously, the fundamental goal of systems engineering is to define, design, and operate an SD system with maximal performance, *i.e.,* a system that provides superior outputs at minimal resource consumption. Hence, an SD system with high performance delivers products with high market value and ensures high profitability due to cheap operation (*i.e.,* low resource consumption).

The attainment of constantly high system performance is more difficult if the system environment is dynamic and the inputs (*i.e.,* market and technology characteristics) vary during the system lifecycle. To deal with a high variance of input parameters, theories aiming to increase the *robustness* of the system were developed (*e.g.,* experimental design [Cochran & Cox 1957], robust design [Taguchi 1993]).

Robustness (*Figure E.7*) characterizes a system's ability to be insensitive towards changing environments [Fricke & Schulz 2005]. Robust systems deliver their intended function under varying conditions without being changed [Taguchi & Clausing 1990, Taguchi 1993, Clausing 1994]. Hence, robustness is the ability of the system to do its basic job in unexpectedly adverse environments [McManus & Hastings 2005].

Though robust systems are capable of delivering outputs with constant performance in the presence of high input variance, changes in inputs often demand respective changes in the outputs. That is, shifting customer's needs can often only be satisfied through the capability of the SD system to deliver a higher variance customized outputs. In this case, parts of the SD system have to be adapted to the changed conditions to increase system effectiveness.

Systems that have the ability to be modified to do jobs not originally included in the requirements definition are called *flexible systems* [McManus & Hastings 2005]. Nilsson & Nordahl

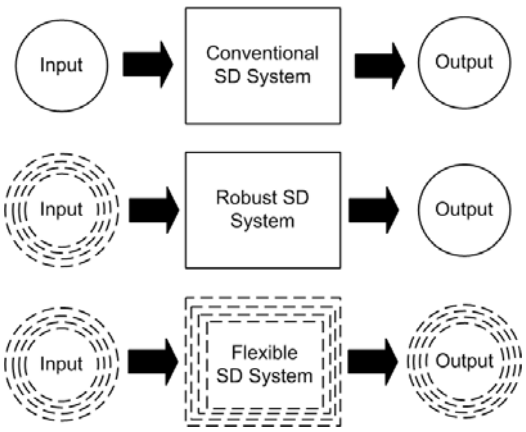


Figure E.7 Different SD systems in dynamic environments

[1995] cite a definition of flexibility as the ability to respond effectively to changing circumstances. Upton [1994] defines flexibility as the ability to change with little penalty in time, effort, cost or performance. Finally Rajan *et al.* [2004] propose that flexibility can be defined as the degree of responsiveness or adaptability for any future change in a product design.

Thomke [1997] reviews a large body of literature dealing with the understanding of the different aspects of flexibility in *manufacturing systems* [Zelenovic 1982, Gerwin 1987, De Meyer *et al.* 1989, Suarez *et al.* 1991, Upton 1994, De Groote 1994], the *economics of firms* [Jones & Ostroy 1984, Carlsson 1989] and *with competitive strategy* [Garvin 1988, Ghemawat 1991]. Furthermore, product flexibility as a *driver of productivity and innovation in the SD* was documented in [Baldwin & Clark 2000, Clausing 1994, Cusumano 1992, Eisenhardt & Tabrizi 1995, Iansiti 1995a, Ulrich 1995, Fricke *et al.* 2000, *etc.*].

Designing flexibility into a system is a strategic decision based on three internal and external SD factors: (1) dynamic marketplace, (2) technological evolution, and (3) variety of environments [Schulz & Fricke 1999, Fricke & Schulz 2005]. In case these three factors of the SD are highly dynamic, the probability of change is considerable and thus it is sensible to invest in implementing flexibility into the system architecture.

Thomke & Reinertsen [1998] argue that flexibility should be viewed as a *parameter in an economic tradeoff*. The extra investments in the flexibility of design aspects and system modules are only worthwhile if the benefits through easy design adaptability and fast response to shifting needs exceed the costs committed to implement flexibility.

However, as the notion of flexibility implies, changes are often difficult to forecast and so, the gains of the committed investments into system flexibility are also difficult to quantify. Thus, incorporating flexibility into systems is often considered an implementation of options into a system that *might but does not have to* be used in the future. That is, flexibility allows the easy adaptation of the system to changed needs, however if these changes do not occur, adaptations are not required either. Hence, flexibility is like insurance, it makes sense to invest in it in case of high consequences of a relatively likely event (*e.g.,* high likelihood of changes in market of technology characteristics).

Options thinking is a new paradigm in engineering design based on economic options theory [Trigeorgis & Mason 1987, Triantis & Hodder 1990, Faulkner 1996, Amram & Kulatilaka 1999]. It uses economic methods to calculate the value of flexibility in present value terms. Measures like *Net Present Value* or *Discounted Cash Flow* are applied to provide the designers with information on the current value of the future change options they can buy through present investments into design flexibility [Baldwin & Clark 2000].

A distinguished method to analyze and quantify the present value of design flexibility is the theory of *Real Options*. De Neufville [2003a] defines real options as elements of a system that provide "rights, not obligations" to achieve some goal or activity. Generally speaking, all elements of a system that provide flexibility can be considered as "real options". Using real options analysis, systems designers can compare the value of flexibility with the cost of acquiring it, and they can make an informed, analytic judgment about whether this flexibility should be incorporated into design [De Neufville 2003b].

E.3.5. Modularity and Real Options

In traditional engineering design theory, technology characteristics and market needs are considered to be stable or predictable during and after the SD project, and the main goal of SD is to reduce technical risk to a minimum. Furthermore, in conventional projects the focus is on reliability and making the best decisions in risky situations. In short, conventional SD is *reactive* to

risk [De Neufville 2003a]. This way, the outcome of the SD project will be a system meaning high value to the customer. In such a context, major changes are not anticipated and the cost invested into flexibility is unnecessary.

On the contrary, options thinking in engineering considers the future to be unpredictable in its entirety. Thus, to think in terms of options alters the way one deals with uncertainty. That is, changes are likely, and the product design and other subsystems of the SD system have to be planned to account for changes. Uncertainty in this context is considered to be something positive, because it may add value to options. Hence, adaptive SD is *proactive* to changes and uncertainty, and strives to create systems that are capable of seizing design opportunities through changes.

Real options in a system mean extra investments during design that allow a higher number of design alternatives the SD organization can benefit from at a later point of time in the system lifecycle. That is, real options do not directly contribute to the performance or operational characteristics of the system, but through real options, the ability is implemented to host future design opportunities. Thus, the value of real options lies in the capability of an SD system to react quickly and at low cost to changes in its inputs.

The modularization of system architecture is an investment that increases the independence of system modules. It was shown in the last section that highly modular systems accommodate changes by reducing their effects to the scope of the module [Baldwin & Clark 2000]. While modularization might even contribute to a decrease of system performance [Smith & Reinertsen 1998], it is only valuable if the gains through cheap adaptations on the module level compensate the likely loss on performance. Hence, modularity is profitable if system flexibility results in a competitive advantage for the company in the long term.

The capability of a system to manage changes effectively while responding to external and internal needs is fundamental for an adaptive SD system. With this capability, the SD system can continuously collect feedback from its environment on the value of its behavior and adapt itself to maximize value. Changes in the system during an adaptive SD can mean minor module-internal adjustments or major design changes where the modular system design architecture is manipulated, *e.g.,* using the six modular operators (*splitting*, *substitution*, *augmentation*, *exclusion*, *invertion* and *porting*) proposed by Baldwin & Clark in [2000].

One way of utilizing the increased flexibility of modular systems is to develop products in evolving increments. Incremental SD has the strengths of the evolutionary development lifecycle and improves it by allowing for the delivery of increments of the system with partial functionality at the end of the incremental mini-projects. Between two released increments, the modules of the system can be developed independently and adapted according to the stakeholders' feedback using the modular operators. Thus, with incremental SD, design options can be kept open until the maturity of the technology achieves the required level and/or the customer is ready to pay for the increased functionality. This allows the SD system quick reactions at low cost.

Figure E.8 depicts the incremental evolution of a flexible, modular design as an output of an adaptive SD system. The architecture of the product is developed and adapted to the market needs in five systematic steps during incremental SD. The product could be *e.g.,* a mobile phone with a modular structure, where the first version or increment is a rather simple one, including the core phone module (red element), the software (light blue element), and a low-resolution color display as a special feature (dark blue element). The boxes with different colors represent the process of adding, substituting, excluding or upgrading modules of the same design, *e.g.,* the second version could include a photo camera (new orange box), the third one a color display with

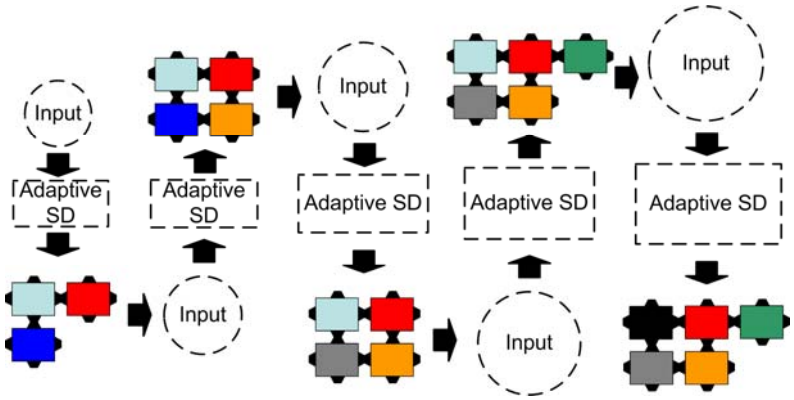


Figure E.8 Adaptive SD delivering incrementally evolving systems

a higher resolution (gray element), the fourth one a new Blue Tooth interface (green element), and the final increment a new operating system for the software of the phone (black box).

However, modularity is not the only requirement for system flexibility and adaptability. Hiding information, a few modules with clear, closely related functions and minimal inter-modular relations are only some of the basic characteristics of flexible systems. For example, to assure the smooth evolution of the incremental system in *Figure E.8*, it is essential to apply clearly defined, common, or standard interfaces and components. While changes or inconsistence in the interfaces between two system increments can lead to major compatibility problems, investments into interface standardization create important real options for the designers.

It is also important to obey a set of system-level design rules during modularization and adaptive development [Baldwin & Clark 2000]. Design rules can appear in form of reference architecture for a certain product or product family containing basic guidelines and constrains for system design. To be able to benefit from incremental evolution, it is important that developers follow the basic flexible design structure, because the range of system flexibility is not unlimited and changes against the design rules are quite expensive and thus not prohibited.

Another important system characteristic is the *synergistic specificity* of the system [Schilling 2000]. Synergistic specificity is the degree to which a system achieves greater functionality as its components are specific to one another. This emergent system characteristic describes the sensitiveness of the performance and functionality of a system towards different system configurations. That is, systems with high synergistic specificity have rigid architecture and design rules prohibiting major changes in the system. On the contrary, low synergistic specificity implies high architectural flexibility, *e.g.,* plug-and-play or mix-and-match modularity with LEGO-like system elements.

Other researchers like Fricke & Schulz [2005] propose that system flexibility can be described through some basic and extending principles. They derive three basic principles for flexibility (ideality, independence, and modularity) from distinguished design methodologies like TRIZ [Altshuller 1984] or axiomatic design [Suh 1990], and extend these by six further principles (integrability, autonomy, scalability, non-hierarchical integration, decentralization, and redundancy). This work can be considered as a first milestone in the long research process of classifying and characterizing generic flexible architectures.

E.4. CHAPTER SUMMARY

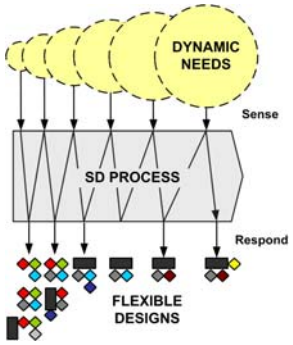
The high complexity of engineering systems is a main obstacle for adaptiveness in SD systems. This chapter showed that model-based system design fosters the definition of alternative system architectures and the selection of the best one for the system objectives. It was found that

adaptive SD systems with the capability of sensing changes and responding to these in form of reconfiguration call for a basically *modular* system architecture. Modularity contributes to high system flexibility by supporting the maximization of hidden information in the design, and thus the independence of the design modules. Flexible systems are designed to implement the capability of efficiently being changed in the system components under high uncertainty or ambiguity. This proactive design technique to deal with the unknowns regarding the future proposes to invest in real options in the actual design and profit from efficient changes resulting in higher value designs later. Adaptable subsystems are the basic building blocks of the adaptive ZOPH+T enterprise system.

F. UNCERTAINTY IN THE SYSTEM DEVELOPMENT – MANAGING THE DYNAMIC SYSTEM CONTEXT

F.1. CHAPTER ABSTRACT

This third chapter in a row describes the characteristics of modern SD projects and discusses the role of uncertainty in the SD. Uncertainty exists in SD projects, because the information available for the developers at the outset and during the project about the key characteristics of the SD system is often inadequate and even dynamically changing. That is, during the period between the documentation of market needs (*i.e.,* system inputs) and the launch of the first piece of product (*i.e.,* system output) the SD context changes, which affects the value of the developed system.



This chapter shows that adaptive SD systems are organized for *learning* that helps profit from environmental changes. Learning is the iterative process between sensing and responding that fosters both the effective reduction of risks and capturing of opportunities. Furthermore, this chapter proposes that the effectiveness of learning can be improved if conducted concurrently on independent subsystems. Finally, it will be underlined that learning is successful if divided into a process of focusing on the systematic generation of *inventions* (technology development) and another process of seeking *innovations* (SD).

F.2. DEFINITION OF UNCERTAINTY

A basic problem during SD project planning is how to deal with uncertainty and "vagueness", which cause risk. Uncertainty concerns the lack of knowledge about a problem at the point of time a decision on the solution of the problem is made.

> *Uncertainty is a condition, event, outcome, or circumstance of which the extent, value, or consequence is not predictable* [INCOSE 1998a].

Uncertainty usually comes in two forms: *foreseen* and *unforeseen uncertainty* (or *ambiguity*). Foreseen uncertainty describes the phenomenon that the *exact values* of the main characteristics of the key outcomes of the project cannot be predicted in the beginning of the SD. That is, at the outset of an SD project, the developers have only vague ideas about the future customer's needs, which define the objectives of the SD. Furthermore, they do not know *exactly* how to design, develop, and fabricate a product suitable for these unclear needs, which technologies will be available and effective in the product, how long it will take to develop and manufacture the product, and how much the total SD project will cost. Thus, they cannot predict the exact characteristics of the final product.

In decision theory, foreseen uncertainty is considered as a problem that can be modeled and solved using the *Probability Theory*. If there is more than one element in the design space with nonzero probability for the outcome of the SD, there is foreseen uncertainty [see Hazelrigg 1996]. That is, foreseen uncertainty means that there is a variation in the possible values of the performance characteristics of the end-product, and the cost, and duration of the SD. Hence, foreseen uncertainty is often referred to as representing the *known unknowns* in the SD, and the goal of decision-making is to maximize the stakeholder utility of the solution described by an objective function including all system objectives as variables.

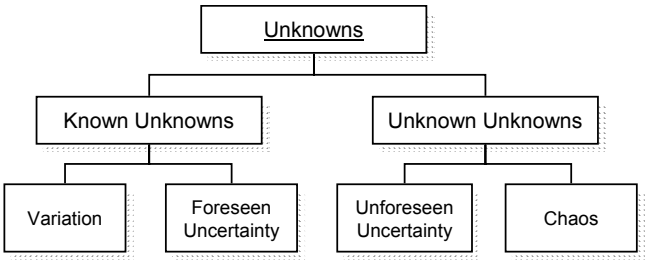


Figure F.1 Types of unknowns (based on [De Meyer et al. 2002])

A form of uncertainty is unforeseen uncertainty or ambiguity referring to the *unknown unknowns* in the project, meaning the absence of knowledge of the planning team about what will need to be done and when. Ambiguity is more dangerous than foreseen uncertainty, because it is more difficult to plan. The planning team is either unaware of an event's possibility or considers it unlikely and does not bother create contingencies (or preventive actions) for it. As De Meyer *et al.* [2002] argue, ambiguity is not always caused by spectacular out-of-the-blue events, but it can also arise from the unanticipated interaction of many events, each of which might, in principle, be foreseeable. Thus, ambiguity is related to the behavioral or dynamic complexity of the SD system, in which known or unknown agents (or subsystems) interact, and produce results that cannot be considered during planning, because of the lack of knowledge about these agents or their interactions. Unforeseen uncertainty can also be caused by unknown distortions, delays, biases, errors, and other imperfections of calculations or measurements [Sterman 2000]. It is particularly dangerous if such biases are caused by the developer's perceptions that constrain their mental models, because it slows down learning and the effective reduction of unforeseen uncertainty.

De Meyer *et al.* [2002] further define unknowns into four categories: (1) *variation*, (2) *foreseen uncertainty*, (3) *unforeseen uncertainty* (or *ambiguity* or *equivocality*), and (4) *chaos (Figure F.1*). These categories are defined on the basis of their relation to project management techniques, and they represent a fair classification of project uncertainty based on its criticality. Furthermore, according to the definitions above, both variation and foreseen uncertainty belong to the main category of known unknowns (*i.e.,* uncertainty), and unforeseen uncertainty and chaos are forms of unknown unknowns (*i.e.,* ambiguity).

Research in other areas provides similar classifications for uncertainty in the SD. For example, the body of work in organizational decision-making recognizes three types of decisions with regard to uncertainty: decisions under (1) *risk*, (2) *uncertainty*, and (3) *ambiguity* [*e.g.,* Daft & Macintosh 1981, Daft & Lengel 1986, Einhorn & Hogarth 1986]. Here, decision-making under risk constitutes the condition where information is not available, but a probabilistic description of the missing information is available (*i.e.,* variation at De Meyer *et al.* [2002]). Decision-making under uncertainty, in contrast, involves decisions where distributions are unknown. In this situation, less knowledge exists than compared with decision-making under risk (*i.e.,* foreseen uncertainty). Finally, decision-making under ambiguity involves a still more profound lack of knowledge, meaning that the functional form is completely unknown, and often that the relevant input and output variables are unknown. Hence, ambiguity here refers to both kinds of unknown unknowns in SD. Others distinguish merely between uncertainty and ambiguity [*e.g.,* Sarbacker & Ishii 1997, Haeckel 1999, Pich *et al.* 2002], or uncertainty and imprecision (*i.e.,* variation in design parameter values and uncertainty in design decision) [Antonsson & Otto 1995], or deal only with uncertainty [*e.g.,* Hazelrigg 1996, De Laurentis & Mavris 2000, DoD 2002, Ulrich & Eppinger 2004] as a global term for the problematic concerning the lack of knowledge when planning, designing, modeling, or making decisions.

Conventional systems engineering and design theory differentiate between two types of uncertainty: *variation* in the output of a design or manufacturing process due to known and

unknown noise factors [*e.g.*., Taguchi & Clausing 1990, Thornton 2001], and technical uncertainty that causes *risk* in SD [*e.g.,* NASA 1995, Shishko *et al.* 2004]. Both categories describe uncertainty regarding known, vague aspects of the SD that are then attacked using statistical design methodologies (*e.g.,* robust design [Taguchi 1993]) or standard risk management approaches.

Thus, in engineering too often the false assumption is made that all possible problems and uncertainties are known at the outset of the project, and they can be effectively handled through systematic design work and risk management. However, the growing system complexity often results in unforeseen events during SD (both in a positive and negative sense), which cannot be handled effectively using conventional techniques. Furthermore, sudden, unanticipated changes in the customers' needs that remain undiscovered by the SD organization can result in a situation where the wrong system is developed right, *i.e.,* a product is developed for invalid requirements. Hence, the adaptive SD framework proposed in this thesis accounts for both main categories of uncertainty in the SD: uncertainty and ambiguity, and explores how project management and systems engineering can jointly overcome these difficulties.

F.3. TYPES OF UNCERTAINTY

Uncertainties in a project that are the main sources of risks—and also opportunities—can be classified into different categories. The following basic types of uncertainties were collected after reviewing relevant literature on uncertainty and risk management [Browning 1998a, 1999a, 1999b, Huchzermeier & Loch 2001, Thomke 2003]:

- *Development cost uncertainty*: It indicates the uncertainty in the ability of a project to develop an acceptable design within a given budget. Causes can be cost attentiveness, available budget, quality of budget planning, resource availability, performance uncertainty, schedule uncertainty, and schedule rate change [Browning 1998a]

- *Schedule uncertainty:* It implies the uncertainty in the ability of a project to develop an acceptable design within a span of time. Causes of schedule uncertainty are, *e.g.,* number of intentional and unintentional iterations in the SD process, activity set completeness, activity flexibility, resource availability, iteration scope, durations and variations of constituent activities and/or sub-processes, available time and the unknown unknowns in the project [Browning 1999b]

- *Performance uncertainty:* It refers to the uncertainty in the ability of a design to meet desired quality criteria (derived from the performance objectives of the system). Furthermore, it is the uncertainty in the ability of the activities in the SD process to deliver the desired design on time and within the defined budget limits. Thus, performance uncertainty comprises both technical and process performance uncertainties. Technical performance uncertainty usually arises from the exploration of solutions that have not been used before, have not been combined in "this" way before, or have not been miniaturized in such a way before, *etc.* [Thomke 2003]. Additionally, Browning collects causes for both kinds of performance uncertainties, *e.g.,* inadequate design, development, and decisions; improper design evaluation; high product complexity; the distribution of risks across the system; schedule uncertainty; technology uncertainty and development cost uncertainty [Browning 1998a].

- *Technology uncertainty:* It is a subset of performance uncertainty meaning the uncertainty in the capability of a technology to provide performance benefits. Technology uncertainty can be a result of uncertainty in technology maturity, technology obsolescence, technology effectiveness, appropriateness of a technology for a given purpose and thus technology robustness, or technology capabilities [*e.g.,* Schulz *et al.* 2000]. Browning [1998a] defines

further causes: uncertainty in *e.g.,* technology and system coupling and sensitivity, familiarity of the design personnel with the technology, reliance on technology supplier, and ease of regulatory approval.

- *Market uncertainty*: It refers to the uncertainty in the anticipated utility or value to the market of the chosen "design to" specifications. It usually stems from severe market dynamics that make the market characteristics unpredictable.

- *Business uncertainty*: It implies the uncertainty in political, economic, labor, societal, or other factors in the business environment. If the political and business environment is not stable or predictable, the long-term operation is usually not profitable for the SD organization in that environment.

- *Production uncertainty*: It refers to the lack of knowledge about the producibility of a design. Some designs may work in the laboratory or even for production in small quantities, but it may not be feasible or cost-effective to ramp up production.

- *Need uncertainty:* It refers to uncertainty concerning the quality, completeness, and validity of the gathered customer's needs. Need uncertainty exists because customers are rarely able to articulate all their needs specifically, because they either face uncertainty themselves or cannot define needs on products that do not yet exist.

These eight categories cover the possible types of uncertainties a project manager has to account for during project planning and might face in the course of the project. *Figure F.2* depicts the interactions between the eight categories of uncertainties.

Environmental factors describe the characteristics and dynamics of the SD environment that affect the value of the product and the whole SD as well. In case the environmental conditions are uncertain due to frequent changes, the effectiveness of Marketing in the SD organization becomes vital. Furthermore, if the environmental characteristics are ambiguous, *i.e.,* the future changes are unpredictable, the SD organization has to move towards adaptive SD techniques that support the systematic collection of feedback from the stakeholders and allow easy adjustment of the system objectives. Since ambiguity in the market is for most SD organizations rather difficult to handle, the capability of developing and producing high value products in an ambiguous SD environment in the long term means a great opportunity for a company and can lead to growing market share and increasing profit.

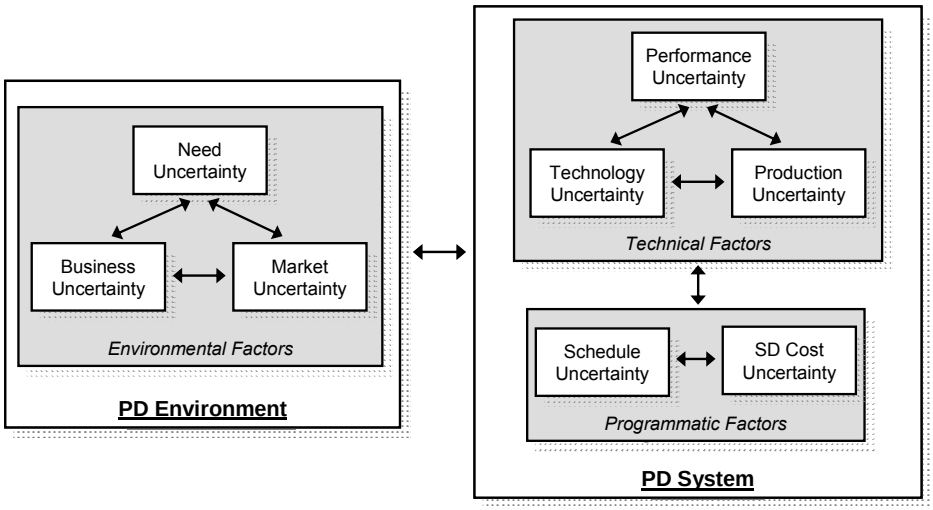


Figure F.2 Interrelations between uncertainties

The internal factors of uncertainty in *Figure F.2* can be grouped into two main classes: technical and programmatic factors. The bi-directional relationship between programmatic and technical uncertainty represents an important aspect of project management. That is, the tradeoffs made in the beginning of the project between technical and programmatic SD goals drive the uncertainties and thus risks and opportunities in the project. This means that projects with a low budget and schedule allocation have a limited space of possible technical achievements. It is simply not possible to achieve ambitious technical targets without appropriate investments. If project managers are not aware of that fact, and the technical performance targets are kept at an "ultra-high level" with low budget and schedule, the technical factors of uncertainty will increase and pull the programmatic factors with them. In this case, programmatic uncertainty has a positive feedback effect on technical uncertainty, *i.e.,* when programmatic uncertainty grows, technical uncertainty increases as well.

F.4. EFFECTS OF UNCERTAINTY

Uncertainty in SD means that the developers cannot accurately predict the final results of the project. Uncertainty in the probable final outcomes of the system performance is depicted in *Figure F.3*. The *x*-axis in *Figure F.3* depicts system technical performance as an aggregated measure of the key design aspects represented by technical performance parameters. It is used here to illustrate that the uncertainty in the overall system performance of the product is a range of possible outcomes. However, in parameter-based SD, PDFs for the overall system performance are rarely used. Usually, each design aspect is tracked individually, and only the value or risk of the overall system performance is calculated. Later parts of the thesis present the definition and tracking of technical performance parameters and the derivation of statements regarding the overall system performance and value. It is important though that uncertainty for all parameters describing the design and system technical performance can be demonstrated the same way as in *Figure F.3*.

The probability density function (PDF) in *Figure F.3* represents all likely outcomes of the overall system performance. Thus, as *Figure F.3* shows, uncertainty can mean that the final outcomes are inadequate (*i.e.,* they represent a lower performance level than the customer desired), but it is also possible that the performance of the final product exceeds the expectations.

The first case means possible negative events: *risk*, *i.e.,* possible profit loss for the SD organization due to lower performance than what the customer desired. This can result in lower sales volumes or lower prices and thus lower profit. The second possibility concerns positive events *i.e.,* if the performance deviation is positive and thus the product is better than expected, it is called *opportunity*. Better performance can be sold better and thus opportunities mean possible profit gains for the company. Opportunities have to be effectively identified and captured during

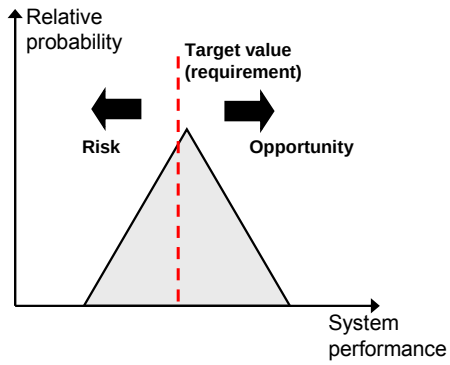


Figure F.3 Uncertainty in the system performance

the project, because they are the main drivers of innovation and thus profit in the SD. The following sections deal with these two phenomena.

F.4.1. Risk

The goal of SD is to define a set of criteria that represents the customer's preferences, and then design and develop a product that satisfies these criteria. Thus, *every activity in SD works to reduce the gap between the actual and the target system performance and thus to reduce risk*. Due to the inadequacy of traditional top-down engineering development models (*e.g.,* waterfall lifecycle) to support the organization and management of effective and efficient development work, conventional SD projects are characterized by an ongoing firefighting effort to reach the system objectives within the least possible budget and schedule overruns. In such projects, the ultimate goal is to find methods that help understand the SD system and thus effectively reduce and control technical risks that are the main drivers of programmatic risks.

As *Figure F.3* depicts, the left tail of the distribution representing unacceptable system performance outcomes below the target value means risk for the project. Risk is defined in technical standards as follows:

> *Risk is a measure of the potential inability to achieve overall program objectives within defined cost, schedule, and technical constraints. Risk has two components: (1) the probability or likelihood of failing to achieve a particular outcome, and (2) the consequences or impacts of failing to achieve that outcome* [DoD 2002]

The consequences are usually expressed in monetary terms, and thus the risk of a negative event is possible monetary loss. As risk is inherent in every SD project, SD can be considered as a process of uncertainty reduction and risk management [Browning 1998a]. All the functions of systems engineering strive to reduce uncertainty in the different disciplines of SD and decrease the impacts of inadequate outcomes. Risk is usually calculated with the following equation [*e.g.,* Meredith & Mantel 2003]:

$$R=PI \qquad \textbf{(F-1)}$$

where R is the measure of risk, P is the probability of an inadequate outcome, and I is the impact of that outcome. It is important for every engineer to understand the notion of risk, *i.e.,* risk is a product of two independent variables: probability and impact. Thus, in case a negative outcome of an event has a high impact (*e.g.,* because it can lead to an accident with fatal consequences), the risk can be still medium or low if the probability that such an event happens is low. Similarly, if a negative event happens quite likely in the future, but it does not significantly affect the behavior of the system (*i.e.,* the impact is low), then the risk is not significant either. Hence, risk management focuses on identifying the sources of risk (*i.e.,* the uncertainties) and seeking ways to reduce either the probability or the impact (or both) of a negative event to a level that reduces

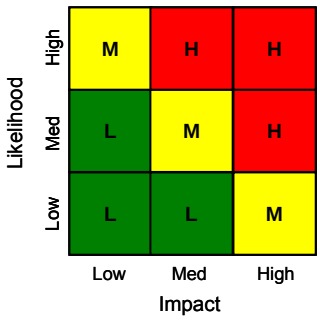


Figure F.4 Risk categories (modified from [DoD 2002])

the overall risk to an acceptable level.

The level of risk can be ranked on the basis of the combination of probability and impact *e.g.,* using a risk matrix depicted in *Figure F.4*. Risk matrices are effective tools to understand and classify risks in a project. However, such matrices provide only qualitative information on the risk status (*e.g.,* the risk is low, medium or high), and developers and managers usually prefer quantitative information (*e.g.,* the risk might lead to a possible profit loss of € X if not mitigated).

Risk can be quantified more exactly with *Equation (F-1)*. However, experts warn that the mere multiplication of uncalibrated values of probability and impact might lead to information that will at best be misleading, if not completely meaningless, resulting in erroneous risk ratings [DoD 2002]. As this kind of information is a main source of ambiguity, risk calculations always have to be based on calibrated values of probability and impact. One effective way is to use numbers between 0 and 1 for both probability and impact. This way, the product is also between 0 and 1, and it can be easily ranked according to a standard scale (*e.g.,* 0 - 0.33 low; 0.34 – 0.66 medium; 0.67 – 1.00 high).

F.4.1.1. Technical Performance Risk Calculation

Browning *et al.* propose a generic approach for risk calculation in [2002]. The proposed *risk value method* combines the basics of *Engineering Decision-making* [*e.g.,* Hazelrigg 1996], *Utility Theory* [*e.g.,* Fishburn 1970], and *Taguchi's Quality Loss Theory* [Taguchi & Wu 1980] in one effective method for risk estimation. The concept of the risk value method is that the customer value lost due to inadequate product quality is a function of the deviation of the actual from the target product quality. Thus, the *impact* of a deviation from the target value can be calculated as the expected stakeholder *utility* at the target minus the *utility* at the actual value. The left diagram in *Figure F.5* depicts a triangular PDF representing the possible outcomes of one key technical performance aspect of the design (*i.e., "TPM_1"*) and two different kinds of impact functions representing the consequences of the differences for parameters where *the larger is the better*. The right diagram shows the related utility curves representing the customer utility of the possible outcomes of the TPM.

Taguchi & Wu [1980] propose that the function of quality loss due to lower performance is quadratic as shown in *Figure F.5* (function *"A"* on the left). However in reality, quadratic functions are often difficult to define and thus simpler, linear (*"C"* in *Figure F.5*), or piecewise linear functions (*"B"* in *Figure F.5*) are applied for the definition of impact or utility as a function of system performance.

The risk of a possible TPM outcome can be then calculated using *Equation (F-2)* as the product of the probability and the impact of an outcome. In case of more than one possible TPM outcome, the overall risk is calculated as the sum of all possible outcomes, each weighted

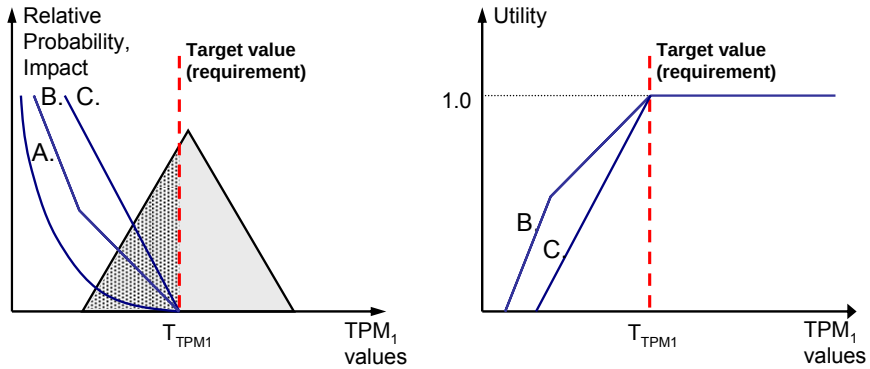


Figure F.5 Risk calculation using the possible outcomes of the system performance and impact functions

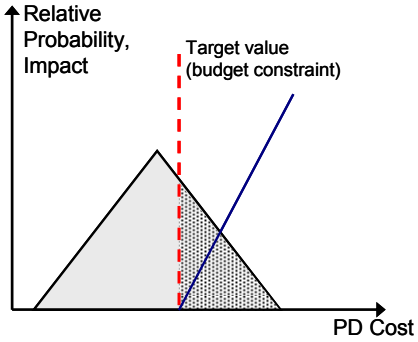


Figure F.6 Risk calculation using the possible outcomes of project cost and impact function

by its impact.

$$R_{TPM_1} = \sum_{-\infty}^{T_{TPM1}} f(x_{TPM\,1}) I(x_{TPM\,1}) \qquad \textbf{(F-2)}$$

where R_{TPM1} is the risk, x_{TPM1} are the possible outcomes of TPM_1, where *the larger is the better,* $f(x_{TPM1})$ is a discrete form of a PDF representing the probabilities of all x_{TPMi} outcomes, T_{TPM1} is the target or requirement value of TPM_1, and $I(x_{TPM1})$ is the impact function. In case the possible outcomes are represented by a PDF, the sum becomes an integral:

$$R_{TPM_1} = \int_{-\infty}^{T_{TPM1}} f(x_{TPM_1}) I(x_{TPM_1}) dx \qquad \textbf{(F-3)}$$

The result of this integral is the expected value of the impact function on the range of the possible outcomes. At technical parameters, where the nominal is the best, usually symmetric impact functions are used, where impacts can have nonzero values on both sides of the target values. Here, the ranges of the integral are set from -∞ to ∞. The integral in *Equation (F-3)* is an application of the *von Neumann-Morgenstern Expected Utility Theorem* [von Neumann & Morgenstern 1944] for risk calculation, where risk means the expected loss in stakeholder utility due to lower product quality.

The overall technical performance of a product can rarely be described by one, single technical performance measure. Thus, usually a vector of the most important performance parameters (*e.g.,* TPMs) is used to capture the key dimensions of the product that mean value to the customers. This vector of most important product attributes is used for the calculation of the overall technical performance risk as well. That is, the overall technical performance risk is a function of the risk of the k single TPMs:

$$R_{TP} = f(R_{TPM1}; R_{TPM2}; ...; R_{TPMk}) \qquad \textbf{(F-4)}$$

This function can take various forms. For example, Hazelrigg [1996] proposes different forms of the Multiattribute Utility Function (MAUT) to compute the overall value of a thing described by independent attributes, *e.g.,* linearly additive utility (or weighted average), multiplicative utility, log-linear utility, and lexicographic preference ordering. Browning & Hillson [2003] also discuss the applicability of different methods to acquire the value of overall technical performance risk (*i.e.,* weighted average, MAUT, and geometric mean), and conclude that the simplicity of the weighted average outscores the benefits of other sophisticated methods.

In this thesis, the weighted average of the risks in the different areas of technical performance is applied:

$$R_{TP} = \sum_k w_{TPM_k} R_{TPM_k} \qquad \textbf{(F-5)}$$

where R_{TP} is the overall technical performance risk, w_{TPMk} is the weighting showing the relative importance of the TPM value for the customer, and R_{TPMk} is the risk of the TPM_k.

F.4.1.2. Cost and Schedule Risk

During project planning, an SD process is defined, which is capable of delivering a product that satisfies the requirements. The planning of this process usually starts *top-down*, where the planners define the major deliverables and the cost and schedule targets for the main SD phases, and then decompose them into activities on various hierarchy levels. Once all activities and steps of the project are defined including their deliverables and the required resources, a *process model* is built by integrating the process *bottom-up*, using the activities and their deliverables as building blocks [*e.g.,* Browning 2002]. The result of the bottom-up process integration is then evaluated against the targets and constraints set during the top-down project decomposition. Cost and schedule risk can be captured during planning if the SD process delivered by the bottom-up planning requires more resources than defined during top-down planning. Furthermore, cost and schedule risk arise in the project when the actual cost and duration of the activities exceed the estimated values and thus threaten the planned overall project cost and duration.

This phenomenon is depicted in *Figure F.6*, where the possible outcomes of the SD cost include values above the target. As project cost and schedule are parameters where the *smaller is the better*, the outcomes that are higher than the target have an impact on the project. The impact of both SD cost and schedule overruns can be significant for a company, because it reduces the profit due to costly extra development effort, contractual obligations, critical market launch date due to scarce competition, *etc.*

Thus, during project planning, the three main dimensions of SD (*i.e.,* product performance, SD cost, and SD schedule), have to be balanced to achieve high system value. As the three factors are often conflicting, *i.e.,* high product performance usually contradicts with low cost and schedule, systems engineering management has to make compromises between the SD goals and find a way to achieve the highest product performance at the lowest cost and schedule *possible*. Since the optimum for the system is rarely the optimum for the single parameters (*i.e.,* the highest product performance is usually not the cheapest), when looking for the optimal balance for the project, the project team has to make many *tradeoffs* among the three SD dimensions. The balance is the place in the design trade space where the *overall SD risk* is the lowest.

The overall project risk is the function of the risks incorporated in the cost and schedule of the project and the technical performance of the outcome of the SD. A possible way to calculate overall SD project risk is simply to calculate the sum or the weighted average of the risks in the three SD dimensions. In this thesis the latter alternative is used, *i.e.,* the weighted average of the cost, schedule and technical performance risk of the project:

$$R_{PD} = \sum_i w_S R_S + w_C R_C + w_{TP} R_{TP} \qquad \textbf{(F-6)}$$

where R_{PD} is the overall SD risk, w_S is the relative importance of the SD schedule or duration, R_S is the schedule risk, w_C is the relative importance of the SD cost, R_C is the cost risk, w_{TP} is the relative importance of the product technical performance, and R_{TP} is the technical performance risk.

F.4.2. Opportunity – Design for Uncertainty

Risk management is a central task of every SD project. As Browning & Hillson [2003] remind us, *`project management is risk management`*. That is, the risks inherent in SD have to be identified and reduced to a level that enables the total fulfillment of the defined customer's needs. Thus, the possibility that the outcome of the SD shows a negative deviation from the customer's preferences and the impact of this difference have to be reduced to an acceptable minimum by the end of the design process. This kind of SD project management focuses only on the negative side of the distribution of the possible outcomes, and strives to minimize the possible deviation meaning lower product performance than desired.

However, due to scarce global competition, companies have to plan for changing customer's needs and strive to exploit all opportunities within the available resources to maximize the overall lifecycle value of the system. Thus, during design, the whole range of possible SD outcomes has to be considered, *i.e.,* the whole design space that is specified by the actually available and future technologies, and the available resources. As conventional engineering development attempts to deliver a design for a product specified by the customer's needs, developing the product specifications based *only* on the predicted future customer's needs limits the design space. Given the fact that customer preferences, market characteristics, the political and business environment, and the available technologies all include uncertainties and even ambiguities, limiting the characteristics of the design for a set of unclear values defined at the outset of the project often leads to sub-optimal designs and thus lost profit at the end. Hence, effective SD has to consider all possible design options for a certain product or product family even beyond the predicted future market needs. Furthermore, it must be possible to make design changes if shifts in the characteristics of the SD environment cause reductions in the lifecycle value of the actual design.

To enable long-term SD effectiveness, a paradigm shift in engineering thinking is required, and a new way of dealing with the effects of uncertainty has to be introduced for the design and development of engineering systems [De Neufville 2004]. In this new *design for uncertainty* philosophy, the goal is to design products and organize SD systems that are capable of reacting to foreseen and unforeseen future changes in the SD environment by adapting to the changed conditions. The premise is that just as engineers need to guard against failure, they should also enable the exploitation of unexpected opportunities that may be associated with the projects and products, those that drive innovation in the SD [De Neufville 2004].

An opportunity is considered to be the opposite of a risk, and thus opportunities are *possible positive events* that lead to profit gain (and not to profit loss). Opportunity is the combination of a *probability*, *i.e.,* the likelihood that a positive event will happen some time in the future, and an impact or in this case *benefit*, *i.e.,* the profit gained if the event happens. Each source of uncertainty, as discussed before, incorporates opportunities, *e.g.,* lower time-to-market or project

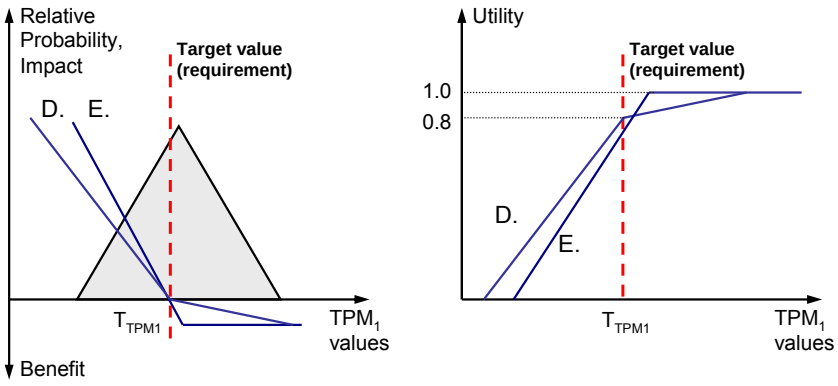


Figure F.7 Stochastic modeling of opportunity and risk

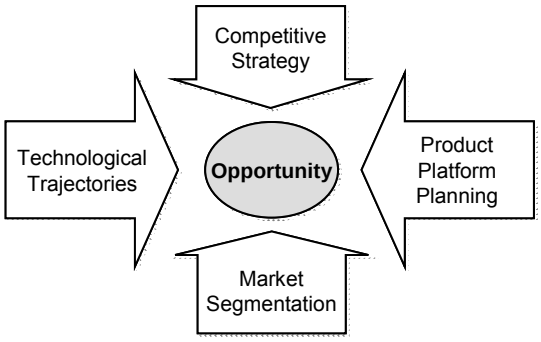


Figure F.8 Four basic perspectives that affect the value of opportunities

duration than expected, lower SD cost, better technical product performance, better market situation than expected, etc. All these various opportunities increase the overall system value to a certain extent.

Technical opportunity can be calculated stochastically using the same approach applied for the risk calculation before. If the whole range of outcomes is considered in the SD, and outcomes with higher performance than required mean value for the customer, it is deliberate to consider these outcomes during decision-making. Thus, beside the impact of inadequate outcomes also the benefit of outcomes that exceed the performance targets can be calculated. This is shown in *Figure F.7*, where the target value for TPM_t is set lower than the performance value with the highest stakeholder utility. Such a decision can have many reasons; usually it is the result of a tradeoff between the capabilities of the SD system, the resources that constrain the project, and the customer's desires regarding the overall system performance. However, it often happens that characteristics of the SD change during the project (*i.e.,* due to emerging new technologies or the discovery of novel design solutions), and it becomes possible to exploit design opportunities within the original project budget and schedule. As the exploitation of sudden opportunities demand continuous project control, quick reaction, and changes in the SD system, effective project management and flexibility in all ZOPH+T subsystems is required. Adaptive SD supports project management by continuously generating information on the performance of the SD process including the actual risk status and the stakeholder value of the discovered opportunities.

Opportunities regardless of their sources are worth capturing if they bring enough benefits to the SD organization. As uncountable opportunities arise during the various projects of a company, it is vital to evaluate their value and allocate resources only to the most promising ones. Ulrich & Eppinger [2004] propose that there are four basic perspectives that have to be considered during the evaluation and prioritization of design opportunities. These four areas depicted in *Figure F.8* define the dimensions of the SD strategy of the company. Thus, every opportunity chosen for exploitation has to fit in this strategy space and lead the company to the desired direction.

Furthermore, the value of an opportunity for an organization depends on the company's strategic flexibility as well. This aspect is depicted in *Figure F.9*, where

- the bold arrow in the middle represents the basic strategic direction of the company. The direction of the arrow is defined by the characteristics of the four perspectives affecting the value of an opportunity (*Figure F.8*) that describe the multi-dimensional strategy space of the company.

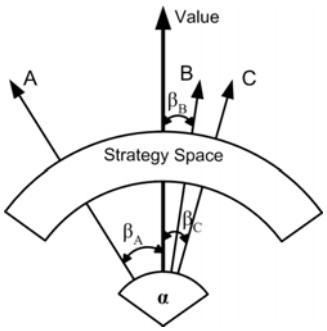


Figure F.9 Evaluation of opportunities

- "α" represents the range of the company's strategy landscape, *i.e.,* the range of opportunities that are valuable for a company regarding its strategic direction (*i.e.,* α is the strategic flexibility of a company);

- the arrows indicate different opportunities;

- "β" is the difference between the direction (*i.e.,* characteristics) of the opportunity and the main strategic direction of the company (*i.e.,* how the opportunity fits in the company strategy);

- the size of the arrow is the absolute stakeholder value of the opportunity;

- v*cos*β shows the relative value of the opportunities regarding the company's main strategic direction and product portfolio, *i.e.,* the value of the opportunity for the company.

Thus, β is related to market uncertainty, *i.e.,* the higher β is, the higher is the uncertainty that the realization of an opportunity adds value. The parameter α in the opportunity model represents the strategic flexibility of the company, *i.e.,* the range of uncertainty a company is ready to deal with in order to seize an opportunity.

Generally, it can be stated that the larger β is, the higher is the uncertainty regarding the value of the possible outcomes, because a large β means a basically new strategic direction and thus new markets, new customers and/or new products for the company. However, since uncertainty has good and bad sides with positive and negative impacts, a long $\underline{v}$ vector with large β (*i.e.,* high value) can incorporate great opportunities for an SD organization.

F.4.3. Net Present Value of Opportunities

Besides stakeholder value, which brings benefits to the SD organization, another important aspect of the evaluation of opportunities is the determination of the investments that are necessary to exploit an opportunity. Investments comprise the various costs required to design and implement a technology or the related product component from the discovered opportunity. The size of investments required developing a product or technology from an opportunity varies on a large scale depending on the characteristics of the opportunity and different aspects of the SD system and its environment.

Major design opportunities that enable the development of products guaranteeing long-term customer satisfaction are sought in SD to maximize stakeholder value. These opportunities are the fundamental drivers of innovation in new SD projects. However, not all SD cultures permit the effective discovery and exploitation of design opportunities. As the implementation of innovative ideas, the main source of design opportunities demand extra development effort and

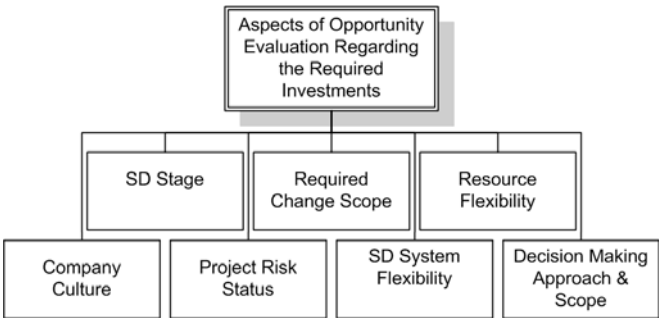


Figure F.10 Aspects of opportunity evaluation regarding the required investments

result in possible changes in the SD process, SD project environments with high resistance to changes reduce the ability to exploit opportunities.

A basic measure to estimate how much a company would gain through the exploitation of an opportunity is the *net present value (NPV)* of the opportunity [Wöhe 1996]. This measure is a powerful valuation method used in the capital markets to determine the future gains achievable through a present investment. The same method can be applied for design opportunities, where the total cost of the project to exploit an opportunity including SD, production, marketing, distribution, *etc.* is considered as the investment. The value of the project is then described by the benefits through marketing and selling the product. Thus, the measure NPV provides a means to quantify the future benefits of an actual investment (*i.e.,* decision on an investment). This is possible by comparing all cash outflows and inflows in the course of the project. The NPV of a design opportunity can be calculated as follows:

$$NPV = \sum_{t=0}^{n} \frac{V_t^j - I_t^k}{(1+i)^t} \qquad \textbf{(F-7)}$$

where V_t^j is the present value of all cash inflows (*i.e.,* earnings) and I_t^k is the present value of all cash outflows (*i.e.,* investments) regarding an opportunity, n is the total length of the project, t is the amount of time (*e.g.,* years), and i is the cost of capital. Thus, the higher is the NPV of an opportunity, the more profit the company can gain through investing in it[5].

Net present value of a design opportunity can be also calculated during an SD project, in case the outcomes of an SD activity, experiment, sub-process or even a whole lifecycle phase involve promising results. Opportunities discovered in the SD process can be seized within the scope of either the actual or a later project. The decision when to seize an opportunity involves many SD facets the project manager has to consider. The drivers of project benefits gained through an opportunity are depicted in *Figure F.8* and summarized in the measure: *value*. The other side of NPV is the *investment* required to seize an opportunity. The aspects of investments regarding the exploitation of an opportunity are depicted in *Figure F.10*. These investments are usually linked to the status of the project, the culture of the organization, and the available resources in the project.

An important aspect for this decision is the organizational and decision-making culture of the company, which constrains the considered decision options during decision-making. If risk reduction is the major project objective, capturing opportunities has low priority, and thus it is

[5] An alternative approach to NPV for the calculation of the future value of an opportunity is *real options* [*e.g.,* De Neufville 2003a, 2003b, 2004, De Weck *et al.* 2004, Shishko *et al.* 2004]. An advantage of real options is that is accounts for uncertainty and flexibility after the project decision. However, both NPV and real options can be effectively used to estimate the future value of an opportunity.

usually moved to forthcoming projects. This philosophy might be successful for risk reduction; however, it reduces the chances of developing superior products.

Another aspect of the in-project evaluation of opportunities is the actual project stage. The basics of the *rule-of-ten* [*e.g.,* Ehrlenspiel 1995, Swift *et al.* 1998], the well-known principle of quality management, are applicable here, too: the later a change is made in the SD, the higher are the consequences. That is, the cost of seizing opportunities found in a late lifecycle phase, at high design maturity are so high that it is not worth any more to invest in them in the scope of the actual project.

The risk status of the project is another important decision aspect. In case of high project risk, a design solution or technology that would reduce risk in the critical areas is always welcome and even sought by the developers, since it steers the project to the right direction. However, opportunities, even with high stakeholder value, that improve areas not in risk have to be evaluated thoroughly. On the one hand, these opportunities increase the lifecycle value of the system and thus foster product superiority. On the other hand, the manager of a project in high risk has to invest in risk reduction, and not in the improvement of design aspects not in risk. If the product and the SD structures are modular ensuring independent module development and even independent module development budgets, then the decision is easier. However, the changes in an integral product architecture that the exploitation of an opportunity might cause increase risk that is not desired in case of a high-risk project status.

The scope of the change caused in the SD system by the exploitation of a design opportunity is twofold. Whereas *direct changes* include the effort required to develop a mature product or technology from an opportunity, there are also indirect changes in the SD system due to this extra development effort, *e.g.,* due to change propagation from the improved area to other areas.

The direct cost of a change depends on two major aspects: the *affected design areas* and the *degree of innovation* in these areas. During the decision on investing in capturing the opportunity, the project manager has to make a tradeoff between the two factors. On the one hand, a change affecting many areas, or basic design characteristics and rules (*i.e.,* the design architecture); even a small adjustment can have enormous cost. On the other hand, if the improvement scope concerning the degree of innovation is high (*i.e.*, due to the discovery of a breakthrough innovation), it usually requires large investments. If it also affects many design aspects and modules, the cost of improvement might be too high to implement it in the actual project. This aspect, the affordability of opportunities, is captured in *Figure F.11* showing how resources limit the scope of changes.

Affordability also depends on the indirect changes an improvement action causes in SD. If a

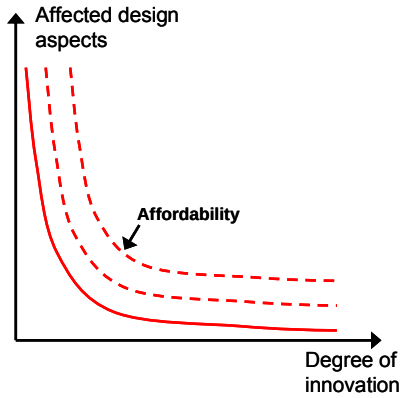


Figure F.11 Affordability as a function of change scope

change in one component or design aspect affects many related components or aspects due to high interrelatedness among system components, then the cost of the many small adjustments might sum up to a rather high total improvement cost that might not be affordable in the actual project. Thus, the high system visibility of the components in integral system architectures (solid line in *Figure F.11*) increases the indirect cost and thus risk of even slight changes. Hence, integral product architectures limit the scope of the in-project exploitation of opportunities.

Flexibility is a characteristic of highly modular system architectures [*e.g.,* Baldwin & Clark 2000]. In such systems, the affordability and controllability of the changes due to improvement actions is higher, because the effects are hidden in the modules. That is, the design aspects affected directly or indirectly by the change are limited to the size of the module that limits the risk of undesired change propagation. High flexibility in the product architecture also allows increased flexibility in other ZOPH+T subsystems of the development enterprise required to react quickly to foreseen and unforeseen changes in the SD. For example, high process flexibility driven by modular product designs enables even late, module-level changes in SD [MacCormack *et al.* 2001], which is one of the main risk drivers for traditional SD projects developing integral designs. The second part of the thesis discusses flexibility means in the different ZOPH+T subsystems.

Flexibility in resource constraints and management reserves also supports innovation and capturing opportunities in a project. On the one hand, resource flexibility requires the openness of the company management towards changes and design opportunities that increases the project manager's decision scope in a certain project. On the other hand, greater investments in the design of both product and process architectures foster SD flexibility [MacCormack *et al.* 2001]. Increased planning effort and the application of improved process modeling methods delivering flexible processes support the allocation of adequate management reserves to the project. Hence, a flexible SD process with realistic budget and schedule constraints considering design iterations required to achieve breakthrough SD results is fundamental to cope with changes and support innovative design work.

The next section reviews SD facets that support the handling of uncertainty and thus foster the effective reduction of risk and exploitation of opportunities in the SD. These facets comprise iteration and learning, experimentation, V&V and frontloading, technology development, concurrent engineering and overlapping, and finally SD lifecycle models.

F.5. TECHNIQUES TO DEAL WITH UNCERTAINTY

F.5.1. Iteration and Learning in SD

SD is a problem-solving process [von Hippel 1990] *and as such, it is fundamentally iterative in nature* [Yassine & Braha 2003]. During the iterative SD process, the developers collect customers' desires, specify a design problem, generate ideas and concepts to solve it, and analyze and

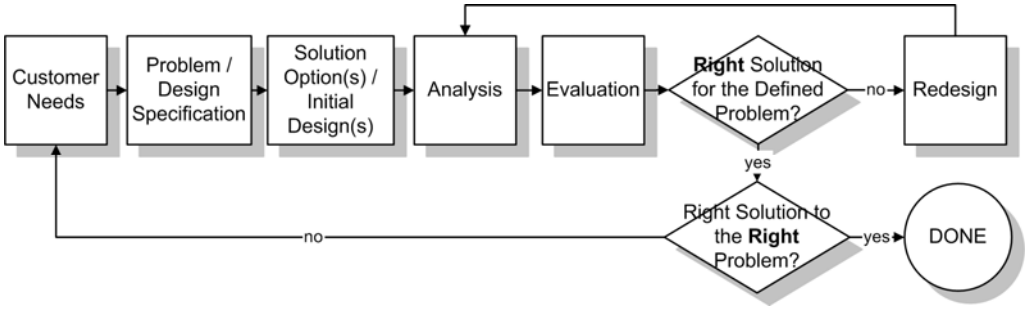


Figure F.12 Iterative SD process

evaluate these ideas against the defined problem and the customer's desires to determine the quality and value of the solutions. If the quality is inadequate (*i.e.,* they did not solve the *problem right*), they redesign, *i.e.,* generate new ideas and concepts to improve the solution. In case the value of the solution is too low (*i.e.,* they did not solve the *right problem*), they collect feedback from the customers, redefine the problem, generate new solutions, *etc.* [Thurston & Locascio 1994]. (Note that the basic framework of iterative design is similar to the systems engineering procedure in *Figure B.3*).

Ford & Sterman [1998] define four basic activities that describe the fundamental elements of SD in their systems dynamics model: initial completion, quality assurance, iteration, and coordination. This model includes similar elements for the iterative design process in *Figure F.12*. *Initial completion* is the finishing of a design activity that provides an initial design. *Quality assurance* means the analysis and inspection of this initial design for defects. That is, quality assurance evaluates the design to decide if the right solution was found. *Iteration* means work on previous activities to improve the initial design and correct the found defects (*i.e.,* redesign). Finally, *coordination* is the integration of the SD project among phases, *e.g..,* when designers work with marketers to refine specifications (*i.e.,* are we looking for the solution to the right problem?).

During this process, the developers acquire information on the design problem, its description, and the possible solutions. Thus, at the end of such a problem-solving cycle, their knowledge and understanding of the problem is higher than in the beginning. That is, the developers *learn* during the process of SD.

Learning is fundamental in the SD, because the knowledge of the developers about the exact outcomes of their work is inadequate at the outset of the SD project. Hence, learning is a basic means to resolve uncertainty and increase the confidence in the characteristics of the final product. While each SD activity improves the product design, they also contribute to the knowledge of the SD organization with regard to the SD problem. Thus, with the completion of every activity, not just the design, but also the design knowledge of the development team evolves. The effectiveness of this knowledge evolution and thus organizational learning defines the final value of a project.

Learning depends on feedback. Developers make decisions to change the "real world", implement these decisions, and gather feedback on the effects of these decisions in the "real world". Using this feedback, they revise their understanding about the "real world" and the decisions they make

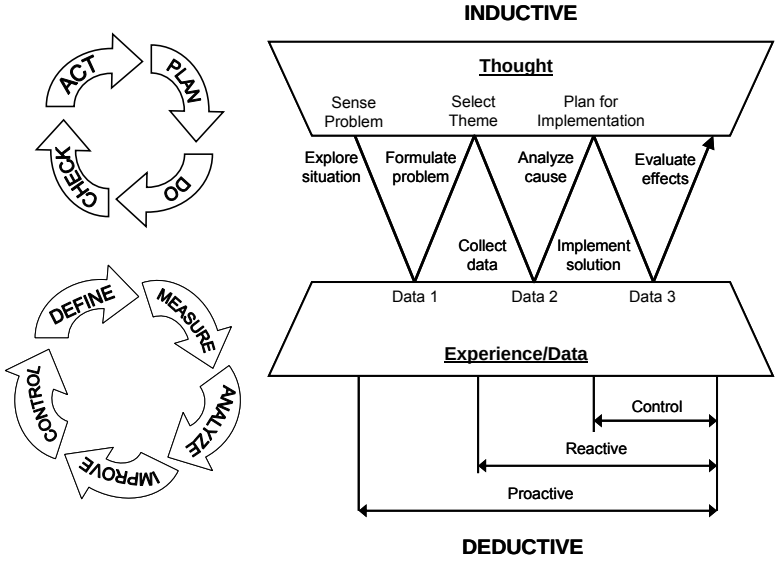


Figure F.13 Iterative quality improvement processes

to bring their perception of the state of the system closer to its goals [Sterman 2000].

Learning as an explicit, iterative feedback process is also the basis of quality and process improvement philosophies. The *PDCA* (plan-do-check-act) or *"Shewhart Cycle"*, the *WV* (or *zigzag*) framework and the *DMAIC* (define-measure-analyze-improve-control) cycles are analogous methods to capture a generic framework for the improvement of a process or system.

The PDCA process shown in *Figure F.13* depicts the basic structure for quality and process improvement in the Total Quality Management (TQM) philosophy. PDCA has four key steps: (1) planning an improvement, (2) making the improvement, (3) checking the improvement results, and (4) acting and replanning if necessary. PDCA is considered a cycle because successive rotations (analogous to iterations) make progress possible. Similarly, the WV framework, which is named after the zigzag pattern that models it, alternates between thought and data as it moves forward to solve a problem [Shiba et. al. 1993, Kleim & Ludin 1997].

DMAIC is also an iterative technique for incremental process improvement using the Six Sigma methodology. The five steps of DMAIC comprise: (1) *define* the customers, their Critical to Quality (CTQ) issues, and the Core Business Process involved; (2) *measure* the performance of the Core Business Process involved; (3) *analyze* the data collected and process map to determine root causes of defects and opportunities for improvement; (4) *improve* the target process by designing creative solutions to fix and prevent problems; and (5) *control* the improvements to keep the process on the new course [I-Six Sigma Online Website].

An analogous model, the iterative *experimentation cycle,* was developed by Thomke [1998, 2003] to encapsulate the fundamental characteristics of the relationship between design and test activities during design iteration. Similar to the other iterative cycles, this model can also applied at various hierarchy levels of the project, *i.e.,* iterations can have a small scope involving only a few activities, but also major SD stages or even the whole project can be considered as an iterative cycle. In the latter case, SD projects are viewed as experiments, where the goal is to capture design opportunities and develop high value products.

Traditional iterative design and quality loops are quite effective means to reduce foreseen uncertainty and variation in the design, and thus improve the *quality* of the system (*i.e.,* increase the degree of the fulfillment of requirements). However, in the dynamically complex SD environment, high quality and high value are often different. Thus, it can happen that the final outcome of the SD fully satisfies the requirements defined in the beginning of the project, but the user's preferences have crept during the project affecting the final value of the product. That is, the real market situation differs from the estimated situation. This can be prevented by extending the learning loop by including external feedback from the SD environment that enables the continuous valuation of the design.

Sterman [2000] proposed a systems dynamics model for idealized learning shown in *Figure F.14* based on the theory of *double-loop learning* [Argyris 1977, Argyris & Schön 1978]. Hence, the idealized learning process includes two major feedback loops that create a relation, one loop with the *real world* and the other with the *virtual world.* On the one hand, the feedback on our previous decision from both "worlds" lead to new decisions that are affected by the mental models of the decision-makers and the organizational culture. On the other hand, in a dynamic, complex world, where the developers' knowledge about certain aspects of the design is missing or ambiguous, new information generated in the feedback loops also affects the developers' mental models about the reality (*i.e., we change the system, the system changes us*). Thus, as the feedback information from the world changes the developers' mental models, they change the architecture of the systems and create different decision rules or strategies.

The model of double loop learning provides the basic idea of the *learning or adaptive organization*. That is, the SD organization has to collect feedback continuously from its internal and external environment, and react to this feedback by adapting itself to the changing needs [Dove 2001]. This ability of an organization to respond to the changes is the main driver of adaptiveness and agility.

The double feedback structure in *Figure F.14* includes an important element that enhances learning effectiveness: the virtual world. The elements of the virtual world are abstractions or models of the real world. As the developers' knowledge of the final product and the future environment of the SD is limited, they have to make assumptions about the unknown aspects during design work, and test these assumptions to see if they were right. Modeling helps this learning process by providing an environment for controlled experimentation. That is, in the virtual world, a limited amount of design aspects can be analyzed and proven in a controlled process, under realistic conditions. This way, uncertainty can be systematically reduced by continuously increasing the number of design aspects (*i.e.,* experimentation scope) analyzed during the iterative experimentation cycles.

According to organizational theory, learning occurs at different levels, at multiple levels of abstraction [*e.g.,* Argyris 1977, McKee 1992]. MacCormack *et al.* [2001] differentiate also between detail-level or *context-specific learning* and system-level or *generational learning*. Whereas, context-specific learning results in knowledge that can be directly applied for the solution of specific design problems within the context of a project, generational learning comes from a longer process involving multiple projects and results in documented lessons learned and experienced by

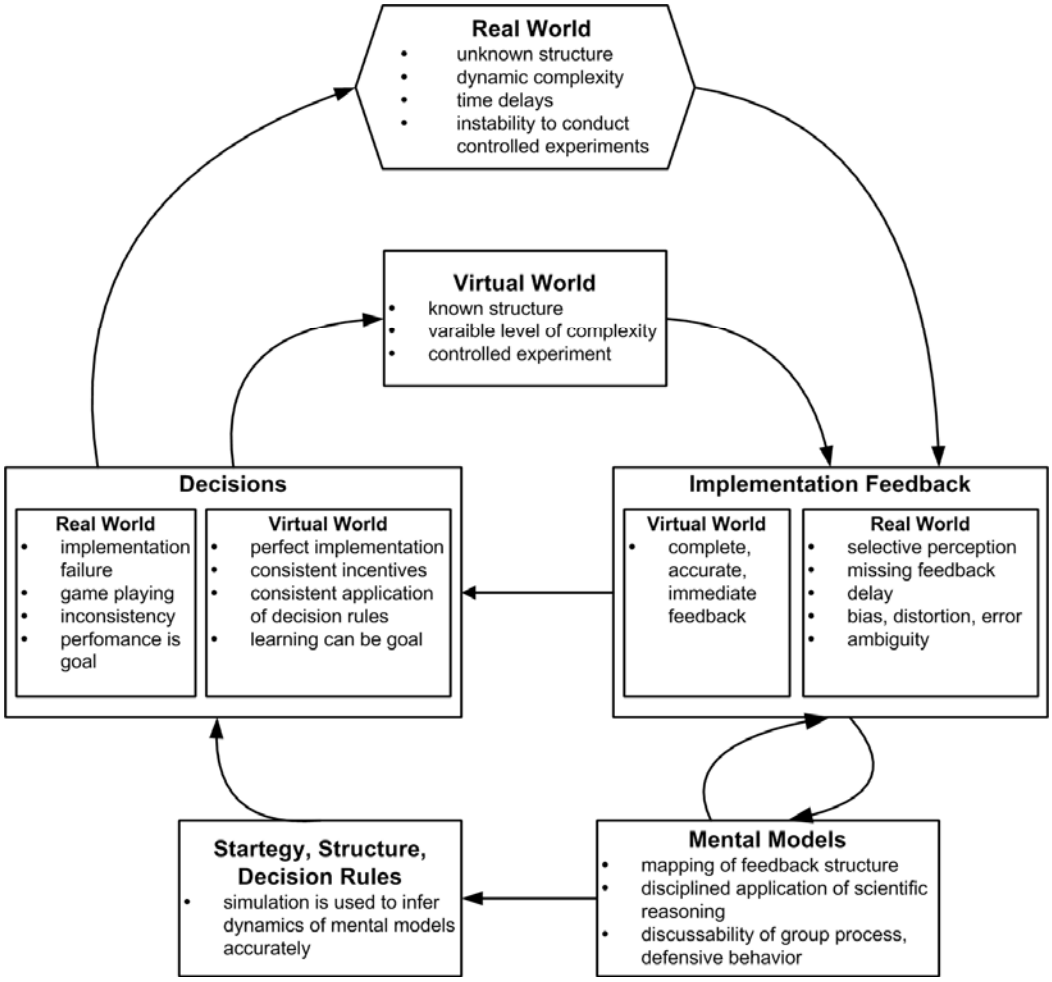


Figure F.14 Idealized learning process (adapted from [Sterman 2000])

developers.

Both system-level and detail-level learning are essential to project success in uncertain and dynamic environments [MacCormack *et al.* 2001]. Thus, the SD project must provide an ideal environment for iterative problem solving and experimentation at all levels of the organization. How structured experimentation contributes to learning will be discussed in the next section.

F.5.2. Experimentation and V&V

Learning involves the *detection and correction of error* [Argyris & Schön 1978]. Hence, experimentation cycles following a *trial and error* working scheme are excellent means for iterative learning in SD. *Figure F.15* depicts the experimentation cycle proposed by Thomke [1998, 2003], where iterative design and evaluation steps form a closed feedback loop of learning. In such loops, ideas are tested against requirements and customers' needs to find better designs and correct design errors. Thus, experimentation cycles are places where context-specific learning occurs in the projects. Due to the scalable nature of the experimentation cycle model, projects can also considered as experiments when major opportunities are captured and evaluated iteratively, using existing knowledge and experience from previous projects. In this sense, experimentation cycles also support generational learning.

Thomke [1998, 2003] argues that experiments during the design of a new product can be conducted in different modes, and the main management goal is to find the optimal switching points between different modes. When it comes to the decision on switching between two experimentation modes (*e.g.,* from computer simulation to rapid prototyping), the effectiveness of further experimentation in a certain mode is compared to the programmatic aspects of the experiment. That is, after each experimentation cycle the following questions are raised: (1) Did the experiment deliver the required outputs in terms of expected level of knowledge / design maturity? (2) Does the design have the required level of maturity to effectively perform the next planned experiment mode? (3) Are the required resources available to perform another experimentation cycle in the current mode?

Modes are experiments with similar purposes, but different fidelity (*i.e.,* number of design aspects considered), cost and duration. As the design evolves during the SD process, the fidelity of the consecutive experiments grows. Hence, the level of abstraction of the *virtual world* (*i.e.,* the network of controlled experiments) decreases during SD, and the overlap and similarity between the virtual and real worlds increases. That is, the design aspects included both in the models (or prototypes) and the simulated virtual world (*i.e.,* the fidelity of the experiments) increase towards the end of the SD, result in the implementation of the final experiments in the real world, and thus qualify and certify the feasibility of the design.

A deliberate experimentation or prototyping strategy is vital for continuous uncertainty and risk reduction and thus learning in the SD. Furthermore, experimentation cycles foster the systematic generation and testing of innovative new ideas, and thus enable companies to create and refine their products and effectively capture design opportunities [Thomke 2001]. In this experimentation system, different modes of experiments with growing fidelity are organized

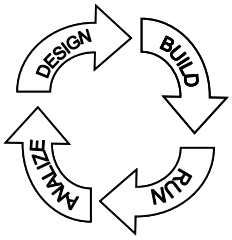


Figure F.15 Experimentation cycle (adapted from [Thomke 2003])

sequentially or parallel to *maximize the effectiveness of learning* in the SD.

The consideration of the SD process as a network of iteration cycles that together cover every important aspect of the SD is an important element of adaptive SD projects. The reason for this is that experimentation cycles are sub-processes of the SD process with a distinct purpose, *i.e.,* a clear network of activities with measurable deliverables. The project benefits from this, because partitioning a process into fairly independent tasks or sub-processes reduces process duration [von Hippel 1990]. Furthermore, experimentation cycles transform the rather sequential traditional SD projects with few decision and control points at the major milestones or stage gates into a modular network of problem-solving cycles with controllable, interrelated deliverables. So, considering SD as a chain of experiments helps the planners break down the project goals into clear sub-goals that can be effectively controlled during the project.

This phenomenon is depicted in *Figure F.16*, where the difference between traditional end-of-the-phase testing is compared to planning for fast iteration cycles. The second process in *Figure F.16* is the result of DSM partitioning conducted during the first process. Hence, both experimentation strategies in *Figure F.16* include the same number and type of activities; however, the structures of the processes are different. While the first strategy includes long delays between design and testing that leads to ambiguity in the SD process [Sterman 2000], the second strategy ensures fast feedback to the developers on the results of their work and prevents unnecessary work on erroneous designs. Thus, the second strategy helps the design team find failures right after they have been included in the design.

In traditional SD, the project strategy is often to develop high quality products *right at the first time*. In such an SD project, the complete design is finished first and then evaluated through expensive system tests. This SD philosophy does not account for the iterative nature of technical work and proposes to increase the precision of SD work and strive to achieve *zero defects* [Taguchi & Clausing 1990]. While the philosophy of zero defects or error-free work is applicable for production processes, *SD cannot be done right at the first time*, because sequential work reduces the effectiveness of learning. Furthermore, a fundamental problem in sequential SD is that design failures are usually found too late in the project when changes have high consequences on the project schedule and budget.

On the contrary, evolutionary SD strategies realize the importance and effect of iterative learning from innovation and account for iteration during the planning of the SD process. The premise of evolutionary or experimentation-driven SD is that the effort of reducing risk of the first failure is much higher than the cost of correction [Thomke 2003]. Hence, in such projects, the goal is not to eliminate these valuable iteration cycles, but to reduce their scope to the required minimum based on the process logic and information flow, and thus support effective and efficient experimentation [*e.g.,* Denker *et al.* 2001, Browning 2003].

As *Figure F.16* depicts, experimentation and iteration cycles usually comprise a set of

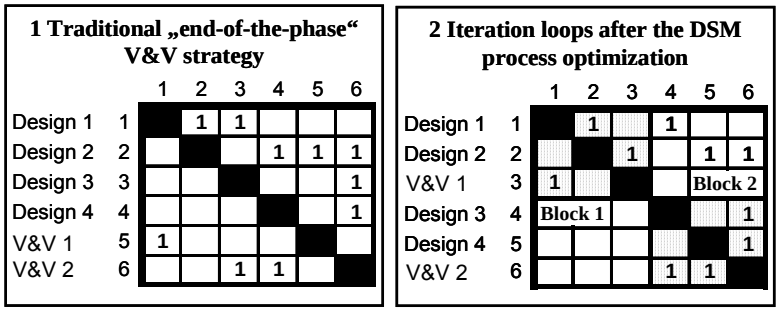


Figure F.16 Traditional and modular process design for experimentation

interrelated design and V&V activities. The role of design activities in experimentation is to generate innovative and creative ideas to solve the problem defined by the scope of the experimentation. The V&V activities in the experimentation cycle must be then appropriate to effectively analyze the solution for the defined problem provided in the form of the deliverables of the design activity and provide valuable information on the feasibility, quality, and value of the design solution. Furthermore, the information delivered by the V&V activities is the main source of information for the decisions after the experimentation cycles. As a consequence, systematic experimentation does not only reduce uncertainty by increasing learning effectiveness, but it also does it by reducing SD complexity and fostering enhanced project control.

Furthermore, experimentation cycles can be considered as fairly independent process modules of coupled design and V&V activities. Depending on the modularity of the product design, these process modules are more or less independent of each other. In case of high independence, experiments can be conducted in parallel with the modules of the system, which reduces process duration significantly.

F.5.3. Frontloading of Experiments

Loch *et al.* [2001] propose that there are three basic factors that influence V&V strategies: the *cost* and *duration* (or feedback time) of the V&V activities and the *learning* between the activities. Obviously, the goal of project planning is to define a V&V strategy that maximizes the effectiveness of learning and minimizes feedback time and cost. While V&V and design activities are closely connected in the SD process, V&V strategy planning can be only effective, if it is integrated with project planning.

In this thesis, V&V strategy planning is considered as one distinct step of the integrated systems engineering planning effort conducted at the outset of the project. The goal of systems engineering planning is to provide the project with a structure that enables continuous innovation and learning and the timely validation and verification of the requirements and products of the SD. Further, systems engineering has to guarantee the continuous evolution of the design by optimizing the information flow among SD activities and reducing the probability of undesired rework in the SD process.

Experimentation and V&V are effective problem-solving methods that are inevitable in every SD project. As the previous section showed, V&V and testing are most effective if scheduled right after the relevant design activities, because the *delay* between the output and information on the feasibility of the output (*i.e.,* feedback on the result of the design work) can be considerably reduced this way. The programmatic benefit of organizing closely related design and V&V

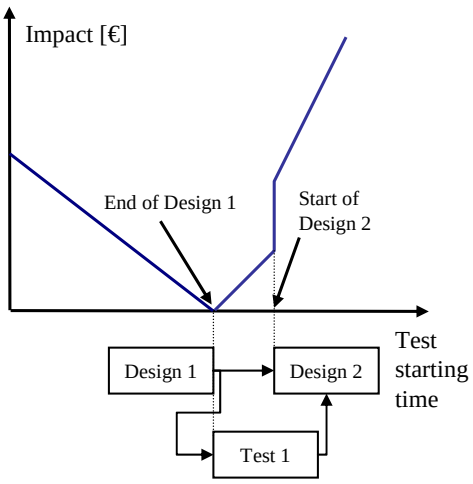


Figure F.17 Stylized impact of a separate test starting time

activities in experimentation cycles is depicted in *Figure F.17* showing the impact of V&V activity starting time on the project budget. The best time to start a V&V activity is immediately after the output of the design activity is delivered. If V&V starts at any different point in the SD, it can have financial consequences.

If V&V is done too early, the deliverables to be evaluated are not mature enough, and the V&V results will be poor. This causes the repetition of the V&V effort and thus higher project cost. On the other hand, late V&V provides late feedback on the design, which reduces the effectiveness of learning and innovation in the SD, and increases rework probability.

Another system-level aspect of V&V strategy planning besides the reduction of feedback delay between design and V&V activities is that the V&V strategy has to foster the effectiveness of problem solving by providing the developers with the right answers in a timely manner. As mentioned before, in conventional SD projects the work is done in consecutive stages, and the main testing effort is done late in the project, mainly on the integrated physical prototype of the system. For example Boehm [1981] showed in his study of several large software projects that the relative cost of correcting software errors increases significantly as a function of the phase in which the corrections or changes were made. This phenomenon is often called the *rule-of-ten* referring to the observation that the cost of solving a design problem or making a design change increases by the factor of ten with each elapsed phase after the "inclusion" of the failure.

Many researchers realized that the erroneous experimentation philosophy of the waterfall SD model is a main cause for major rework efforts in the projects that drive cost and schedule risk; and proposed to reengineer SD process structures and *frontload* problem-solving activities (V&V and testing) in the SD [*e.g.,* Boehm 1981, Clark & Fujimoto 1991, Assmann 1998, Fricke *et al.* 2000, Thomke & Fujimoto 2000, *etc.*]. Thomke & Fujimoto [2000] define *frontloading* as

> *a strategy that seeks to improve development performance by shifting the identification and solving of [design] problems to earlier phases of a product development process.*

Frontloading strategies attempt to improve conventional SD by proposing the application of innovative technologies to reduce uncertainty early in the SD project. Novel experimentation technologies such as computer modeling, virtual prototyping, and digital mock-ups (DMU), computer simulation using the finite element method (FEM), rapid prototyping, *etc.,* enable the cheap design and fabrication of prototypes early in the SD. Using these technologies, low-cost, iterative trial-and-error learning can be conducted in the early SD stages, where uncertainty is still rather high.

Hence, virtual experimentation fosters system evaluation and knowledge generation before key decisions on the system design are made, and it reduces technical uncertainty concerning the critical design aspects. With the enabling technologies of frontloading, the development of a complex engineering system receives a new dimension. That is, the architecture and behavior of the developed system can be completely designed and evaluated virtually and thus, most of the major design failures can be captured and corrected before having built a single piece of hardware.

As a consequence, frontloading triggers rapid innovation and effective failure detection by offering a novel experimentation environment (*i.e.,* the virtual world), where prototypes can be generated quickly and at a low price. Further, while virtual prototypes do not have any material cost, they can be redesigned and evaluated rapidly until a feasible, mature design that guarantees maximal customer satisfaction has been found.

The achievement of high design maturity and performance early in the SD results in better SD performance [Thomke 1998], and reduces SD time and cost and thus frees up resources to be more innovative in the marketplace [Thomke & Fujimoto 2000]. Thus, the frontloading of experimentation increases both the effectiveness and efficiency of the SD process and thus the total value of the SD endeavor.

The next section discusses concurrent engineering, a project management technique to reduce SD cycle time.

F.5.4. Concurrent Engineering

A well-known technique for the compression of cycle time and thus for the reduction of schedule uncertainty is the application of concurrent engineering [Imai *et al.* 1985, Takeuchi & Nonaka 1986, Clark & Fujimoto 1991]. As the companies in different industry areas realized in the 1990s that time-to-market is a key source of competitive advantage [*e.g.,* Rosenau 1990, Blackburn 1991, Wheelwright & Clark 1992, Cusumano & Selby 1995, Sabbagh 1996], the search for methods started that help effectively shrink project duration. The date of market launch is particularly important in dynamic markets with highly innovative products. Vesey [1991] showed in his study on high-technology SD projects that products entering the market six months later than their competitors, but otherwise within budget, earned 33% less in a five-year period than they would have if they had entered the market on time. On the other hand, similar companies having entered the market on time, but 50% over budget reduced the firm's profitability with regard to that product by only 4% [Eisenhardt & Tabrizi 1995].

Hence, low process duration is not just an important factor of SD, but in many industries the source of long-term market success and profitability. Browning & Eppinger [2002] remind us that process effectiveness (*i.e.,* high value products) and efficiency (*i.e.,* low process duration and budget) greatly depends on the process architecture. Industry studies also show that companies

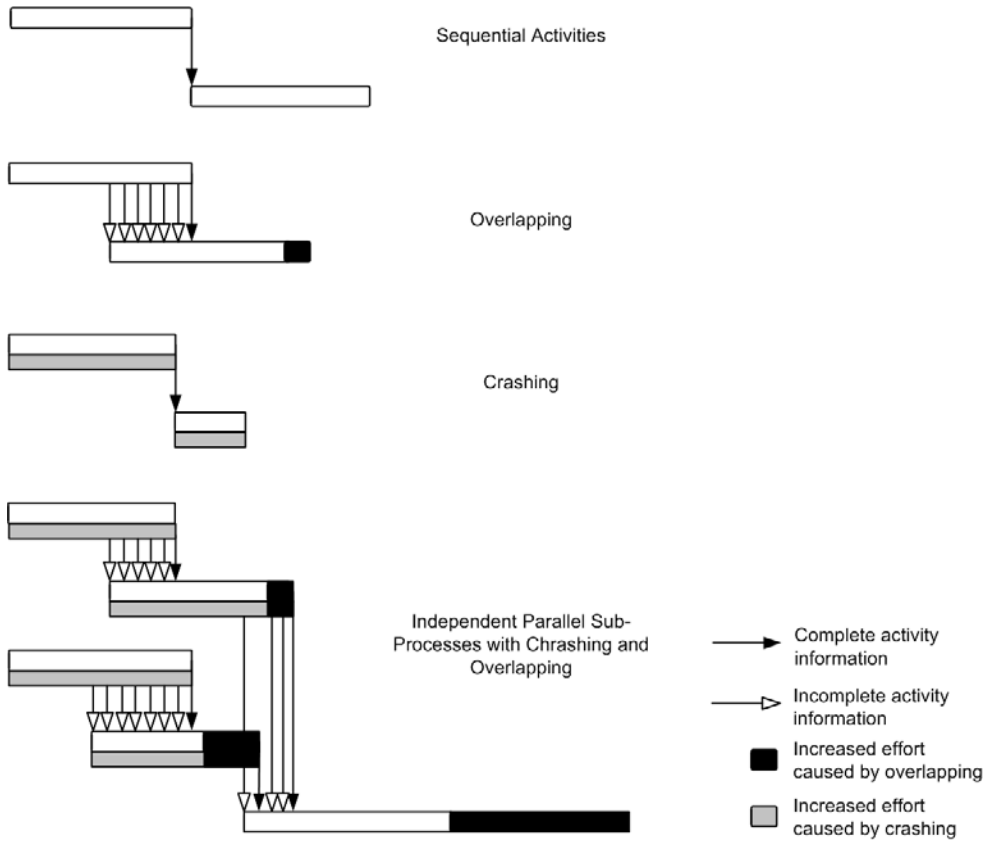


Figure F.18 Sequential, overlapping and concurrent development activities

with SD processes with parallel or overlapping activities or even sub-processes achieve great reductions in process duration [Takeuchi & Nonaka 1986, Clark & Fujimoto 1991].

There are three basic techniques for cycle time reduction through concurrent engineering in SD. These three techniques (overlapping, crashing, and parallel execution of independent sub-processes) are compared to the traditional, strictly sequential activity execution in *Figure F.18*, where four different processes are depicted using Gantt-charts. The bars represent the working effort required to finish a certain task, the full arrows deliver complete and the empty ones incomplete deliverables. While the black, full parts of the bars represent the additional activity durations due to activity overlapping, the grey bars show required additional working effort due to the compression of activity duration.

The principle of *sequential activity execution* depicted in the first chart is that each step of problem solving starts only if the logically previous step has finished its job and delivered the adequate information. The philosophy for strictly sequential processes is that the completion of one activity unequivocally means qualitatively complete deliverables. Thus, all activities wait until the predecessor activity is finished, because it means perfect information for the developers. However, this assumption is rather idealistic. In reality, fully complete information rarely arrives at the activities, and thus the developers have to start their work with assumptions, anyway.

In case the outcome of SD is not an entirely novel product, where every design aspect is completely new (and usually this is the case), developers can use existing knowledge from previous projects (generational learning), estimate parts of missing information, and start activities with an incomplete set of input products. That is, they start working with incomplete information and make adjustments on their work when the complete set of data arrives from the finished predecessors. If the additional working effort these adjustments require is smaller than the time saved through *activity overlapping*, than the project can benefit from reduced time-to-market through overlapped activity execution [Krishnan *et al.* 1997].

The benefits achievable through overlapping are illustrated in *Figure F.19* in case of two activities. The total cycle time reduction is the function of the degree of overlapping. On the one hand, the higher the degree of overlapping, the lower the maturity of information provided by the upstream activity and thus the higher also the caused (direct and indirect) additional SD work (note that *Figure F.19* considers only two activities and thus only additional direct SD work). On the other hand, Krishnan *et al.* [1997] point out that a too low degree of overlapping might also cause rework due to reduced process flexibility. The cause of this quality loss is that upstream activities finalize their output information too early and thereby limit the scope of possible changes by the downstream activity, which can also lead to rework. Hence, during project planning, the correct overlapping ratio has to be determined; otherwise the project suffers instead of gaining from concurrent engineering.

The third diagram in *Figure F.18* depicts *crashing*, a different strategy for time compression [Rosenau 1988, Stalk & Hout 1990, Cordero 1991, Roemer & Ahmadi 2004]. The underlying assumption in this strategy is that all required steps of an SD project are known *a priori*, and SD projects can be accelerated by simply cutting the original activity durations [Eisenhardt & Tabrizi 1995]. Crashing can even be motivating if applied in a well-understood SD project, and if it comes together with appropriate rewards for faster work.

The fourth chart in *Figure F.18* presents two parallel, crashed sub-processes with overlapped activities. Since the two sub-processes do not have any information dependencies, they can be conducted concurrently in the project. The fifth and lowest activity in the diagram depends on both sub-processes and integrates the deliverables of the concurrent process parts. The black, full part of the lowest integration activity shows that overlapping in this case might cause risk in

the project. Thus, if the activity starts working, and does not have sufficient information, the inputs coming irregularly from the predecessors might lead to undesired additional work, even beyond the reduction in process duration gained through overlapping.

Cycle time reduction through concurrent development comes at the cost of increased process and organizational complexity [*e.g.,* Clark & Fujimoto 1989, Wheelwright & Clark 1992, Krishnan *et al.* 1995]. Additionally, concurrent methods often increase the frequency and number of information transfers between project phases [Clark & Fujimoto 1989, 1991]. More tasks begin with incomplete or preliminary information, boosting the number of iterations and thus leading to oscillations in the process [Yassine & Braha 2003]. Hence, if concurrent engineering is not planned and handled appropriately, it can lead to chaos and high programmatic risk in the project. The next part discusses planning aspects of concurrent engineering in SD projects.

F.5.4.1. Characteristics of Concurrent Engineering

The cost of concurrent engineering is additional design iteration or rework [*e.g.,* Ford & Sterman 2003]. Rework means iterative refinement of activities to account for changes in their inputs [Browning 1998a]. These changes can stem from new (additional or changed) information and/or failure to meet design objectives [Smith & Eppinger 1997b]. The causes of rework can be threefold [Browning 1998a]: (1) changes in upstream (previously worked) activity outputs as a result of external changes (*e.g.,* changes in customer's needs); (2) concurrent activities changing shared assumptions; or (3) changes in downstream activity outputs due to failures and incompatibilities discovered during V&V.

All three kinds of iterations can appear in an SD process intentionally (*i.e.,* planned iterations) or unintentionally (*i.e.,* unplanned iterations) [*e.g.,* Browning 1998a, Clausing 1994]. A main objective of project management is to reduce and even *eliminate unintentional iteration* in the SD process, because it is a main source of ambiguity and thus project risk. In this section, the characteristics of SD environments are reviewed concerning the applicability of concurrent engineering.

The application of concurrent engineering involves a *calculated risk* [Smith & Reinertsen 1998], *i.e.,* the additional amount of rework due to working with incomplete information. However, the amount of rework generated through different techniques of concurrent engineering differs on the basis of the characteristics of the SD project. Both crashing and overlapping of activities are effective in principle with mature and stable environments [Cordero 1991, Eisenhardt & Tabrizi 1995], since they require the thorough understanding of the SD process and the presence of low

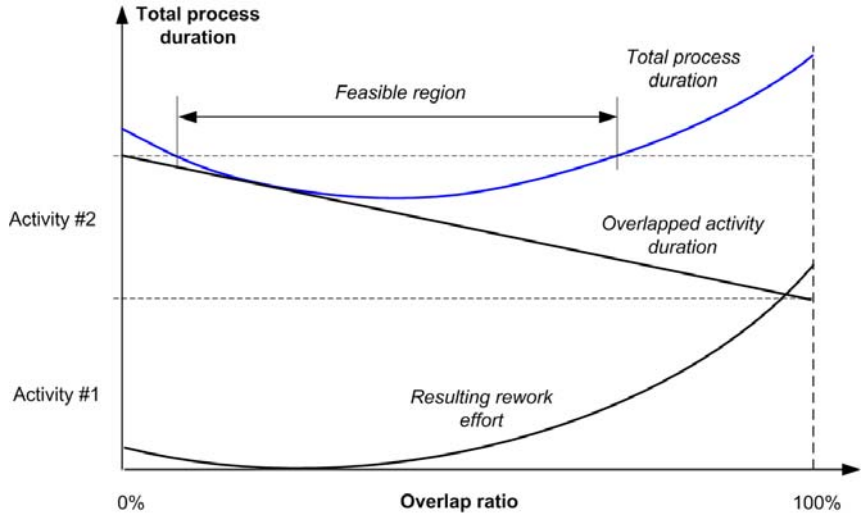


Figure F.19 Stylized function of achievable benefits through activity overlapping

uncertainty in the SD environment. That is, in projects dealing with product upgrades or the development of designs with a low degree of innovation, the developers work in a familiar SD environment and thus they can complete their tasks faster or even in parallel. However, for highly dynamic SD environments, crashing and overlapping are rather ineffective causing a high amount of rework and chaos in the project [Eisenhardt 1989].

The effectiveness of overlapping is also affected by other SD characteristics. First, as overlapping causes iterations, it is ineffective for highly complex processes due to the effect of change propagation in the process. That is, if many interrelated activities overlap, the scope of the resulting iterations might cause chaos in the project. Hence, modular product and process architecture supports overlapping on the process-level by reducing process complexity and thus the scope of propagating iterations. While modules in a modular design are relatively independent, they can be designed and developed independently, enabling parallel SD work [Baldwin & Clark 2000] and an increased degree of parallel testing [Loch *et al.* 2001]. Thus, work on the modules can be conducted with overlaps or even fully in parallel without affecting the work on the other modules as long as the system-level design rules and requirements are obeyed.

Second, overlapping on the activity-level (*i.e.,* the work within the modules) can be done if it yields benefits for the project. As already discussed, a tradeoff between overlapping and the resulting rework has to be made during planning (*Figure F.19*). The manageable overlap ratio depends on many aspects, *e.g.,* SD process uncertainty, the type of exchanged information [Krishnan *et al.* 1997, Terwiesch *et al.* 2002], upstream uncertainty resolution [Terwiesch & Loch 1999], or evolution and downstream iteration sensitivity [Krishnan *et al.* 1995, 1997].

The type of exchanged information between overlapped activities is the first factor contributing to the effectiveness of overlapping. An activity can have various input needs, *e.g.,* customer's needs, component specifications, virtual or physical prototypes, simulation or test results, *etc.* Parts of this information can be exchanged in a preliminary form (*e.g..,* as assumptions) and another part of it cannot. The task of project managers here is twofold. First, they have to determine how the outputs of the activity can be decomposed into information packages. Then, they have to find out which packages of information can be estimated with high certainty and thus transferred preliminarily.

There is always a degree uncertainty *a priori* about the exact characteristics of the activity outcomes. Foreseen uncertainty can be handled by estimating the expected value of the PDF of the possible outcomes. However, in the presence of ambiguity (*i.e.,* when the activity outcomes cannot be estimated), overlapping is dangerous. Thus, in highly dynamic, ambiguous SD environments, overlapping is not recommended [Cordero 1991, Eisenhardt & Brown 1998]. In such projects, effective techniques for time compression are fast iteration, short phases, and multiple milestones.

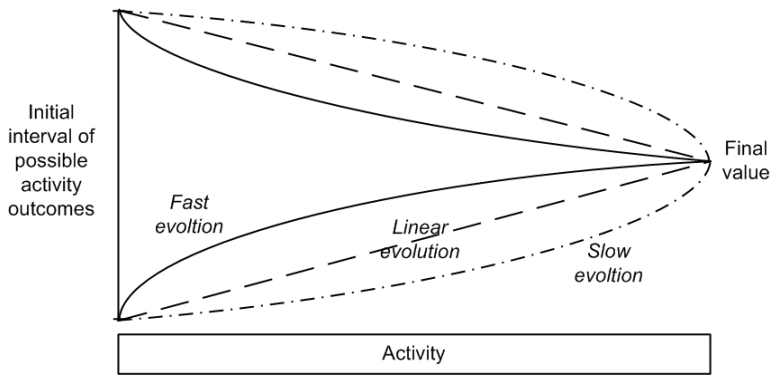


Figure F.20 Evolution of the final outcome of an upstream activity (adapted from [Krishnan et al. 1997])

Another factor that affects overlapping effectiveness, besides the determination of the preliminary information packages to be exchanged, is the amount of iteration and rework caused by certain preliminary information packages. Two aspects contribute to the amount of iteration caused: *upstream information evolution* and *downstream iteration sensitivity* [Krishnan *et al.* 1997]. *Evolution* is defined as the speed at which the interval converges to a final upstream solution. *Downstream sensitivity* is defined as the duration of a downstream iteration to incorporate upstream changes associated with the narrowing of the interval [Loch & Terwiesch 1998].

The information evolution of the upstream activity depends on the characteristics of the activity referring to the way uncertainty concerning the exact outcomes is resolved during the execution of the activity. *Figure F.20* depicts three types of information evolution: *fast*, *linear,* and *slow evolution*. While for fast evolution fairly certain information is obtained early in the activity, for linear and slow evolution uncertainty is resolved rather late. Hence, for fast information evolution it is possible to define a high overlapping ratio. On the contrary, the upstream activity outcomes for linear or slow information evolution remain uncertain and even ambiguous long after the beginning of the activity.

Downstream sensitivity means the sensitivity of the downstream activity to changes in its input. Downstream sensitivity is low for computer-based design and evaluation activities in the first part of the SD, because the effort of making slight changes in computer models is rather low. However, later in the SD, during the implementation of the design in hardware components, changes get more expensive. During this part of the SD, the use of existing SD experience (*e.g.,* existing designs and lessons learned) is helpful to resolve uncertainty and reduce the probability of major changes during overlapping.

F.5.4.2. Planning Aspects of Concurrent Engineering

With the above-described aspects of activity overlapping and concurrent engineering, planning aspects for effective concurrent processes are identified in this part. As mentioned before, the logic and structure of the SD process is a key aspect of overlapping effectiveness. While product and process modularity supports information hiding in the system, it increases the independence of SD sub-processes working on different modules. Thus, in modular SD processes, concurrency can be implemented on various levels of the process architecture (*e.g.,* system-level, module-level, component level, *etc.*). That is, due to the lower level of interaction between sub-processes, module-level SD processes can be accomplished independently and thus overlapped at regular information exchange among module developer teams.

If the SD work is done in various hierarchy levels in the project, the exchange of information among developers has to be also organized in various levels. While communication is a key aspect of concurrent engineering [Clark & Fujimoto 1991, Clark & Wheelwright 1993], the task of systems engineering is to facilitate effective information exchange among team members by good project organization. That is, the task of systems engineering is to define a hierarchic communication network of developers, organize them in teams or clusters, and define appropriate communication channels. Due to the hierarchic nature of SD projects, each member of the development team can be assigned to various development groups relevant to their work areas on various hierarchy levels.

Information exchange among developers in the various levels of the SD project is done using different channels of communication. While the closest colleagues in a design group work together every day, separate groups conducting overlapped design work might only meet weekly, and developers involved in design review teams might meet irregularly only for the major milestones. That is, the basic rule for communication is to define a manageable communication scope and frequency for each member of the hierarchical SD organization to support effective

SD work. The organization and interdependence of hierarchical milestone meetings in an SD project is depicted in *Figure F.21*. Meetings at the lowest level might be held weekly, and higher-level meetings only after major experimentation cycles or lifecycle phases of the project, where the achievements of the work done is compared to the planned performance to decide on the success of the project.

Meetings among team members conducting overlapped design work can be supported by state-of-the-art multimedia applications enabling *e.g.,* effective communication between regionally separated SD sites through video conferences, real-time integration of the results of overlapped modeling teams, evaluation of the work results, and adaptation of the team tasks to the actual SD status. A successful method for such collaborate meetings and learning exercises is the *design center* approach [Shishko 2000, Wilke *et al.* 2000, Wilke 2003, Finkel *et al.* 2002, Finkel & Burazanis, 2004].

Design centers are collaborative working environments where the members of interdisciplinary design teams (IDPs) responsible for different subsystems of a certain product work together and integrate their subsystem models to evaluate technical and budgetary aspects of the developed product. Such SD environments foster collaborative learning and experimentation by enabling real-time problem solving and decision-making among the team members of different disciplines. Design centers are also effective means to control the early experimentation with different design alternatives through the application of virtual prototypes.

To summarize, concurrent engineering can be highly beneficial for SD projects, however, the project manager has to be aware of the characteristics of the internal and external SD context, and the consequences of concurrent engineering (*i.e.,* increased probability of rework). *First*, the effective application of concurrent engineering requires an increased planning effort and greater investments in architectural design. *Second*, the success of concurrent engineering mainly depends on the product design structure. While for modular designs, a high degree of activity concurrency might still result in low rework, integral design architectures contribute to high activity interdependency on the system level and thus prohibit high overlap ratios. *Third*, for high technical uncertainty, activity overlapping and crashing increases programmatic risk in the SD project. *Fourth*, a highly flexible SD process architecture supports the implementation of concurrent engineering even in uncertain environments. *Fifth*, the utilization of generational project experience fosters the selection of the right concurrent engineering strategy and the decision on the optimal overlap ratio. *Sixth*, the definition of data packages containing incomplete information to be transferred fosters the effectiveness of overlapping and parallel activity execution. *Seventh*, enhanced communication is a basic requirement of concurrent engineering.

On the contrary, for highly dynamic SD environments and extremely innovative project goals,

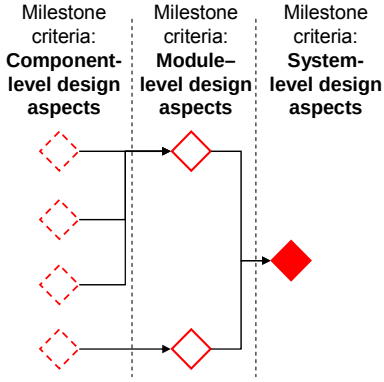


Figure F.21 Organization of milestone meetings for various hierarchy levels in the project

fast iteration cycles [Eisenhardt 1989], frontloading of design evaluation activities [*e.g.,* Fricke *et al.* 2000, Thomke & Fujimoto 2000], experiential SD strategy [Eisenhardt & Tabrizi 1995], and frequent milestones [Gersick 1988, 1994, Weick 1993, Ha & Porteus 1995] are recommended instead of overlapping.

F.5.5. Separation of Technology Development and System Development

Short SD lifecycles ensure early market launch and higher profitability. Furthermore, short lifecycles are also effective against high market uncertainty. That is, in the presence of high uncertainty and ambiguity in the market, fast reaction to the emerging market needs and technologies reduces the risk of releasing products with obsolescent functions. On the contrary, short SD cycles often inherit high technical risk, because the developed products cannot reach the desired technical maturity and reliability within the reduced SD time.

In case multiple core technologies have to be integrated in complex new products, reduced cycle time might drive technical risk and cause increased delays in the project [Meyer & Utterback 1995]. Furthermore, Mansfield *et al.* [1972] show in their study in the pharmaceutical industry that firms concentrating on product performance and quality instead of cycle time can also prosper even with high schedule overruns in the project. Hence, Clark & Fujimoto [1991] argue that simply trying to accelerate cycle time without first achieving simplicity and efficiency can be the road to ruin for a firm. Particularly high technical uncertainty requires managerial attention and the application of effective risk reduction strategies (*e.g.,* high degree of experimentation [Meyer & Utterback 1995], frontloading of design and testing activities [*e.g.,* Urban & Hauser 1980, Thomke & Fujimoto 2000], *etc.*).

An effective method to deal with both high technical and market risk is to separate SD and Technology Development (TD) in the SD system and thus *transfer* a main part of the uncertainty from the SD to the research teams. This reduces the scope of the SD problem to be solved and provides the SD team with mature solutions for the emerging market needs. Another reason for separation is that SD and TD have fundamentally different characteristics. On the one hand, the goal of SD is to introduce a product in a *previously chosen market segment, at a fixed date of introduction* and thus SD is done in a *disciplined style* and has a *tight schedule*. On the contrary, TD requires a *creative environment and sufficient time* because it *seeks to achieve generic improvements available for the entire product program*. Thus, TD projects operate under *highly ambiguous circumstances* and with *uncertain cost and budget constraints* [Clausing 1994, Schulz *et al.* 2000].

Thus, the output of TD is a generic product (*i.e.,* a technology) applicable in various circumstances, not a solution to a specific problem or need of the market as with SD projects. The outputs of TD are flexible and robust technologies, which can be used in various products of the company portfolio [Clausing 1994]. In this sense, SD is a set of problem solving efforts that translate information on technological possibilities and market needs into a set of detailed designs, instructions, and other information assets required for production [Iansiti 1995b]. Hence, technologies are enablers of the cost efficient realization of customer's needs in product functionalities for which the customer is willing to pay money.

The effective integration of TD and SD projects is a key to success [Wheelwright & Clark 1992, Iansiti 1995b, 1995c, Iansiti & West 1997]. It is even more important in adaptive SD projects, where the SD system frequently produces prototypes and product increments, which are then validated directly with the customer. The continuously evolving incremental products demand close interaction between SD and TD groups as well as a careful selection of new technologies that will add value to new products. Additionally, the developed technologies have to be transferred to the SD in a timely manner and be able to be smoothly integrated in existing

technologies [Iansiti 1995b]. Hence, TD objectives have to be defined as part of the corporate strategy and synchronized with the strategic objectives of the product portfolio.

Different authors proposed various frameworks for TD during the last years [*e.g.,* Clark & Fujimoto 1991, Wheelwright & Clark 1992, Clausing 1994, Metz 1996, Stanke & Ulbricht 1997, Zehnder 1997]. One TD approach, the Total Technology Development (TTD) framework proposed by Schulz *et al.* [2000], based on earlier work by Clausing [1994], was developed according to the fundamental principles of systems engineering. The TTD framework is a separate process for technology development with clearly defined interfaces to SD, and embedded into an integrated framework of methods and tools, tailored to the needs of TD providing great benefits in terms of technology superiority, maturity, robustness, and flexibility [Schulz *et al.* 2000]. TTD is a holistic approach attempting to consider every important aspect of TD and to integrate them in a TD framework that provides the SD with superior, flexible technologies at the right time.

The four main steps of TTD are depicted in *Figure F.22*. In the core of the TTD framework is an *integrated technology strategy* that incorporates market-based and technology-based requirements for next generation technologies. Since TD delivers only the technologies that will enable future customer satisfaction in the long term, technology strategy planning has to deal with high uncertainty and ambiguity. Thus, the requirements and guidelines included in the integrated technology strategy delivered by the first step of the TTD framework have to be flexible to accommodate future changes [Lowe 1995].

The TD strategy represents the company's vision concerning the functionalities of the products of the future. Furthermore, the TD strategy determines which technology areas will be supported by the SD system in the next period, and how the system architecture of the future will look like. Flexible product platforms serving several generations of incremental product versions are good reference points for a TD strategy. Platforms and reference architectures define rules for modules and interfaces that promote the steady evolution of the product towards desired strategic goals. Thus, the reference modules of a system support the decomposition of TD tasks into module-relevant sub-tasks and similarly to SD projects; technologies can be also developed independently for the modules of the system. This is particularly important for adaptive SD systems, where high SD and TD complexity often hinders the capability of a system for quick response.

The technology strategy of a company includes a set of strategic directions for the SD system concerning the portfolio of technologies to be developed in the future. These are the main inputs for the next step in TTD, where technology concepts are generated, analyzed, and selected to ensure competitiveness and high company profit in the future. While TD is an especially creative and innovative process, the output of concept generation includes the concepts of technologies that might bring market leadership to the company. However, the technical feasibility of the concepts is also essential for future success. Thus, during concept selection superior technology concepts with a high feasibility are selected, which fit in the strategic direction of the company, can be integrated with current technologies, and can be developed at acceptable cost and within the desired timeframe.

In the next step of TTD, the main goal is to develop the technology and achieve the maturity

Figure F.22 Total Technology Development framework (adapted from [Schulz et al. 2000])

required by SD and the robustness that fosters its implementation for various products in different circumstances. The third step of TTD is the last one done separately from the SD program. The exit criteria for a technology from this step are high maturity and robustness that allows its direct application at various SD projects. Due to the reduced technology lifecycles and the relatively open industrial system, the development of technologies with a low rate of evolution or low robustness is not viable in the long term. Emerging technologies in these areas are more cost-effective to buy from outside the company, which leaves more space for TD in the "core competence" technology areas.

In the last step of TTD, developers and researchers (*i.e.*, TD and SD teams) work together to *select* the most suitable technologies for the SD needs, and to *transfer* them to the SD to *integrate* them with existing technologies. Iansiti [1995b] argues that technology integration relying on a system-focused approach has the highest impact in product and SD performance. In a system-focused approach generational system-level knowledge of experts is applied to decide which technologies bring the highest benefits for the developed product.

The main challenge for technology selection and integration is to decide which of the technologies that properly work under laboratory conditions will also work as part of a product. Furthermore, the question is how novel and existing technologies can be integrated to achieve a synergy that increases the value of the product. As *Figure F.23* depicts, the decision on technology transfer can be integrated in design reviews and milestone meetings, where the actual

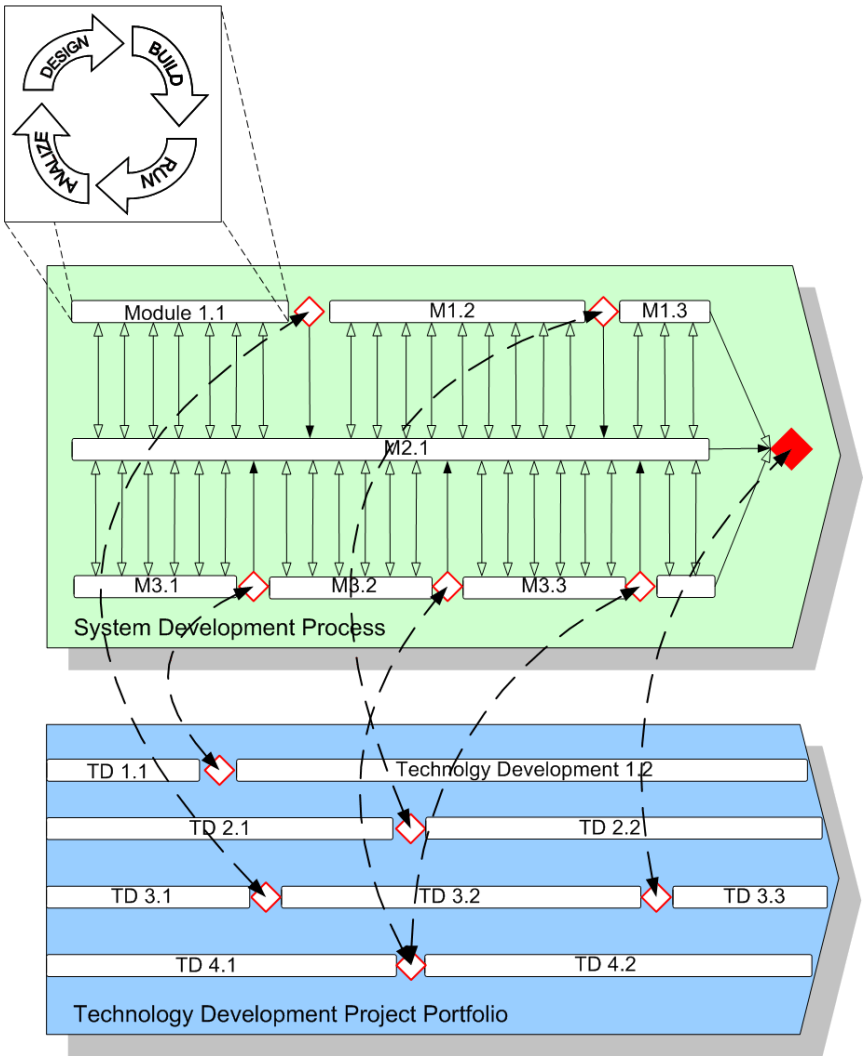


Figure F.23 Separated technology and system development processes

state of the SD project is analyzed, and decisions on more effective risk reduction and opportunity capturing actions are made. In case TD has mature technologies that could be used to capture a discovered design opportunity or reduce a type of technical risk more effectively, the effects of the new technology on the design has to be evaluated and the possibility of technology transfer has to be investigated promptly.

While effective technology transfer and integration is vital for the response ability of the adaptive SD system, continuous communication, and feedback on the results of TD from the "customers" on the SD side have to be organized. Furthermore, the joint strategic goals of TD and SD have to be understood and implemented systematically to achieve long-term system success.

F.6. CHAPTER SUMMARY

This chapter introduced the term uncertainty describing the known and unknown unknowns of SD. Uncertainty is a main driver of risk in conventional SD projects. However, as the chapter showed, besides the downside part of uncertainty that means the probability of inadequate product characteristics in the end product, there is also an upside part referring to system performance exceeding the customer's preferences. While the ability to exceed market needs leads to superior products, a main challenge of adaptive SD systems is to reconfigure their architecture to accommodate uncertainty and profit from the ability to sense changes and respond to them adequately.

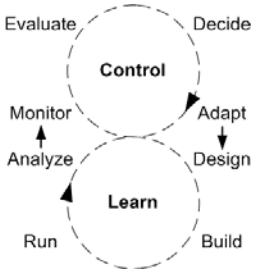
This chapter proposed that learning is the main driver of technical uncertainty reduction. Learning is the application of generational project knowledge and creativity to solve new problems in an SD project. Thus, learning is the main driver of innovation. Learning is most effective if the SD process is conducted iteratively. The various iterative problem-solving loops discussed in this chapter transform conventional waterfall SD practices into short processes of cyclic problem solving. The experimentation cycle, one of the distinguished approaches, is an SD process framework where *trial and error* experimentation is the driver of the SD work.

Other techniques to reduce risk in the SD were also discussed in this chapter. Frontloading, concurrent engineering, and the separation of TD and SD are all effective ways of risk reduction; however, thorough, deliberate planning and effective project control is required to profit from these methods; otherwise they can be major project risk drivers.

G. MANAGING THE ADAPTIVE SYSTEM DEVELOPMENT SYSTEM

G.1. CHAPTER ABSTRACT

This chapter builds on the previous, theoretic chapters of the thesis and proposes a framework for adaptive SD systems. This adaptive SD framework is a *double loop of learning and control* that provides the SD process with a flexible structure, but ensures rigorous control at the same time. The core of adaptive SD is controlled experimentation and learning supported by iterative SD work, continuous verification and validation, frequent milestones and prototypes, and risk and opportunity-driven decision-making. The adaptive SD framework creates a work*state*-driven SD environment, where the actual characteristics of the internal and external SD contexts define the system objectives and drive the evolution of the SD system architecture.



G.2. PROJECT MANAGEMENT IN ADAPTIVE SD SYSTEMS

A *project* is a temporary endeavor undertaken to create a unique product or service. *Project management* is the application of knowledge, skills, tools, and techniques to project activities in order to meet or exceed stakeholder needs and expectations from a project [PMI 1996]. SD organizations organize work in projects to develop and manufacture market products and thus generate value to the society.

The success of SD projects under dynamic circumstances lies in the capability of sensing and interpreting changes in the system context, and responding to them quickly in the form of efficient system adaptations. The task of project managers is twofold in this context: (1) they have to *implement adaptability* in the SD system through enhanced planning, and (2) they have to *control* the project and foster effective decision-making on system adaptation at the right time and place in the SD process. Hence, the management of projects in adaptive SD systems requires an increased planning effort and high management attention during the projects.

While changes in conventional SD projects with inflexible structures are undesired; the *leverage of adaptive SD systems lies in their capability to reorganize and profit from changes*. Hence, adaptive SD systems are *masters of changes* as they are organized to sense and respond to the shifting SD context. That is, the investments to implement the capability of adaptiveness in the SD system are committed because architectural adaptability and quick reaction to shifting technology and market characteristics assure superior products and market leadership in the actual market environment. Furthermore, the structure of adaptive SD systems fosters enhanced innovation and learning through iterative SD work. Thus, changes are not just possible, but are a natural part of adaptive SD projects, and they enable the effective capturing of design opportunities resulting in better products, which even exceed the customer's preferences. That is, *changes drive the generation of value and maximization of stakeholder satisfaction in adaptive SD projects.*

The difficulty of adaptive SD management is that the project manager has to *proactively* plan for uncertainty and ambiguity in order to design an SD system architecture that has the strengths of traditional SD and the additional capability of easy restructuring to respond to environmental changes. This demands a good understanding of the strategic vision of the company; and the broad perception of both the managerial and technical aspects of the ZOPH+T subsystems and

their interactions. Without this comprehensive basic knowledge about the characteristics of the SD system, flexibility, and adaptiveness cannot be implemented.

Haeckel [1999] argues that the operation of adaptive organizations is driven by an *emergent strategy* rather than a precise plan of the future. Though the emergent strategy involves a minimal number of design and decision rules, it is basically determined by the emergent behavior of the system. Strategy in this sense is *a design for an adaptive structure* that describes the basic guidelines for the adaptive SD work.

A further management challenge is to enable the sensing of internal and external changes (*i.e.,* deviations from the assumptions in the plans) and the correct interpretation of the effects of changes on the SD system [Dove 2001]. This requires *continuous measurement and control* in the project and *clearly defined, frequent decision points*, where the project team evaluates the actual project state using the measured data, and makes informed decisions on the required further actions towards maximal stakeholder satisfaction.

On the one hand, a major goal of managing adaptive SD projects is to create a *flexible project architecture*, where the direction of innovation defines the information flow in the process, and creative work is not constrained by strict plans and constraints. On the other hand, adaptive SD requires *enhanced project control*, because it aims to maximize value by continuously evaluating (*i.e.,* verifying and validating) the SD results and thus constantly collecting feedback from its environment on the fitness of its behavior. This is an obvious contradiction between free innovation and project flexibility on the one hand and rigorous control on the other. *The successful management of these conflicting project requirements demands a novel way of management thinking about SD projects, where the role of project control is not to constrain SD work, but to support the continuous adaptation of the flexible project plan to maximize overall project value.*

The adaptive SD framework in *Figure G.1* provides an effective means to handle this management problem. The **double adaptive loop of learning and control** is based on existing adaptive lifecycle models (*e.g., collaborate-speculate-learn* lifecycle by Highsmith [2000], *sense-interpret-decide-act* loop by Haeckel [1999], *idealized learning process* by Sterman [2000]), and improves them by providing a systems engineering construct for effective planning and control, and thus the successful implementation of adaptive SD systems. Additionally, the adaptive SD framework in this thesis makes a clear distinction between how sensing and responding occurs at the technical and management levels of the SD project.

The adaptive SD framework is a *process-centric SD model* developed to support deliberate decisions on fast system adaptations as a response to environmental changes. To reach this goal, the adaptive SD framework fosters the design of high flexibility into the SD process on the one hand, allowing the continuous, parallel evolution of the product design and the SD process architecture with regard to the shifting characteristics of the SD context. On the other hand, the adaptive SD framework is designed to facilitate the continuous collection of internal and external

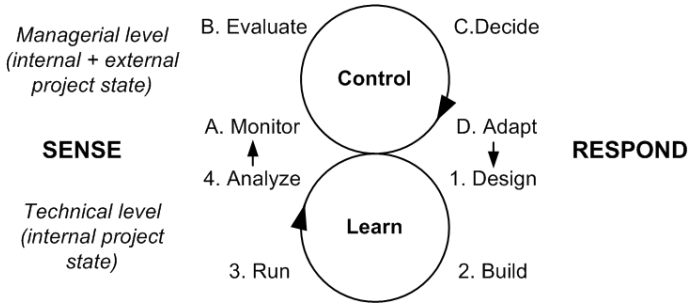


Figure G.1 Adaptive SD framework

feedback on the project performance, and to make informed decisions on the adaptation of the flexible process architecture possible.

The adaptive SD framework decomposes the SD work into effective process modules with defined working scope followed by review meetings. Hence, in adaptive SD, the goal is not to simply frontload testing in the SD process, but *to conduct systematic V&V continuously in a network of experimentation cycles with growing scope and fidelity*. A further goal is to *complete each step of SD with a verified deliverable and each major SD step with a validated prototype*. This ensures continuous internal and external feedback on both the quality and value of SD and thus effective project control.

The role of V&V as an integral part of the experimentation cycles is not only to prove performance, feasibility, and reliability; but also to support learning and innovation by providing prompt information on the results of the design activities. Rapid experimentations benefit from quick feedback on the output of the design activities without long delays that reduce the effectiveness of learning.

The process view of the adaptive SD framework in *Figure G.2* depicts the double adaptive loop of learning and control using conventional process modeling elements (*i.e.,* activities, deliverables, and milestones). During the learning stage of adaptive SD, iterative experimentation cycles are conducted according to the four-step *design-build-run-analyze* model proposed by Thomke [2003]. In case the product design has a modular architecture, the experimentation cycles on the design modules can be executed relatively independently from each other providing the developers with high freedom in their module-level work.

The deliverables of experimentation cycles are verified models or prototypes of the design representing a set of design aspects and functions desired by the customer. The performance and maturity of the process deliverables are monitored and evaluated in the second loop (*i.e.,* in the control loop) conducted at the decision points in the SD process. The goals of the control loop are to determine the overall quality and value of the designed system, decide on adaptations, and implement them in the system.

Hence, the notion of the adaptive SD framework is to break down the project into a clear network of decision points that are fed by verified information from the iterative learning loops of design-build-run-analyze sub-processes. The decision points include quantitative decision criteria representing the evolving maturity of the developed system that support continuous project control and deliberate decisions on the required design changes and system adaptations.

The evolving process states assigned to decision points in the SD process provide the project with a basic structure. This flexible deliverable-driven process structure increases the flexibility of the SD project, because the decision points and their contents (deliverables and decision criteria) are defined at the outset of the project and adapted constantly according to the actual process performance. That is, the *exact specifications* of a certain experimentation cycle are kept open until the input information with the required maturity for the design of the experiment is available

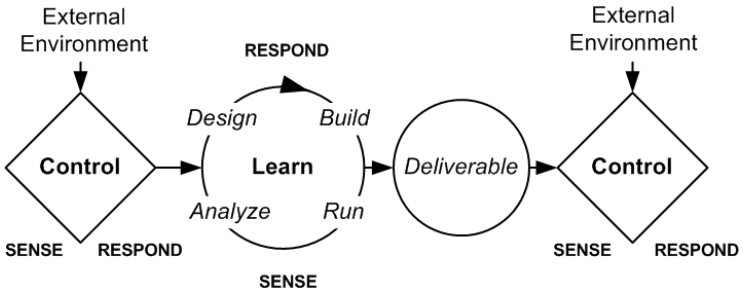


Figure G.2 Adaptive SD framework (process view)

from SD process. This moment can be as late as the start of the experimentation.

G.2.1. Adaptive Experimentation Cycles

The adaptive SD framework differs from conventional SD management philosophies, because it supports work*state*-driven [Highsmith 2000] and experimentation-driven [Thomke 2003] project management instead of traditional work*flow*-driven or plan-driven management. That is, as *Figure G.3* depicts, the iterative learning cycles (*i.e.,* the experimentation cycles) represent a precise, result-oriented main SD task whose deliverables are evaluated to decide on the success of the sub-process and thus on the required further SD steps towards maximal stakeholder satisfaction.

The experimentation cycles between two decision points (milestones or decision reviews) are scalable process elements with clear deliverables that can be tailored and applied at any level of the SD process based on the required outputs. As the deliverables vary from a computer model for thermodynamic simulation to a full-scale physical prototype for integration test, experimentation cycles are perfect basic building blocks of evolutionary and incremental SD projects. That is, an experimentation cycle is a flexible process pattern, where certain representations of the system are designed, built, tested, and analyzed to prove definite design aspects and requirements. Thus, experimentation cycles generate and offer verified solutions for certain portions of the SD problem. The solutions are then validated with the relevant stakeholders; and the effects of the process results on the overall system value are evaluated to determine the actual project status.

The model of the experimentation cycle originally proposed by Thomke [2003] was extended in this thesis to include the decision on system adaptation before and after the cycles. This extended model is shown in *Figure G.3*, where solid lines represent the original cycle, and the dashed lines depict the new process element *"step zero: control"*. The goal of the *control* step is to *monitor* process performance by collecting actual information from inside and outside the project, *evaluate* this information, and determine the actual risk and opportunity status regarding the main requirements and constraints of the SD. This information provides an excellent basis for *decision-making*, because the actual risks and opportunities represent the probable profit losses and gains the management can anticipate without changing anything in the project plans. To reduce risks and capture opportunities the management can decide to *adapt* the process to better deal with the actual SD needs.

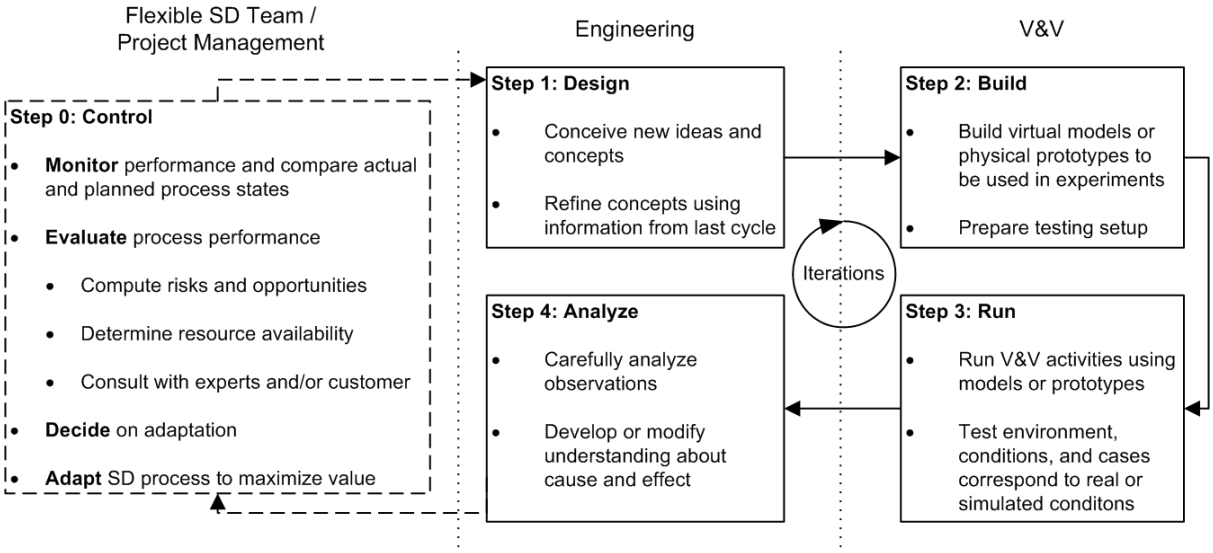


Figure G.3 Adaptive experimentation cycle (modified from [Thomke 2003])

Adaptations aiming to change system behavior and thus improve the fitness of the outputs of the SD system can be twofold. On the one hand, adaptations can concern the actual experimentation cycle to reduce risk by correcting the failures found during V&V or improve the design to capture design opportunities discovered in the cycle, and achieve the exit criteria of the sub-process. On the other hand, once the exit criteria of the cycle have been achieved and no improvement options are open, the downstream part of the process is adapted according to the results of the actual sub-process. This means moving to the next experimentation cycle with a different scope; reconfiguring it based on the actual process state and customer feedback collected during the team review; and conducting the next experiment.

In this sense, experimentation cycles in an SD process can be considered as a network of process steps consisting of atomic process elements (design and V&V activities) designed to fulfill certain process objectives. Further, the objectives of the consecutive experimentation cycles include a growing number of design aspects with increasing target values representing the planned *technical performance profile* of the SD process. During project planning and control, SD activities are assigned to these main SD process steps with attributes and functions contributing to the required evolution of product performance (*i.e.,* design and performance parameters) in the SD process. Since there are many ways to produce a definite SD deliverable, there is also more than one SD activity that can be conducted to fulfill the objectives of a certain experimentation cycle. Hence, the task of planning in adaptive SD projects is to find the possible SD activity options relevant to the project objectives and document these in the SD plans. These selected SD activities provide then the process options evaluated at the decisions on process adaptation.

The next section introduces the procedure of project decomposition that provides the required information for project adaptation.

G.3. PARAMETER-BASED PROJECT DECOMPOSITION

Parameter-based SD supports the planning and adaptation of the SD process composed of a network of experimentation cycles. That is, during planning and adaptation, the definition of actual process needs and the selection of suitable activities to fulfill these needs are done with a technical performance parameter system representing both the status and target values of the most important project and product requirements. This section presents how parameter-based project decomposition works, and how it supports project planning and the decisions on process adaptation.

Figure G.4 shows an improved version of the project decomposition procedure by Cho [2001] applicable to the planning of adaptive SD projects. During this procedure, the basic process elements of the adaptive SD project are defined successively. *First*, the main milestones of the project are set, and the deliverables for each milestone are identified. *Second*, the various product representations and development products contributing to the development of the major phase deliverables are specified. These products include the end and enabling products of the SD as depicted in *Figure G.5*.

The goal of the *third step* of project decomposition is to model the product (*i.e.,* the system of enabling and end products) as a parameter system representing every key product characteristic as a measure. This parameter-based product representation can then be used to specify the required maturity of the deliverables for the milestones and reviews, and derive quantitative milestone criteria (*i.e.,* target values) for each decision point. Conventional hierarchical measures that systems engineering measurement defines at the outset of the project based on the main project goals (*e.g.,* Key Performance Parameters (KPPs), Measures of Performance (MOP), Technical Performance Measures (TPMs)) are applicable as decision criteria [DoD & US Army 2000].

The *fourth step* concludes the first stage of project decomposition, where the detailed project control structure is defined including all technical reviews in a certain phase. The system product structure and the technical performance parameter system provide the main input for the definition of the contents of the decision points.

The output of the first stage of project decomposition is a system of decision points that breaks down the main SD objectives into a network of deliverables with quantified performance requirements. The sum of these performance requirements represents the required maturity of the design at each decision point of the SD process. Furthermore, the relations among the decision points and deliverables show the *change propagation paths;* and thus define the scope of

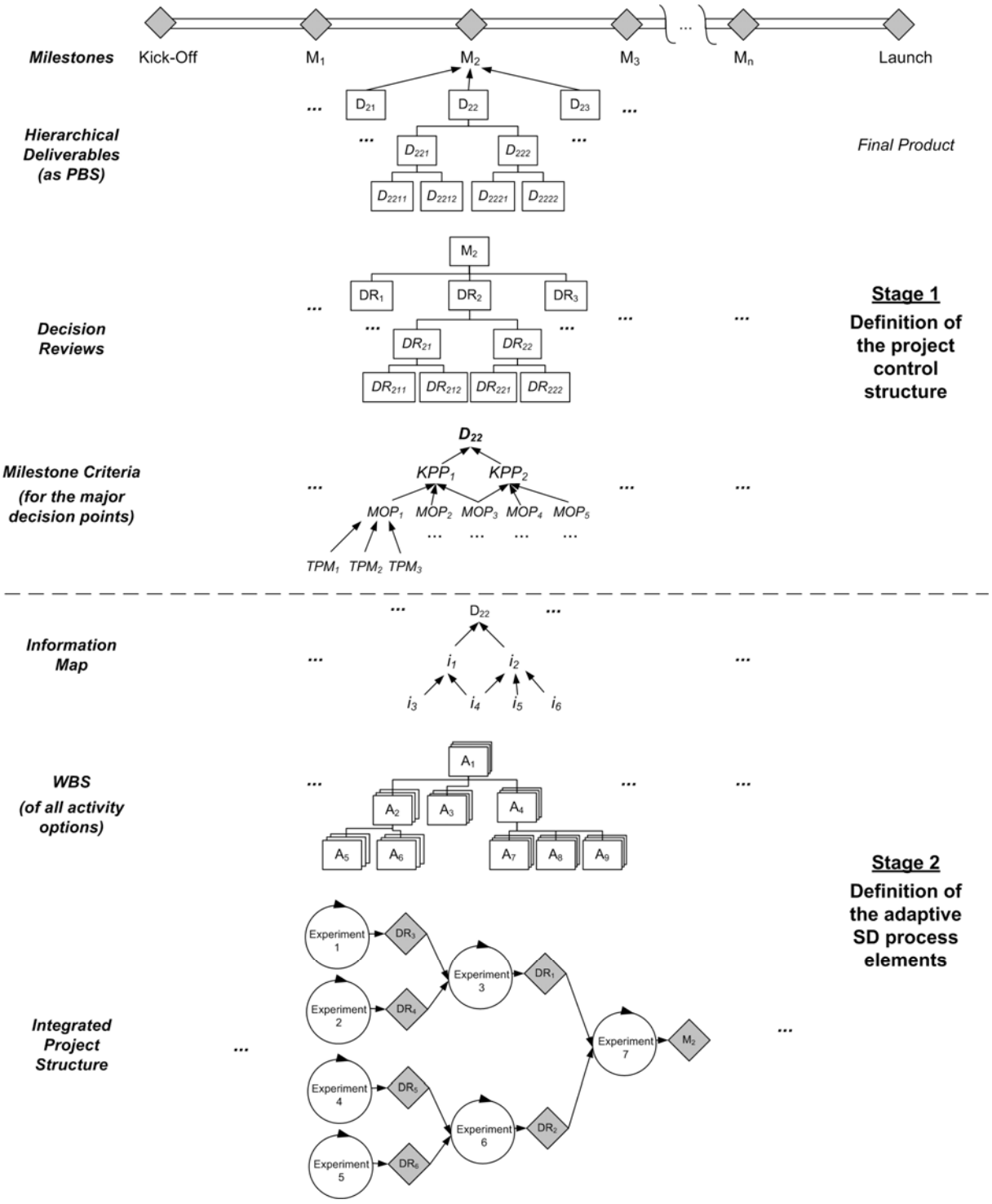


Figure G.4 Parameter-based project decomposition during adaptive SD project planning

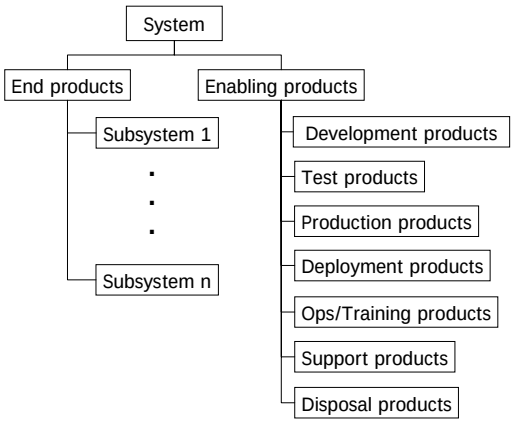


Figure G.5 System product structure

process adaptation at each decision point. That is, the network of decision points and deliverables indicate which contents of which decision points are affected by the process adaptation at a certain decision point.

In the second stage of project decomposition, once the structure of the decision points has been set in the project, the required activities to reach the milestone criteria are defined. This starts with *step five*, the creation of a list of information pieces needed to produce each milestone deliverable. In parameter-based project planning, the contents of these information pieces are modeled using relevant measures from the hierarchical measurement system (also used as milestone criteria). *Fifth*, causal diagrams are drawn between the information pieces to determine the underlying information structure of the process. The information flow defined in this step shows how the parameter values representing the actual performance of the design evolve between two SD process deliverables.

Based on the information flow in the process, the activities that provide the various pieces of information are identified and organized in a hierarchical Work Breakdown Structure (WBS), in *step six*. Effective methods to obtain and document the effects of SD activities on product properties, and thus customer desires are *e.g., Quality Function Deployment (QFD)* [*e.g.,* Hauser & Clausing 1988], *Value Analysis* [Fowler 1990], or the *Earned Quality Method* [Paquin *et al.* 2000].

The Houses of Quality (HoQ) in QFD depict the relationships between customer requirements, product part characteristics (or engineering characteristics), processes and process control methods. This way, it is possible to analyze how changes in one part of the HoQ (*e.g.,* engineering characteristics) affect the other parts (*e.g.,* customer requirements).

A special application of QFD proposed in [Chao & Ishii 2003] is shown in *Figure G.6*, where the relationship between customer requirements, engineering metrics, and SD tasks defines the HoQ. This method, derived from traditional QFD, is applied to classify SD tasks based on two factors: (1) the type of measures they affect with their deliverables, and (2) the severity of the impact on the measures.

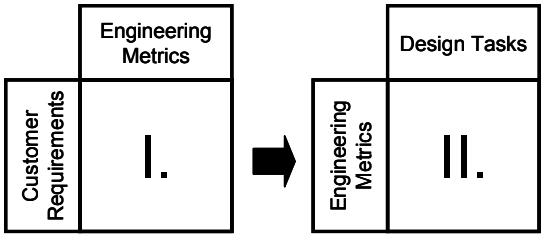


Figure G.6 QFD to determine task – metrics relations (adapted from [Chao & Ishii 2003])

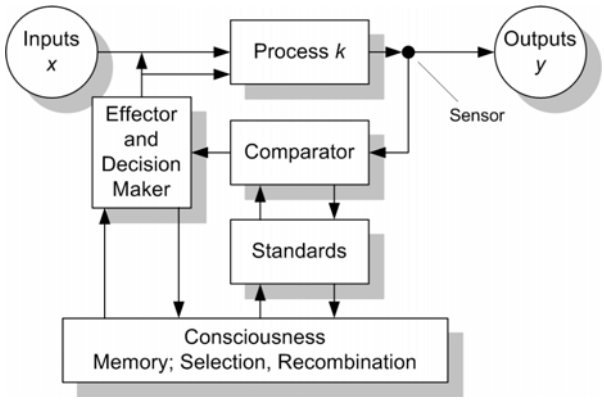


Figure G.7 Third-order feedback system of cybernetic control (adapted from [Meredith & Mantel 2003])

According to the outputs of QFD, the required information to fulfill certain process objectives can be associated with SD activities. To foster effective decision-making on adaptation, project planning in adaptive SD systems collects all possible SD activity options and *activity modes* that provide the pieces of information required to produce a deliverable and organize these in an extended WBS. In such a WBS, the branches include *all identified activity options* that can be conducted to generate a certain kind of information (*Figure G.4*).

Finally, in *step seven*, the activities in the WBS are organized into experimentation cycles based on their input needs and output products to define a first integrated project structure. These activities provide the required deliverables at the decision points in the project. While the deliverables and required target values clearly define the objectives, scopes and required fidelities of the experimentation cycles, the best activity options from each branch of the WBS can be assigned to the relevant sub-processes.

The information generated during project decomposition is a key input to process modeling during adaptive SD project planning and project control. The next section discusses the procedure and theory of adaptive project control. Then, a parameter-based, work*flow*-driven, stochastic process modeling method, the *VVT Process Modeling* method is presented in *Chapter I* and validated *in Chapter J* to discuss the actual state-of-the-art in process modeling techniques. This thesis improves the state-of-the-art in process modeling by proposing a model for adaptive SD project planning in *Chapter K*. This process modeling technique, the *Adaptive System Development Process* method, implements the double loop of learning and control in a simulation algorithm applicable to the planning of highly flexible, adaptive SD projects.

G.4. PROCEDURE OF ADAPTIVE PROJECT CONTROL

In this section, the three stages of the procedure for adaptive project control are described. This procedure integrates project monitoring, risk management, and project control to process internal and external project information describing the actual state of the SD to support deliberate decisions on project adaptation. The three stages of the procedure are presented in this section in detail; and then validated in an industrial environment in the next chapter.

G.4.1. Adaptive Project Control

Meredith & Mantel [2003] suggest that the two fundamental objectives of project control are: (1) the regulation of results through the alteration of alternatives; and (2) the stewardship of organizational assets. Hence, project control requires continuously updated information from the project and decision alternatives in case changes are needed.

The third-order feedback loop of cybernetic control by Meredith & Mantel [2003] in *Figure G.7* is a basic model of adaptive system control. The model depicts an intelligent control system capable of individual learning and dealing with unforeseen situations or system states. While conventional project control systems work according to preprogrammed response patterns (*e.g.,* documented in company work policies), intelligent control systems can change both the goals and system architecture if changes in the project needs are sensed. Thus, in adaptive control systems, the role of the gathered actual project information from *both the internal and external project environments* is inevitable for effective system operation. Furthermore, the effective control of adaptive SD projects demands a *highly flexible system architecture* that can be modified and reconnected to respond to the shifts in the system environment.

Thus, two important systems engineering disciplines have key functions in the *procedure of adaptive project control* in *Figure G.8*. Systems engineering measurement collects hierarchical, internal SD process data, which is then transformed into input information adequate for risk calculation. This information is then passed on to risk management that plays the role of the *"Comparator"* in the procedure of adaptive project control. In the second stage, the actual risk and opportunity status of the project is determined according to actual *Standards* (see *Figure G.7*), *i.e.,* the project objectives derived from the actual stakeholder needs. The output of risk management is detailed information on the actual project status translated into managerial language, *i.e.,* possible profit losses (*i.e.,* risk) and gains (*i.e.,* opportunity) by considering the consequences of the discrepancies between the actual and planned SD progress. This is the main input information for the decision on project adaptation in the third stage. The output product of the adaptive project control procedure contains the adapted project plans.

As *Figure G.8* shows, the procedure of adaptive project control is a risk- and opportunity-driven decision-making procedure. Thus, the outputs of the risk management stage are used for various purposes in the procedure. *First*, this information is applied to determine the actual project status, and provide the management with detailed information on the programmatic and technical project performance and their effects on the profitability of the project. *Second*, the risk and opportunity values are used to define the optimal team constellation for the review meeting. As the risks and opportunities point to the weak and strong points of the project, experts from relevant technical and managerial areas can be invited to the meetings to make deliberate decisions on the best next actions. *Third*, the risk and opportunity information is key input for the adaptations of the project plans, because it represents the design aspects demanding particular management attention and extra SD effort.

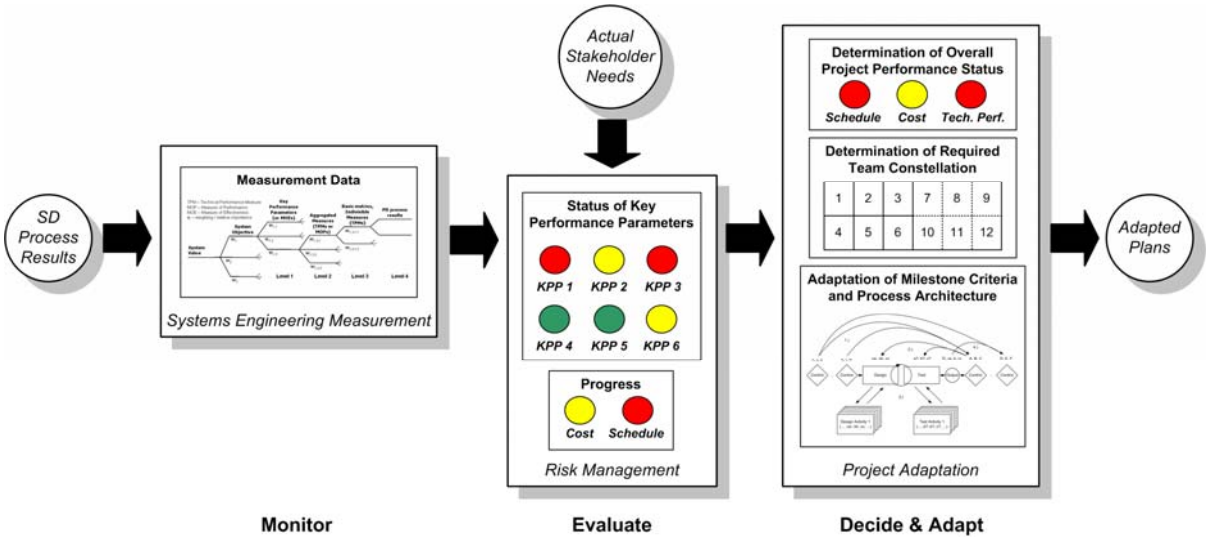


Figure G.8 Procedure of adaptive project control

The procedure of adaptive project control enhances conventional project control mechanisms by establishing tight relationship with systems engineering measurement and thus considering both technical and managerial criteria for the decisions. Furthermore, it introduces risk management functions in the project control procedure that provides the project control team with detailed information on the actual project status. Here, the basic input information for the decisions during project control is not the mere variance of the actual and planned project performance aspects, but the variances *and* the consequences thereof. That is, the data provided by risk management provides insight into the impacts of both technical and programmatic performance deficiencies on the final project success and profitability. Hence, this valuable information highlights areas of improvement needs "rank ordered" according to their effects on stakeholder value. The next sections describe the procedure of adaptive project control.

G.4.2. Project Monitoring and Systems Engineering Measurement

The first step of adaptive project control that provides important input information for the decisions is systems engineering measurement. The main instrument for defining the expectations (*i.e.,* milestone criteria in *Figure G.4*) for a process or sub-process (*e.g.,* an experimentation cycle) in a clear and transparent way is the application of systems engineering measures. These measures represent the most important product requirements that are combined with the programmatic constraints of the project to facilitate the identification of critical technical areas; the determination of the expected value added through the implementation of a change option; and the monitoring of available resources for the implementation.

During systems engineering measurement, a metrics system is established that associates measurement data (*e.g.,* maturity of SD activity outputs) with the critical requirements and constraints of the project. The goal of this activity is to define measures that provide the greatest insight into critical product and process aspects at the lowest cost. *Figure G.9* shows how the system lifecycle value can be associated with system objectives and then linked to systems engineering measures and measurement data.

The selection and specification of measures that address the critical aspects of a certain project is a complex task. Different measures have different application scopes, and address different product and process aspects. To make complexity manageable at each level of the project, few global measures are defined on the system level (KPPs), which represent the major project objectives and can be tracked in all the system phases. These measures are then broken down into lower level technical measures (MOPs) that stand for the key design and performance aspects of the modules and components comprising the system. Based on the degree of system complexity, these key technical measures are further detailed to obtain lower-lever technical measures (TPMs) that can be directly measured or associated with process data.

The basic structure of the performance measures system applied during the decision on

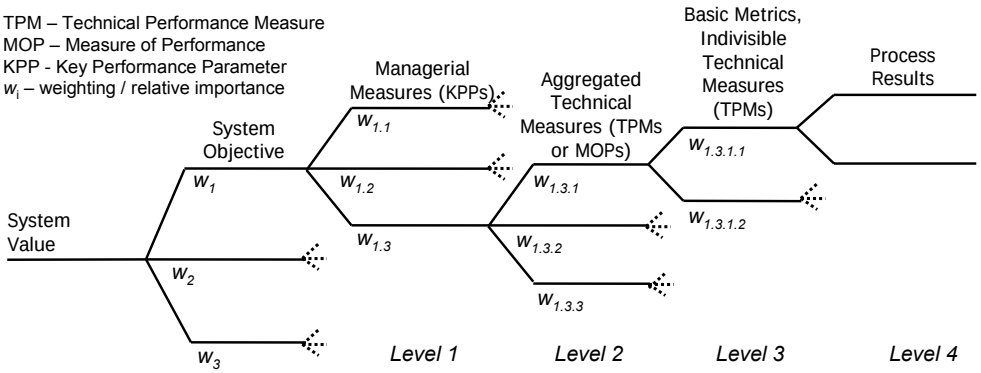


Figure G.9 System of key parameters as decision criteria

system adaptations is shown in *Figure G.9*. The tree structure of the system represents the hierarchic relations between the various kinds of measures. The different measures at each level serve as success criteria for the experimentation cycles. In the following, the four identified hierarchy levels of decision measures are described:

- *Level four:* Raw process data or raw data after processing. This data is usually the direct output of V&V activities.

- *Level three:* Basic metrics and technical performance measures (TPMs). In some cases, these measures are "indivisible", *i.e.,* they cannot be broken down into lower level measures. These measures can be directly derived from the raw test data. In some cases, even the post-processing V&V data is used as TPMs. In other cases, these TPMs can be low-level, aggregated measures that indicate the key characteristics of a component or a subsystem. Thus, the statistical evaluation method applied for the V&V data processing has a key effect on the quality and usability of the measures.

- *Level two:* These aggregated measures derived from the lower hierarchy level measures represent the key technical characteristics or properties of the overall system. During aggregation, high-level technical measures are generated from the lower level measures based on hard, statistical, or empirical rules. These key indicators of product technical performance (also called Measures of Performance (MOP)) are the main drivers of system-level technical decisions.

- *Level one:* These measures represent the critical project requirements and the highest-level project goals and constraints. These operational and managerial measures of success (also called Key Performance Parameters (KPPs)) are closely connected to the system objectives and thus to the overall customer satisfaction and project value. Based on the states of these measures, the management can decide on the further progress of the project (*e.g.,* "go – no go" decisions).

The tree structure of the decision support measures (*Figure G.9*) defines the hierarchic (or vertical) relationship between the systems engineering measures. That is, one can depict the hierarchic structure of the product requirements in a system of measurable product parameters. However, one main drawback is that the measures tree does not contain horizontal relations between measures. While vertical relations between measures are important for the interpretation of the SD activity data for requirements tracing, horizontal links foster the analysis of the propagation of the effects of failure and performance variance through the system structure. That is, the consequences of inadequate product performance in one area can be determined in other, related areas.

In case of a high number of interlinked decision measures, it is reasonable to capture also the horizontal links between the measures to determine closely related groups or clusters. *Figure G.10*

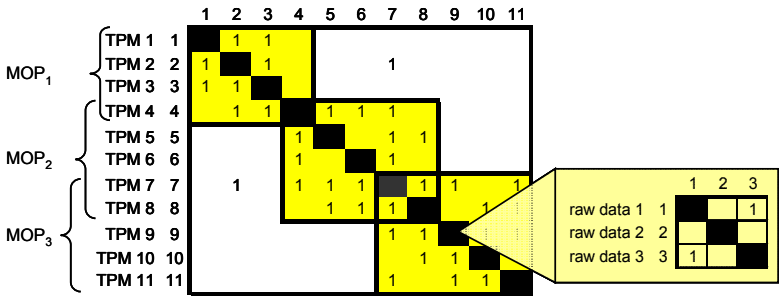


Figure G.10 Representation of the interdependencies of measures in a DSM

shows a matrix representation of *level three* of the decision measure structure using a parameter-based DSM. The matrix in *Figure G.10* showing the related blocks (or clusters) of measures can be obtained through matrix manipulation using a DSM clustering algorithm. The goal of clustering is to find subsets of DSM elements (*i.e.,* clusters or modules) that are mutually exclusive or minimally interacting.

The measures included in the identified clusters are closely related to each other and have minimal links to the measures in other modules. Thus, it usually makes sense to assign a higher-level measure to a cluster of TPMs that describes the actual performance of the cluster. This characteristic of the clusters can also be exploited during the evaluation of process performance, *because the severity and scope of the discovered performance discrepancies can be determined by identifying the other affected performance requirements*. Furthermore, the activities applied during failure correction actions (see *Figure G.18*) can be selected on the basis of their estimated effects on the design performance.

The next section describes how project risk and opportunity is calculated using the measurement data.

G.4.3. Determination of the Actual Risk- and Opportunity Status

An essential task of adaptive project control is the determination of the actual project status, because the discrepancies between the actual and planned values of the key performance indicators and the consequences thereof generate the needs for system adaptation. If the actual process and design performance cannot be determined precisely, critical areas requiring improvement efforts might be neglected reducing the value of the change actions during the decisions on system adaptation.

In order to increase the transparency and applicability of system engineering measurement data, the effects of the variances of every lower-level measure are calculated on the overall system value. The question here is: what is the system-level effect of lower product or process performance on the module or component level? Thus, a method is required that is capable of translating the discrepancies of lower-lever technical measures into impacts on the system-level managerial measures.

The hierarchical systems engineering measurement system in *Figure G.9* includes measures that break down the managerial and system-level technical requirements into lower-lever technical goals. The measures together in each level represent the overall performance of the product in growing detail. Stakeholder value depends on how well the product meets the preferences for these measures. Hence, knowing the interrelations among measures and the stakeholders' preferences for the measures, the shifting overall system value can be calculated from the positive or negative discrepancies between the actual and target values of measures at any level of the structure.

In the presence of uncertainty, the overall system performance depends on the possible outcomes of the technical and managerial parameters represented by the high-level KPPs of the system. Since the stakeholders' expectations are different for each parameter, it is important for the decision-makers to know the effects of the performance discrepancies of each KPP on the overall stakeholder value. The goal of decision-making at each level of the SD project is to maximize the stakeholder value provided by the product by maximizing the value of system performance characteristics and minimizing SD cost and duration.

Utility functions describe how stakeholder value changes as a function of the possible parameter outcomes. To support decisions during *design for uncertainty*, stakeholder utility is transformed into impact (*i.e.,* possible profit loss through inadequate performance) and benefit

(*i.e.,* possible profit gain through exceeded stakeholder preferences). *Figure G.11* depicts three basic types of impact functions derived from the *Kano* taxonomy for customer utility [Clausing 1994].

The first KPP type in *Figure G.11* represents a *must-have* need. Such needs are usually met by current technology, and every new product must satisfy these needs [Dahan & Hauser 2000]. Furthermore, some government and industry regulations refer to this type, and have to be fulfilled by the product (*e.g.,* package sterility in food packaging discussed later in the case study). In case the required KPP value is not met in the product, it results in high profit loss, however exceeding this need does not bring any extra profit for the company.

The second KPP stands for a *more-the-better* need. If such a need arises in an industry segment, companies strive to focus technology development efforts to be able to increase this very characteristic of product performance. Typical KPPs of this kind are, *e.g.,* the speed of computer processors, resolution of cameras and color density of displays in mobiles phones, mass of aerospace systems, fuel consumption of cars, *etc.* The limitation for maximizing such KPPs is usually the project budget and the resulting product unit cost.

The third main stakeholder need represented by KPP_3 is a *delighter*. This special class of needs refers to needs which customers have difficulty articulating or rarely expect to have fulfilled [Dahan & Hauser 2000]. Many innovative product features that come from technology development are delighters aiming to provide the product with a special flair and differentiate the specific product from others. It is important to realize that the result of *technology push* strategies is that delighter needs become must have needs in the market if they are to be successful. Adaptive SD is capable of exploring delighter needs and capturing opportunities through incremental product releases. Various safety features in automobiles (*e.g.,* airbag, ABS, ESP), the graphical user interface in computer software, or Internet access in hotels, airports, airplanes, *etc.,* started as delighters and became standard product features in the long term [Dahan & Hauser 2000].

The application of the impact-benefit functions in *Figure G.11* has multiple advantages for decision makers. *First*, it is a good basis for the evaluation of decision alternatives by providing the value of the whole range of possible parameter outcomes. While conventional impact functions depict only the effects of negative outcomes, the representation of the benefits of better products opens new dimensions during decision-making towards the identification and capturing of design opportunities.

Second, the functions in *Figure G.11*, which are continuously updated by Marketing, depict the critical areas (*e.g.,* must have needs) that require particular attention from the developers and innovative design aspects that drive the stakeholder value of the system. *Third*, the impact-benefit functions are an effective means to translate between the technical staff and the management showing the expected profit variances as a function of technical system performance.

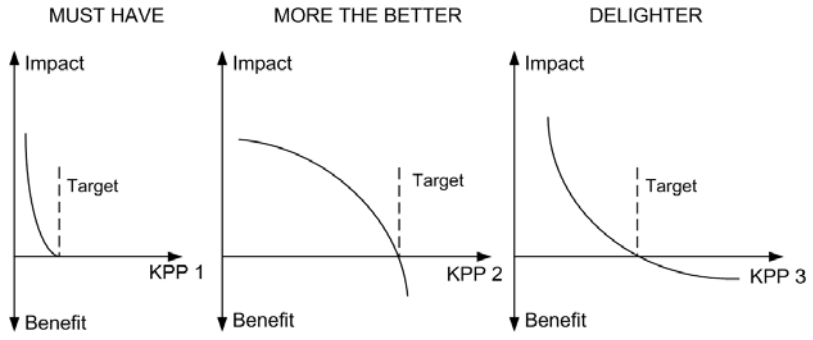


Figure G.11 Impacts and benefits derived from customer utility based on the Kano taxonomy

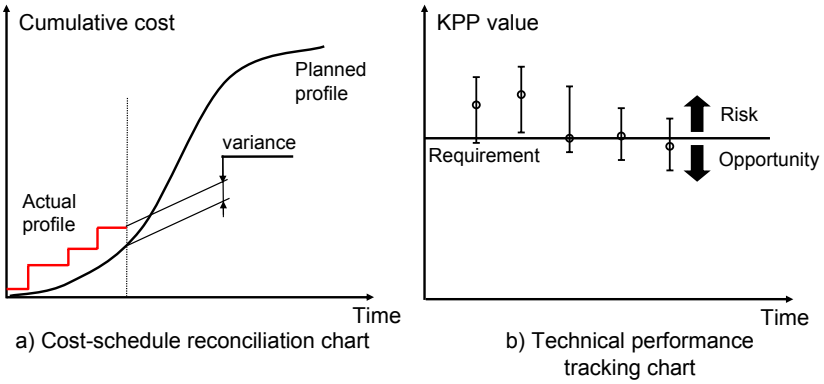


Figure G.12 Project performance monitoring charts

As the actual process performance values, the measurement system, and the impact-benefit functions are known, the project management can determine the actual project status embodied by the single KPP risk and opportunity values. This information can be applied for different purposes: (1) the overall project risk and opportunity status can be determined using the single KPP risks and opportunities; (2) as the single KPPs represent the major design aspects, the required composition of the review team for effective decision-making can be revealed on the basis of their status; and (3) the single KPP status can be used to define the activities with the highest magnitude on the critical areas for the SD process adaptation.

Monitoring charts are used to get an insight into actual versus planned performance. The *cost-schedule reconciliation chart* and the *earned value chart* that arose from it are well-known project management tools to capture the earned value of the work carried out (value completed) for the tasks performed in the project [Meredith & Mantel 2003]. While comparing the actual with the planned project performance, the difference between the two is calculated to evaluate the status of the project (*Figure G.12a*). Earned value charts capture the difference between planned and actual project performance, but they do not provide any information on the risk implied by this difference—often, this risk will grow as a non-linear function of the difference. A more effective method for performance monitoring is to compute risks from the performance deviations and use the risk values for the decisions.

Technical performance tracking charts (*Figure G.12b*) are used to forecast probable outcomes of the key performance aspects (*i.e.,* KPPs or contributing TPMs), record estimates, and actual values, compare them to the projected ones, and calculate technical risk in a project [*e.g.,* Pisano 1995, INCOSE 1995, DoD 2001a]. Thus, tracking technical measures effectively supports project control. Furthermore, technical performance tracking assists decision-making by relating information from the SD process to high-level project goals, thereby providing management insight to the technical state of the project. This also facilitates the adaptation of the SD process to the changing internal and external needs.

Project performance monitoring in the adaptive SD framework integrates the methodologies of earned value and risk management to estimate the consequences of performance variances on the overall project value. Furthermore, besides risk, opportunity is calculated to account for the situation, when the information from systems engineering measurement shows that for some aspects of the design it might be possible to even exceed the stakeholders' expectations, if adequate actions are made.

Performance profiles like the ones in *Figure G.12* and the milestone criteria for the decision points assist continuous risk management by providing reference points for risk calculation. That is, the target values and impact functions model the stakeholders' preferences for the product, which can be combined with the performance data from the project to determine the actual risk

and opportunity status. This is depicted in *Figure G.13*, where the actual value of a TPM, where *smaller is better*, is illustrated by the PDF, the desired performance (*i.e.,* target value) by the dashed line, and the estimated impacts / benefits of negative / positive performance discrepancies from the target value is depicted by the solid line (*i.e.,* by the impact / benefit function).

During the evaluation of project performance, the values of two representative measures are calculated from the functions in *Figure G.13* (*i.e.,* risk and opportunity). Risk representing the consequences of negative outcomes (*i.e.,* negative performance discrepancies) is computed with *Equation (F-3)* for KPPs, where *smaller is better*:

$$R_{KPP_k} = \int_{T_{KPP_k}}^{\infty} f(x_{KPP_k}) I(x_{KPP_k}) dx \qquad \textbf{(G-1)}$$

where R_{KPPi} is the risk, x_{KPPi} are the possible outcomes of the KPP, $f(x_{KPPi})$ is a PDF representing the probabilities of all x_{KPPi} outcomes, T_{KPPi} is the target or requirement value of the system performance, and $I(x_{KPPi})$ is the impact function.

For the calculation of opportunity, a similar equation is applied:

$$O_{KPP_k} = \int_{-\infty}^{T_{KPP_k}} f(x_{KPP_k}) B(x_{KPP_k}) dx \qquad \textbf{(G-2)}$$

where O_{KPPi} is the opportunity, and B_{KPPi} is the benefit function (*i.e.,* the negative side of the impact function). In case of KPPs, where *larger is better*, the equation applied here for risk calculation provides the value of the opportunities, and *Equation (G-3)* is used for risk calculation, respectively.

The illustrative outputs of the risk calculation stage are depicted in *Figure G.14*. The first four charts are actual values of four KPPs together with the targets and impact / benefit functions. While each KPP has different characteristics, the PDFs, the target values and the impacts / benefits are all different. Hence, the calculated risks and opportunities are the values that depict the *real project status*. While two performance aspects in *Figure G.14* inherit low risk (*"KPP_2"* and *"KPP_3"*), two others (*"KPP_1"* and *"KPP_4"*) are in *yellow*, representing moderate risk. Furthermore, opportunities were discovered in two areas (*"KPP_1"* and *"KPP_3"*); however, only one of the two aspects (*"KPP_3"*) has real potential for the project (the one in *green*).

Besides technical performance, programmatic risks and opportunities are also analyzed during project control. However, the determination of cost and schedule risk requires more

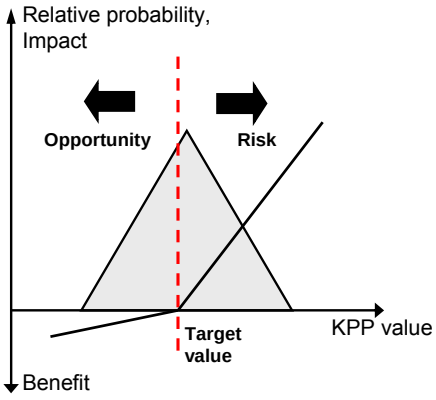


Figure G.13 Chart for risk and opportunity calculations

sophisticated techniques. The goal of programmatic project control is to take the actual project cost and duration, forecast the overall values for the project, and decide on the criticality. Meredith & Mantel [2003] review various analytic methods for project control, *e.g.,* earned value analysis, benchmarking, QFD, variance analysis, trend projection, *etc.* Even though these methods are not risk-based control methods, they can be modified to provide probabilistic outputs applicable for stochastic risk calculation.

The last two charts in *Figure G.14* show two examples for acquiring inputs for stochastic risk calculation for programmatic aspects. The first one is a modified version of the *cost-schedule reconciliation chart* in *Figure G.12a*, which has been improved to allow for risk calculation. The lowest dashed line in the chart is the cost target profile, and the two dashed lines above it represent limits obtained from historical project data. These functions are proxies for risk calculation that include statistical estimates for the probability of missing the final target because of the actual project status. That is, if the actual value is above the first (target) profile and below the middle dashed line, the probability of failing the cost target is low. If the actual value is between the middle and the top line, the probability is moderate, and probability is high above the top line, respectively. These values can be associated with the right impact values to calculate the risk inherited in the actual project cost value.

A better way to estimate programmatic risk is the application of stochastic network methods or process modeling techniques. These methods, described later in *Chapter I* and *K* in detail, stochastically estimate the possible outcomes of technical and programmatic process attributes. That is, the project manager enters the actual project status in the network or process modeling tools for each decision, and approximates the possible outcomes of the performance of the process attributes under the modified circumstances. The last chart in *Figure G.14* depicts how the output of stochastic process analysis is applied directly for risk and opportunity calculation. Thus, stochastic process analysis techniques provide effective means to support the determination of the actual project state during the procedure of adaptive project control. The next section describes how the risk and opportunity data are utilized before and during decision-making.

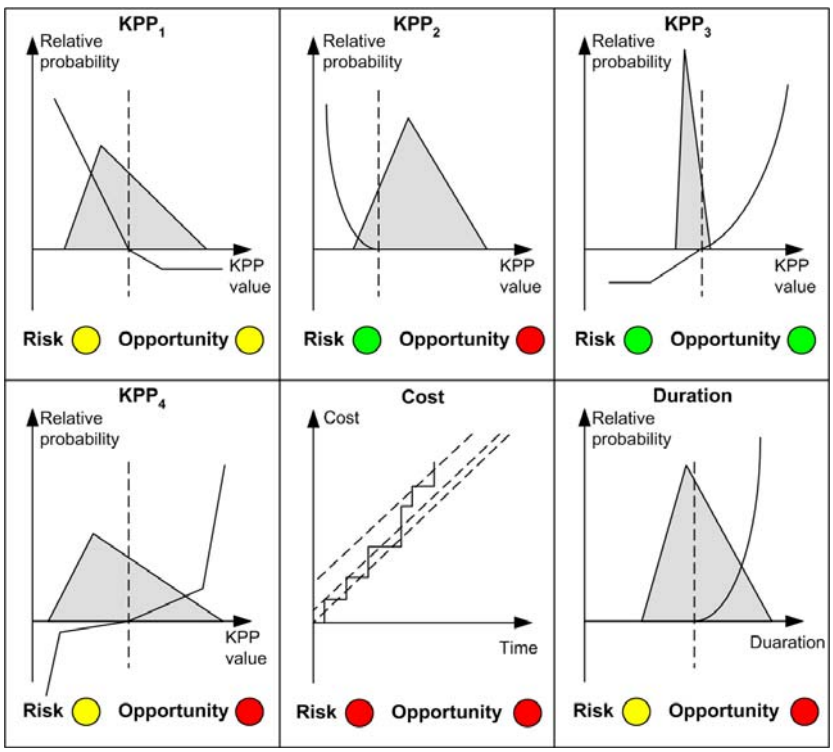


Figure G.14 Illustrative outputs of risk calculation

G.4.4. Project Adaptation

The third stage of the procedure of adaptive project control uses the single risk and opportunity values as input information and first, it calculates the overall project risk, second it determines the optimal team constellation for the review meeting, and finally, it adapts the SD system to maximize value.

G.4.4.1. Determination of the Overall Project Performance Status

In the first step of the project adaptation stage, the overall risk and opportunity values are calculated for the project. These numbers are representative values for the overall project status regarding the fulfillment of the final project success criteria. The customers determine the overall value of a project through the characteristics of the final project output (*i.e.,* the product). If the product characteristics fulfill their previously articulated (and not defined) needs, and the product is available when they need it at a price they can afford, the product means value for them, and thus they will acquire it.

Therefore, the ultimate task of project management is to fulfill and exceed stakeholders' needs by maximizing the effectiveness and efficiency of the SD work, and thus deliver the best product at the lowest resource consumption possible. Both the stakeholder' needs and the managerial objectives of the SD project have three main dimensions: the *technical performance* of the product, the *financial aspects* related to the product (*i.e.,* price of the product, or the cost of the SD project), and the *timeliness* of the product introduction (*i.e.,* time point of product launch, delivery date, or project duration).

Risk calculation in the previous stage provides the actual risk and opportunity values for each project measure relevant for a certain review meeting. Now, the task of the project manager is to calculate one risk and one opportunity value for the overall project. Overall technical performance risk and opportunity are computed as the weighted average of the risks and opportunities of the single KPPs:

$$R_{TP} = \sum_{k} w_{KPP_k} R_{KPP_k} \qquad \textbf{(G-3)}$$

where R_{TP} is the overall technical performance risk, w_{KPPk} is the weighting showing the relative importance of the KPP value for the customer, and R_{KPPk} is the risk of the KPP_k.

The overall technical performance risk is then combined with the risks in the other two dimensions of project performance to acquire the overall SD project risk:

$$R_{PD} = w_S R_S + w_C R_C + w_{TP} R_{TP} \qquad \textbf{(G-4)}$$

where R_{PD} is the overall SD risk, w_S is the relative importance of the SD schedule or duration, R_S is the schedule risk, w_C is the relative importance of the SD cost, R_C is the cost risk, and w_{TP} is the relative importance of the product technical performance, R_{TP} is the technical performance risk.

Overall project opportunity can be calculated the same way, using the single values and weightings for opportunity in *Equation (G-3)* and *(G-4).* Overall project risks and opportunities are recorded after every decision, and they are tracked throughout the project to keep the project management up-to-date about the SD performance.

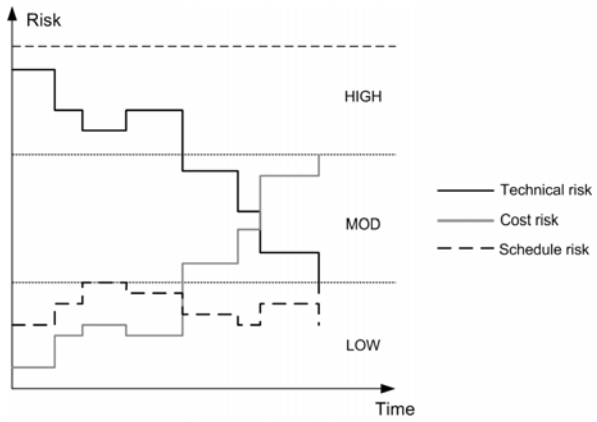


Figure G.15 Risk waterfall chart for the three key dimensions of project performance

Typical risk waterfall charts for the three key dimensions of project performance are depicted in *Figure G.15*. These charts support both context-specific and generational learning in the SD if documented regularly in every project. This way, the project manager can estimate performance, risk, and opportunity profiles during planning, and forecast the project evolution in terms of risk reduction or opportunity capturing. Then, the project can be planned by assigning SD activities to the estimated risk reduction and opportunity capturing profiles based on the capability of the activities of moving the risk and opportunity values to the right direction. This supports *process design for uncertainty* and thus risk- and opportunity-driven project planning and decision-making on adaptations.

G.4.4.2. Determination of the Optimal Team Structure

The actual risk and opportunity status of the project highlights the areas where improvement is needed to achieve the project and system objectives. These areas can be both known and unanticipated risk areas of the project as well as foreseen and unforeseen opportunities emerged during the SD work. Now, the goal of project management is to organize an efficient meeting where experts from all critical competencies make collaborative decisions on the required actions to maximize system lifecycle value.

The quality of the team structure is important for both the effectiveness and efficiency of decision-making. Thus, the constellation of the review teams is defined on the basis of the actual project status. The risk- and opportunity-driven set-up procedure of a flexible review team is presented in *Figure G.16*. The review team has an adaptable structure of six fixed team members (*i.e.,* the core team), who participate in each meeting with similar purpose; and six free slots for additional team members who are always invited depending on the actual project status.

The interdisciplinary review team includes six core members from all project-relevant technical and business disciplines. That is, besides hardware and software development, manufacturing, systems engineering and testing; marketing also has a fixed seat in the meetings. This is important, because technical risks and opportunities can be evaluated better if experts who are aware of the customer's voice also take part in the decisions. Though interdisciplinary work is difficult, because experts from each discipline have a different point of view about the system and speak a slightly different language, these meetings are the best places to learn about the system under development, and solve system-level problems cooperatively.

The free slots in the flexible review team structure can be filled with experts from relevant company SD and TD disciplines based on the actual risk and opportunity status and the character and experience of the available team members. For instance, in case a technical opportunity is found in the SD process, experts from relevant areas of the technology development are invited to decide how emerging technologies can be selected and integrated with

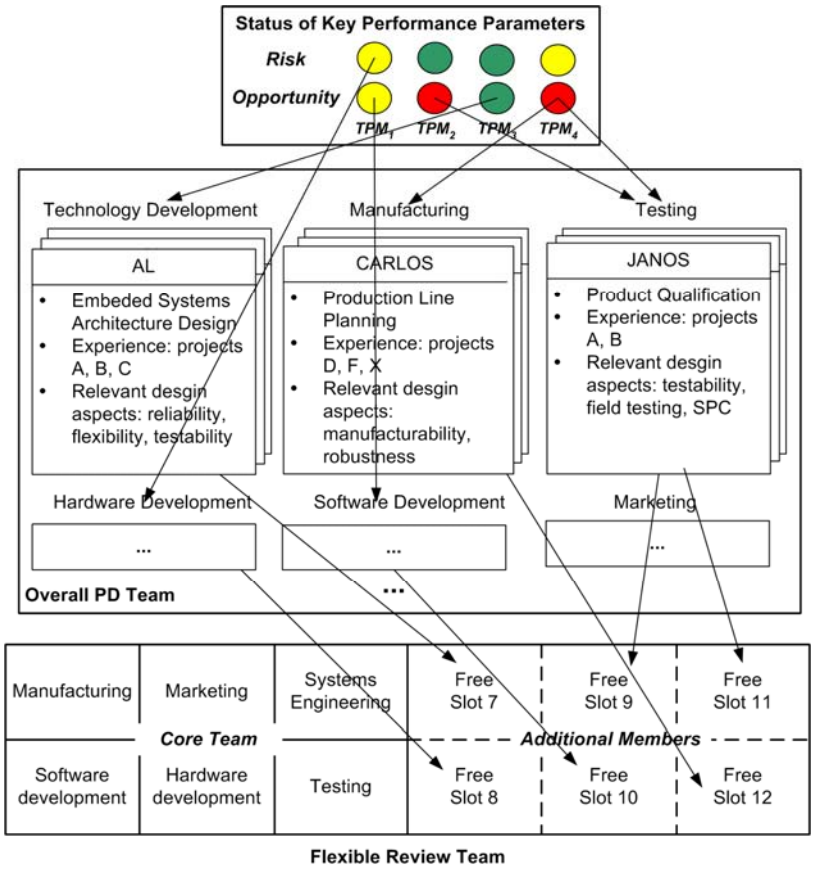


Figure G.16 Risk /opportunity-based determination of a flexible review team structure

the existing ones to capture the discovered opportunity. Furthermore, marketing experts assist decision-making by providing first estimates for the value of the different design alternatives aiming to capture the opportunity.

The combination of planning interdisciplinary team structure with risk- and opportunity-based project control assists decision-making by guaranteeing the optimal team size and quality. This enhances the effectiveness of collaborative learning and problem-solving, and supports the elimination of barriers among company competencies and departments. Furthermore, if software tools support project control and risk calculation, then meeting scheduling can be linked, reviews can be planned, and experts can be invited automatically, which prevents schedule conflicts and unnecessary misunderstandings in the SD organization.

G.4.4.3. Adaptation of Milestone Criteria and Process Architecture

After the overall project risk and opportunity status has been determined and the optimal team structure designed, the team analyzes the actual situation in the project and decides on the required next steps.

The decision framework for system adaptation is depicted in *Figure G.17*. The goal of the system adaptation procedure is to evaluate the risks and opportunities with regard to the actual plans and objectives, and decide if changes are required to assure final project success. During the decision-making procedure, all critical design areas are analyzed, the problem scopes are drawn, and improvement alternatives are generated aiming to increase the system lifecycle value. While all improvement options have both positive and negative effects, the review team has to make tradeoffs between the key project objectives, and find the process option (depicted by the small DSMs in *Figure G.17*) that adds the highest stakeholder value to the project. The evaluation

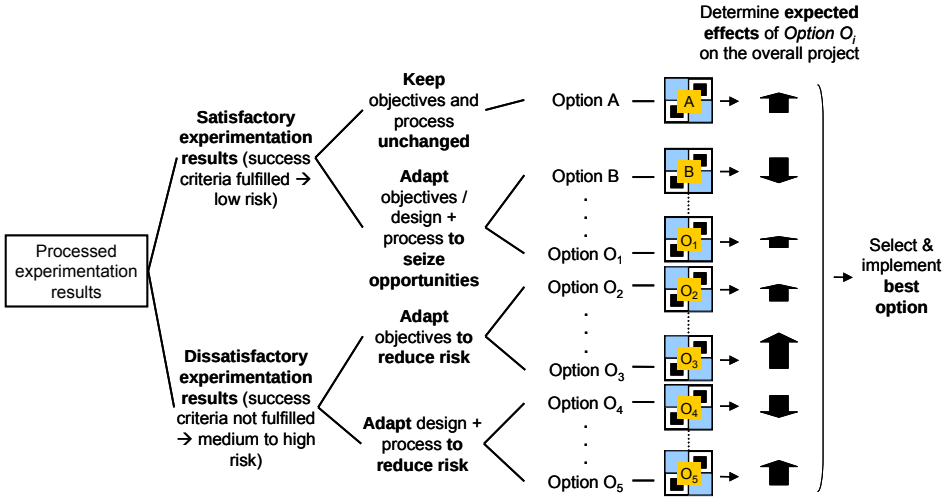


Figure G.17 Decision framework for system adaptation

of process alternatives is done on the basis of the capability of the SD activities to reduce risk and capture opportunity in the critical technical performance areas.

Figure G.18 shows how process adaptation works in a parameter-based SD context. As the arrows *(1)* and *(4)* depict, the results of finished experimentation cycles might modify the objectives (*i.e.,* target values) of the subsequent cycles and adjust the focus of system development by moving it to the critical design aspects (*i.e.,* TPMs). The critical design aspects and shifting targets determine the characteristics of the design and V&V activities required in the experimentation cycles (arrows *(2)* in *Figure G.18*).

As *Figure G.18* depicts, SD activities are required with high effects on the TPMs *a*, *b*, and *c* in the actual process state. Relevant activities are selected from the WBS and the flexible project plan, and the best improvement option is identified for the process following the decision framework for process adaptation (arrows *(3)* in *Figure G.18* represent the results of the decision-making procedure in *Figure G.17*). As the SD process needs are shifting during the SD project according to the actual design performance and changes in the market needs, the value of the SD activities for the project also changes. That is, SD activities which are quite effective in one process state might loose their importance for the project in a later state with changed conditions. Hence, the goal of project adaptation is to continuously adjust the SD process (and product) design and select SD activities (and product modules) from the company knowledge base and the flexible plans that are the best for maximizing system lifecycle value.

Process adjustment is particularly valuable in case of iteration and rework in the project. Even though Eppinger *et al.* [1997] define iteration as *the repetition of tasks to improve an evolving SD process*, in reality the same versions of activities are seldom repeated or reworked in the SD. That is, during iteration, assumptions or failures of previous activities are corrected, which usually means modifications in the design, but *not the repeated, complete execution of finished SD activities*. The reason is that every activity increases the design knowledge, and thus the process state after an activity is different from the state before it. Hence, this different process state generates different process needs that can only be satisfied with different actions and not the repetition of activities adequate for prior process needs.

The term *rework* stems from plan-driven project management, where the parts of the process already executed are repeated to account for new information coming from later parts of the project. Thus, rework has a rather negative meaning referring to an undesired process failure mode that causes rework on tasks already completed in earlier parts of the process. This negative meaning is due to the false assumption of conventional project management that projects can be

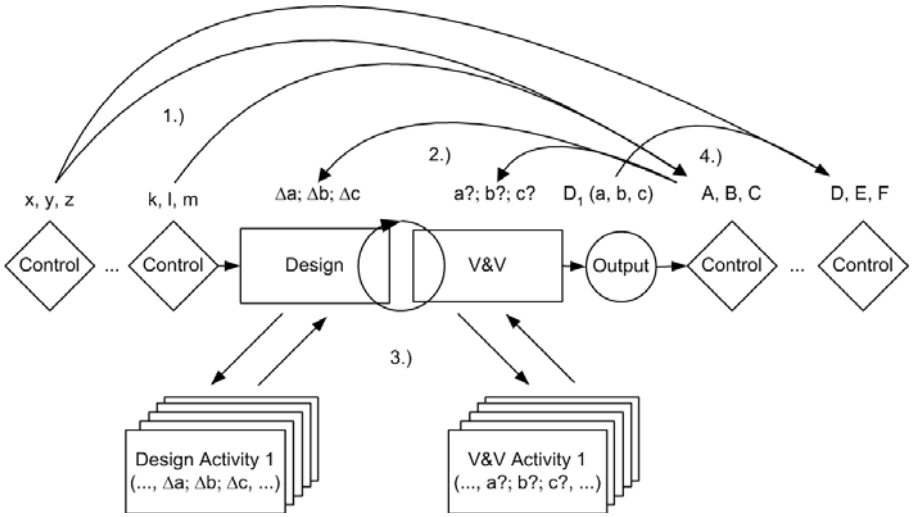


Figure G.18 Procedure of process adaptation

planned completely at their outset, and that the planned activities are capable of delivering the desired product right-at-the-first-time.

On the contrary, adaptive SD projects are organized to have the capability of adjusting the original direction if unforeseen SD process states involve new information that changes the overall value of the project. *The discovered new information does not mean that the work done before was wrong, but it points out that some aspects of the SD that require particular attention were not considered during planning*. Thus, the scope of the original SD problem has slightly changed, the solution (*i.e.,* project plan) proposed for the original problem is no longer suitable, and the simple rework of the original tasks does not solve the new problem effectively and efficiently.

Hence, the term rework does not apply to iteration cycles in adaptive SD. Here, iteration is considered as an innovative sub-process to improve the design areas with inadequate performance using SD activities that are most suitable for the purpose of iteration. While in adaptive SD, the process is designed for high flexibility, the architecture of the iteration cycles changes in each cycle to adjust the scope of the SD work and re-concentrate resources on productive and value adding areas.

G.5. CHAPTER SUMMARY

Adaptive SD is *process-oriented*. Hence, the behavior of the SD process and the value generated for the society by this behavior shape the flexible subsystems of the ZOPH+T SD enterprise in the project. The *adaptive SD framework* presented in this chapter described how SD work could be organized to augment the effectiveness of learning, and how the SD process could be controlled to guide the developers towards high value. While enhanced learning increases process effectiveness, and adaptive project control improves efficiency, the application of the adaptive SD framework contributes to maximal stakeholder value in the whole system lifecycle.

The second part of the chapter introduced the *procedure of adaptive project control* to show how decision-making in the control loop of the adaptive SD framework works. Because *the goal of system adaptation is to change the SD system to move to a state with higher overall value*, the main input to adaptive project control comes from systems engineering measurement. Measurement tracks project performance against actual market needs to obtain measures for estimated stakeholder value (*i.e.,* risk and opportunity values); and provide decision-making with criteria for the determination of the best value adding strategy for the actual process state.

H. CASE STUDY I – DECISION-MAKING IN ADAPTIVE SYSTEM DEVELOPMENT AT TETRAPAK CARTON AMBIENT

H.1. CHAPTER ABSTRACT

In this chapter, the implementation and validation of the *procedure of adaptive project control* and the *decision framework for system adaptation* is demonstrated in a real industry environment through a case study. The case study at TetraPak Carton Ambient in Italy demonstrates how parameter-based project monitoring and risk- and opportunity-driven decision-making can be implemented in the food packaging industry SD processes, and how it can be integrated with conventional SD methodologies.

This chapter focuses on the validation of the control loop, and applies data from two pilot projects conducted at TetraPak. The objective of this chapter is to set up a parameter-based measurement system and a related decision-framework that supports deliberate decision-making at each level of the SD project. After the successful implementation of these project control methods, the next case study on process modeling will show how adaptable SD processes can be planned to enable the effective application of the adaptive SD framework at TetraPak.

H.2. STRUCTURE OF THE CASE STUDIES IN THIS THESIS

The case study was conducted in one of the pilot projects of the EC Fifth Framework project *SysTest*. Eight partners from different countries and industrial segments of the EU gathered in the EC research project SysTest to develop a generic Verification, Validation, and Testing (VVT) Methodology for the whole product lifecycle. The new VVT Methodology was intended to revolutionize Testing and Verification & Validation (V&V) methodologies by offering a *systems engineering framework for the VVT strategy planning and VVT planning procedures*. The new VVT Methodology evolved from widely used industry V&V standards and methods (DoD, ASME, ECSS, SAE, ISO, IEEE, *etc.*) and merged these with current European industry practices.

The VVT Methodology developed in SysTest supports parameter-based SD planning and control, proposes enhanced process modeling as a basic planning tool for SD projects, and fosters the strong integration of V&V with other systems engineering disciplines (*e.g.,* requirements engineering, risk management, systems engineering measurement, project planning, configurations management, *etc.*) [Lévárdy *et al.* 2004b]. Thus, the SysTest pilot projects, conducted to prove the validity of the new VVT Methodology, provided an excellent environment also for the validation of the theoretical results of this thesis, *i.e.,* the adaptive SD framework, the decision and control procedure for adaptive SD projects, the VVT Process Modeling procedure and the *Adaptive System Development Process* method discussed in the following chapters.

Thus, first, a brief overview on the implementation of the VVT methodology in SysTest is given in this chapter to prepare the later, more thesis-relevant parts of the case study. A further goal of this chapter is to describe the TetraPak Carton Ambient SD environment. The SysTest pilot project structure conducted at TetraPak Carton Ambient (TetraPak) in Modena, Italy, is depicted in *Figure H.1*. According to a typical research project schema, the TetraPak pilot project in SysTest started with the identification of improvement needs, and the definition of project

objectives based on these needs (*Stage 1*). These basic needs and the derived research objectives defined both the focus of the research project and the tailoring needs regarding the new methods to be implemented. The overall goal of TetraPak, as an SD organization, at the introduction of the new VVT methodology was to select the methods that are the most adequate in addressing the actual problems of the company SD projects, tailor them to the company SD characteristics, and integrate them with the current company methodologies. These tasks were done in *Stages 2* and *3* of the pilot project structure.

In *Stage 3*, during integration, the new parts of the tailored VVT Methodology were integrated with the current company V&V and SD methodologies. The goal here was to experiment with the new methods and evaluate their feasibility in the TetraPak SD environment. During this first pilot project, existing company data sources (databases, outputs of existing SD and VVT methods, formalized planning and decision procedures, *etc.*) were analyzed to determine if the input information required by the new methods is available in the current SD environment. Further, the links between new and exiting methods and tools were established, and channels that allowed gathering the missing pieces of input information were defined.

While the goal of *Pilot Project I* conducted in *Stage 3* was to implement systems engineering methods in V&V planning, and integrate these with the current company methodologies, *Pilot Project II* (*Stages 4, 5,* and *6*) aimed to optimize the effectiveness of the implemented methods and improve the SD project environment to increase the performance of the new methods. During the *Pilot Project II*, effects of V&V on SD process performance were analyzed, and methods to increase these effects were introduced. On the one hand, the decision-making procedure after V&V activities and experiments was documented and improved to define a *generic framework for SD process adaptation*. On the other hand, a *process modeling method and tool* (*i.e.,* the VVT Process Modeling (VVTPM) procedure and tool) developed in SysTest and introduced in *Pilot Project I* was further improved and incorporated in the current TetraPak planning methodology. *Stage 6*, depicted by dashed lines in *Figure H.1*, was a *virtual* pilot project conducted in the scope of this dissertation. That is, data from *Pilot Project II* used to validate the VVTPM tool was applied to validate the *Adaptive System Development Process (ASDP)* method proposed in *Chapter K* of the thesis. Finally, the results of *Pilot Project I* and *II* were evaluated from the industry point of view to determine the overall feasibility of the implemented systems engineering methods at TetraPak.

Further sections of this chapter introduce the company TetraPak Carton Ambient and highlight some aspects of the *Stages 1, 2,* and *3*. These stages are not discussed in detail, since they are not altogether relevant to this thesis. On the contrary, *Stage 4* is the main subject of this section, where the case study on the decision and control procedure for adaptive SD projects was conducted. The following chapter describes the role of process modeling for adaptive SD project planning. The methods introduced in the next chapter were validated in *Stage 5* and *6* of the TetraPak pilot project. These stages and the pilot project evaluation in *Stage 7* are described the

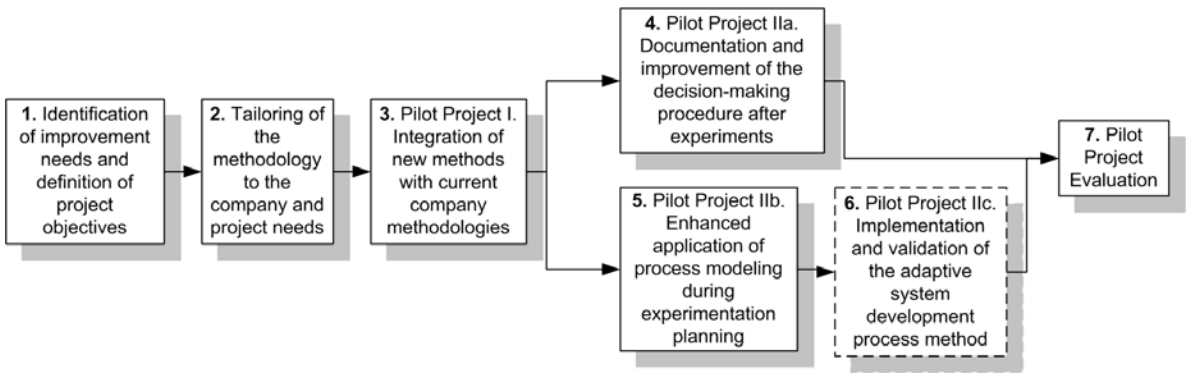


Figure H.1 Pilot project structure at TetraPak

Figure H.2 TetraPak packaging line

last part of the thesis.

H.3. TETRAPAK PILOT PROJECTS – OVERALL OBJECTIVES

TetraPak Carton Ambient, a business unit of TetraPak, is the world leader in the development and production of packaging systems for liquid food products that can be stored at room temperature (*Figure H.2*). TetraPak Carton Ambient operates in six countries with 2,100 employees. It is mainly situated in Lund, Sweden; and Modena, Italy; both important centers for SD and production. The case study was conducted at the TetraPak Carton Ambient S.p.A. facilities in Modena.

The flagship product of TetraPak is the food container package (*Figure H.3*). Different types of packages can differ with regard to the enclosed volume or shape. Furthermore, the same package can be produced with different packaging materials (there are today nearly one hundred different specifications available) depending on the needs of the customer and the peculiarities of the packed product. A product is usually defined by properly combining different configuration variables such as: filling machine system type, package volume, package type, package shape, opening device type (if applicable), cooling system, headspace unit (if applicable), filling machine working frequency, *etc.*

Due to the specific conditions in the food packaging industry, the production of the package and the filling of the nutritional product into it are done at the same place. Consequently, the final product quality (*i.e.,* the quality of the packaged food) highly depends on the quality of the overall packaging system. To assure the highest quality products, TetraPak offers packaging solutions covering the whole infrastructure of the packaging process including the machinery, such as the packaging lines, filling machines, and downstream equipment, as well as the package including packaging materials, opening systems, and sales and distribution solutions. *Figure H.2* shows a TetraPak filling machine developed and produced in Modena.

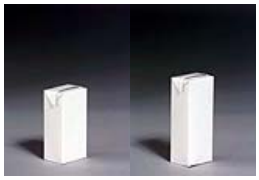


Figure H.3 Two TetraPak packages with different volumes

The most important customer requirement on food containers is *human health protection*. Thus, the main properties of the product (*i.e.,* the package) that are relevant for the definition of a proper SD process are the ones involved in the creation and maintenance of packaged product sterility and in the retention of sensory and nutritional properties during shelf life. The *sterility* of a product can be defined as follows: *a product is sterile if it is free from all microorganisms*. Since microorganisms multiply in accordance with the composition of the product, it is often not necessary to achieve complete sterility in order to avoid spoilage.

The more general term above can then be substituted by the term of *commercially sterile product*: a product free from pathogens, free from toxins, and free from microorganisms which are capable of multiplying inside the food during the intended shelf life in non refrigerated conditions. The quality limiting factors for commercially sterile products are then chemical, not microbiological. A definition of aseptic packaging can accordingly be given as a packaging process where microorganisms are prevented from entering the package during and after packaging. This definition applies irrespectively of the packed product type, so aseptic packaging is possible whether the product is sterile or not. Technically speaking this definition implies that the packaging material, the product to be packed, and the surroundings of the area where the product is packed need to be sterilized. Furthermore, the process has to ensure that the product is transferred to a package maintaining aseptic conditions, and that the package is tightly sealed after filling. Since every step in the process affects the overall package sterility, which in turn is related to consumers' health safety, the complete packaging solution involved in the process is safety critical.

The main quality requirements on the packaged food include the areas of microbiological safety, safety from non-microbiological contaminants, commercial sterility, minimal organoleptic product change during packaging, minimal interaction between food and packaging material, and sufficient barrier properties of the package towards external chemical agents. These basic quality requirements apply globally to all the developed products and fundamentally affect the SD methodology applied at TetraPak.

The main characteristics of a typical SD project in packaging system development are shown in *Figure H.4* using general SD lifecycle phases and the standard V-model. The first two phases, where the project is defined, the customer's needs are elicited, and the system architecture and defined specifications are usually performed at the system developers' site. Then, subsystem development and fabrication is conducted in collaboration with various suppliers. While the suppliers play a critical role in the development of the packaging system, the success of the collaboration with the suppliers has major impacts on the quality of the final product and the duration of the SD project. Suppliers require unambiguous and complete specifications to deliver

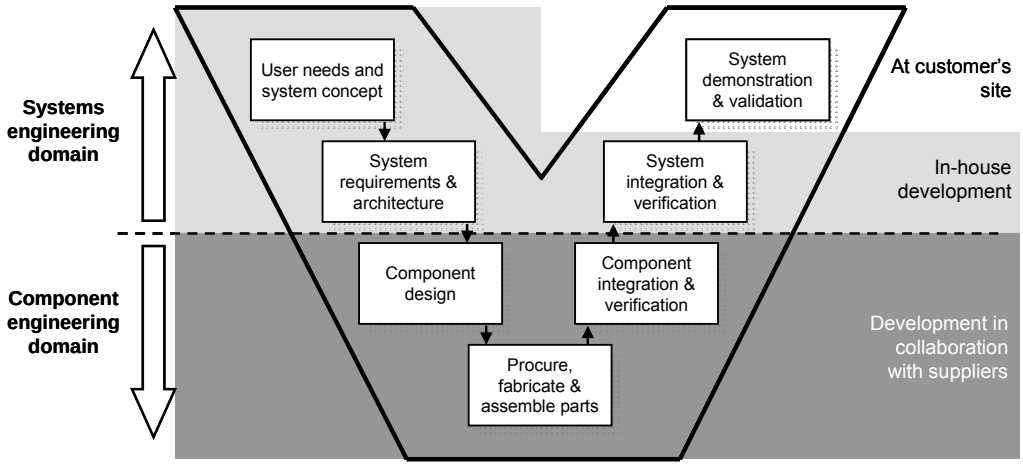


Figure H.4 Typical lifecycle of food packaging system development projects

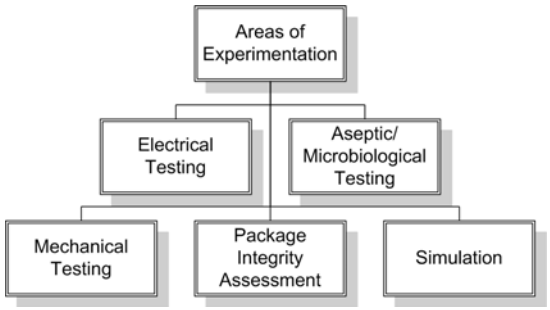


Figure H.5 Experimentation areas at TetraPak

a subsystem desired by the customer. As subsystem verification is done at the suppliers' site, a well-established quality control system and enhanced communication with the suppliers are essential for the final project success.

Another interesting key characteristic of SD projects in the food packaging systems industry is that system verification and validation are spatially separated. Whereas system verification can be performed in-house at the system developer's plant, the final system validation and qualification has to be done at the customers' site, sometimes at a remote location on the other side of the planet. The reason for this is that the whole functionality and quality of the developed technology or system can only be validated in the intended environment. This on-site validation phase is quite expensive, it is critical for the customer satisfaction, and delays in this part of the project can have huge financial consequences. Hence, *a main goal of the TetraPak technology and product development projects is to achieve the highest possible technical system maturity until the end of system verification to ensure a short and smooth system validation and qualification phase.*

The TetraPak SD lifecycle is simpler than the one depicted in *Figure H.4* consisting of only four main lifecycle phases: *Project Definition*, *Concept Development*, *Prototype Development*, and *Product Qualification*. Certainly, the structure of the SD projects at TetraPak follows the same logic, but the SD tasks and milestones have a slightly different structure from the generic lifecycle depicted in *Figure H.4*.

Figure H.5 shows that the bulk of V&V activities and experiments done during a typical TetraPak project concerns physical testing. This means that several testing rigs are designed and constructed during the SD, which increases the complexity of the projects. Additionally, the majority of the V&V activities are done on physical prototypes produced at a late part of the SD. Such domination of physical testing can raise programmatic risk for the project. That is, if major design deficiencies are found in the physical prototypes, the rework effort to correct these failures can concern the early stages of the SD project, which makes iterations quite expensive and time-consuming. An effective strategy to reduce technical risk early in the project is the frontloading of computer modeling and simulation in the SD. This way, design failures can be found early and corrected at low cost before proceeding to the costly physical experiments.

It is important to note though that in a conservative company culture with a lower rate of overall product change and innovation for new products, the development and application of computer models or rapid prototypes can be time-consuming and ineffective. Furthermore, if industry regulations require a high amount of physical V&V, and the historic information from the V&V process is well recorded and easily accessible for the V&V teams, the impact of the frontloading of evolutionary product evaluation techniques (*e.g.,* simulation, digital mockups, rapid prototyping, *etc.*) can decrease substantially. Thus, *during project planning, a tradeoff has to be made between the frontloading of computer-based experimentation and traditional physical testing based on the cost-benefit ratio of the methods.*

H.3.1. TetraPak Project Goals in the SysTest Pilot Project

After the description of the special characteristics of TetraPak and the food packaging industry, this section describes the main goals of TetraPak in the SysTest pilot projects. These goals concern parts of the corporate strategy regarding the planning and control of experiments and V&V activities, and the improvement areas identified for the SysTest pilot project. Thus, the following project goals represent SD needs that can be fulfilled through the application of a systems engineering methodology for enhanced experimentation and testing. The general pilot project goals are [Hoppe, Lévárdy *et al.* 2004a]:

- Standard food packaging system development projects are usually *small or medium in size with low complexity*. Thus, existing knowledge, historical project information stored in databases, and best practices can and must be utilized during project planning. Process modeling is an adequate method to integrate and store project plans and measured information for the long term.

- TetraPak strives to *standardize products and projects* where tailoring requires only the adjustment of specific product properties during project planning. Other project characteristics must be derived from the historical data of previous projects.

- All products are human-health critical and thus, a set of highly procedural V&V activities must be applied in order to fulfill food production regulations. *These activities must be integrated with the new V&V methods developed during SysTest.*

- An effective method to control the key product properties during SD is essential for the satisfaction of the customer's needs and the human health safety regulations. Thus, *the implementation of a technical performance control method that associates SD process results with the key product requirements is required in the pilot project.*

- Physical testing, particularly in the intended environment, is very important, but it entails great expenditures and is a key cost driver of the V&V process and the project. Hence, *to achieve considerable reductions in V&V and experimentation cost, the amount of physical testing must be reduced.*

- *The implementation of simulation-based VVT activities in the early SD lifecycle phases is required* to generate early information on the design and support later requirements verification activities through physical testing. Further, the information generated during simulation and early experimentation must be utilized better *to improve the effect of learning in the early stages of SD.*

These high-level goals defined in *Stage 1* of the pilot project gave a good basis for selecting new methods and tools to strengthen the weak points of the company SD and V&V processes.

H.3.2. The Results of Tailoring

H.3.2.1. Improvement of Project Planning and Control

During *Stage 2*, the systems engineering VVT methodology was tailored to the company needs, new planning aspects, and V&V methods were introduced at TetraPak. One new, implemented technique was the systematic application of systems engineering measures for the planning and tracking of system technical performance and the estimation of technical performance risk in the project. The method for parametric technical performance tracking could be integrated with the existing *TQM*- and *Six Sigma*-based quality management system of TetraPak to support the process and product quality control—a main element of both

methodologies. Further, the collection of relevant data and best practices from previous projects during tailoring showed that all the required information for the establishment of a technical performance measurement system was available at the company.

Furthermore, a thorough analysis was conducted in the *tailoring stage* to determine the difficulty of introducing enhanced process modeling for integrated planning of SD and V&V processes. The results of the analysis showed that the necessary basis for process modeling is given at TetraPak, and the required input information for the effective application of a parameter-based process planning approach is also available.

H.3.2.2. Selection and Tailoring of Relevant V&V Methods

The VVT Methodology includes a large set of V&V methods and activities collected and developed during the SysTest project. Among these methods, each partner company could find relevant, new ones for their industry segment, and implement these after having communicated with SysTest partners who have experience in the application of the given methods.

The results of the screening and selection of V&V activities during tailoring is depicted in *Table H.1*. During tailoring, the V&V activities and methods were assigned to two main classes:

V&V Activities			
screening		**tailoring needs**	
already in application	satisfactory	applicable as described	**12**
		tailoring required	**1**
	improvement required	applicable as described	**31**
		tailoring required	**15**
new	required	applicable as described	**10**
		tailoring required	**2**
	recommended	applicable as described	**2**
		tailoring required	**1**
	nice to have	applicable as described	**3**
		tailoring required	**0**
	not applicable		**3**
Total			**80**

V&V Methods			
Screening		**tailoring needs**	
already in application		applicable as described	**7**
		tailoring required	**3**
new	Required	applicable as described	**3**
		tailoring required	**0**
	Recommended	applicable as described	**7**
		tailoring required	**3**
	nice to have	applicable as described	**0**
		tailoring required	**0**
	not applicable		**8**
Total			**21**

Table H.1 Tailoring needs on V&V activities and methods at TetraPak (adapted from [Hoppe, Lévárdy et al 2004a])

the ones that had been known and applied before at TetraPak, and the ones that were completely new, thus they had not been applied before in the company. In case of the activities already in use, it had to be decided if the current version of the V&V activity was satisfactory, or whether it could be improved on the basis of the activity description in the VVT Methodology. As depicted in *Table H.1*, 59 of 80 methods part of the VVT Methodology had already been applied at TetraPak in some form, and the description of 12 of them could be applied directly in the company V&V processes. Further 31 activities already in use could be enhanced according to the descriptions, and 16 had to be tailored to the food packaging industry environment.

The second group of V&V activities includes the ones that had not been applied before, but which were desired to be used in the future. Twelve of the remaining 21 V&V activities were considered to be very important for the improvement of the company V&V processes, three were classified as *"recommended"*, medium-term options, another three as *"nice to have"* and finally, three activities were *"not relevant"* at all for food packaging.

The lower part in *Table H.1* shows the same classification for V&V methods. Seven of the ten methods already in use were applicable as described in the VVT Methodology, and three had to be tailored to the TetraPak environment. Three of the remaining eleven new V&V methods were classified as *"required"* for the company V&V. Ten new methods were *"recommended"* for future application and eight were irrelevant for food packaging system development. According to the overall pilot project goals described above, most of the V&V methods identified as relevant for the TetraPak SD processes were parametric experimentation planning methods (*e.g.,* FMEA, DoE) or computer-based experimentation methods (*e.g.,* methods for enhanced computer modeling and simulation).

The classification of V&V activities and methods was done in collaboration with the core competence teams described in *Figure H.6*. Every activity performed during the development of the food packaging solution can be associated with a core competence in the organizational structure of TetraPak. That is, the core competencies are responsible for the development of a certain subsystem of the overall packaging system, and thus a certain part of the SD process. Thus, experts involved in these core competencies plan and perform the relevant V&V activities in the SD process. These experts were the main stakeholders of tailoring, and thus they were involved in the tailoring process to support the deliberate methodology adaptation with their inputs.

Once a decision had been made on the applicability and implementation of the V&V

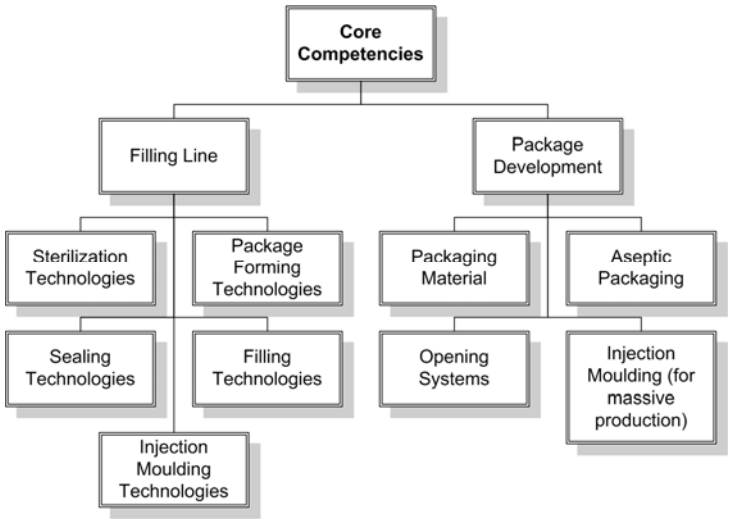


Figure H.6 Core competencies at TetraPak

methods and activities at the company level, the required tailoring of the VVT Methodology was accomplished, and the descriptions of the V&V activities and methods were incorporated in the company knowledge base, which included all the necessary information to execute a TetraPak SD project.

H.4. PILOT PROJECT I – INTEGRATION OF NEW METHODS WITH CURRENT COMPANY METHODOLOGIES

Stage 3 of the TetraPak pilot project structure refers to the integration of new methods with the current SD and V&V methodologies. Even though this stage included the introduction and improvement of various V&V and planning methods, this section only concerns the following planning aspects: *the definition of a systems engineering measurement system and the assignment of milestone criteria and impact functions to the main milestones of the project*. The reasons for selecting these aspects are twofold: (1) the implementation and improvement of parameter-based project control and decision-making was one main goal of TetraPak, and (2) these aspects have high relevance for the methods developed in this thesis.

The first pilot project at TetraPak focused on the upgrade of an existing packaging line. The improvement of the capabilities of the actual packaging line was characterized by the following three main critical technical system objectives [Hoppe, Lévárdy *et al.* 2004b]:

- The first main goal of the pilot project was to *exceed the limits of the actual technology in terms of flexibility*. Flexibility means that the same packaging line and in particular its core, the filling machine, can produce packages characterized by different shapes, volumes, and opening type.

- The second goal was to *drastically reduce the time to change production for the complete packaging line*. The success of the project depended on the ability of the same packaging line to produce different packages characterized by TetraPak quality, and to improve the *efficiency* significantly by minimizing the time for change of production.

- The third goal was to *minimize known weaknesses of the packages to be introduced*. Market studies had shown that a new appearance of the produced packages could increase the market share of the customer company. New appearance concerned the type of the packaging material used and/or the form of the package.

As these three main project goals show, the pilot project at TetraPak was related to the upgrade and redesign of an existing packaging line by the introduction of packages with new volumes, and by increasing the efficiency of the change between the productions of similar volumes using the *QuickChange*^{tm} technology developed by TetraPak. The possibility to introduce new volumes and switch quickly between the productions of two different packages improves the flexibility of the packaging solution and increases the overall lifecycle value of the system. As the product design and spectrum of the packaged food producing companies vary on a regular basis, flexibility in package size, form, material, and opening are key characteristics of an appealing solution, and guarantees long-term customer satisfaction. Since the retail of packaging material is one of the core businesses of TetraPak, the flexible production of various packages on the same machine is a deliberate, long-term strategic decision to increase the profitability of the company.

The identification and communication of the critical project goals are essential before starting to plan, because these goals provide the main direction for the project. In case of TetraPak, the pilot project was classified as a *system upgrade project*, *i.e.,* the system had already been familiar to the project team, and the identified system weaknesses specified clear project goals and improvement areas. Since the introduction of new planning and V&V methods alone had

already entailed a high risk in the project, it was a deliberate decision to select a project with a low rate of innovation and high degree of existing project knowledge.

Based on the above system objectives, the planning team identified key requirement areas concerning the technical performance of the packaging system. The following list includes the identified critical requirement areas:

- Flexibility / adaptability of the packaging line
- Food safety
- Container functionality and protection of the food product
- Public health
- Maintenance of the organoleptic properties and preservation within shelf-life
- Environmental performance
- Opening / closing performances
- Efficiency
    - of the packaging solution
    - of the packaging line
    - of the subgroups
- Market appeal: convenient and attractive

These key characteristics of product performance have key effects on customer satisfaction, thus these requirements had to be satisfied by the final product in any case. Furthermore, these key product characteristics specified the organizational structure of the project V&V. That is, the requirements indicate which core competencies had to participate in the project and to what extent.

Based on the defined key performance requirements, the planning team set up a systems engineering measurement system to (1) associate product requirements with SD activity outputs,

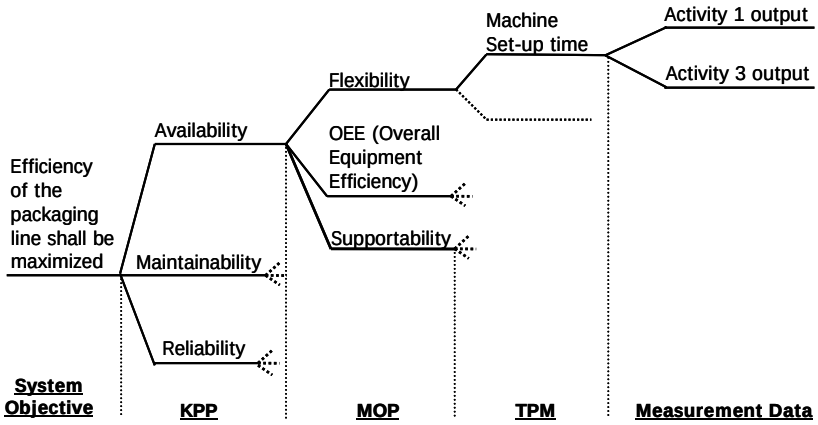


Figure H.7 Systems engineering measurement system defined at TetraPak in the first pilot project

and (2) support project control at every hierarchy level of the project. One branch of the systems engineering measurement system is depicted in *Figure H.7*, where the system objective *"efficiency"* is broken down into three KPPs (*"availability"*, *"maintainability"*, and *"reliability"*) and further lower-level measures. The KPP *"availability"* depends on three system-level performance parameters (or MOPs): *"flexibility"*, *"overall equipment efficiency (OEE)"*, and *"supportability"*. Finally, one TPM is depicted on the lowest level contributing to the system flexibility: *"machine set-up time"*. This TPM can be directly measured in the SD process, or it can be derived from experimentation results to acquire information on the status of the SD.

Once the planning team had set up the systems engineering measurement system, the measures were assigned to the various milestones in the SD project to establish quantitative milestone criteria for the project. That is, the experts at TetraPak collected historical data from previous projects to define *target values* for each parameter at each project milestone. These targets together represent the required overall design maturity at a certain milestone that has to be achieved to guarantee the final project success.

Based on the kinds of the measures, the target profiles can have different forms. For some measures (*e.g.,* sterility at TetraPak) even a slight difference from the requirement value means high risk for the project. Thus, the target profiles of such measures are usually *constant functions* representing the final requirement value. On the contrary, other measures contributing to the technical performance of the system evolve together with the design. The target profile of such a measure (*i.e.,* machine set-up time) is depicted in *Figure H.8*.

The target profile of *"machine set-up time"* in *Figure H.8* depicts a basic trait of SD upgrade projects. The starting point of the profile (*1*) symbolizes the state-of-the-art performance value of the machine set-up time at the outset of the project. The goal of the *Concept Development* stage is to develop a design that fulfills the requirement for improved performance (*i.e.,* reduced machine set-up time). Thus, the target value for this phase (*2*) is the requirement value, and the phase-internal targets (*e.g.,* targets for design reviews) show a decreasing trend for the TPM value.

The third, *Prototype Development* stage has an interesting characteristic, *i.e.,* the TPM values are rather high in the beginning of the phase (*3*), meaning deficient technical performance for the design. The reason for these phenomena is the switch from virtual or paper-based design during *Concept Development* to the physical implementation of the design in hardware components during *Prototype Development*. That is, the design that works as a computer model has to be implemented in functioning hardware. The deficient early design performance indicates that the results of the first physical experiments are usually lower than the final requirements. However, the trend shows that the target for the final prototype is again the requirement value (*4*), which can be achieved through iterative experimentation.

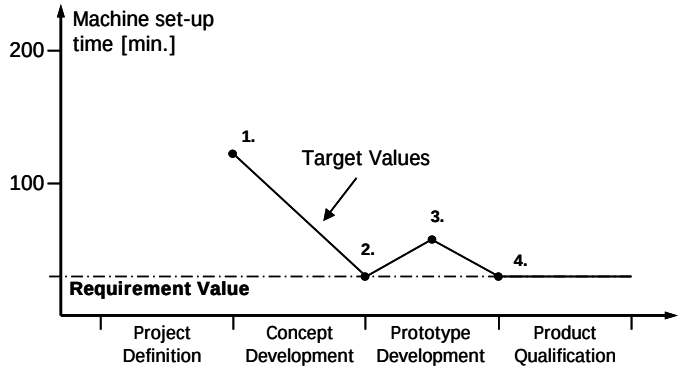


Figure H.8 Target profile for machine set-up time as a TPM at TetraPak (adapted from [Hoppe, Lévárdy et al 2004b])

After having established the target profiles for the project, the planning team at TetraPak defined impact functions for each milestone criterion represented by the systems engineering measures in *Figure H.7*. Impact functions support decision-making for the milestones by converting the actual project performance data into managerial measures showing the estimated deviations from the expected company profit achievable through the project.

The functions in *Figure H.9* depict the impacts of inadequate technical performance for the four major milestones of the TetraPak standard SD lifecycle. The *x*-axis in each function shows the final requirement value for *"machine set-up time"* as a TPM and two other reference values, *i.e.,* the set-up time for similar, conventional machines (*i.e.,* before the upgrade) and the set-up time for the whole packaging line.

The first value is an important reference point for the developers, because the new design had to improve technical performance attributes compared to the similar, conventional machines, otherwise the investments in this design aspect do not generate any profit. The second value represents the set-up time of the whole production line that increases in case the set-up time of a single machine is too long. This situation had to be prevented in any case.

The values of the stepwise impact functions on the *y*-axis are assigned to these three representative TPM values. The impact values during *Project Definition* are low due to the high number of open design alternatives. Developers at TetraPak attempt to keep design options open

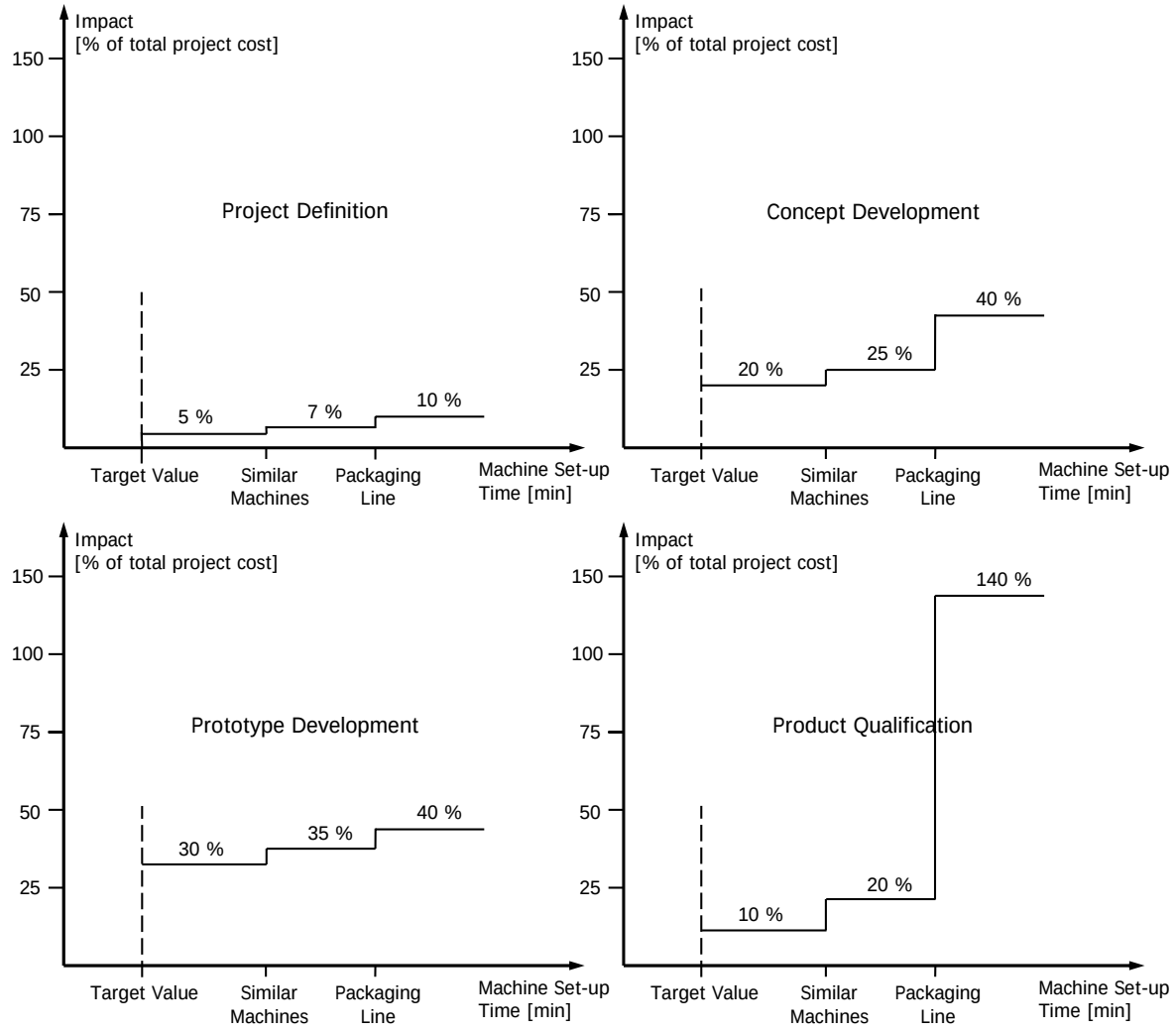


Figure H.9 Impact functions for machine set-up time as TPM at TetraPak (adapted from [Hoppe, Lévárdy et al. 2004b])

Figure H.10 Generic transformation process in the food packaging industry

as long as possible, and iteratively improve performance to prevent suboptimal design solutions. Thus, a lower performance value at this point of the project does not incorporate high risk.

The second chart shows that the impacts of low performance increase as the project proceeds towards the second milestone. The values are higher here, because existing technologies might constrain system performance, and the development or insertion of new technologies are always costly endeavors. The impacts of the third chart referring to the consequences of lower performance at the end of *Prototype Development* are rather high due to the increased cost of rework on physical prototypes in this phase. Major rework actions in this phase can boost the project budget and schedule significantly.

The last chart depicts the financial effects of lower technical performance during *Product Qualification*. As this phase is conducted at the customer's site, only small variances in system performance are acceptable that can be repaired with small adjustments on the machine. Extremely poor performance at this stage of the SD has immense effects on the project, because in this case entire SD phases have to be repeated. *Such technical problems are extremely rare at TetraPak.*

In the next section of this chapter, *Stage 4* of the pilot project structure is discussed, where the decision and control procedure for adaptive SD projects proposed in the previous chapter is validated.

H.5. PILOT PROJECT IIA – VALIDATION OF THE DECISION AND CONTROL PROCEDURE FOR ADAPTIVE SD PROJECTS

The following section describes *Pilot Project IIa* from the TetraPak pilot project structure. In this pilot project, the developers at TetraPak had already been familiar with parameter based SD project planning and control, and process modeling. Note that the same pilot project was applied for the validation of the *adaptive SD project planning methods* described in the following chapters. Thus, this section deals only with adaptive project *control* and *decision-making*.

The project selected at TetraPak to evaluate and validate the procedure of adaptive project control and the decision framework for system adaptation was an SD project from the liquid food industry environment. The selected project dealt with a typical industrial transformation process, where semi-manufactured goods fed one generic transformation process affected by external noises and a finished product was released (*Figure H.10*).

The critical project objectives were to evaluate the correlated effects of raw materials, the transformation process, and the intrinsic variability of the noises (*i.e.,* variability in product and process characteristics) versus the nominal geometrical dimensions, the appearance defectiveness, and the nominal tare weight of the container. The project goal was to reduce the variability (and thus increase stability) of the parameter values and increase confidence in the characteristics of the final product.

The high-level project goals represented by the KPPs in the systems engineering measurement system were:

- Container tare weight
- Container appearance defects
- Container geometrical dimensions

The following sections provide definitions for these measures.

H.5.1. Container Tare Weight

Due to the extremely high number of produced containers, tare weight is a key cost driver in the food packaging industry. Hence, the variability of this parameter is a main risk source concerning the fulfillment of the final product requirements. The main causes for variability in the product parameters originate from the variability of the characteristics of the raw material and the filling machine.

In case of the development of a new container type for liquid food, the project goal was to reduce the weight of the container. That is, a new type of light plastic bottle had to be developed with a capacity of 0.5 liter. A major project goal was to reduce the tare weight of the new bottle from the existing 27 to 21 grams. The variance of the container tare weight using existing material and technology were 18-27 grams, which meant two options for the project: (1) improve existing raw material and technology to reduce the variance of the produced containers to 18-21, or (2) develop new raw material or technology to fulfill the requirements.

As the tare weight of the bottle was a critical product requirement, the characteristics of the raw material and the material manufacturing technology were critical issues in the project.

H.5.2. Container Appearance Defects

Certainly, the lighter the container is, the more vulnerable it is, as well. Thus, it had to be guaranteed that the container could be produced, filled, packed, distributed, and sold without damages that affected the quality of the food in the container and the appearance of the container. Package appearance is a key quality parameter of the final customer, who does not buy deformed containers or containers without labels.

H.5.3. Container Geometrical Dimensions

Besides the effects on the container appearance, the geometric dimensions of the container have a major influence on the volume of the contained food (due to industry regulations) and the distribution and packaging of the container. As food containers are distributed in sales units (carry sleeve) and distribution units (*e.g.,* cardboard, wrap-around boxes, film, palletizing, *etc.*) containing 10+ containers, the tolerance of the geometrical dimensions of the containers have to be limited. Otherwise, it can happen that the containers do not fit on the standard size distribution units. This measure is a generic requirement on all produced container types and measured as the standard deviation of the dimensions of the tested samples.

H.5.4. Case Study Goals

The above-described *Pilot Project IIa* was selected as a case study to validate the feasibility of the proposed procedure of adaptive project control and the decision framework for system adaptation. The same project was also applied for the validation of planning methods for adaptive SD projects in *Pilot Project IIb* described in the next chapter. Thus, the goals of *Pilot Project IIa* related to this case study were:

- The definition of a hierarchical structure of system engineering measures for project monitoring.

- The assignment of these measures to decision points in the project to support decision-making.

- The development and evaluation of a decision framework that helps keep development risks under control and capture design opportunities using the system of measures.

Figure H.11 shows the methodological scheme of the project. Since TetraPak applies the Six Sigma quality management philosophy, the terms in *Figure H.11* are part of the standard Six Sigma terminology. Thus, for each term, the definition is provided from a well-known Six Sigma website [I-Six Sigma Online Website].

The goal of Six Sigma is to increase profits by eliminating variability, defects, and waste, which undermine customer loyalty [I-Six Sigma Online Website]. In the first step of the project schema, historical statistical data is used to determine the characteristics of the reference container volumes though *statistical process control (SPC)*.

SPC is the application of statistical methods to identify and control the special cause of variation in a process [I-Six Sigma Online Website]. This preliminary data is the main input to two further planning activities: (1) *"Design of Experiments (DoE)"*; and (2) *"repeatability study"*. During the two analysis activities, different concepts and configurations of the possible transformation process and container types are developed and assessed to determine which alternative is the most feasible one to achieve the project goals.

DoE is a structured, organized method for determining the relationship between factors (Xs) affecting a process and the output of that process (Y) [I-Six Sigma Online Website]. The outputs of the DoE analysis in the case study project are the weighted averages and confidence levels for the key characteristics of both the transformation process and container configurations. This information is then used to determine the acceptance criteria for the transformation and container configurations. These acceptance criteria can be unique or the same for each configuration. Using the defined acceptance criteria, the *process capabilities* are assessed to determine the quality of each configuration.

Process capability refers to the ability of a process to produce a defect-free product or service in a controlled manner of production or service environment. Various indicators are used, some address overall performance, some potential performance [I-Six Sigma Online Website]. This data is used together with the outputs of the repeatability study in the next step during experimentation and V&V planning supported by a process model.

Repeatability is the variation in measurements obtained when one person measures the same unit with the same measuring equipment [I-Six Sigma Online Website]. During the repeatability study, the contribution of the estimated variability of the tare weights of the transformation and container configurations to the overall dispersion of the process data is assessed. High dispersion of the overall data means risk for the final product quality, thus it has to be minimized.

The outputs of the process capability analysis and the repeatability studies feed into further experimentation and V&V planning steps, the determination of the hierarchical structure of decision support and project control measures, the definition of the decision structure and the definition of the SD process in the process modeling tool (see *Pilot Project IIb*). The output of this complex activity is the estimated cost, schedule, and technical performance risk incorporated in the planned development project for volumes.

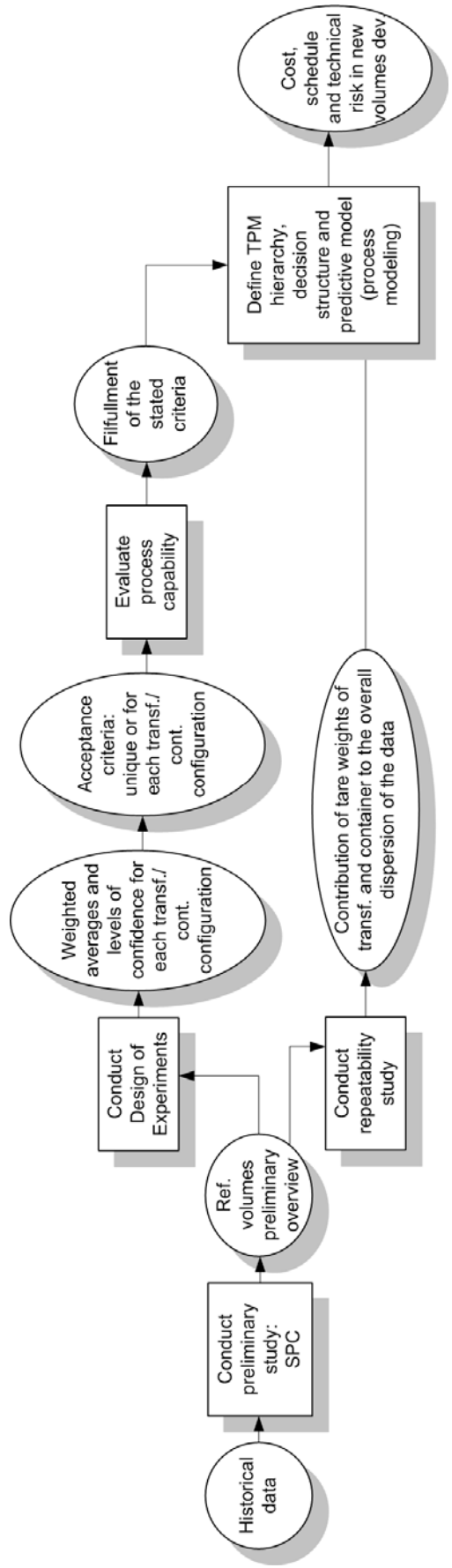


Figure H.11 Methodological Schema of the Case Study Project

The last activity in the methodological schema in *Figure H.11* is where the proposed, new decision support methods are implemented. In the scope of project planning, a *decision support*

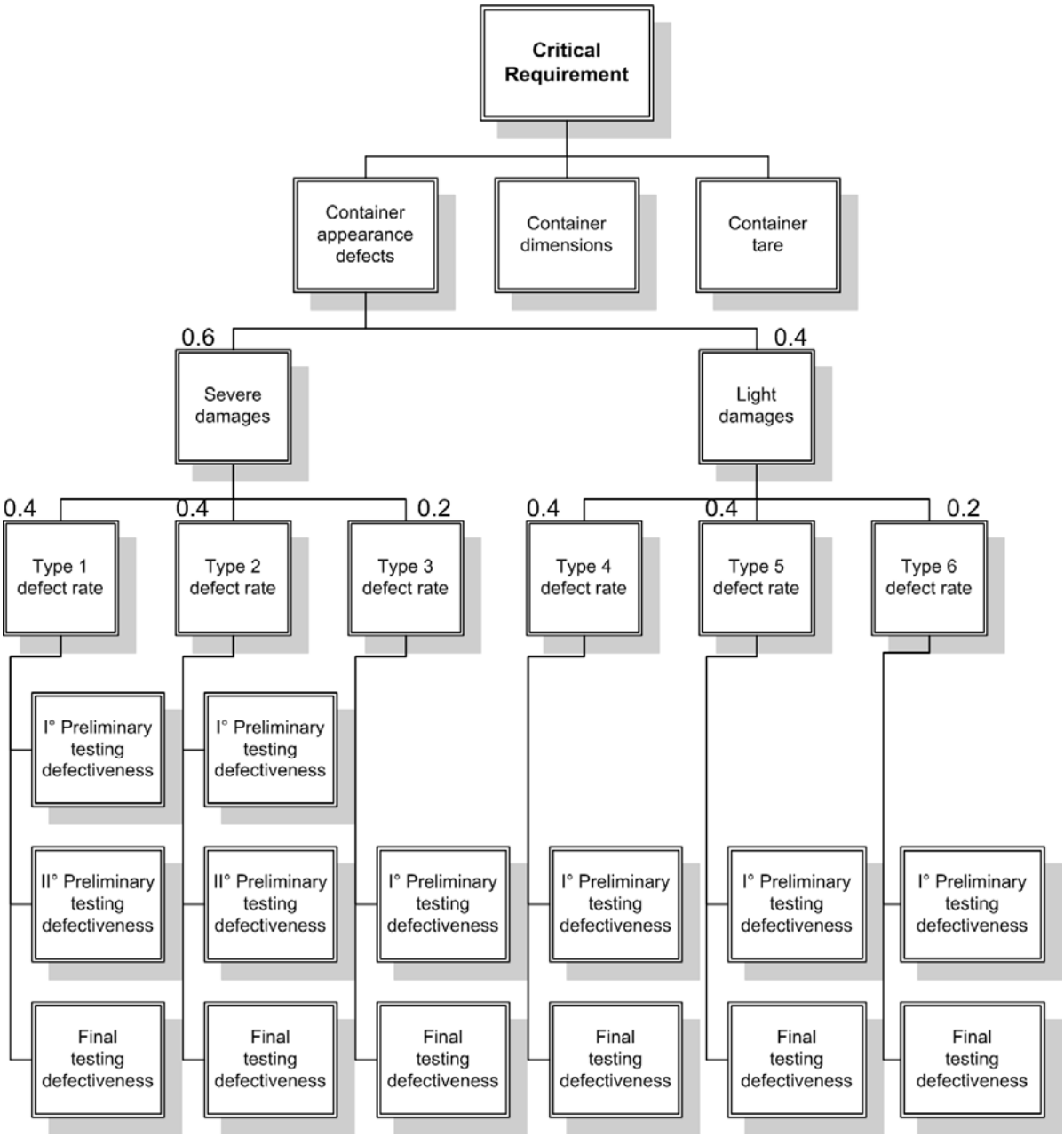


Figure H.12 First branch of the system of measures

framework is defined to support decisions in the course of the SD project with clear decision steps, success criteria, and comparable data from the experimentation cycles. The defined decision procedure is a tailored version of the previously defined *decision framework for system adaptation* in *Figure G.17*, and shall provide answers to the following questions in the case study raised by the TetraPak project management:

- Is it possible to forecast, on the base of the preliminary tests, if the technical risk concerning the performance targets is so high that a revision of the experimentation plans is indicated?

- Is there an optimal point of the change control structure that can be addressed in case of a failure in the later stages of the development?

- Which is the optimal decision point to be addressed for certain failures in V&V activity results?

H.5.5. Definition of the Systems Engineering Measures

A key component and information source of the decision framework applied in the case study project were the decision criteria represented by a hierarchical system of systems engineering measures. The definition of such a measurement system helps the developers collect and organize project data, connect the low-level raw activity data with success criteria derived from high-level requirements and project goals, and thus evaluate the actual state of the project.

Furthermore, this system of measures supports the assignment of experiments with product characteristics, and the definition of the estimated and documented effects of an activity on certain technical measures. Since TetraPak applies Six Sigma and TQM quality management philosophies in the SD projects, the developers were familiar with the Quality Function Deployment (QFD) method, where the relationships among customer's needs, product requirements, design parameters and technical performance measures, and SD activities (both design and V&V activities at TetraPak) are defined and documented. The Houses of Quality in QFD are key inputs for planning, decision-making and project control and thus the adjustment of the project plans and experimentation cycles to the actual project needs.

Using the experience from *Pilot Project I*, the development team at TetraPak defined a more detailed measurement system for *Pilot Project II*. The first branch of this structure is depicted in *Figure H.12*. The three main KPPs on the top of the structure that represent the main system objectives (*"container tare weight"*, *"container appearance defects"*, and *"container geometrical dimensions"*) are broken down into further technical measures (MOPs) and these further into TPMs. *Figure H.12*

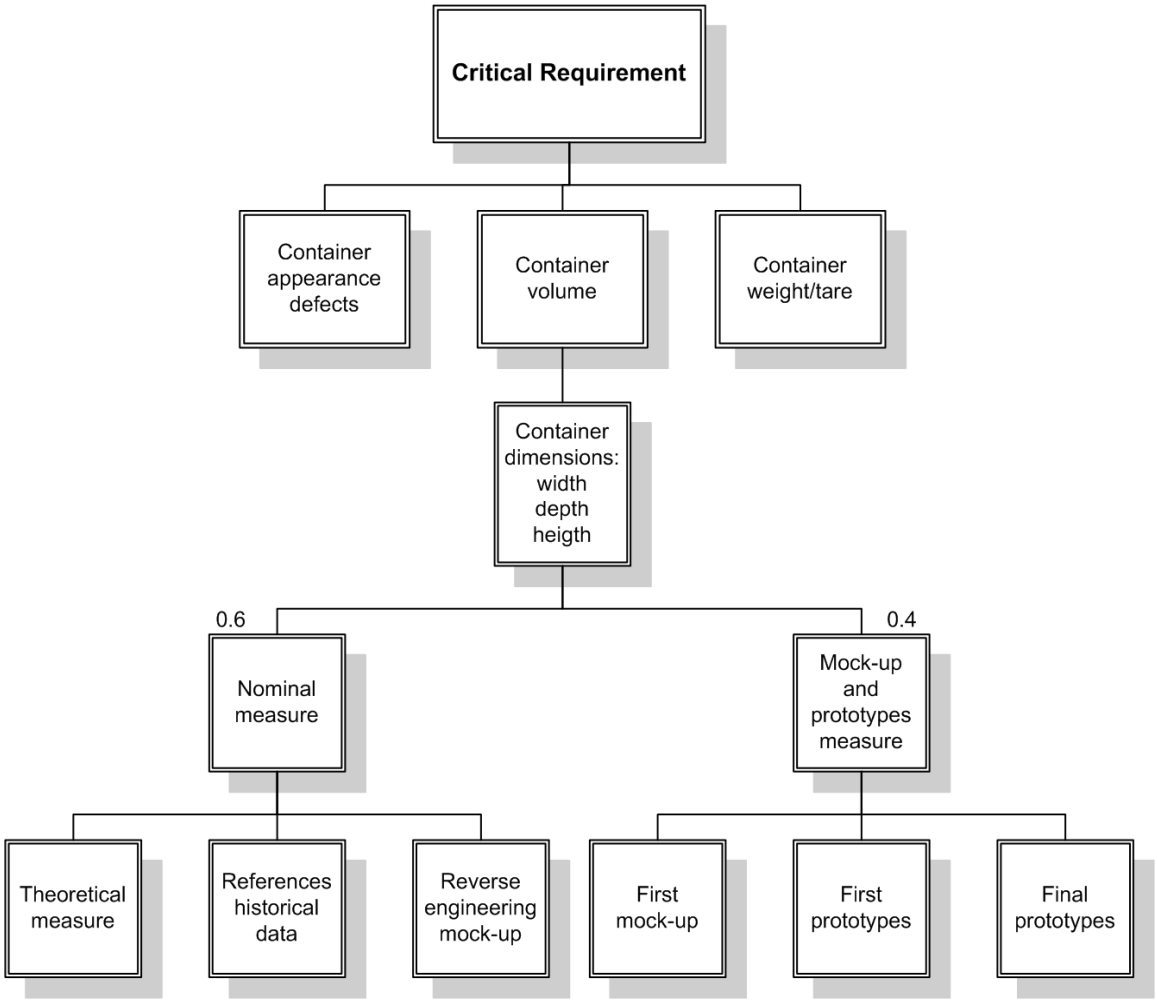


Figure H.13 Second branch of the system of measures

shows the hierarchy of measures for the KPP *"container appearance defects"*. MOPs for this measure are *"severe"* and *"light damages"* on the container. The likely damages in both categories had been documented and classified in previous company projects and thus the definition of TPMs could be done using existing company data. Further, the V&V activities suitable to verify the design and find the certain types of failures can be selected this way.

The lowest hierarchy level of the structure of the measures is the data to be provided by the experiments and mainly the V&V activities in the SD process throughout the project. Since at the end of each of the four main TetraPak SD lifecycle phases the same project goals were verified, the success criteria at each phase were also the same. It means that the same kind of information was needed from the key V&V activities in the SD process in each phase while evaluating the different representations of the design in the different phases (*e.g.,* requirements, function models, virtual prototypes of the design, physical subsystem and system prototypes, and final system assembly) to fulfill the same phase objectives. That is, the main deliverables at the end of each SD phase had the same content, but were different in maturity, in the level of detail, and in the type of representation.

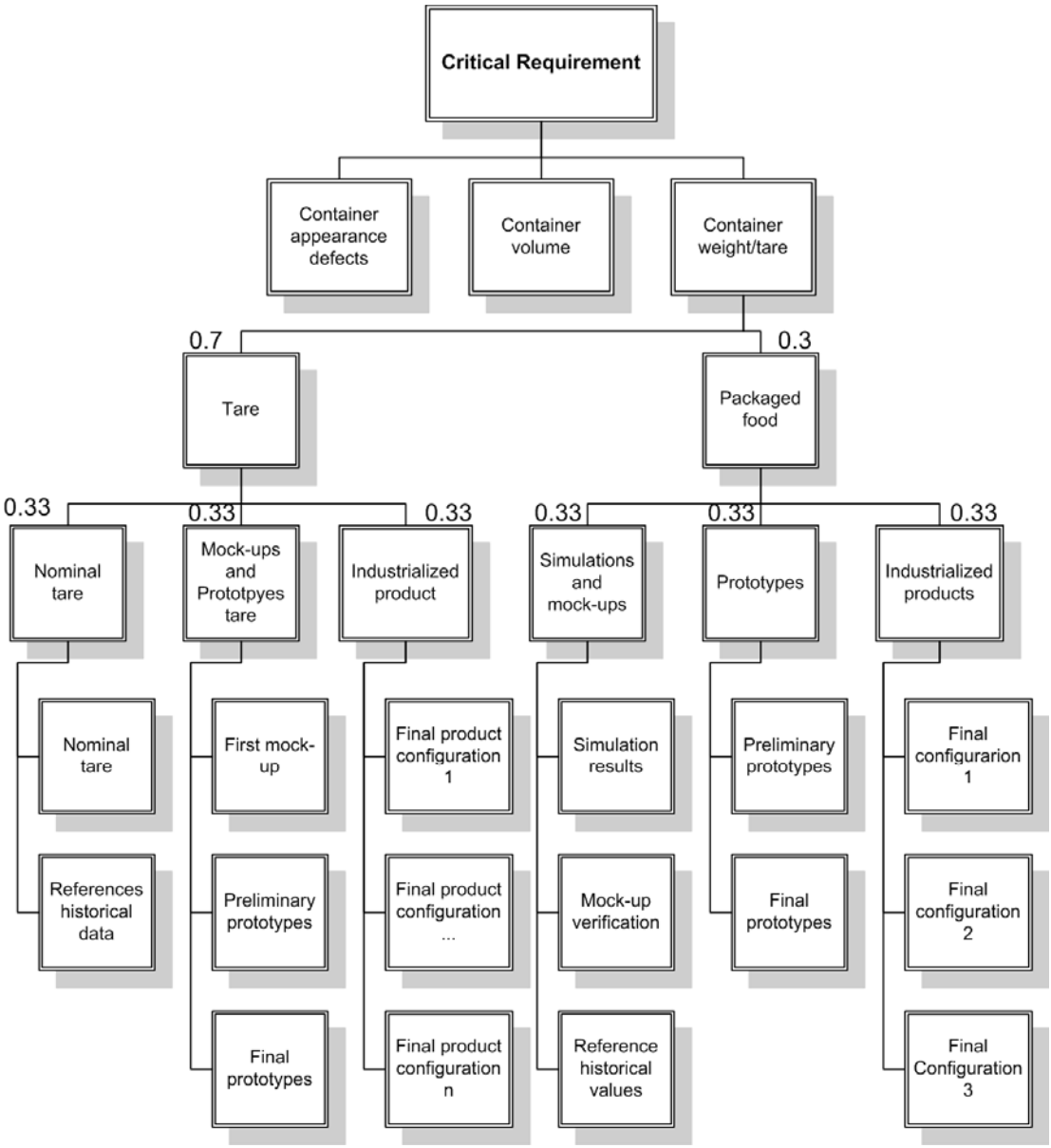


Figure H.14 Third branch of the system of measures

During project planning, the competencies had to make sure that in each phase, design and V&V activities with similar purpose, but different characteristics were planned to cover the same design aspects and ensure the required maturity of the system delivered by the phase. This is indicated in *Figure H.12*, where V&V activities with similar purpose, but growing fidelities are planned for each phase based on the technical measures and the activity data outputs. Thus, the measures support experimentation planning by providing an overview on the required and planned V&V steps to fulfill the goals of each lifecycle phase in the project.

The measures structure in *Figure H.13* shows the second branch of the overall system of measures. The logic is similar here to the first branch; however, the measures are different. The examination of the *"geometrical dimensions of the food containers"* produced by the packaging line requires experiments and V&V activities that are different from the verification of defects. The activities here involve computer modeling and the creation of digital and physical mock-ups that ensure that the dimensions of the packages are within the tight tolerances.

The third branch of the measures system is depicted in *Figure H.14* showing how the parameter *"container tare"* can be broken down into further measures and linked to V&V activity outputs. Two main aspects of the overall container weight are the *"pure tare weight"* and the *"weight of the food"* filled in the package. Both measures were verified in various representations of the product including preliminary calculations and simulation, digital mock-ups, preliminary and final prototypes, and industrialized products. The lowest level of the measures structure indicates the experiments and V&V activities required for the complete verification.

H.5.6. Definition of Milestone Criteria

Similarly to *Pilot Project I*, the step following the establishment of the measurement system in *Pilot Project II* was the definition of parameter based milestone criteria for each milestone. In order to support deliberate decision-making at the milestones, the milestone criteria included the target values for the relevant decision measures and the impact functions showing the consequences of the deficiencies of the actual values from the targets.

The target values for the programmatic parameters and one main technical MOP are depicted in *Figure H.15*. The profiles of planned project cost and duration show that the largest amount of the SD work was planned for *Prototype Development*, where the design was implemented in hardware components and integrated in a functioning system. Furthermore, as the third chart

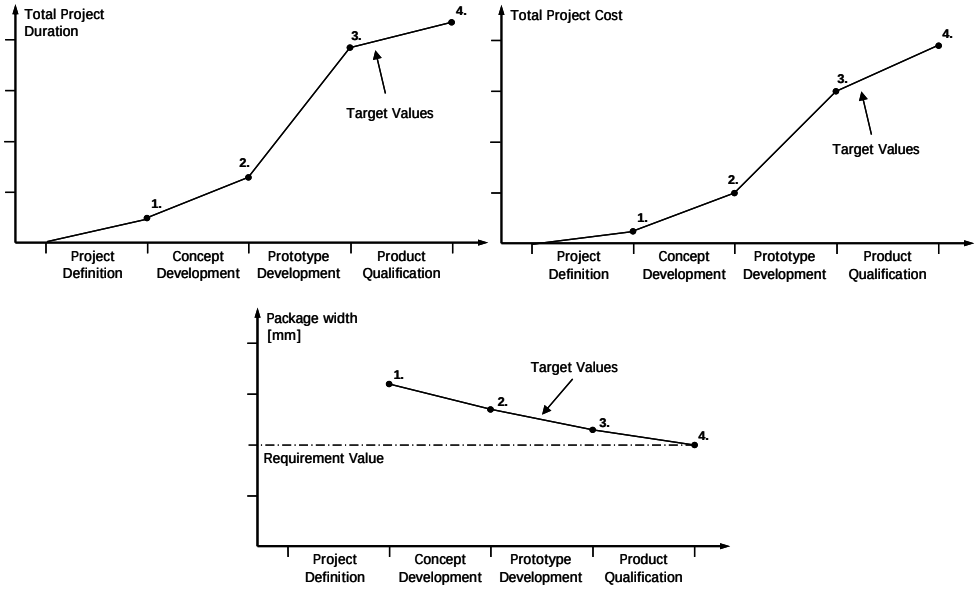


Figure H.15 Target profiles in Pilot Project II

indicates, the package dimension *"width"* evolved steadily throughout the SD process, and the milestone targets were planned to reach the final requirement value only at the end of the SD. This is usually the result of the iterative experimentation strategy at TetraPak, where developers attempt to maintain continuous performance evolution even across SD phases, instead of trying to reach the final values right in the beginning of the project. They could afford this in the pilot project, because they were familiar with the product and they effectively utilized documented generational project knowledge and experts' experiences. Further, the separated technology development projects delivered the SD with mature technologies that reduced technology risk. That is, the knowledge-driven SD system at TetraPak accommodates uncertainty and allows timely decisions on the key design aspects.

The impact functions in *Figure H.16* and *Figure H.17* depict another characteristic of the TetraPak pilot project. Considering the forms of the impact functions, the results of an SD phase can be assigned to three basic categories. In all three cases, the impacts refer to the cost of corrective actions to meet the phase targets and reduced profit due to later delivery or higher SD cost.

The first case (indicated by *1.* in *Figure H.16* and *Figure H.17*) is when the actual performance met the target values, the milestone criteria was achieved without concerns, and thus the phase goals were fulfilled. This means *green light* for the project performance aspect.

In the second case (*2.*), between the target value and second inflection point of the impact functions, the milestone criteria were not fulfilled, but the actual values are close to the target. This case can have twofold consequences based on the values of the parameters in the three quality areas (*i.e.,* project cost, schedule and product technical performance). In case of technical performance deficiencies, further work in the design area has to be done in either the actual or

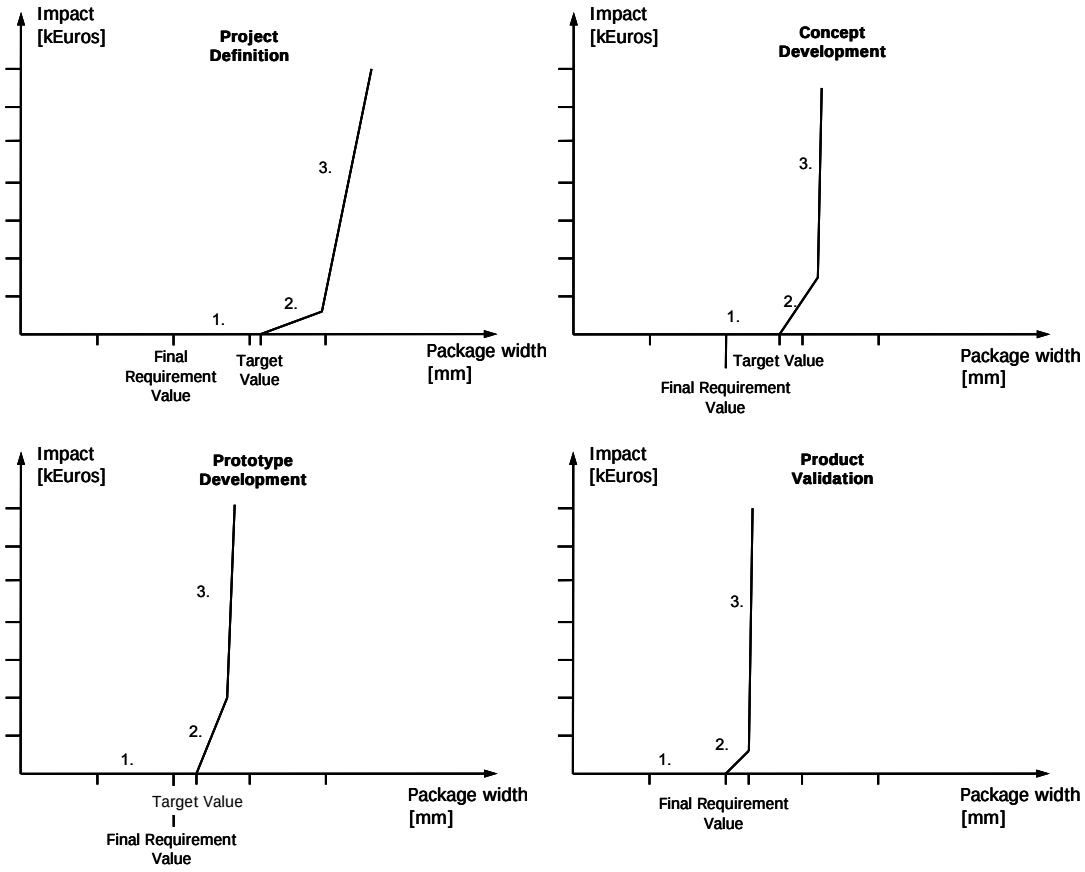


Figure H.16 Impact functions for package width in Pilot Project II

the next phase. If the programmatic aspects allow it, the failures can be corrected in the actual phase through rework. However, if the programmatic aspects are also in a critical state, the technical performance areas move to the focus of the project, and extra resources in the next phase are allocated to improve the inadequate performance areas. Thus, the first parts of the impact functions represent the *yellow light* areas of project performance.

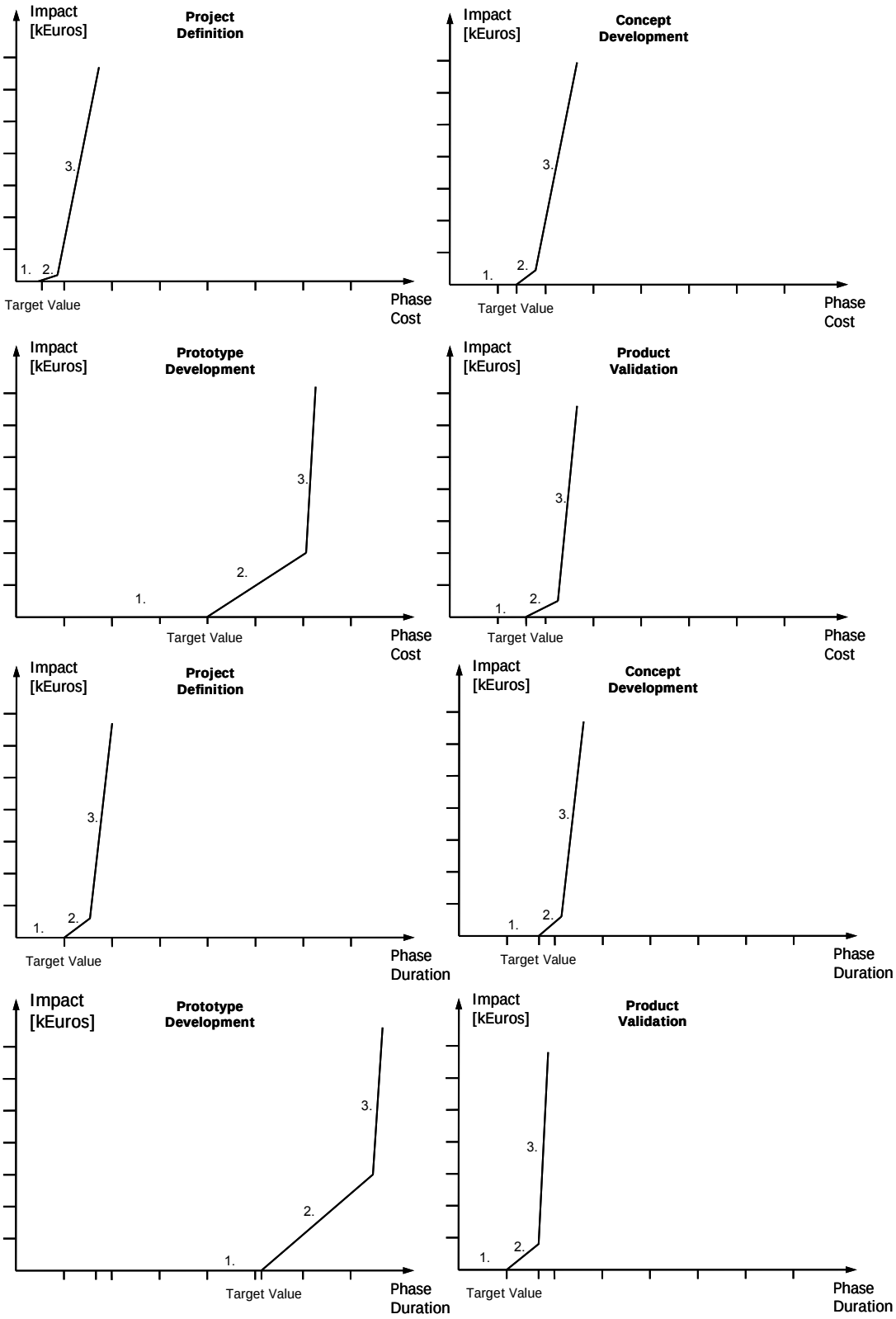


Figure H.17 Impact functions for project cost and duration in Pilot Project II

The third part of the impact functions with a high gradient (*3.*) refers to a rather undesired situation in SD, meaning that one (or more) project performance area is (are) at high risk. If the TPMs are in *red*, this means that some basic failures were included in the design that could not be caught in the previous phases. Red light in a technical performance area usually means the repetition of complete phases to reduce the possible loss of the company reputation towards the customers. In such cases, the only goal is to deliver the desired product with the required performance at any price to keep the customer. Such situations are quite rare at TetraPak.

H.5.7. TetraPak SD Project Structure and Logic

The structure of V&V activities in the measures system has already indicated the logic of TetraPak SD projects. However, it is important to take a deeper look into it and understand this logic. *Figure H.18* shows the four standard TetraPak SD lifecycle phases. In the first two phases, the project is defined, and then the design of the developed system is conducted on system and subsystem level. Due to the previously described key requirements of the case study project, effective cooperation was required between raw material manufacturing and container production in the pilot project. Raw material manufacturing had a main contribution to the quality of the final product, since the characteristics of the raw material realized during the material manufacturing process significantly affected the fulfillment of the final product requirements. Thus, one of the key verification goals was to evaluate the different types of raw materials suitable for the project and provide reliable information for the decision on the best material for the project.

During system design, usually different design concepts are developed and evaluated in parallel against the requirements represented by the defined assessment criteria. In the case study, the design concepts developed during *Concept Development* included configurations of different types of raw material and transformation processes. On the one hand, the possibilities to improve existing types of raw material were to be evaluated, and their behavior with existing or improved transformation processes was planned to be assessed. On the other hand, the analysis of the development of new types of raw material was planned to fulfill the improved product requirements. The assessment of these new types of raw material in the transformation process environment was also planned.

During *Concept Development,* the main task was to generate and evaluate various design concepts and select the best one(s). At the end of the second phase, at *Milestone 2* (MS2), a key decision was to be made regarding the design of the system. At MS2, the project team makes a commitment towards the stakeholders to realize the developed design in a system that satisfies the customer's needs. Usually at this point of time only one design concept exists, but in some cases, where more than one design is still feasible, the decision can be kept open and the team can continue working with two design alternatives. However, the SD team has to be sure that the design(s) are capable of fulfilling the defined customer's needs and product requirements.

Hence, the most important task in the first part of the SD project ending with MS2 is to

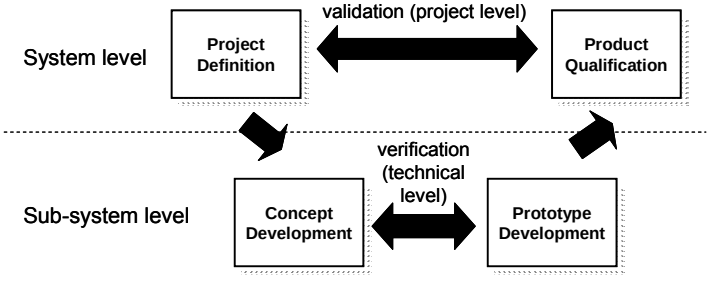


Figure H.18 Generic TetraPak SD lifecycle phases

Phases	Experimentation	Measures used	Phase focus on measures	Usual sample sizes at tests	Requirements status at phase end
Project Definition		Levels 1-2	Level 1	-	defined
Concept Development	Modeling & simulation	Levels 1-3	Level 2	20	frozen
Prototype Development	Physical testing	Levels 1-4	Level 2	300	verified
Product Qualification	Physical system qualification	Levels 1-4	Level 1	3x3000	qualified

Table H.2 Relationship between lifecycle phases, experimentation modes, and measures

define realistic requirements that satisfy the customer's needs, and to develop a robust design that is capable of fulfilling the requirements. As *Table H.2* shows, the requirements are frozen at the end of the *Concept Development* phase. Thus, a key V&V goal for the first part of the project is to verify that it is possible to realize the defined requirements in a feasible design, and validate that the defined requirements fulfill the needs of the customer. In case there is residual technical risk in the project after the commitment at MS2, it can lead to quite costly design changes that could jeopardize the outcome of the whole project.

Therefore, the important strategy aspects of this case study project were (1) to reduce uncertainty in the first part of the SD to the desired level through identifying suitable V&V activities; (2) organize these activities in effective experimentation cycles; (3) define correct success criteria for the experiments; and (4) implement a decision support framework that highlights the major design problems and helps define a way to correct them.

One main trait of experimentation cycles in similar TetraPak SD projects is that the sample sizes for the prototyping activities gradually increase in each phase. Thus, during preliminary testing during *Concept Development*, a rather small sample size of 20 is taken. At TetraPak, testing is often considered an *exam that must be passed,* and in case of a failure (*e.g.,* 1 incorrect result out of 20), usually small adjustments are made on the system under experimentation, and the test is repeated. Due to the small sample sizes, it happens often that retesting provides a positive result (0/20 defects) and the SD project continues after a successful test.

However, a main concern at TetraPak is that a small adjustment on the system during experimentation often does not correct the design problem, and the test results are only good due to the low test fidelity and the small sample size. In case, a design failure caused the errors during preliminary testing, the V&V activities in the experiments during *Prototype Development* might find the failure, However, in case of an insignificant number of defects during testing (*e.g.*, 2/300) it might happen that the failures disappear again after small adjustments on the tested system. Since experimentation and V&V are considered as examinations that the tested system (and the test team) has to pass, adjustments are made until the system passes the test (0/300). The problem becomes quite significant if the design failure remains in the product after the adjustments, and it is only found in the *Product Qualification* phase, where 3x3000 samples are taken. A failure found in this part of the project can have fatal consequences for the project.

Hence, a main goal in the case study project was to increase the effectiveness of learning during early V&V and foster the better analysis of V&V activity results. Due to the immense consequences of late failures, the effects of failures found during preliminary testing had to be evaluated and the causes thoroughly analyzed. Even if this analysis costs extra time and money, the effort invested in early SD is always cheaper than rework at the end. Furthermore, the success

criteria of the experiments during *Concept Development* had to be set more strictly, so that the failures found became the required attention, and the correction actions were not restricted to small adjustments, but covered all relevant design aspects. These aspects of the case study are discussed in detail in the description of *Pilot Project IIb* in a later section.

Now, after the description of the case study and definition of the systems engineering measures, the decision support framework to evaluate the data provided by the V&V activities in the experimentation cycles and decide on the further track of the project is described in the next section.

H.5.8. Definition of a Generic Decision Support Framework

The previous section came to the conclusion that experimentations drive the success of SD projects at TetraPak, and the results of these experimentations can be linked to technical and managerial measures to control the achievements of the projects and compare these to the plans.

The next step of the case study was to analyze the company decision-making procedures and determine how a parameter-based decision support system can be implemented at TetraPak. The analysis had three main findings:

- Though at TetraPak V&V activities delivered vital inputs for decisions at different hierarchy levels of the project, *a standard generic decision support framework did not exist that could be tailored vertically to the various hierarchy levels of decision-making and horizontally to the various phases of the evolving SD process.*

- Experimentation strategy and V&V results had crucial impacts on product and process changes and thus rework actions during decision-making. The use of the system of systems engineering measures implemented in both *Pilot Project I* and *II* supported this decision-making procedure by providing transparency on the state of the development work and maturity of the design. However, *the analysis found a lack of instructions or guidelines how these measures could be used to support decision-making.*

- Decisions after experiments could be made on the basis of the estimation of risks and opportunities calculated from the difference of the activity outcomes from the success criteria (*i.e.,* target values) and the impacts/benefits of these differences. Systems engineering measures and design parameters could be used to facilitate effective decision-making, because the measures fostered the identification of problem areas and the estimation of risk and opportunity regarding the quality of the product and the programmatic constraints (cost and schedule). Furthermore, *the development of a method was required to link design failures and their causes with the certain part of the design process where the failures were committed.* Such a method would have fostered deliberate failure correction through the identification and implementation of effective corrective actions.

Based on these findings, a decision was made to apply the adaptive SD framework proposed in the last section and tailor it to the TetraPak SD process needs. The result, the systematic decision-making framework is presented here.

H.5.8.1. Description of the Decision Support Framework

During the case study in *Pilot Project IIa*, four levels of decision-making were identified in a general TetraPak SD process, *i.e.,* the *Test Method level*, the *Project Level*, the *Milestone Review Team Level* and the *Toll Gate Level*. These are the four hierarchy levels, where decisions in the SD process concerning the results of V&V activities and experimentation cycles are made. In the following part, the documented and improved decision procedures are demonstrated for each

level, including the embedded block(s) of the previous level(s). Furthermore, the decision scope, the evaluation criteria, the stakeholders, the general results, and the method of risk assessment are described for each level. The described decision support framework follows the structure of the control loop of the adaptive SD framework and assigns the relevant decision-making steps to different hierarchy levels of the project. This way, the four single steps of the generic control loop were transformed into a complex decision support procedure covering all aspects of project control.

H.5.8.1.1. Test Method Level

This lowest level of decision-making involves *informal decisions on the validity, completeness, and correctness of the results* (*i.e.,* if the test procedure has been implemented properly and the result is comparable with the referenced type of test). Decisions on this level are based on highly standardized procedures.

Test Methods Level decisions are made immediately after the V&V activity by test engineers using raw V&V activity data or post-processed data to decide if the experiment was performed correctly and if the V&V activity was done according to the test procedure. In case of dissatisfactory results, mainly retesting and sometimes rework on relevant design activities is done. The need for failure correction and thus rework can be discovered at an early stage by showing the discrepancies of the V&V results from the success criteria. The amount of rework can be reduced if V&V activities are scheduled optimally and decisions are supported by clear success criteria.

The main *stakeholders* of the decision are test executors, the laboratory support, and eventually suppliers and consultants.

As *Figure H.19* shows, the scope of change actions on this level is limited to the repetition of the V&V activity, sometimes together with the related experiment. The *cause of this rework action is usually the incompleteness, insufficiency, or incorrectness of the measured raw test data*, *i.e.,* the V&V activity was not performed adequately or the results are not sufficient for further analysis. The main success criteria for good V&V results are *data completeness, relevance, and comparability*. That is, the evaluation of the raw data is possible because enough data are gathered with the required level of quality. Thus, the V&V results can be forwarded or communicated to the involved stakeholders to make a technical decision on the maturity of the design based on the contents of the raw V&V results.

In case a decision was made to repeat a V&V activity, adjustments are made to increase the quality of the activity outcomes. These adjustments can be made on either a trial-and-error basis or by using the *design of experiments* method (DoE). During experimentation and V&V planning, the suitable method to conduct the test is assigned to the V&V activity. This decision can be

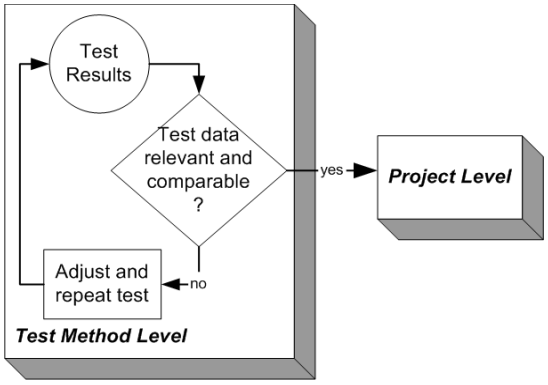


Figure H.19 Decision procedure for the test method level

revisited here, and the method can be changed during the adjustment of the activity.

Systems engineering measures from *level 4* (raw test data) and *3* (TPMs) are applied as success criteria on this decision level (see *Figure G.9* for the levels). During the definition of the success criteria, historical project data is used to define the values to be achieved using statistical methods. If the measured raw data (*level 4*) are directly applied in the decision-making procedure, the goal is to decide if the evaluated product properties represented by low-level TPMs or raw V&V activity data are achieved / minimized / maximized, *e.g.,* estimated mechanical efficiency of the product derived from (estimated) mean time between failure (MTBF) and mean time to repair (MTTR).

In other cases the raw V&V activity data is processed on a hard, statistical, or empirical basis to generate aggregated measures describing the quality of the product, *e.g.,* fraction of successful food container openings according to the specific TPM = xx.x% with a worst case + xx.x + x.x% at 95% confidence level.

This decision is made under high uncertainty, which is usually compensated by planning higher amounts of resources than required. As not the quality or performance of the design, but the quality of the V&V activity results are evaluated and a decision on the fulfillment of the activity exit criteria is made, it is not a risk-based decision.

H.5.8.1.2. Project Level

In case the success criteria in the *Test Method Level* are fulfilled and the data from the experimentation cycle is sufficient and correct, decision-making proceeds to the next level, to the *Project Level* (*Figure H.20*).

The decision on the *Project Level* is done by the V&V activity responsible and/or project manager after one or more V&V activities or a block of strongly coupled SD activities. This kind of decision is a *lower level formal technical review on the performance and maturity of the subsystem or system design*. The goal of such decisions is to assess the design progress at preliminarily set event-driven points in the system lifecycle [DoD 2001a]. Technical reviews are done after each main block of activities and major stage of development to check design maturity, and review technical risk to decide whether and how to proceed to the next stage of development.

Whereas the previous decision on the *Test Method Level* evaluates the quality of the activity, this decision assesses the performance and maturity of the design. During the decision, a set of evaluated V&V activity data is compared to the requirements to determine the state of the design regarding the fulfillment of the requirements and decide if the goals of the sub-process (or experimentation cycle) are fulfilled. A decision is made on the basis of previous project experience after assessment of historical and standard project documents.

This technical decision is limited to the verification of a specific area of product quality or technical performance. Thus, relevant measures for this decision are basic, critical (sometimes aggregated) TPMs from *level 3* and higher-level aggregated measures from *level 2* of the measurement system.

In case critical aggregated measures from *level 3* are applied, the stakeholders of the decision decide if the specific property/sub-group is built right. That is, it is assessed if the specific property was achieved by the design. A typical measure from *level 3* is nutritional properties preservation, consisting of a set of related measures from *level 4* and *3*.

As this level of decision evaluates a subsystem or a system, high-level technical performance measures from *level 2* play the key role for the assessment of the design. The state of the critical high-level product requirements represented by these measures are evaluated to decide if the required system performance level is attained, and thus technical risk was reduced to an acceptable level. Hence, the *result of this decision is a yes/no answer*. The evaluation considers international, external, internal, and quantitative quality criteria. A typical measure used as a success criterion is the maximum percentage (x.xxx %) of defective packages per test.

The desired outcome of this decision is the assurance that technical risk is controlled and mitigated and the performance measures describing the key properties of the product (*e.g.,* packaging line) are under control.

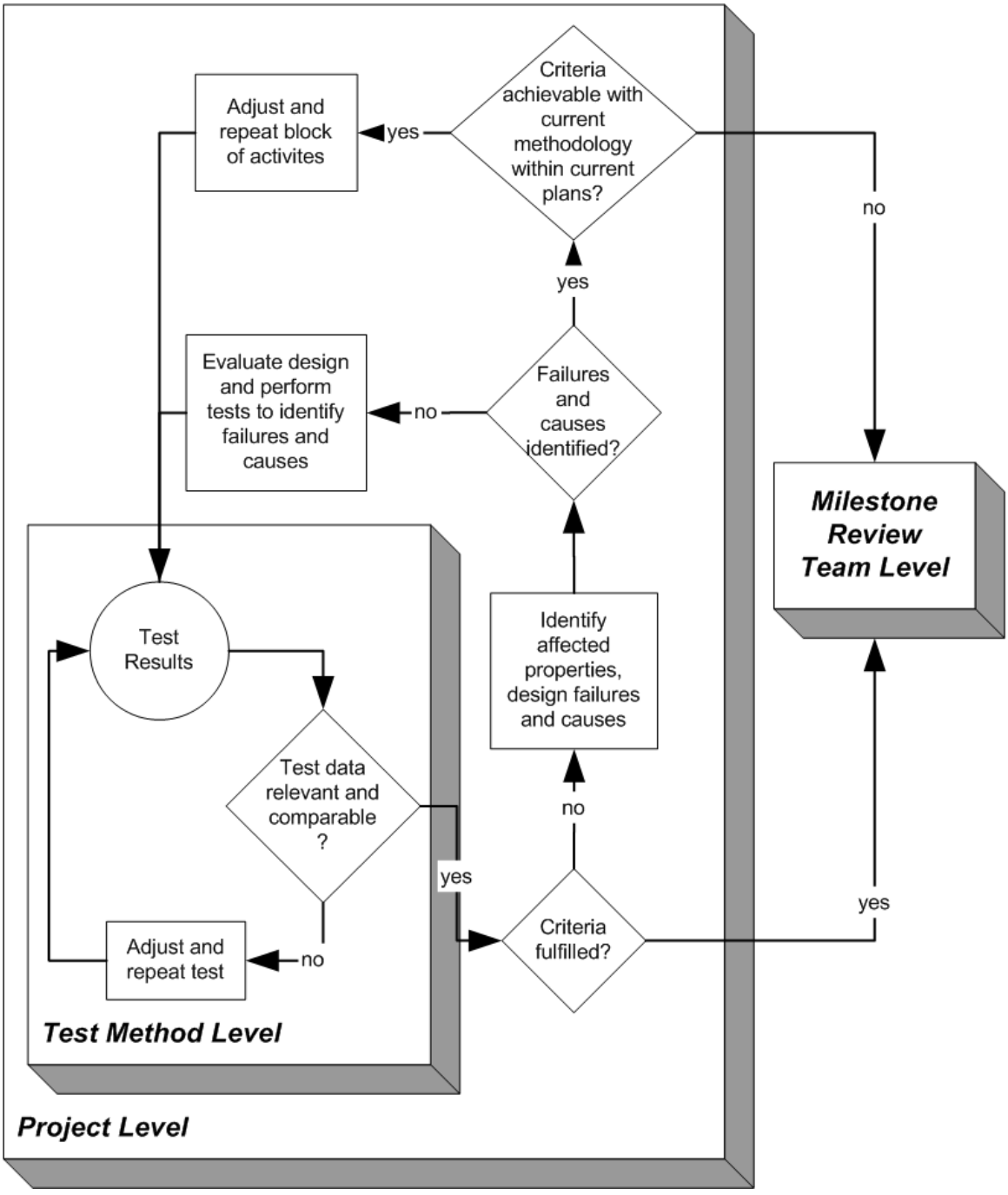


Figure H.20 Decision procedure at the Project Level

As *Figure H.20* depicts, the scope of the decision is the block(s) of activities ending with the technical review. If the exit or success criteria are not fulfilled, *i.e.,* the required design maturity was not achieved, the review team determines which requirements and product properties are affected and identifies the causes for the lower performance. In some cases, particularly during development of new products, the causes are hard to find, and thus the whole design has to be reviewed and thorough design evaluation has to be performed to identify the problems and causes. Good failure documentation, product modeling and simulation, and parameter-based process modeling support failure tracing.

Once the causes of inadequate design performance have been found, the decision-makers determine if the performance requirements can be achieved in the scope of the current methodology, existing technologies, and resource constraints of the project. If it is possible to fulfill the exit criteria through small adjustments in the plan (*e.g.,* through iteration and rework), then modifications in the plan are made and rework is done. In case major changes are required or the resources are insufficient for the change, a decision on the next decision level has to be made. The results of this decision are well documented in a set of standard report documents, which are then added to the project knowledge repository in the Intranet at TetraPak. This way the reports are available on-line for the competent persons and stakeholders.

H.5.8.1.3. Milestone Review Team (MRT) Level

The review team for the main milestones of the project evaluates the overall quality and technical performance of the system design. This decision is a preparation for the *Toll Gate Level* decision, where the management decides if the project can continue or further rework in the problem areas have to be conducted.

The *MRT Level* decision is a *global technical decision on the overall technical performance of the design*. Thus, the success criteria are the full set of *level 2* measures (MOPs). Measures from lower levels are also considered to trace problems (deficiencies between target and measured performance in the key areas represented by the MOPs and TPMs) back to their roots. For example, the packaging solution is technically compliant with international/external and internal regulations. That is, it fulfils all the "basic" properties of the design, *e.g.* a package contains and protects the product, it can be opened and closed, and it must be sold and stored on a market shelf.

This decision is a mere risk-based decision considering the discrepancies of the values of the design parameters from the performance requirements. The overall system is analyzed, and risk incorporated in the quality aspects is highlighted using color indicators for the MOPs and TPMs. Technical performance risk (red – yellow – green; critical – intermediate – passed), based on the difference between the measured and the required values and consequences, is classified into the three categories. The three categories were discussed in detail before when the three parts of the impact functions were explained.

Red means that there is no suitable technical alternative to solve the problem with the achieved product maturity, thus one or more of the previous phases have to be performed again. *Yellow* means that the design aspect is in a critical status, *i.e.,* change in the design or technologies can provide a feasible solution and the problems can be corrected through rework. *Green* is assigned to a technical performance area if the success criteria were met. Based on the determined risk category of the TPMs, decisions are made on the required rework actions and the affected activities / activity blocks / sub-processes / phases.

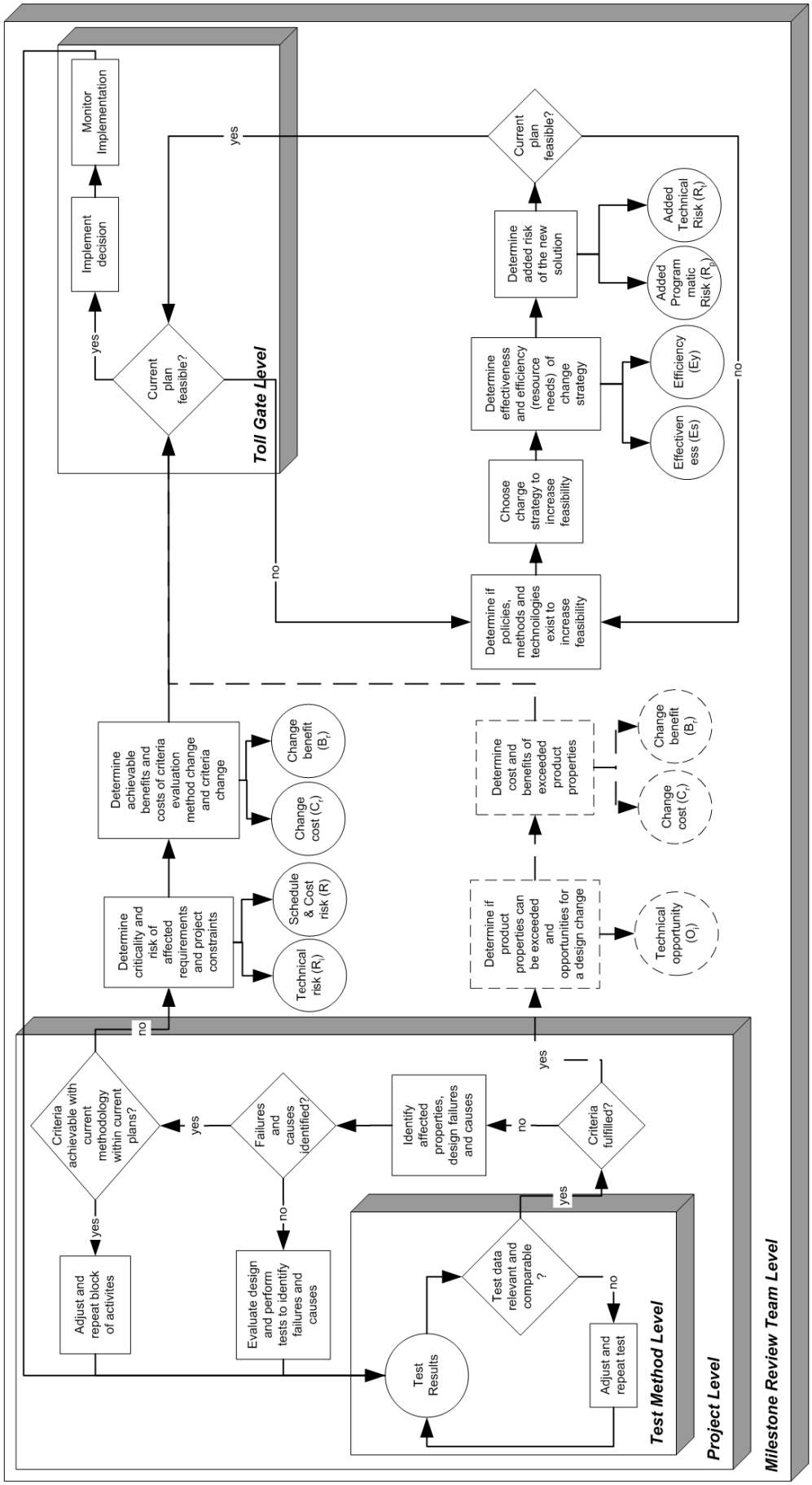


Figure H.21 Complete decision procedure

Figure H.21 shows the MRT level decision process in the context of the complete decision-making procedure. The task of the MRT is to support the highest-level decision makers at the toll gate with evaluated information on the technical performance and maturity of the developed system. This procedure includes the evaluation of both risks and opportunities considering the possible technical outcomes and resource needs of the SD process.

- *Opportunities:* In case the experimentations in the actual SD phase deliver results that exceed the required product performance, it is possible to utilize this opportunity and improve the overall value of the system through the improved features. If the phase results show possible improvements concerning the requirement values, the MRT analyzes the related technical opportunities. Furthermore, the benefits and costs of the development of a product that exceeds the requirements are estimated. After the thorough evaluation of the opportunities, a decision on the utilization can be made at the toll gate level. Technical opportunities found in the early SD might be utilized in the same project, or in case of low resources or late discovery, the opportunities can be transferred to R&D or technology development projects for further assessment. Since TetraPak performs separate technology development projects, it is rare that improvement opportunities are found in SD projects. However, it is also true that the decision culture has not yet included this kind of design evaluation (*i.e.,* opportunities were not sought before). Thus, the inclusion of the two new tasks (with dashed lines in *Figure H.21*) in the MRT decision procedure reduces the risk of overlooking an improvement possibility, and it creates a sound environment for innovation.

- *Risks:* technical risk means the inability of the design to fulfill the stated product requirements. The main causes of technical risk are insufficient design performance due to design failures. V&V activities support the evaluation of the design if it fulfills the product requirements and the identification of design failures that contribute to the unsatisfactory quality.

During the MRT meeting, the decision makers determine the actual risk and opportunity status, evaluate improvement options (including costs and benefits), and make recommendations on the further advance of the project considering technical achievements and risk areas. The prepared alternative change proposals are the main inputs for the managerial decision during the toll gate level. If none of the change proposals is feasible, the MRT continues working until a feasible strategy is developed. This work involves strong communication with the project management, who are the stakeholders of the recommendations of the MRT.

Milestone review procedures use data from both the V&V activities and previous decisions in the project. Thus, the systematic planning of information flow between different decision points

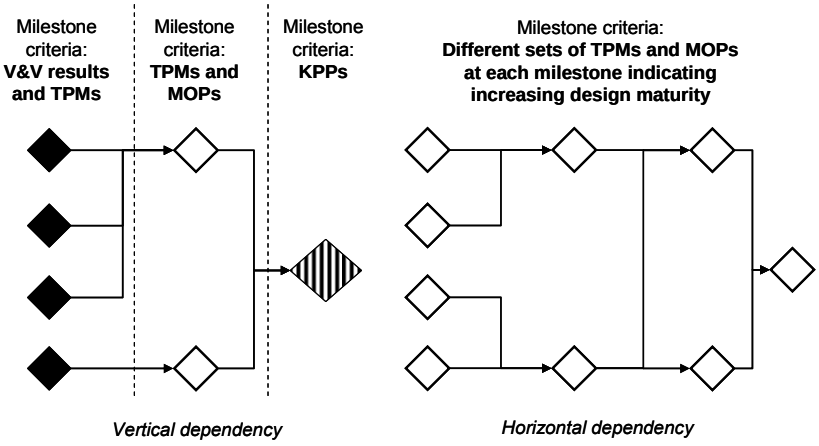


Figure H.22 Vertical and horizontal dependency of decision points

in the various hierarchy levels (*Figure H.22*) and the integration of the milestone structure with the experimentation strategy and the systems engineering measurement process facilitates the decisions in the following:

- It provides clear traceability of causes and effects of technical performance discrepancies in the product development process.

- It supports the identification of the activity blocks to be repeated on the basis of the risk categories of the decision measures and the effected product characteristics and requirements.

- It fosters the estimation of the amount of required change effort based on the parameter risks and effected activity blocks.

H.5.8.1.4. Toll Gate Level

This is the highest-level decision with the main goal to approve the decision proposed by the MRT based on managerial aspects. This decision, also called *milestone decision* in the INCOSE community, is made by the project management considering critical, high-level technical and programmatic aspects of the project.

The goal of this decision is the validation of the developed system and the examination of the feasibility of the project plan. That is, the management and external stakeholders decide if the product will maximally fulfill the satisfaction of the customers (*i.e.,* if the right product is built). The success criteria for these decisions are strictly formalized. The recommendations of the MRT have to cover every important aspect of the criteria to facilitate the transparency during the decision.

Measures that build up the success criteria are mostly from *level 1*. These Measures of Effectiveness or Key Performance Parameters represent the most critical system performance needs from the customer viewpoint. Further evaluation measures are project cost and duration, and their estimated impact on the end-product cost and profitability. Examples: (1) cost/xxxxx packages, (2) Overall Equipment Efficiency (OEE), (3) market segment specific KPPs.

Typical success criteria for the measures for this decision are:

- the packaging solution allows a reduction of 10% in cost/xxxxx packages based on the market request

- the assessed OEE complies with the levels furnished by the company to the customers

- the packaging solution is highly reliable according to the nutritional properties' preservation because it is applied for infants or nutriceutical products

The risks and opportunities concerning the global technical and management measures are categorized on the basis of market-specific medium and long-term global economic estimations. That is, during the decision, the economic value of the current design is estimated and compared to the strategic plans of the company. The objective of this decision is to maximize the value of the design based on existing and estimated, product and project related, technical and economic knowledge.

According to the recommendations of the MRT, two basic decisions are made at this level:

- Based on the achieved technical performance of the design, the area managers decide if the achieved and planned technical maturities of the design are consistent. Based on the evaluations and recommendations of the MRT, deficient technical areas requiring further improvement are identified and decisions on rework actions are made. The level of risk indicated by the color of the traffic light symbol assigned to the relevant design parameters designates criticality of the discovered problems and the amount of rework required.

- If fundamental technical problems are found, *i.e.,* if one or more KPPs are in red, major changes in the project have to be made. This usually means that the toll gate criteria were not achieved and the project cannot proceed to the next stage. Now, the task of the project management and MRT is to find a feasible solution to the problems and save the project. In such situations, change proposals are generated by the MRT to focus on the problem areas with all available resources and reduce risks to the level that the toll gate can be passed. The management then evaluates the change proposals. The main goal of the management for such decisions is to find the most effective and efficient solution that improves the product to the required performance level.

TetraPak decision name	**Test method level**	**Project level**	**MRT level**	**Toll gate level**
INCOSE nomenclature	Informal reviews with standardized procedures	Formal *technical review* on the subsystem level	Highly formal *technical review* on the system level	*Milestone decision*
Purpose	Demonstrate completeness of test and correctness of results → V&V activity quality verification	Demonstrate performance and maturity of design and decide on rework → sub-system level verification	Demonstrate overall design quality and performance and prepare recommendations for the toll gate → system level verification	Accept / discuss fulfillment of high-level managerial and quality measures → system level validation
Product	Validity, completeness of the V&V activity results	Fulfillment of the performance targets of the evaluated block	Global technical maturity of the product incl. identified technical risk areas	Global system and project validation from managerial viewpoint
Decision	Retest or small amount of rework	Rework on the evaluated block of activities	Color indicators for the technical measures → identification of the areas of improvement and rework need	Go – no go decision on the possibility to proceed to the next project phase
Scope	V&V activity or experiment	Block(s) of activities, sub-system (or system) level of the product	Lifecycle phase, system level of the product	Overall project and related parallel projects
Evaluation Criteria	Level 3 and 4, *e.g.*, fraction of successful openings, mechanical efficiency	Level 2 and 3, *e.g.*, nutritional properties preservation,	Level 2, 3, and 4, *e.g.*, number of defective packages	Level 1, *e.g.*, Cost/xxxxx packages, overall equipment efficiency
Stakeholders	V&V activity executors, laboratory support, suppliers, consultants	V&V activity responsible and/or project manager	Milestone Review Team and project manager	Area management and project manager

Table H.3 Summary of the characteristics of the decision levels in the decision support procedure

The scope of the decision concerns the entire project considering also the state of other, parallel projects. At this level of decision, the stakeholders require a compact, high-level view on the state of the system under development. Management attention has to be drawn to the main achievements and deficiencies in the project. The decision on further actions, including system change proposals can be facilitated by the evaluation of required investments, provided benefits and added risk.

The actual state of the development process can be easily captured by highlighting the risks and opportunities incorporated in the development system (product, process, organization and goals), together with their costs and benefits. The calculation and visualization of the value added through changes is an effective tool to make the effectiveness and efficiency of further actions transparent and obtain management attention for these actions.

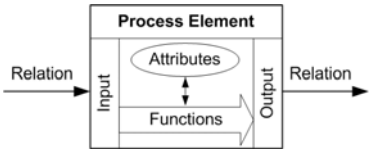
H.6. CHAPTER SUMMARY

This chapter introduced TetraPak Carton Ambient, a global player in the food packaging industry. The most important products, the SD and V&V environments, and the company culture were introduced to understand every important aspect of the case studies in this thesis. Furthermore, this chapter showed how parameter-based project monitoring and risk- and opportunity-driven decision-making was implemented in *Pilot Projects I* and *IIa*. The *procedure of adaptive project control* from the previous chapter was tailored to the TetraPak SD context, and a *decision support framework* was defined that facilitates decision-making on process adaptation. Four levels of project control were identified, where the decision support framework introduced a procedure for project control and adaptation, and described every important characteristic of decision-making (see *Table H.3*). This comprehensive decision-making model assists project planning for the definition of the structure and criteria of decision points in the project; and it supports project control by comprising the required aspects for and the *best way* to make a decision.

I. WORK*FLOW*-DRIVEN PROCESS MODELING – THE VVT PROCESS MODELING PROCEDURE AND TOOL

I.1. CHAPTER ABSTRACT

This is the first of four chapters on process modeling, an effective tool for project planning. The four chapters present the way from traditional network techniques, over work*flow*-driven process modeling—representing the state-of-the-art in the industry—to work*state*-driven methods that might be the next generation of project planning tools.



Two parameter-based stochastic process modeling methods are described and validated in the next four chapters to illustrate the difference between conventional, work*flow*-driven, and the novel, adaptive, work*state*-driven approaches for project planning. The VVT Process Modeling (VVTPM) procedure and tool developed and implemented in an industry environment in the SysTest project is presented in this chapter.

I.2. HISTORY OF PROCESS MODELING

The goal of project planning is to facilitate the accomplishment of the project work and thus support the generation of deliverables desired by the customer. The more sophisticated the product to be delivered as output of the SD project is, the more important it is to support the smooth cooperation of the different functions involved in the project by adequate plans.

Project planning establishes a network among the subsystems of the ZOPH+T enterprise model, associates the relevant elements of the different areas with each other, and organizes these in a system that continuously receives inputs from its environment and transforms them into outputs that generate value to the society. This system (*i.e.,* the SD project) represents a complex undertaking with a certain objective. The system objective describes a future state of the world that is different from the state of the world in the beginning of the system operation. The difference between the two temporally different states of the world is often called a *problem* [Sterman 2000, Haberfellner *et al.* 2002], and the job of the project as a system is to *solve this problem* under the constraints defined by the capabilities of the SD enterprise.

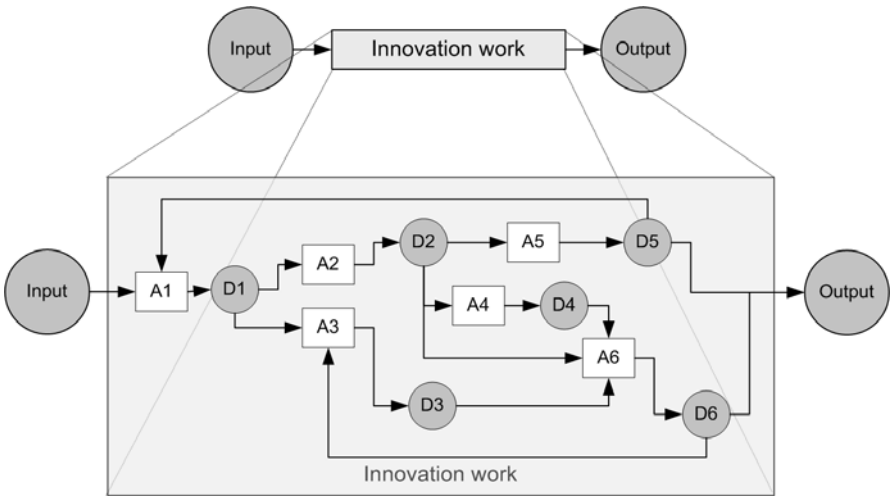


Figure I.1 Partitioning of the innovation work into activities

The problem-solving effort in an SD system is usually partitioned into smaller tasks to organize innovation work effectively and efficiently [von Hippel 1990]. The tasks or activities an SD project is broken down into, enable the distribution of problem pieces among the members of the SD organization to solve these in a systematic manner (*Figure I.1*). The network containing the logical sequence of the problem-solving tasks that produces results of stakeholder value is the *SD process*. The activities representing smaller parts of the process are accomplished by people from the organization system, using methods and tools from the technology system, and they produce results that give the product system according to the goal system. Thus, the SD process is the backbone of the SD system, which holds it together and describes the intended behavior of the system over time.

The fitness of the SD system behavior is described by the *performance* (*i.e., effectiveness* and *efficiency*) of the SD process. Thus, the success of an SD project depends on the technical performance of the delivered product, and the timeliness and resource consumption of the work conducted to develop the product. The elements of a process and their pattern of interaction define the *process architecture* [Browning & Eppinger 2002], which is an important process variable [von Hippel 1990]. While diverse SD process architectures consist of different process elements that are organized in different patterns of interaction, they behave differently and have different process characteristics (*e.g.,* performance of the delivered product, cost and duration). Thus, the main goal of project planning is to find the SD process architecture that maximizes the fitness of SD system behavior. That is, the task of project planning is to partition the innovation work into a network of activities that allows the fulfillment of the project objectives with maximal effectiveness and efficiency.

The selection of the best process alternative for given objectives requires the *understanding* of the process [Whitney 1990]. Thus, project managers need a tool that facilitates the description and exploration of the SD process space and fosters the adequate definition and sizing of the required SD activities and process deliverables. Process modeling supports these efforts by representing the SD work as a network of SD activities that consume deliverables of previous SD activities and produce deliverables for subsequent SD activities.

The two basic building blocks of conventional process models (*i.e.,* process elements and deliverables) make it possible to describe every process. The process elements (first drawing in *Figure I.2*) represent any package of work that produces an output or result [Browning 2002]. To deliver the intended *output*, the process element requires *inputs* for its work. How inputs are transformed into outputs depends on the *attributes* and *functions* of the process elements. These two characteristics describe the capabilities of a certain process element.

The deliverable elements are the inputs and outputs of the process elements representing artifacts (*e.g.,* information, documents, material, models, prototypes, drawings, decisions, *etc.*) that process elements produce as output and consume as inputs (second drawing in *Figure I.2*). The transfer of deliverables represents the *information flow* among process elements and thus defines the logical structure of the process.

The first attempts to support project (and program) management by model-based planning approaches that consider the project as a network of activities and deliverables developed various process representations. Two of the most common ones are the *Precedence Diagramming Method*

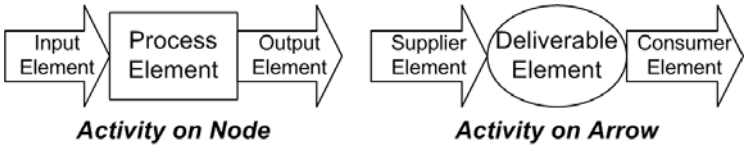


Figure I.2 Basic process representations (adapted from [Browning 2002])

(PDM), also known as *Activity-on-Node (AON)* method, and the *Arrow Diagramming Method (ADM)*, or the *Activity-on-Arrow (AOA)* framework depicted in *Figure I.2* [PMI 1996]. While PDM uses nodes to represent the activities and connects them with arrows that show the dependencies (first drawing in *Figure I.2*), in ADM arrows stand for activities, and nodes for deliverables (second drawing in *Figure I.2*). In both frameworks used for process sequencing or scheduling, the most important process attribute is *activity duration* denoting the time required to conduct a certain activity.

Due to the restricted availability of ADM for schedule analysis (it supports only finish-to-start activity dependencies) and the technical difficulties with ADM modeling (it may require the use of dummy activities to define all logical relationships correctly), most of today's process scheduling techniques and tools use PDM representation. Another reason for the broader application of PDM is that the project management goal of project scheduling is to transform the activity network into an *operating timetable* that supports project monitoring and controlling [Meredith & Mantel 2003]. In this sense, deliverables are of secondary importance representing only the dependencies among activities.

To support project scheduling, the first process modeling methods had twofold requirements: (1) Define the logical sequence of activities in the SD process; and (2) determine the required time (*i.e.,* process duration) to finish the SD work assuming that the modeled network of SD activities delivers the product desired by the customer as final output. Based on the input requirements and output products, the project manager was able to determine the precedence relationship among SD activities from the project WBS and organize them into a network. This way, the decomposed project could be integrated into a network model and the key process characteristics could be analyzed.

Process analysis techniques like the Critical Path Method (CPM) and the Program Evaluation and Review Technique (PERT) were developed and applied to identify the process elements with the highest contribution to the overall process duration and to compute the required time to finish the project. Through mathematical analysis, network methods calculate the range of possible start and finish dates for each SD activity. Once this is complete, the chain of dependent activities which has the longest total duration is identified and designated as the *critical path*. Any activity along this path that starts late or takes longer than planned lengthens the whole project schedule.

Both CPM and PERT calculate the critical path of the project; however, while CPM works with deterministic process attributes, PERT requires the definition of three point estimates for

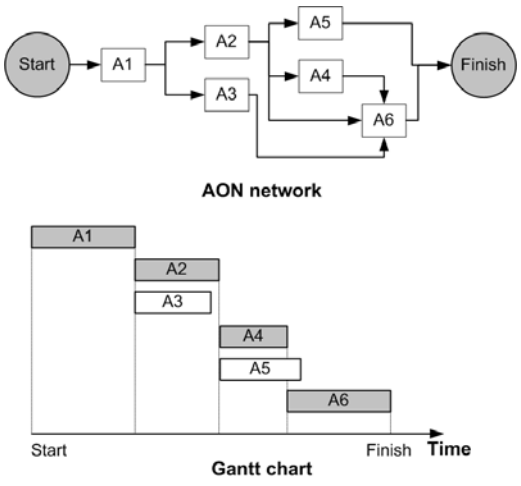


Figure I.3 AON network diagram and Gantt chart of a project

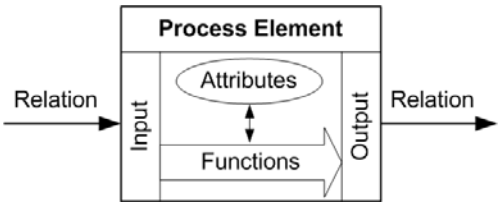


Figure I.4 Generic IPO process element for workflow-driven process modeling

the possible duration of each process element to use these values to estimate the *expected total duration* of the process elements on the critical path [*e.g.,* Igenbergs 2000]. Regardless of the kind of method applied for process analysis, the results of network techniques (*i.e.,* the structured process schedule) are usually translated into a master schedule in a graphical format easy to read and interpret for project managers.

The Gantt chart representation of an activity network is the most important tool of a project manager for the tracking and controlling of the project schedule. The AON network and Gantt chart view of a project is depicted in *Figure I.3*. The bars in the Gantt chart represent the activities, where the length of a bar is the linear function of the activity duration. The activities in grey are on the critical path, so a deviation in the duration of these activities from the planned value affects the total project schedule. On the contrary, the activities in white are with slack, so these process elements can be somewhat delayed without lengthening the project completion time.

The main task of process scheduling is to minimize the overall duration of the project, thereby reducing slacks to the minimum and dispersing resources equally throughout the complete project length. While network diagrams and Gantt charts became model-based project plans, the need arose to fill the process models with more project-specific information, *e.g.,* input requirements, entry criteria, output products, exit criteria, estimated cost and duration of the activity, activity responsible, supporting tools, metrics, resources, *etc.* [*e.g.,* Negele 1998, Pall 2000, Browning 2002]. Based on the purpose of the process model, different frameworks can be applied hosting only relevant parts or the complete set of this process information.

One of the most common workflow-driven process modeling frameworks is the input-process-output (IPO) method [*e.g.,* Negele 1998]. The IPO process element is constructed by merging the input and output products with the activity, and thus reducing all important process attributes except for activity relations to one process element (*Figure I.4*). The notion of IPO is to describe input and output deliverables linked to an activity as part of the element and consider the element interactions generated by the input needs and output products as mere activity relations. While integrating the I/Os with the process elements supports process integration, the deliverables completely disappear from the process schedule moving the scope of SD projects and the focus of manager attention from fulfilling technical system objectives to accomplishing managerial project schedules. This is a main source of planning problems and inadequate project plans that hinder effective innovation work.

I.2.1. Modeling Design Iteration

Most process modeling and network techniques are oriented toward production and business systems, where processes are repetitive without interwoven iterative loops [Browning & Eppinger 2002]. These characteristics do not apply to SD processes, where one main goal is to provide proven solutions to a problem and thus to verify and validate all assumptions and concepts, and iteratively correct failures until a feasible result is found.

Thus, a main problem with conventional network techniques is that they were developed to analyze deterministic networks and thus are not applicable to model evolving systems like an SD

process. That is, PERT and CPM are abstractions of the reality that do not account for the iterative manner of engineering design work. Whereas an SD process architecture might provide the optimal process schedule according to CPM, the best process option might have completely different characteristics if the possibility of rework and design iterations are considered in the model. As a consequence, the best process schedule obtained using CPM or PERT is useless if inputs of the activities on the critical path are delayed, because parts of the SD work had to be repeated due to inadequate product performance, a process variable not considered in conventional network techniques. Project managers usually include time buffers in the schedule to account for such problems; however, the determination of buffer sizes is often done arbitrarily based on the project manager's own experience.

To allow for iterations in process models, extensions to conventional network techniques and process modeling frameworks were proposed that are more suitable for modeling and analyzing iterative activity networks. The Graphical Evaluation and Review Technique (GERT) is an improved version of PERT including probabilistic branching, rework modeling, and process analysis using simulation [*e.g.,* Pritsker & Sigal 1983, Neumann 1990]. Other models use signal flow graphs [Eppinger *et al.* 1994, 1997, Andersson *et al.* 1998] or systems dynamics models [Ford & Sterman 1998] for iterations modeling. Important contributions come from research on concurrent engineering, where activity overlapping has main effects on rework and iteration. For example, Ha & Proteus [1995] proposed a model for optimal review timing to minimize total expected completion time. Further, Krishnan *et al.* [1997] developed a framework for overlapped sequential tasks, where the optimal overlapping strategy is based on upstream information evolution and downstream iteration sensitivity. Finally, Roemer *et al.* [2000] discussed time-cost tradeoffs in multiple overlapped tasks.

A larger body of literature on design iteration modeling applies the design structure matrix (DSM) method. Besides the proposed theoretic iteration models for concurrent [Smith & Eppinger 1997a, Ahmadi & Wang 1999], sequential [Smith & Eppinger 1997b], and hybrid processes [Smith & Eppinger 1998], the DSM community conducted extensive research work on both the analysis and optimization of iterative SD processes. On the one hand, partitioning algorithms[6] were developed that manipulate the process structure to reduce the scope of rework and thus reduce the effects of rework on the process schedule [*e.g.,* Steward 1981b, Gebala & Eppinger 1991, Kusiak & Wang 1993, Tang *et al.* 2000]. On the other hand, simulation and optimization algorithms were proposed to account for the probabilistic modeling of design iteration during process analysis [*e.g.,* Browning 1998a, Browning & Eppinger 2002, Cho & Eppinger 2005, Meier 2005].

DSM can be easily combined with an AON or IPO network where the number of modeling elements is reduced to the process elements and their relations. The goal of DSM application in process modeling is to obtain a mathematical form (*i.e.,* adjacency matrix) of the network structure, and manipulate it to improve the project schedule. While a process, as a kind of system, derives its added value from the relationship among its elements (*i.e.,* the activities) [Rechtin 1991, Browning 2002], the leverage of DSM is in its ability to improve the project schedule through the manipulation of process sequence based on the activity dependencies.

An AON with probabilistic branches, its DSM view, the partitioned DSM, and a Gantt chart view of the probable project schedule including rework activities is depicted in *Figure I.5*. Decisions on iteration are made in reality based on the comparison of actual and desired design performance, and after the consideration of available resources. Since workflow-driven project management and process analysis traditionally do not include product performance as process attribute, scholars developing process models had to find another way to account for iteration.

[6] See *Section E.3.3.3.4* for more details on DSM partitioning

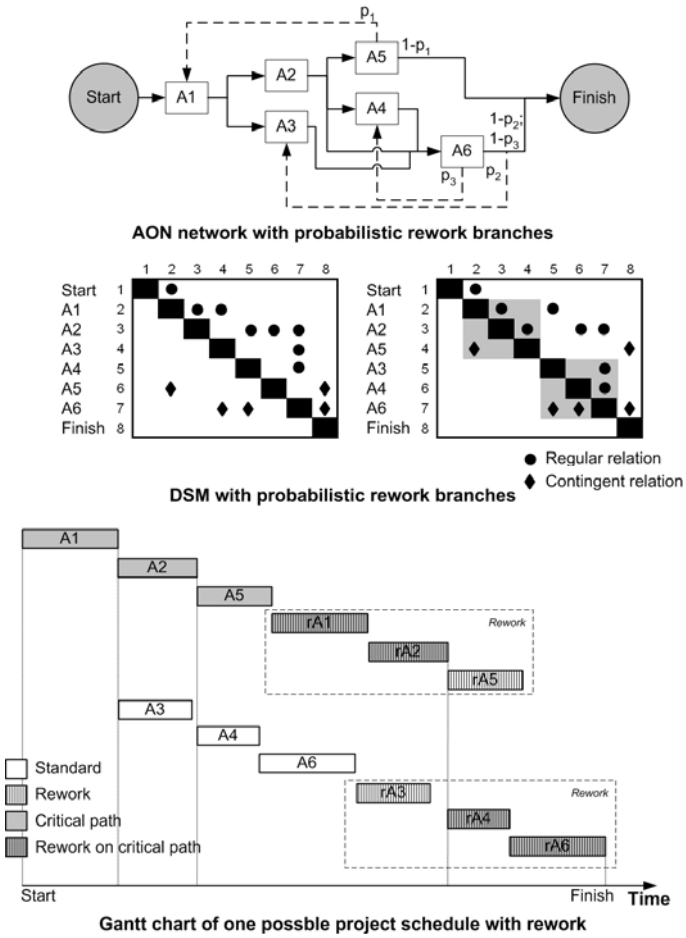


Figure I.5 Different views of process models with probabilistic branches

Because the experts could tell which activities had been the main triggers for iteration and rework in the projects (*e.g.,* V&V activities), and it was possible to derive statistical probabilities for rework for a certain activity from existing project documents, probabilistic iteration modeling became a popular technique [*e.g.,* Belhe & Kusiak 1996, Smith & Eppinger 1997b, Browning & Eppinger 2002, Cho & Eppinger 2005].

The simplest way to model the decision on design iteration during process simulation is depicted in *Figure I.5*. Here, for the contingent relations in the activity network (represented by the diamonds "♦" in the DSMs), the simulation algorithm decides to iterate with a probability of *p*, or continue the process with a probability of *1-p*. In case of a decision on iteration, the affected parts of the process are repeated and the project schedule is modified. The Gantt chart in *Figure I.5* shows how the critical path creeps as a consequence of design iterations. Activities *"A4"* and *"A6",* originally critical activities (see *Figure I.3*), are replaced by *"A5"* and the rework versions of *"A1"* and *"A2"* (*"rA1"* and *"rA2"*) on the critical path. Thus, the project schedule accounting for iterations is not simply longer, but the criticality of the activities shifts.

An important step in the evolution of design iteration models is the work conducted by Browning [1998a] and Browning & Eppinger [2002]. The proposed method is the first DSM-based simulation model that analyzes design iteration in a more generalized project network than the previous analytical models. To allow for more realistic process modeling than *e.g.,* the DSM in *Figure I.5*, Browning and Eppinger propose to calculate the risk of rework at each iteration branch in the process, where rework risk is a function of the probability and its consequences. The DSMs in *Figure I.6* show the probabilities and impacts of rework that are multiplied during simulation to obtain the risk of rework. The basic idea behind the model by Browning and

Eppinger is based on the observation that the main cause of iteration is that input information (assumptions, models, requirements, *etc.*) of an activity (produced by another one) changes during the process. If this happens, corrective actions (or rework) are made on the activity to account for changes in the input and produce adequate outputs. However, rework usually does not affect a single activity, but it propagates through the process in the form of second-order rework causing the repetition of activities linked to the first reworked task.

While both the probability and scope of rework are usually uncertain variables, they are considered as random variables in the model. Thus, the first DSM in *Figure I.6* representing the *volatility* of the activity (probability of changes in inputs), and the second DSM showing the *sensibility* of the activity (probability of changes if input shifts) in case of changes, are set up during modeling. The combination of these values shows the risk of rework for each activity.

Another aspect of the model by Browning & Eppinger is the consideration of learning through improvement curves in the model. While rework often requires only small adjustments on the original output products, rework activities do not require the same effort as the original activities. How learning affects the cost, duration, and effectiveness of SD activities during rework depends on the characteristics of the activity.

Thomke & Bell [2001] analyze how experimentation strategies contribute to the effectiveness and efficiency of design iterations and find that both factors decrease in consecutive experimentation cycles. Furthermore, they show that the cost of design iteration depending on the required test fidelity and the cost of building a prototype for the experimentation can be relatively low for virtual prototypes and extremely high for full-fidelity physical prototypes. This affects iteration efficiency as well, because while physical prototypes often have to be built again (*e.g.,* automotive crash tests) for retesting, computer models require only slight modification for a rerun of the simulation.

The consideration of rework probability and impact, and the effect of learning in the model by Browning & Eppinger move conventional workflow-driven iteration models towards adaptiveness. As the Gantt chart in *Figure I.6* shows, the process architecture is different from the one in *Figure I.5* in three aspects: (1) rework activities are shorter than regular ones, due to the effect of learning; (2) the architecture of the second rework loop differs from the original one, *i.e.,* activity *"A3"* is not part of the rework loop due to low rework risk; and thus (3) the critical

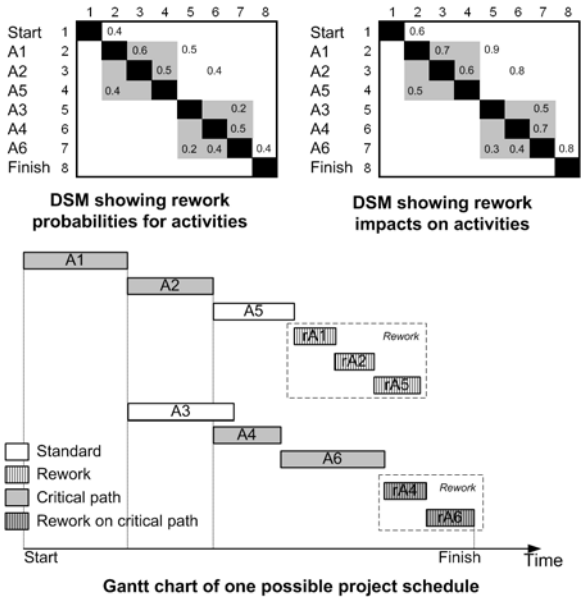


Figure I.6 DSMs for rework risk calculation and one possible project schedule

path crept again and now it involves *"A1", "A2", "A4", "A6", "rA4",* and *"rA6"*.

As the process architecture might be different in successive simulation runs, the results of the simulation provide an excellent basis for project scheduling, risk analysis, and the definition of realistic management reserves to implement the capability in the project to effectively deal with the forecasted iterations.

The next section describes the Verification, Validation and Testing Process Modeling (VVTPM) Tool developed at the Institute of Astronautics of the TU München under the supervision of Markus Hoppe and the author of the thesis. The VVTPM is an extended application of the above described model of Browning & Eppinger [2002] and the risk value method [Browning *et al.* 2002] for experimentation-driven SD process planning.

I.3. WORKFLOW-DRIVEN PROCESS MODELING – THE VVT PROCESS MODELING TOOL

I.3.1. Objectives of the VVTPM

The VVTPM[7] is a parameter-based, stochastic process modeling tool for SD project planning with a special focus on V&V and experimentation planning. The VVTPM is a product of the SysTest project and thus, the theory behind the VVTPM is based on the VVT Methodology discussed before in *Chapter H*. The partners in SysTest recognized that V&V in an SD project can be considered as a means to reduce technical uncertainty and thus risk regarding the project objectives. Thus, one main requirement on the VVTPM was to allow for the modeling of the effects of V&V on technical project risk and use this variable as evaluation criterion for the adequacy of project plans. Obviously, the best project plan is then the one that reduces technical risk most effectively at minimal resource consumption. Further goals were to model design iteration as an expected consequence of V&V in the SD process and provide a means to improve process schedule by reducing the effects of rework on the process duration.

The VVTPM tool provides a planning environment, where a project plan can be generated in an iterative manner that guarantees the systematic and controllable implementation and evaluation of system requirements in a product that maximally fulfills the customer's requirements. Thus, the VVTPM approach integrates the output products of project planning, risk management, systems engineering measurement, V&V planning, and configuration management to provide a project plan as output that considers the main aspects of each discipline.

The three dimensions of project objectives (project cost and duration and technical product performance) are the process variables in the VVTPM. To support detailed planning, the desired overall technical performance of the product is broken down into the most important KPPs representing the key technical performance aspects of the product. Methods that successfully dealt with the implementation of parameter-based process modeling, risk management, and project control were important inputs for the development of the VVTPM theory. For example, the technical performance measurement approach [Pisano 1995, DoD 2001a] provided the basic model for parameter-based technical performance modeling and tracking as well as risk-driven decision-making and project control.

The theory of technical performance measurement fosters project planning and control by modeling product performance as a vector of attributes with quantified target values. These

[7] Publications of the author on the VVTPM: Lévárdy *et al.* 2003, Lévárdy & Hoppe 2004, Hoppe, Lévárdy *et al.* 2003, Hoppe, Lévárdy *et al* 2004a, Hoppe, Lévárdy *et al* 2004b, Meier, Hoppe & Lévárdy 2004

technical parameters are assigned to the major decision points in the project to assist the measurement and control of design evolution. Clearly, the variables project cost and duration had to be incorporated in the risk-driven project control theory to support deliberate decisions.

While the basic theory of technical performance measurement was adapted in the VVTPM procedure, it was modified in one important point. To account for better, integrated risk calculation results for all process variables and satisfy the needs of project management, the original concept of technical performance measurement was associated with the risk value method [Browning *et al.* 2002]. The risk value method considers the SD process as a value generating system, where the amount of risk reduced and opportunity captured describes the overall system value. Thus, the most important attributes of SD activities in this model are their effectiveness (capability to reduce risk and capture opportunity) and efficiency (required cost and duration to provide effectiveness). Furthermore, what measures the value of the process on the activity level, also works on the project level, *i.e.,* the overall value of the SD process is a function of its capability to efficiently reduce the risks and capture the opportunities required to guarantee maximal stakeholder satisfaction.

Therefore, SD process options in the VVTPM procedure are evaluated on the basis of the deviations of the estimated overall process performance (cost, duration, and product technical performance) from the target values and the financial consequences thereof. Because alternative SD processes employ different kinds of SD activities in different networks to reach the project objectives, the estimated programmatic and technical risks incorporated in an SD process are also different. Thus, the goal of the project manager during planning is to take the evaluated process characteristics and choose the best one according to the corporate and project objectives.

I.3.2. Structure of the VVTPM Procedure

Process modeling is a systems engineering procedure to define, evaluate, and select process alternatives for given project objectives translated into evaluation criteria in the language of the model. Hence, the basic structure of the VVTPM procedure follows the *systems engineering procedure* in *Figure B.3* [Igenbergs 2000, Haberfellner *et al.* 2002]. The original five-step procedure for iterative, systems engineering decision-making is depicted by the main building blocks of the VVTPM procedure in *Figure I.7*. The following part of the thesis describes these main steps of

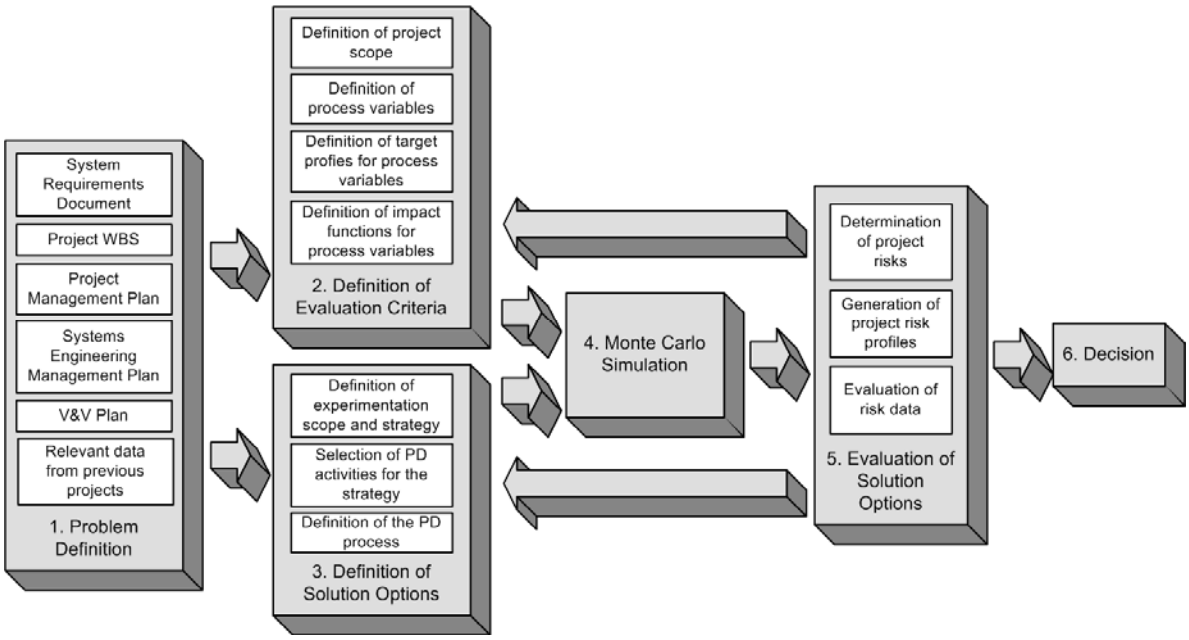


Figure I.7 VVT process modeling procedure

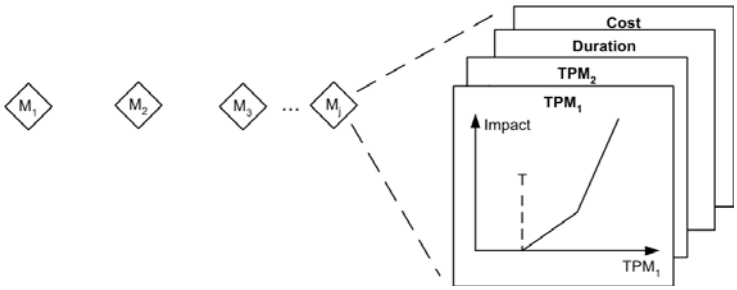


Figure I.8 Definition of evaluation criteria in the VVTPM procedure

the VVTPM procedure and the VVTPM tool implemented in MS Project.

I.3.2.1. Problem Definition

The first step of the VVTPM procedure is the generation and collection of relevant input documents from the different disciplines of project planning. It is important to have all the information from project management and systems engineering, because during process modeling, the actual project plans are specified and implemented in a model.

System requirements and the customer's needs documented in various forms define the objectives and scope of the project in qualitative and quantitative terms. This information is vital for the definition of milestone criteria and the selection of SD activities for the project. The project requirements and constraints combined with the milestones in the Project Management Plan (PMP) and the Systems Engineering Management Plan (SEMP) provide the basic control structure of the project.

The Risk Management Plan includes the anticipated, prioritized risks and criticalities of the project that assist both the definition of impact functions for the main milestones and the refinement of the project scope regarding the main risks. Furthermore, the Risk Management Plan includes the planned mitigation actions to handle the identified risks and criticalities.

The PMP contains the project WBS including the SD activities, the project management defined during project decomposition. The experimentation and V&V strategies contained in the SEMP and V&V plans include the main requirements for experimentation in the project phases and the proposed experimentation strategy.

The task of process modeling as a function of systems engineering is to integrate the often separately defined tasks of the project in one working project schedule, and establish links between the single groups of developers. In addition, during process modeling, the behavior of the SD project is simulated and analyzed to understand the complex SD task to be accomplished. Thus, process modeling is a learning exercise, where the puzzle pieces of the project are integrated in one picture to verify the feasibility of the project specified in different documents. Process modeling also serves as a tool to specify and size the SD activities using experts' opinions and existing documentation of previously accomplished projects.

I.3.2.2. Definition of Evaluation Criteria

In this second step of VVT process modeling, the SD problem to be solved is specified in the VVTPM tool using the language of the model. The structure and contents of the main milestones that begin and end the phases of the project are specified in this step. The milestone criteria including the target profiles for each process variable (cost, duration, and TPMs) represent the evolving states of the SD towards final project success.

To support project planning, risk is calculated at each milestone of the project in the VVTPM tool to show the actual project performance at the major decision-points. To enable risk calculation, in this step the project manager has to define the impact functions for each process variable at each relevant milestone (*Figure I.8*). This data is used together with the phase targets of the variables to evaluate the achievements of the phases and compute risk to highlight the criticalities during simulation. The risk values at the major milestones serve as *traffic lights* showing the project manager if the planned effort is adequate to fulfill the defined phase objectives. This information is then used to improve the plans until risk is reduced to the desired level.

I.3.2.3. Definition of Solution Options

The previous step provided the project with a clear structure and the process model with the evaluation criteria. This is the point where top-down project decomposition turns into bottom-up planning. Here, the components of the project are integrated in a process model to analyze the feasibility of the project plan.

During process integration, the activities from the project WBS are organized in a network based on their input needs and output products. Activities in the VVTPM procedure are considered as information processing units in the project, which receive information from previous activities and pass on improved information to subsequent ones. Hence, the goal of project planning is to organize the activities in a network that allows that each activity starts with the information required for its effective operation.

The function of an activity in a process is to add value to the design by transforming the received inputs into outputs with improved characteristics. As *Figure I.9* depicts, a generic VVTPM element has the following attributes:

- *Method applied with the activity (m):* the method or methodology applied to conduct a certain design or V&V activity. For example, an experimentation to design and verify the aerodynamic characteristics of an aircraft can be computer-based using the finite element method (FEM); it can be conducted on a low scale physical prototype or on a full size prototype in a wind tunnel. Hence, the method applied for the activity affects all other activity attributes (cost, duration, fidelity, and effectiveness).

- *Performance level (p):* the precision or the level of performance with which a design or V&V activity is conducted. The criticality and level of innovation of the design aspect to be implemented and evaluated define the required precision of both the design and V&V activities. Besides the design or V&V method applied with the activity, the level of precision is the other factor that affects all other activity attributes (cost, duration, fidelity, and

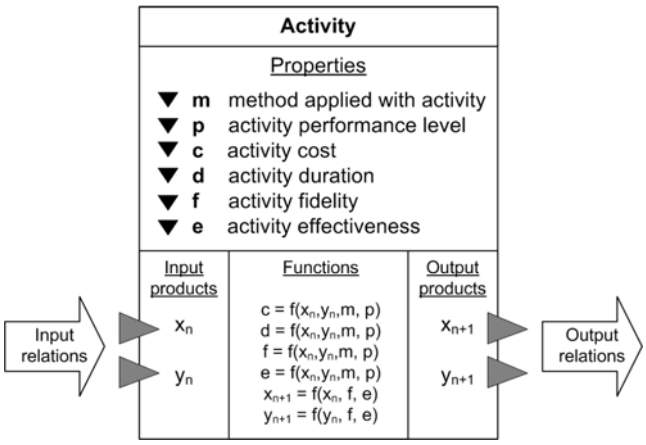


Figure I.9 VVTPM element

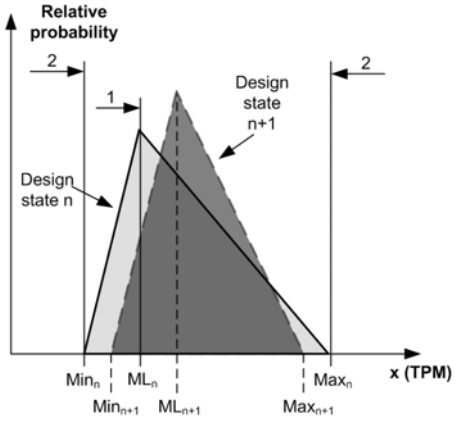


Figure I.10 Activity effects on "TPM x" in stochastic, parameter-based process modeling

effectiveness).

- *Inputs (x_i):* the information required to conduct an activity. Inputs are characterized by the type of information (*e.g.,* kinds of TPMs), its value or performance level, and its precision (*i.e.,* confidence or maturity).

- *Cost (c):* the typical or expected cost of the activity mode, including cost of equipment, installation, execution, and analysis; alternatively, this can be a three-point estimate (optimistic, most likely, and pessimistic).

- *Duration (d):* the typical or expected duration of the activity mode; alternatively, this can be a three-point estimate (optimistic, most likely, and pessimistic).

- *Fidelity (f):* the typical or expected amount of fidelity defined as the coverage of design aspects or the number of technical parameters (*i.e.,* TPMs) influenced through the activity.

- *Activity effectiveness (e):* the expected quality of the output related to its input. VVTPM activity effectiveness is measured in two ways: (1) performance improvement, represented as the shifting of one or more TPM values in the direction of improvement; or (2) uncertainty reduction, represented as the reduction of the uncertainty bounds around one or more TPMs[8]. *Figure I.10* depicts the two effects of an SD activity on a TPM (x). The two triangles illustrate the changing uncertainties incorporated in *process states n* and *n+1* (*i.e.,* the design performance before and after the activity) due to the activity effects. The overall activity effectiveness (E) can be calculated as the weighted sum of the effects (e_j) of the activity on the individual TPMs:

$$E = w_1 \cdot e_{TPM\,1} + w_2 \cdot e_{TPM\,2} + \ldots + w_n \cdot e_{TPM\,n} \text{ where } \sum_{i=1}^{n} w_i = 1 \qquad \textbf{(I-1)}$$

where the weightings (w_i) represent the relative importance of the TPMs to the project's stakeholders. There are also other, more sophisticated ways to calculate E as a function of the single e_js.

- *Functions:* the dependencies of outputs on inputs. The rate of value added by an activity to the overall project through its deliverable depends on the quality of activity inputs and effectiveness of the activity.

[8] Note that the theory of *orthogonal arrays* in Six Sigma uses the same differentiation between activity effects on product quality (see [Taguchi & Clausing 1990] for more details).

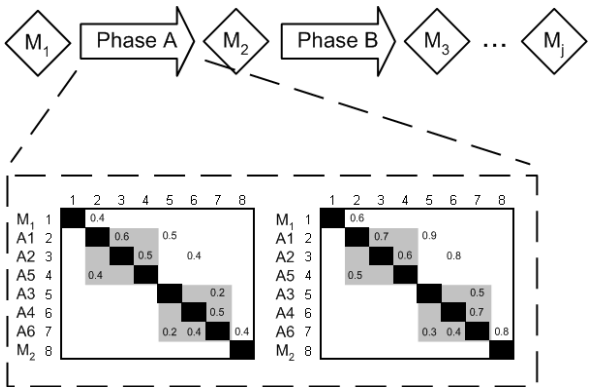


Figure I.11 Definition of solution options in the VVTPM procedure

- *Outputs (y_j):* the deliverables from the activity mode. The output of ASDP elements is information leading to increased knowledge of the product design and its level of performance. The quality of the output depends on the maturity / performance level of the input parameters, activity fidelity (*i.e.,* coverage of TPMs), and activity effectiveness.

  The quality of information comprised by the input and output (I/O) deliverables can be described by various measures (*e.g.,* maturity, accuracy, timeliness, various dimensions of quality, *etc.*). These measures can overlap with system TPMs, or they can be mapped to TPMs to describe a relationship. Based on this relationship, a change in the I/O parameter values caused by an activity can be twofold: (1) magnitude of change and (2) direction of the effect on the value of TPMs and their uncertainties [Browning *et al.* 2002].

- *Input and output relations:* these model elements describe the interaction among process elements. The function of activity relations is twofold in the VVTPM procedure: (1) they symbolize the activity precedence of relationships, which supports activity sequencing in the project schedule; and (2) the relations have probabilistic attributes used in iteration modeling. That is, each activity relation has values for rework probability and impact as described in the previous section.

The result of the step *definition of solution options* in the VVTPM is depicted in *Figure I.11*. This process description includes all relevant aspects of the process architecture for process analysis. This process model is analyzed through Monte Carlo simulation in the next step of the VVTPM procedure.

I.3.2.4. Monte Carlo Simulation

During stochastic simulation, the behavior of a process system is analyzed by investigating the possible effects of the process elements on the process variables. To allow for better process performance estimation, the main activity attributes (cost, duration, and effectiveness on each relevant TPM) are random variables in the VVTPM procedure, and tools represented by triangular probability density functions (TriPDFs). Besides the process structure, these functions are the main inputs for the simulation algorithm.

During stochastic Monte Carlo simulation, the algorithm generates random samples from the TriPDFs for every process parameter for each activity. These discrete values are then used to calculate the overall process performance. In conventional activity networks without iteration, the critical path is determined to compute project duration and the sums of the activity values for every other variable. In case iteration is possible in the process model, the conventional critical path calculation [*e.g.,* Igenbergs 2000] cannot be applied. To overcome this difficulty in the VVTPM tool implemented in MS Project, first the rework risk values are used to generate the

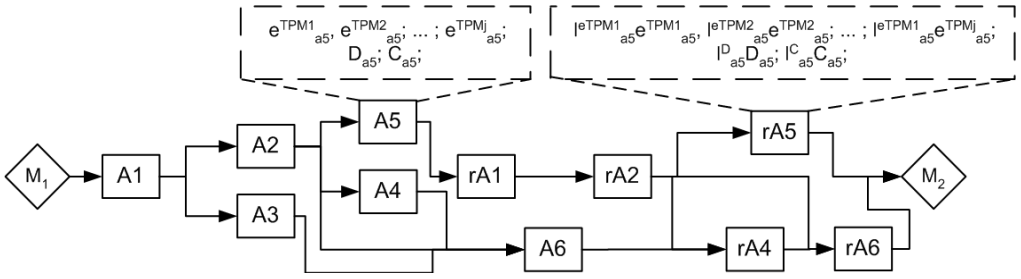


Figure I.12 Activity network generated during one simulation run in the VVTPM tool

actual process schedule including all iteration loops. That is, the stochastic iterative process architecture is transformed into a linear, discrete activity network with a sequence of regular and rework activities (see the Gantt chart in *Figure I.6* for illustration). During this transformation, the iteration loops are unfolded using the probabilistic rework risk values. To account for learning and improvement in the SD process, the probability of rework is linearly reduced in each consecutive iteration round. The degree to which the probability decreases after each rework loop is defined by the project manager based on his or her own experience and historical data.

Learning also affects the activity characteristics during iteration (*Figure I.12*). While the regular activities in the generated networks have the sampled values for each activity attribute, the cost, duration, and effectiveness of rework activities are reduced by the *learning factor*. For example, the duration of the regular activity *"A5"* in *Figure I.12* is multiplied by the factor *"I^{D}_{A5}"* to compute the reduced rework activity duration. The *"I"* factors are between 0 and 1, depending on the activity and process characteristics. Hence, during process modeling, the project manager can decide, "how much of the activity" has to be (or usually is) repeated during rework or iteration. Depending on the kind of activity this number might vary (*e.g.,* whereas some tests have strict procedures, and the same activity has to be used always, for other activities rework means only a small adjustment—*e.g.,* on a computer model—thus only a small part of the original activity—*e.g.,* model building—has to be repeated).

The generated linear activity networks are used to calculate the results of the single simulation runs similarly to a conventional CPM network. Since the values of the activity attributes and the process architecture (due to the rework risk) change in each simulation run, the simulation results are different in each run. These results are then summarized in PDFs after multiple simulation runs to provide inputs for risk calculation.

I.3.2.5. Evaluation of Solution Options

The goal of project planning is to define an SD process capable of delivering the product desired by the customer without risk. To assure a risk-free end product, project control sets regular control gates to evaluate the performance of the major deliverables at concrete points in the project. VVT process modeling combines these two project management functions and supports the definition of an SD process that is risk-free at the end and at the most important project milestones and control gates.

Therefore, during process analysis, the milestone performance values for each process variable obtained through the simulation runs are organized in PDFs to estimate the possible outcomes of each dimension of process performance. The PDFs of a TPM (TPM_1) are depicted in *Figure I.13* together with the target profile and the final requirement value. The PDFs are used to calculate risk to track the estimated performance of the process as *risk reduction system*.

The process sought is the one that accomplishes its task within the available programmatic constraints. Thus, the ideal process is the one with acceptably low technical and programmatic risks for each milestone of the project. It is highly irresponsible and unacceptable for the project

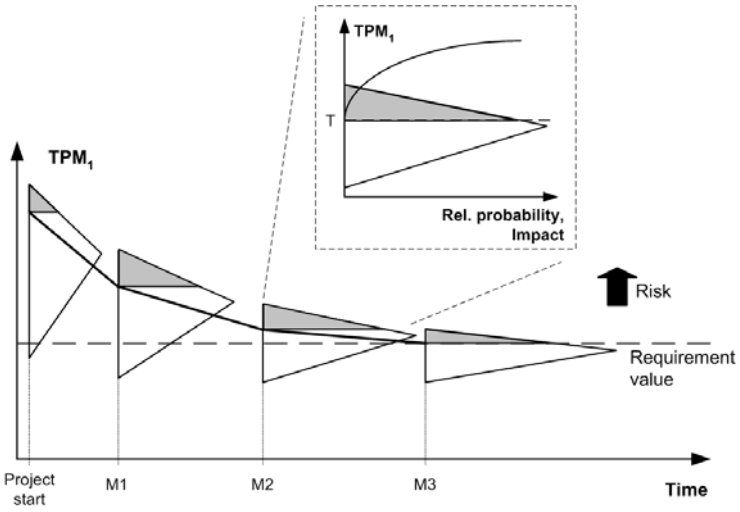


Figure I.13 Results of the stochastic process simulation

management to start a project without having an adequate plan to finish it. Thus, process modeling as a way of project integration serves as a verification tool for both the process and the project objectives. Obviously, project objectives and constraints are only changed if there are no means to fulfill them.

Project definition is done iteratively in the VVTPM procedure. Thus, once a process option has been defined and evaluated, the project manager identifies the weak points (medium- and high-risk aspects) and seeks improvement alternatives to increase process performance. As the different process variables have often contradictory effects on each other, actions improving one aspect may increase risk in another one (*e.g.,* technical risk reduction through extra activities means increased process cost and duration, or increasing design reliability through redundancies also increases weight and fuel consumption). Thus, the project manager has to be able to make tradeoffs between project goals and find a process that satisfies each process performance aspect.

The result of the fifth step of the VVTPM procedure is a set of process options with different risk characteristics. On the one hand, the project manager can select the best of these process options through the analysis of the overall risk values in the next step. On the other hand, the project manager can experiment with different process architectures for possible process scenarios. That is, besides the best plan for given project objectives, alternative plans can be defined for anticipated or unexpected SD project states. The benefit of this kind of planning is that it supports quick reaction to situations the original plan does not consider. That is, in case the unexpected happens, the project manager can select ways to deal with the new situation from preliminarily prepared improvement actions.

I.3.2.6. Simulated Process Schedule in a Gantt Chart

A major innovation of the VVTPM tool is the way the simulation outputs are presented in MS Project. To overcome the main deficiency of existing Gantt chart representations, the VVTPM tool shows the simulation results including the expected rework effort in a bar chart. This is depicted in *Figure I.14*, where Gantt chart representation of the original DSM from *Figure I.12* is presented together with the simulation results. To acquire the second schedule in *Figure I.14*, first, the simulation results (*i.e.,* the regular and rework activities) are organized in a network. Then, based on the recorded (*i.e.,* sampled) activity durations from each simulated run, the *expected duration* of each activity in the new network is computed. That is, the rework activities are integrated with the regular activities in the original network, and the expected activity durations are calculated just as in PERT. This data is shown in *Figure I.14* in the enhanced Gantt chart view.

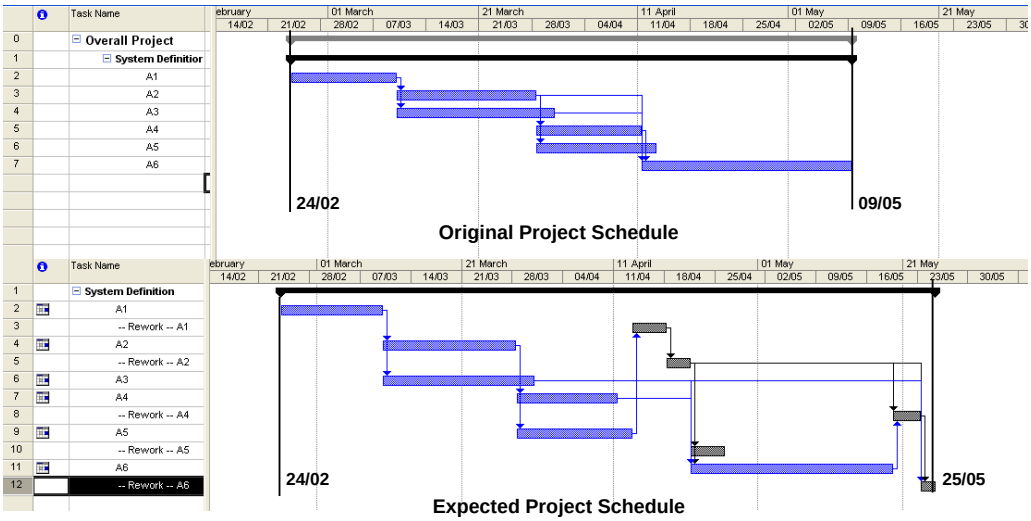


Figure I.14 Gantt chart with expected activity durations in the VVTPM tool

The resulting project schedule is 65 days long instead of the original 53 days. The 12 days extra working effort means an app. 23% increase in project duration. Considering that the quite simple project only included six activities and three rework relations, the benefits of the VVTPM tool are obvious. Of course, project planning assigns management reserves to each projects to be able to cope with the expected rework; however, the application of the VVTPM tool helps identify the critical activities and allocate the extra resources to the places with the increased required working effort. Furthermore, the VVTPM tool fosters the understanding of the causes and effects of schedule risk in an SD project and supports the engineering departments in their negotiations with the project management about their operating budget in a certain project.

I.3.2.7. Decision

The final step of the VVT process modeling procedure is the selection of the optimal project schedule for the SD, which is the one with the lowest overall risk. The first step in determining the measure of overall risk for a certain process option is to calculate overall technical performance risk from the single TPM risks. Usually, Marketing can provide information on how the customer's preferences can be prioritized and how weightings for each key technical performance can be defined. Using the weightings, overall technical performance risk is computed as the weighted average of the single TPM risk values.

Then, to obtain one representative value for each process alternative that describes the overall fitness of the plans regarding the system objectives, the weighted average of the single risk values in the three main dimensions of process performance is calculated. Now, it is possible to rank the process options and choose the one with the lowest overall risk.

I.4. CHAPTER SUMMARY

This chapter demonstrated how the VVTPM tool supported work*flow*-driven project planning at TetraPak by estimating both the programmatic and technical aspects of process performance and the related risks regarding the achievement of the project goals. The VVTPM supported proactive risk management during project planning by highlighting the main risk areas and supporting the planning of adequate mitigation actions. Hence, the main benefit of the application of the VVTPM tool at TetraPak was during the selection and sizing of the right activities for given project objectives, and the definition of realistic buffer sizes for the iterative design work in the project.

J. CASE STUDY II – IMPLEMENTATION OF THE VVT PROCESS MODELING PROCEDURE AND TOOL AT TETRAPAK CARTON AMBIENT

J.1. CHAPTER ABSTRACT

This chapter continues the description of the TetraPak case studies with a special focus on the validation of the stochastic, parameter-based *VVT Process Modeling* tool. This work*flow*-driven process modeling tool turned out to be quite effective at TetraPak for risk-based project planning, since it highlights the effects of various SD process options on the project risks and allows the selection of the best SD process schedule for given project objectives. Furthermore, the VVTPM software environment creates the basis for the long-term implementation of work*state*-driven project management and process modeling methods, like the *Adaptive System Development Process (ASDP)* method proposed in the following chapter.

J.2. PILOT PROJECT IIB – ENHANCED APPLICATION OF PROCESS MODELING DURING EXPERIMENTATION PLANNING

J.2.1. Pilot Project IIb Characteristics

The definition of the main decision points and the decision support framework in *Pilot Project IIa* (see *Chapter H*) was the first step in project planning to describe the structure and logic of *Pilot Project II*. The output of this first planning step was a network of decision points that broke down the SD into phases and sub-tasks with quantified objectives. This was the point in the planning process where top-down project decomposition turned into bottom-up project integration. That is, once the major project deliverables had been quantitatively defined and the decision points representing the backbone of the project had been set, the activities required to proceed from one major process state (*e.g.,* described by Milestone 1) to the next one (*e.g.,* to Milestone 2) had to be selected and organized in a process that made the delivery of the desired system possible.

While there are many alternative tools supporting project integration, process modeling is one of the most effective means for project planning and analysis. To obtain quantitative risk estimates regarding the fulfillment of the main projects goals in the three main areas of process performance (*i.e.,* project cost and duration, and product technical performance), TetraPak chose to apply the VVTPM tool in the case study.

A main goal of TetraPak with the VVTPM tool was to estimate the risks incorporated in alternative SD process plans and select the best risk reduction strategy based on the weighted average of the single overall risk values. Since the case studies were conducted using the VVT methodology, a further goal with the VVTPM tool was to analyze the effects of V&V activities on technical risk reduction and gradually build up a database of reusable project plans including the effects of V&V activities on the key technical performance aspects of TetraPak products.

Because V&V results are one of the main drivers of design iteration, the evaluation of probabilistic iteration modeling using the DSM method in the VVTPM tool was an important

project goal in *Pilot Project IIb*. To support this process modeling aspect, historical project data was collected and analyzed to find the activities that are the main triggers of iteration and rework, and statistical data was derived to define the rework probabilities and impacts for each activity. Furthermore, historical data was applied to estimate the effects of SD activities on the technical performance attributes of the developed packaging solution.

The case study project concerned the production and filling of packages in a typical industrial transformation process in the liquid food industry environment. Semi-manufactured goods (*e.g.,* the packaging material) are the inputs for this generic transformation process affected by external noises, and the output is a finished product. To recall, *Pilot Project II* dealt with the joint development of a liquid food package and the related machinery.

The objective of this generic type of development process is to define a new, improved package solution that enhances existing packages in terms of *geometry*, *material*, or *other characteristics*. During this transformation process, the package and sometimes its material have to be developed, and the machinery handling the package is adapted to the new package characteristics. Thus, *Pilot Project II* was a new SD project for the package and its material, and a system upgrade project for the machinery. This chapter describes how this project was planned using the VVTPM software environment.

J.2.2. Pilot Project IIb Process Description

The VVTPM procedure supports iterative systems engineering planning and the definition and evaluation of multiple project plans for different scenarios. During project planning in *Pilot Project IIb*, four alternative SD processes were designed with slightly different process architectures. These four processes are depicted by the DSMs in *Figure J.1*, where the rows and columns represent the different activities, and the marks in the matrix illustrate the different kinds of relations among them. Regular relations are illustrated by full circles ("•") and contingent ones by diamonds ("♦").

The basic structures of the four alternative strategies in *Figure J.1* are similar, *i.e.,* the SD project is broken down into four consecutive phases (*Project Definition, Concept Development, Prototype Development*, and *Product Qualification*) ending with milestone reviews. As the DSMs depict, the milestones in each phase are directly connected to V&V activities that provide the decision-makers with the main technical information. That is, at the end of each phase, the design work conducted in the specific phase is assessed and compared to quantitative success criteria derived from the product requirements.

As a consequence, V&V results are the major triggers of rework in TetraPak SD projects (see DSMs in *Figure J.1*). Hence, after the main V&V activities, decisions are made on each level of the decision support framework developed in *Pilot Project IIa* (*Test Method, Project, MRT,* and *Toll Gate levels)* to determine the necessity of further design efforts (*i.e.,* iteration or rework) in order to reach the phase objectives. In VVT process modeling, the effects of *test method level* decisions (*i.e.,* retesting) on the project budget and schedule are incorporated in the stochastic PDFs representing the estimated activity characteristics and thus these relations are not included in the DSMs. Decisions on the *project level* that might result in the repetition of activity blocks within the actual phase are represented by the feedback relations between V&V activities and design activities (*e.g.,* the relation between *"4th Preliminary Verification – Tare"* and *"Semi Manufactured Choice/Design"*). Finally, the effects of decisions on the process schedule on the MRT and toll gate levels are depicted by the feedback relations between the milestones and the design activities in the DSMs (*e.g., "MS4 – Formal Review"* and *"Sub-groups Design"*).

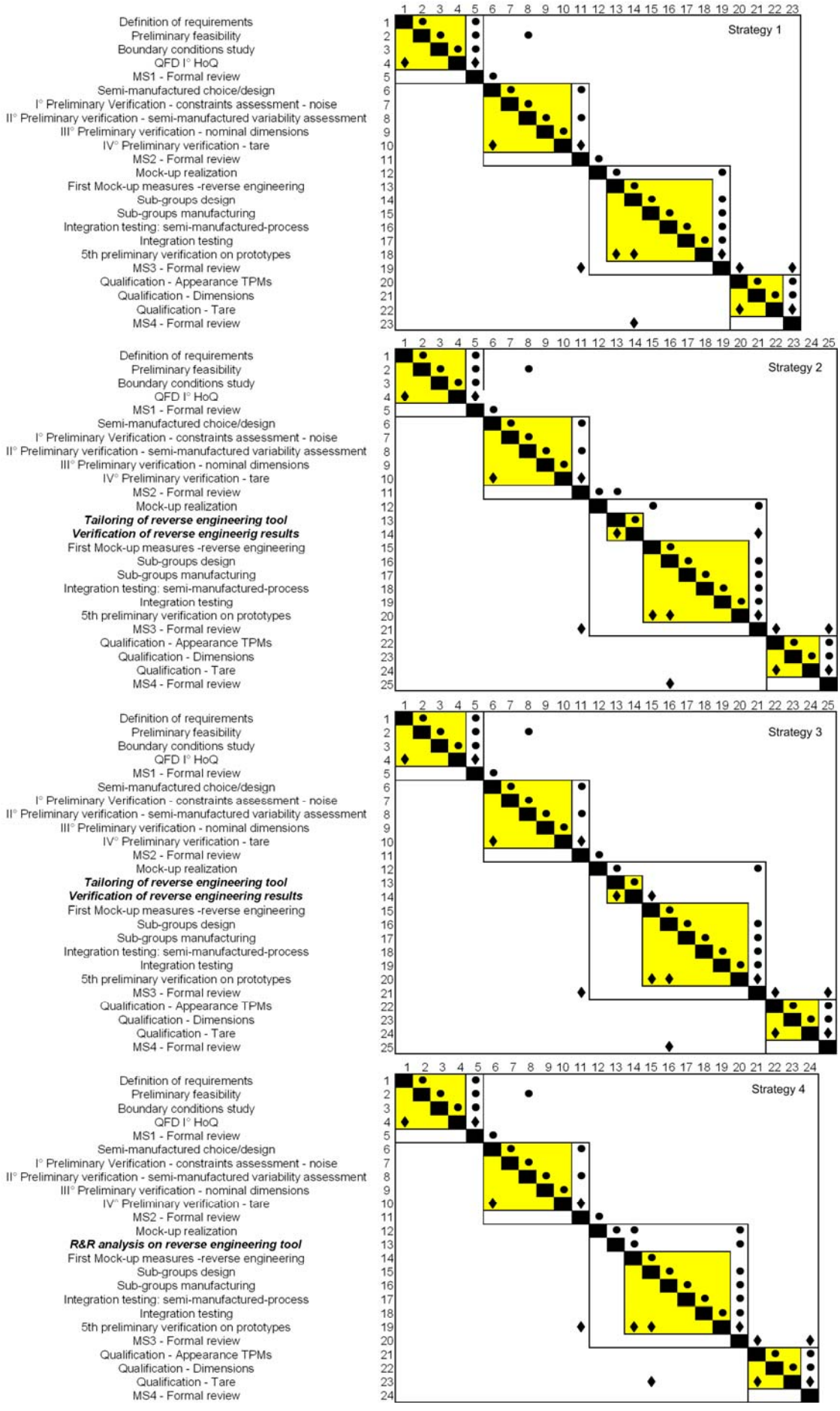


Figure J.1 Four process alternatives in Pilot Project IIb

As discussed before, the process models defined in *Pilot Project IIb* have special emphasis on the V&V strategy of the project, and thus V&V activities are depicted in higher detail and number than design activities. However, to obtain realistic simulation data, the effects of the design work on the product and process characteristics are also summarized in one or a few design activities in each phase.

In parameter-based SD, the V&V activities in each phase are conducted to verify the status of TPMs, and thus prove that the main project goals represented by the three main project KPPs *"container appearance defects", "container geometrical dimensions"* and *"container tare weight"*)[9] have been fulfilled. Thus, the phase V&V goals and scope are similar; the various representations of the evolving design generated in each phase are evaluated through different V&V activities to assess the achievement of the phase objectives in the three key areas of technical performance.

As the four DSMs depict in *Figure J.1*, the four alternative project plans differ only in the architecture of the third phase. Hence, the first phase in each process model includes the same activities that define the system-level requirements of the product under development and evaluate the feasibility of the preliminary design studies and project plans. Thus, the most important V&V activities in this phase are *"Preliminary Feasibility Analysis"*, *"Boundary Conditions Study",* and *"QFD Analysis"* using the Houses of Quality.

The second phase is the *Concept Development*, where the main focus is on the design and selection of the packaging material as well as the assessment of the characteristics of relevant system designs and transformation processes through *modeling and simulation*. Hence, this second phase starts with a design activity that selects the required semi-manufactured goods (*i.e.,* the packaging material) and the packaging system configurations for the analysis. At the outset of the project, it was estimated to be likely that a new type of raw material had to be developed to fulfill the improved package requirements. Thus, the resources allocated to the first design activity enable both the design and selection of novel and existing solutions.

Four preliminary verification activities are included in the second pilot project phase. The goal of these activities is to assess quantitatively the effects of the variations of the semi-manufactured material characteristics on the behavior of the transformation processes and the output product properties (*i.e.,* tare geometric dimensions and container defects). The outputs of the second phase are the selection and specification of the suitable raw material, the feasible package design, and the specified packaging system design.

*Prototype Developmen*t is the third phase in each of the four SD process alternatives. The main goal in this phase is the implementation of the design in a feasible, physical prototype. Design activities in this phase comprise the bottom-up implementation of the design into components and sub-groups that are then integrated in the physical system prototype. The main V&V phase goal is the verification of this physical prototype and the identification and complete elimination of each design failure included in the physical prototypes. Thus, this phase includes V&V activities from the areas of physical testing, integration testing, and system verification.

The four SD process options planned in *Pilot Project IIb* mainly differ in the way the third SD phase is conducted. *"Strategy 1"* is the basic version of the SD process, which is capable of dealing with the technical risks expected at the outset of the project. The other three process variants are pessimistic plans including increased SD efforts in reverse engineering, and repeatability and reproducibility analysis to find and correct design deficiencies in the physical

[9] The systems engineering measures systems that depict the relations between V&V activities, TPMs and KPPs are shown in Figure H.12-14 in Chapter H.

prototype. These SD processes can be activated if unanticipated risks arise during the first two phases and increased effort is required to reach the phase 3 targets.

In *"Strategy 2"* and *"3"* two additional activities were included to tailor, apply, and verify a reverse engineering software tool. The main difference between the two SD process options is that the two additional activities are conducted in parallel in *"Strategy 2"* and serially in *"Strategy 3"* to the original SD activities. In *"Strategy 4"*, one additional V&V activity is included in the original *"Strategy 1"* network, *i.e.,* a *"Repeatability and reproducibility (R&R) analysis on the reverse engineering tool"*.

The final phase deals with the qualification of the product at the customer's site. The three main qualification activities validate the three main KPPs (package appearance and dimensions, and tare weight) of the project sequentially.

J.2.3. Process Definition in the VVTPM Tool

During the application of the VVTPM tool for project planning, the first steps comprise the definition of project scope and the milestone criteria (TPM target values and impact functions) for each phase. These data were entered in the tool using the functions in *Figure H.15-17* in *Chapter H*. For simplicity and confidentiality reasons the process options defined in the VVTPM tool include only one TPM *"package width"*.

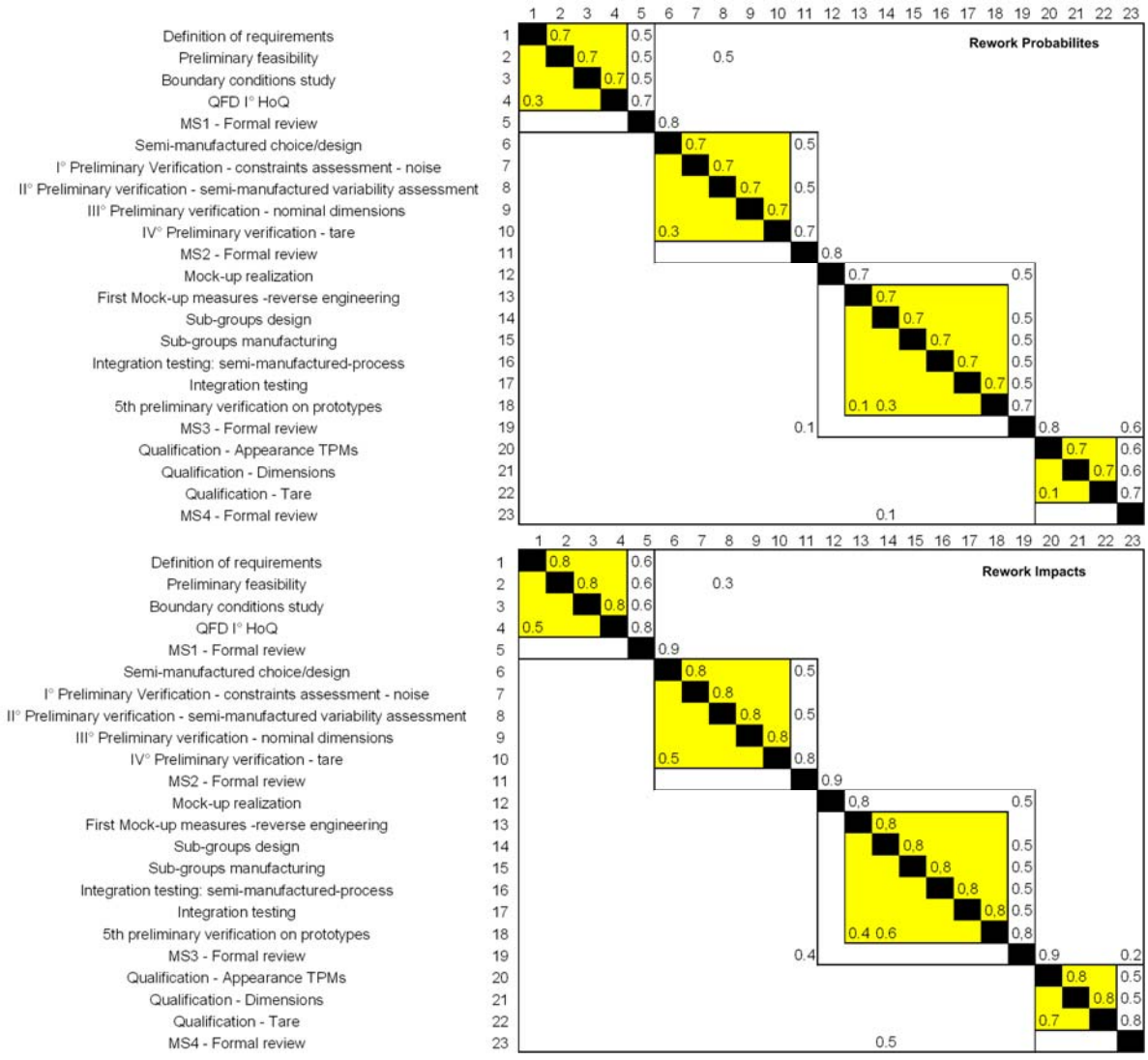


Figure J.2 Rework probabilities and impacts in Strategy 1

The following step in VVT process modeling is the process definition in the tool. The process architecture as well as the rework probabilities and impacts of *"Strategy 1"* are depicted in *Figure J.2* (the charts of the remaining three process options are included in Appendix I in *Chapter N*). As the DSMs show, one major iteration loop is included in phases 1, 2, and 4 and two loops in phase 3. While the pilot project was rated as a medium to low risk project, the probability of iteration within the phases were set to *"low (0.3)"* and inter-phase iterations were rated as *"very low (0.1)"*. Also the probability of major iterations in the final Product Qualification phase was set to *"very low (0.1)"*.

Due to industry regulations, V&V is highly procedural and standardized at TetraPak; and thus, in iteration loops usually the same activities are reworked. *"Strategy 2"* and *"3"* involve one additional small iteration loop with *"low probability (0.3)"* concerning two SD activities. This iteration loop is conducted in parallel with the original process in *"Strategy 2"* and serially in *"Strategy 3"*. *"Strategy 4"* is with the original iteration structure; however, one more V&V activity is included outside the main iteration loops in phase 3.

Each strategy includes two *inter*-phase rework relations starting at milestone 3 and 4 and ending in the previous phase. Although the probabilities that these rework loops take place are very low, the project management has to consider these during planning.

The activity effects of *"Strategy 1"* on the TPM *"package width"* are depicted in *Table J.1*. The twofold activity effects on the TPM values and the activity performance levels are entered in the VVTPM tool first *qualitatively* for a historical project to calibrate the tool. The tool translates these qualitative measures into quantitative activity effects to support a better definition of the actual project. Then, the user defines the actual project and the activity effects using the five qualitative levels (*very low, low, medium, high, very high*) that are now filled with quantitative values. During simulation, the VVTPM tool uses these quantitative effects for the calculation of the estimation of final design technical performance, and the results of Monte Carlo sampling for the calculation of project cost and schedule. Activity cost and duration values are not included in this thesis due to company confidentiality.

Finally, the VVTPM tool supports the consideration of the effect of learning about rework activities during simulation. Due to the lack of adequate data, the effect of learning about the sizes of the activities was estimated on the basis of expert opinions. One main difference between SD work in virtual and real design environment is that corrections of computer models

Activity	Performance Level	width	
		Most Likely	Uncertainty
Definition of requirements	MEDIUM	MEDIUM	VERY LOW
Preliminary feasibility	LOW	VERY LOW	LOW
Boundary conditions study	MEDIUM	LOW	VERY LOW
QFD I° HoQ	LOW	NONE	NONE
MS1 - Formal review	MEDIUM	LOW	NONE
Semi-manufactured choice/design	MEDIUM	LOW	VERY LOW
I° Preliminary Verification - constraints assessment - noise	VERY LOW	VERY LOW	NONE
II° Preliminary verification - semi-manufactured variability assessment	MEDIUM	MEDIUM	LOW
III° Preliminary verification - nominal dimensions	MEDIUM	MEDIUM	LOW
IV° Preliminary verification - tare	MEDIUM	NONE	NONE
MS2 - Formal review	MEDIUM	LOW	NONE
Mock-up realization	LOW	LOW	VERY LOW
First Mock-up measures -reverse engineering	MEDIUM	HIGH	LOW
Sub-groups design	MEDIUM	LOW	VERY LOW
Sub-groups manufacturing	LOW	LOW	VERY LOW
Integration testing: semi-manufactured-process	MEDIUM	MEDIUM	LOW
Integration testing	MEDIUM	MEDIUM	LOW
5th preliminary verification on prototypes	MEDIUM	HIGH	MEDIUM
MS3 - Formal review	MEDIUM	LOW	NONE
Qualification - Appearance TPMs	HIGH	MEDIUM	MEDIUM
Qualification - Dimensions	HIGH	HIGH	LOW
Qualification - Tare	LOW	LOW	LOW
MS4 - Formal review	HIGH	MEDIUM	LOW

Table J.1 Activity effects on the TPM package width in Strategy 1

Phases	PD Work Type	Effect of Learning
Project Definition	Modeling & Simulation	0.3
Concept Development	Modeling & Simulation	0.3
Prototype Development	Physical Implementation & Verification	0.5
Product Qualification	Physical System Testing	0.6

Table J.2 Effect of learning on rework activities in the SD phases

can be made quickly and cheaply. This is why during the repetition of SD activities in the first two phases, where modeling and simulation play a critical role, the average activity efforts and effects are reduced to 30% of the original values (*Table J.2*).

In the third phase, when the design is implemented in physical components, design iteration requires more SD work. While the scope of rework can vary from minor adjustments of the system to complete redesign, an average 50% rework rate for each activity was a realistic estimation. In the last phase, when the physical system is qualified, the tests are executed following strict procedures. However, the fixed cost of equipment integration, installation, and calibration is rather high, which contributes with a 60% average activity rework effort to this final SD phase.

J.2.4. Simulation Results

Once the process alternatives have been defined in the VVTPM tool, stochastic simulations are conducted to obtain the effects of the different process options on project performance. The risk values calculated using the results of 2000 Monte Carlo simulation runs are depicted in *Figure J.3*. Risk in three areas was calculated for each phase and the overall project for each process option. As the diagrams show, the different process architectures in phase 3 contribute to different phase risk values. Hence, these phase 3 risks determine the feasibility of the four strategies for the project.

Table J.3 shows the weighted averages of the overall risk values. Technical performance has the highest priority followed by the duration of the project in the ranking. Project cost has the lowest priority among the three project performance aspects, since deficiencies in product performance or delays in product delivery can lead to particularly high losses in company reputation and profit that must be prevented at any price.

To illustrate the causes for the outstandingly high cost risks in *"Strategies 2 – 4"*, the simulation results were copied from the VVTPM tool to *Figure J.4*. As the charts depict, the high cost risk values stem from the form of the impact functions. That is, the first part of the impact function (*i.e.,* between the target value and the inflection points) contribute to low risk in the project. However, project cost values that pass the inflection point mean extremely high risk for the project. In *"Strategies 1", "3",* and *"4"*, the PDFs depicting the simulation results have to have clearly separated parts, *i.e.,* one with lower and one with higher cost values (*"Strategy 2"* includes the same effects; however, it is not so clearly separated).

	Cost	Duration	Width	OBJ	RANK
Weigthing	0.2	0.3	0.5		
STRATEGY 1	189	7	0	40	1
STRATEGY 4	1338	223	0	335	2
STRATEGY 3	1961	678	0	595	3
STRATEGY 2	7847	288	0	1655	4

Table J.3 Weighted averages of the process risks

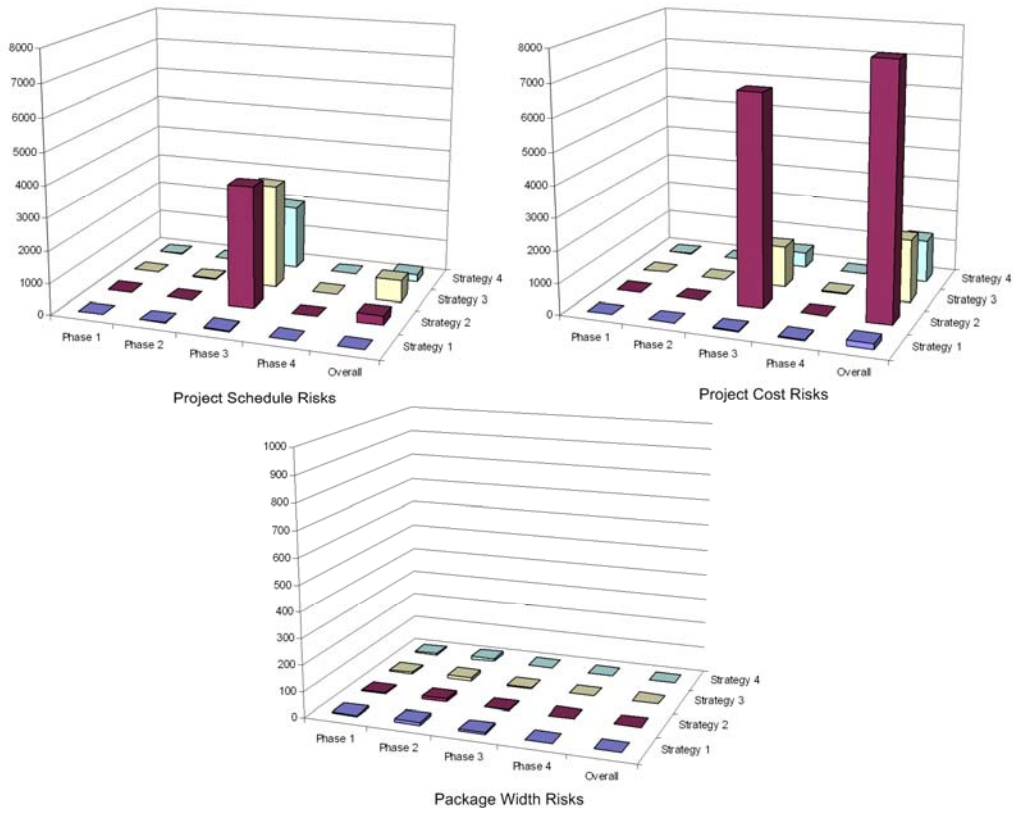


Figure J.3 Risk values concerning the four strategies

These two sets of process outcomes are the results of design iterations in the processes. Due to the probabilistic modeling of iteration in the VVTPM tool, one part of the simulated processes includes iteration(s) and thus has higher project cost, another part is without (or with a lower amount of) iteration with lower project cost. While project cost and duration usually correlate, the simulation results for project duration have the same characteristics.

J.2.5. Recommendations for the Project Manager

The VVTPM simulation results showed that technical performance risk is rather low in the project, and each of the four alternative SD process options includes adequate mitigation actions (in form of design and V&V activities) to reduce this risk to the required level in each lifecycle phase. According to the project description and the VVTPM simulation results, technical risk is

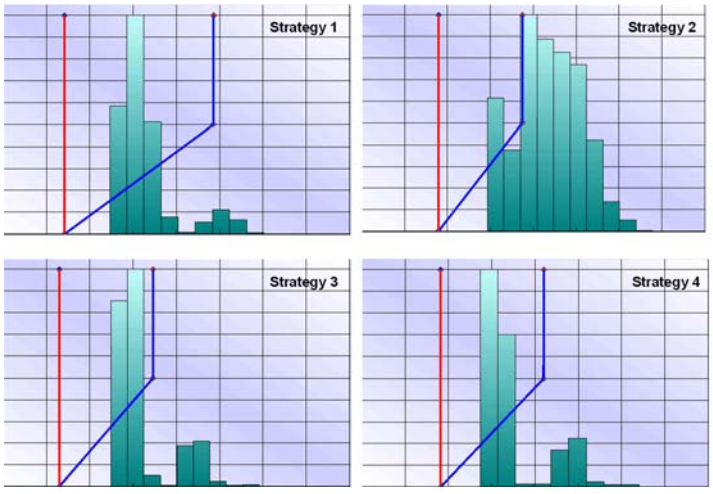


Figure J.4 Simulation results for overall project cost at the four strategies

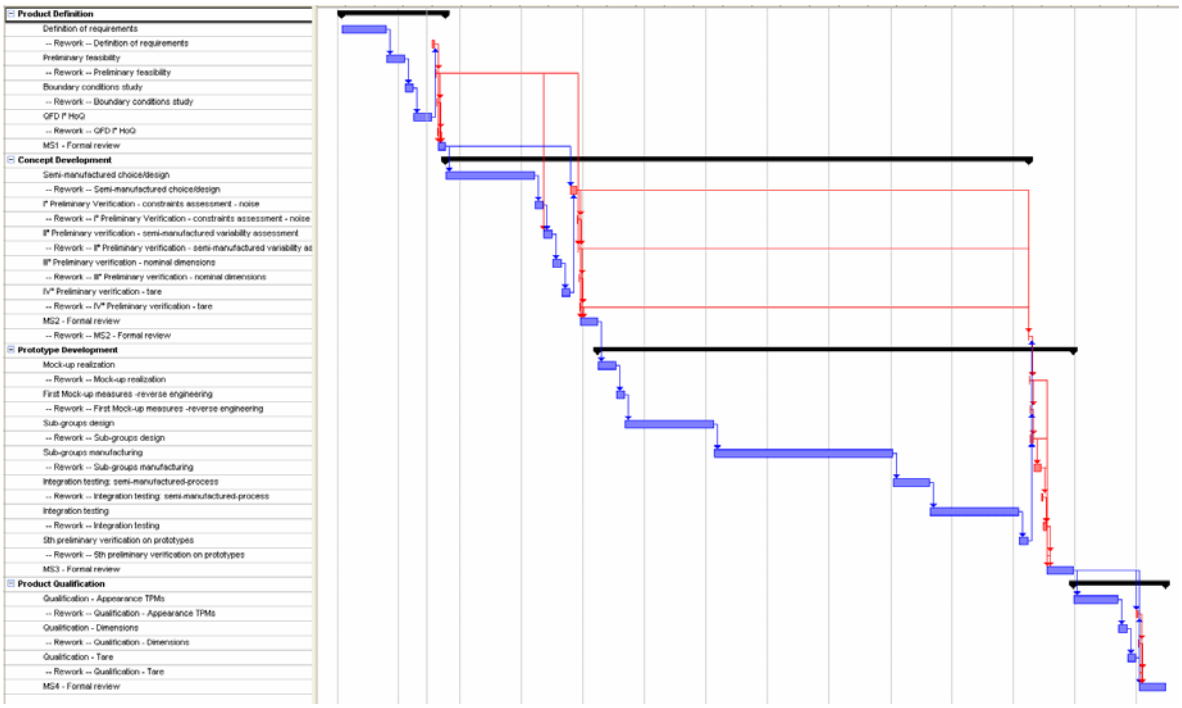


Figure J.5 Estimated project schedule for Strategy 1

expected to come in form of *uncertainty* and *variation* concerning the final values of the product properties. *"Strategy 1"* was found to be appropriate to deal with this kind of technical risk within the limits of the planned project constraints. Hence, the project manager can take the estimated project schedule in *Figure J.5* and finalize the project plan by setting the activity durations and the buffer sizes according to the estimated values.

However, it is important to note that the project schedule in *Figure J.5* could be too optimistic, which might lead to problems during project execution. While even the best of the four project plans entails considerable programmatic risk (see *Figure J.5*) and the planned project budget is rather tight, it might be deliberate to assign higher management reserves as planned to the third project phase. In case of unanticipated problems during *Prototype Development*, the raised resources would enable to switch to one of the alternative plans with additional SD effort in the critical areas. Higher management reserves would also decrease the effect of rework on programmatic risk in the project.

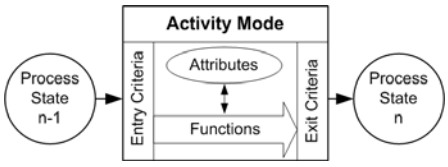
Hence, using the VVTPM tool the need for *increased process flexibility* in the *Prototype Development* phase of the project was discovered. Flexibility can be increased the traditional way, by allocating higher amounts of management reserves to this phase, which can be used in case of SD states with increased risk. Additionally, the consideration of alternative plans with higher SD effort defined at the outset of the project increases the possibility of adequate reactions to situations where technical risk is higher than expected.

J.3. CHAPTER SUMMARY

The application of the VVTPM tool successfully supported project planning in the TetraPak case study. As this chapter showed, the stochastic simulation outputs provide an effective means for process analysis by describing the behavior of a certain process architecture including the extreme outcomes. Furthermore, probabilistic iteration modeling and parametric risk assessment support the detection and elimination of process failure modes during planning that lead to lower risk and more accurate project schedules in the long term.

K. WORK*STATE*-DRIVEN PROCESS MODELING – THE ADAPTIVE SYSTEM DEVELOPMENT PROCESS METHOD AND TOOL

K.1. CHAPTER ABSTRACT



The previous two chapters described the work*flow*-driven VVTPM method and showed how it was implemented at TetraPak. However, the VVTPM method is not adequate for adaptive SD project planning, because it does not account for the modeling of adaptive decision-making. Hence, the VVTPM method was improved to allow for work*state*-driven process modeling, where decisions are made on the basis of the actual state of the design and the process.

The *Adaptive System Development Process (ASDP)* method presented in this chapter implements the *double loop of learning and control* in the VVTPM procedure, and simulates the SD process as an intelligent system that evolves toward maximal stakeholder value.

K.2. WORK*STATE*-DRIVEN PROCESS MODELING

Planning is an aspect of learning [Senge 1990]. Hence, planners need methods to increase their knowledge of the SD project. Process modeling supports project management by providing the project managers with valuable, early information on the estimated course of the project. Thus, the value of process modeling methods for project management can be measured as the *feasibility* of the proposed project plan for the given project environment and its *capability* to fulfill the desired project goals.

A feasible plan provides an effective and efficient means to realize stakeholders' expectations. Assuming that there are numerous ways to develop a product, the *goal of project management is to find the best process with the highest capability to deliver the required outputs within realistic budget and time constraints while considering predictable and unpredictable changes*. As project planning is initially performed at the beginning of a project, reliable methods are needed to identify and quantify risks that can jeopardize the final success of the project. The fields of risk and opportunity management have much to contribute here, but the variety of methods proposed by this area largely fails to integrate decision-making about project cost and schedule risks with technical performance risks.

The way uncertainty and ambiguity are handled in a process modeling method has a major impact on the quality of the delivered project schedule. Parameter-based, stochastic process modeling methods—like the *risk value method* [Browning *et al.* 2002], the *VVT process modeling procedure* already presented and validated in the previous chapters, or the *Adaptive System Development Process (ASDP)* described in this chapter—use technical measures to define the scope of an SD activity and estimate its effectiveness based on the confidence in the design parameters during its execution. Monte Carlo simulation, used by all the above-mentioned methods, provides the possibility to simulate the behavior of the process considering activity effectiveness and efficiency, and estimate the likely process outcomes and resource needs regarding the planned targets.

However, as process modeling frameworks depict the philosophy of project management for a certain company culture, process modeling methods for work*flow*-driven planning and management—like the risk value method or the VVTPM procedure—cannot be effectively used for adaptive SD planning. One major deficiency of these process modeling methods is that the consideration of decision-making in the SD is limited to probabilistic modeling. Nevertheless, SD projects evolve and change as the knowledge of the product and the stakeholders' preferences grow, forming the SD system architecture to respond to the shifting circumstances. Furthermore, adaptive SD projects cannot be planned as precisely as work*flow*-driven stochastic process modeling methods would require.

Therefore, the task of process modeling in adaptive SD systems is to identify all possible process architecture options for the given uncertain and ambiguous project requirements and define a *flexible project structure* that enables both sensing changes in the project context and efficiently responding to these. Furthermore, adaptive process modeling accounts for managerial decision-making on process adaptation during simulation, based on both the programmatic and technical performance objectives of the project. That is, *adaptive process modeling is an intelligent tool*, where not only the values of the process variables vary in each simulation run, but also the architecture and thus behavior of the process according to these parameter values.

Adaptive process modeling techniques like *signposting* [Clarkson & Hamilton 2000, O'Donovan *et al.* 2004] and the ASDP improve traditional process modeling by accounting for both *uncertainty* and *ambiguity*. They do it by simulating the process definition based on a *grand process space*—a network that includes all activity options for a certain project. Adaptive activity selection strives to choose the best activities for the *actual process* by considering the state of the SD process at each decision point.

Therefore, the output of adaptive process simulation is not just the likelihood of the outcomes of process variables (e.g., cost, time, or technical performance), but the likelihood of alternative process architectures delivering slightly different end-products that fulfill the customers needs. The difference between traditional stochastic methods and the adaptive process modeling can be compared to the difference between discrete and continuous simulation algorithms. An adaptive process modeling method, the ASDP, is proposed in the next part of the thesis.

K.3. ADAPTIVE SYSTEM DEVELOPMENT PROCESS METHOD

K.3.1. Adaptive System Development Process Elements

The ASDP method enhances existing methods by incorporating the decision-making steps of the *control loop of the adaptive SD framework* into the simulation procedure. Such decisions aim to increase the overall project value by adding new activities or relationships, or changing existing ones. Thus, ASDP supports work*state*-driven project management by simulating the managerial decisions on process adaptation based on the actual project performance.

In ASDP, the selection of the activities for the process is done adaptively based on *activity attributes* and the *process state*. An activity, modeled as a generic process element using the object-oriented IPO notation [*e.g.,* Negele 1998], is depicted in *Figure K.1*. While IPO is a work*flow*-driven process modeling technique [Pall 2000], it is important to separate two main process elements in the original IPO model to enable the analysis of process work*states* and the effects of the activities on these: (1) the input and output products defined by the actual state of the process parameters (*e.g.,* TPMs); and (2) the activities that transform the design from state *n* to *n+1*.

ASDP can be considered as an improvement of the VVTPM method to enable the stochastic simulation of the behavior of adaptive activity networks in a flexible SD process space. Hence, the main activity attributes in the ASDP model include most of the VVTPM process element attributes (*Figure K.1*). However, some new process attributes had to be defined to support work*state*-driven process modeling. The attributes are:

- *Activity modes:* particular "versions" of an activity with a similar purpose but different characteristics and performance levels (*e.g.,* rework mode for an activity). For example, there may be several ways to conduct a test: a quick, rough way; a moderately thorough way; and a very thorough way. For instance, one might have the option of evaluating a simulation model of the product, testing a quickly fabricated prototype, or testing an actual piece of hardware. Each of these options, or activity modes, requires different inputs, has different entry criteria, and implies different costs and durations. Certain activity modes are only possible when their requisite inputs are available. Of course, the quality of the results varies as well. Hence, an activity's mode of execution influences all its other attributes.

- *Process states:* the process workstates before and after the activity described by the performance level or maturity of the design (represented by the various TPM values) in the activity input and output products, and the actual project cost and schedule values. Process states are described by the actual risk (and opportunity) status of the project; and applied for the selection of the activity with the highest improvement potential (*i.e.,* potential for risk reduction or opportunity capturing) for the actual process state.

- *Entry criteria (EC):* the required performance level or maturity of the inputs to perform the activity in a given mode. (Entry criteria are similar to the parameter confidence levels required to perform an activity in the signposting approach [Clarkson & Hamilton 2000, O'Donovan *et al.* 2004].)

- *Exit Criteria (XC):* the required data quality of the outputs of a certain activity. The required data quality is described by the measures *validity, completeness, correctness, performance,* and *maturity* of the activity results (see *Table H.3* for more measures at TetraPak). In case the exit criteria are not reached, the activity has to be repeated to deliver results at the required quality level.

- *Activity availability (a):* a Boolean value that shows if the activity mode can be performed given the state of the process, *i.e.,* if the entry criteria have been achieved. If not, then the activity is unavailable at this point in the process.

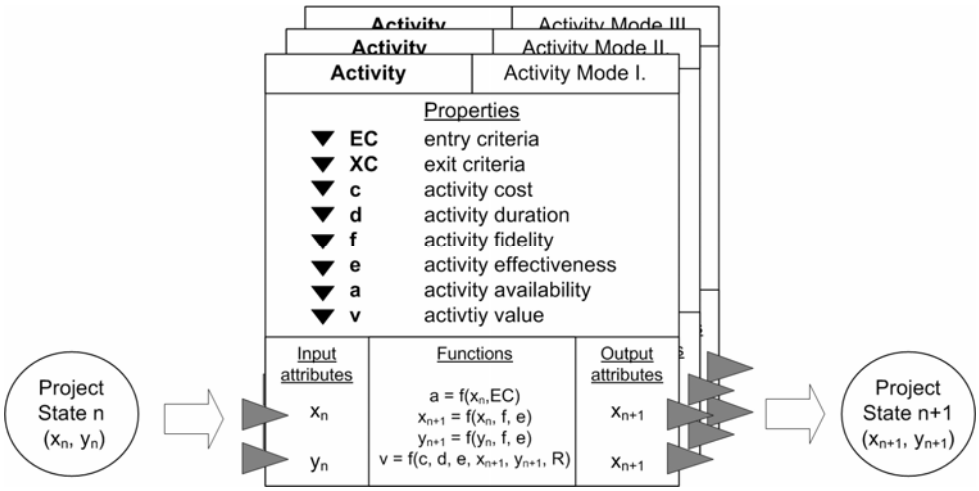


Figure K.1 ASDP process element

- *Activity value (v):* the value of an activity is comprised of two components:

    - The absolute (nominal) value of an activity is determined by its individual effect on design performance and project cost and duration. This theoretic characteristic is independent from the project environment.

    - The relative value of an activity means the activity's contribution to the overall project value, given the project's current status, *i.e.,* depending on the quality of input information available, whether it is ahead of or behind schedule, over or under budget, *etc.* Wrong or inadequate quality input information due to false activity interdependencies reduces activity effectiveness and the value added (garbage in – garbage out). Additionally, input information which comes too early or too late due to poor project schedule also decreases the impact of the activity deliverables on the overall project value. For example, a particular V&V activity may be of great value at one point in a project and a waste of time at another point.

The adaptive process element attributes improve the VVTPM method by supporting the modeling of decision-making on process adaptation. That is, these attributes facilitate in-process project adaptation during simulation following the decision procedure of the *control loop of the adaptive SD framework*.

K.3.2. Activity Calibration and the Selection of Activity Modes

At many SD organizations, the descriptions of SD activities applicable in SD projects are stored in company databases. These activity fact sheets include the information about the SD activities necessary for project planning and execution. That is, a good activity description includes the most important facts about the activity purpose, scope, results and output products, input needs, average cost and duration, methods and tools, and the standard procedures applied during the activity. Hence, these activity descriptions show the *best way* to execute an activity in any company SD project. These company best practices can be considered as the *intrinsic activity modes* and represented by the measure *absolute activity value*.

However, an optimal general solution is usually sub-optimal for a specific problem in a certain problem context. Thus, while activity descriptions are important sources of information during planning, the intrinsic activity modes have to be tailored to the actual project context to maximize activity productivity [MacCormack & Verganti 2003] and provide planning with adequate information. Thus, SD activities can have different versions in different project contexts.

During activity tailoring, the external and internal factors that affect the attributes of an SD activity described as an ASDP modeling element are identified. On the one hand, the uncertainties in the internal and external SD project environment are important to consider during activity selection and tailoring. In *Chapter F.3*, the sources of uncertainty a project has to deal with were classified into eight main groups (*i.e.,* SD cost, SD schedule, performance, technology, market, business, needs, and production uncertainties). These uncertainties affect the characteristics of the project, and thus the type and size of the activities required to apply in the SD process to reach final project success.

For example, the technology or method applied with the activity has main effects on its effectiveness and efficiency. That is, the suitability of the selected technology to solve the SD problem, the familiarity of the SD staff with the technology, or the technology maturity are all key factors contributing to the SD activity value.

On the other hand, there are internal tailoring factors concerning the actual project environment. These factors include managerial and organizational characteristics, the SD philosophy applied in the project, the project type (*e.g., derivative*, *platform* or *breakthrough* projects [Wheelwright & Clark 1992]), product type (*e.g.,* test procedures applied during an activity might be different for different products), process type, make-or-buy decisions regarding the activity owners (*i.e.,* outsourcing), *etc.* For example, the level of innovation required to reach the system objectives, the geographical environment of the project, or the SD team constellation (team size, experience and learning rate, cohesion, motivation, *etc.*) have an important influence on the effectiveness and efficiency of the activity.

Experienced project planners know which "screws" (*i.e.,* activity attributes) have to be adjusted to improve the activity value in a certain SD environment. However, in the presence of uncertainty and ambiguity during planning, it often happens that the planning team cannot set the *exact size* or *type* of an activity required to fulfill a certain SD goal. In such situations, the planners can decide to define alternative plans like in *Pilot Project IIb* at TetraPak, or they can include various activity modes in the project plan and analyze the resulting *grand process space* using the ASDP simulation algorithm.

Planners include real options in the SD process by defining various ways to fulfill certain project goals, and keeping the option to choose the right activity until adequate information from the SD process arrives. Hence, activity modes can include activity versions with alternative design or test methods, different activity fidelities, and customized procedures for special purposes. For example, the effectiveness of design iteration can be improved by planning activity modes with reduced efforts for the correction of anticipated design failures. The investment concerning these new activity modes is limited to the definition and adjustment of standard activity descriptions and procedures during planning. However, the benefit can be high if the project arrives at a process state where high savings can be achieved through the application of the special activity modes.

K.3.3. Project Planning using Activity Modes

In order to illustrate the philosophy of the ASDP method, a simple example for the modeling of overlapping activities is depicted in *Figure K.2*. The first diagram in *Figure K.2* shows how conventional activity overlapping works in case of a design and a V&V activity. The empty arrows represent preliminary information packages that are exchanged between the activities on a regular basis. Krishnan *et al.* [1997] argue that not all kinds of information can be exchanged in a preliminary form, so they recommend the identification of adequate information types and the definition of packages with information of increasing maturity for the process of overlapping.

It can be assumed that the preliminary information packages require a definite amount of SD effort, which can be accomplished by a "certain part" of the originally planned design activity. Furthermore, it is also likely that the completion of the first and second information packages

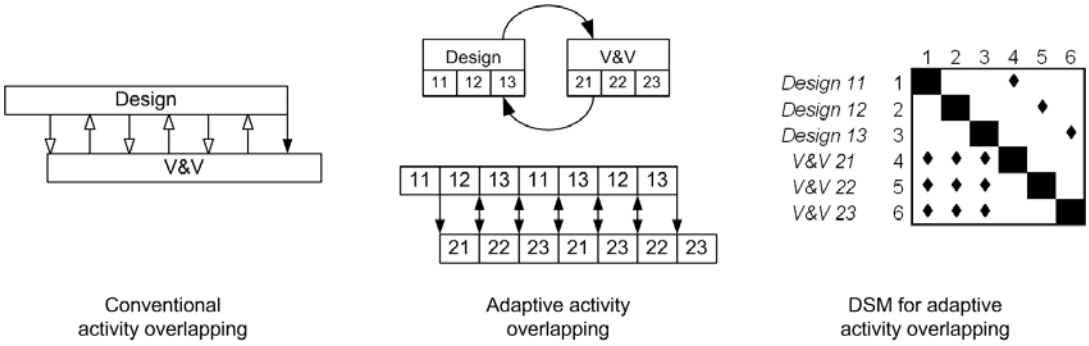


Figure K.2 Activity overlapping in adaptive process modeling

require different SD steps. Hence, the specific part of SD work represented by activity *"design 1"*, can be described by a network of smaller SD tasks, *e.g.,* by activity modes *"design 11"*, *"12"*, and *"13"*. Furthermore, the information included in the outputs of the three design activity modes have to be analyzed through slightly different analysis procedures, *e.g.,* represented by activity modes *"V&V 21"*, *"22"*, and *"23"*.

This way, the two long overlapping SD activities can be broken down into short iterative design-test cycles, where the upcoming activity modes are selected according to the results of the previous cycles. This enables the control and adaptation of overlapping activities through atomic activity modes that can be varied and recombined on the basis of the actual process needs. In case of the process in *Figure K.2*, each design activity mode is connected to a V&V activity mode based on the I/O characteristics of the activities. However, the V&V activity modes can be followed by any of the three design activity modes according to the actual process needs.

K.4. SIMULATION IN THE ASDP METHOD

K.4.1. Parameter Sampling

The ASDP model is explored with stochastic (Monte Carlo) simulation. The main parameters are cost, duration, and product technical performance (represented by the main TPMs of the product). These parameters are random variables in the model represented by triangular probability density functions (TriPDFs). During the stochastic simulation, the parameter values are selected randomly from the TriPDFs, changing the characteristics of each activity and thus the overall process in each run. This aspect of stochastic simulation can be utilized to select the best activity for the process as the project unfolds. The changing activity characteristics facilitate experimentation with different activity options in each simulation run and the selection of the most suitable (robust, valuable) one for the final project plan.

K.4.2. Risk and Opportunity Calculation

The objective of adaptive process modeling is to select the best activity for the process when the time comes to do it. The best activity is the one that adds the highest value to the process.

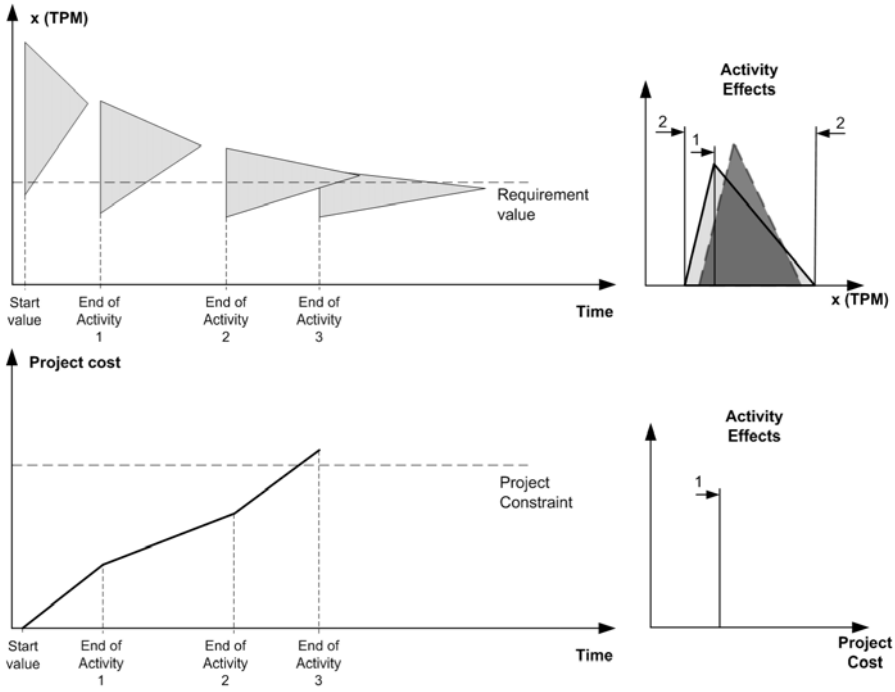


Figure K.3 Activity effects on process parameters during discrete event simulation

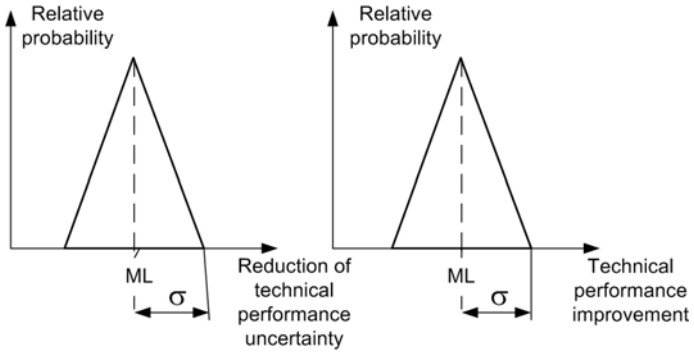


Figure K.4 Stochastic activity effects on design technical performance

Nevertheless, the effect of the activity on the overall project is not a static variable. It changes depending on the state of the project. For example, the type and criticality of the failures found during testing affect the type and amount of iterations to be performed.

The state of the project can be determined based on the difference of the planned and actual values of the three key process performance areas: cost, duration, and technical performance. To obtain meaningful, comparable information on the effects of these differences on the final project outcomes, risk and opportunity are calculated from the differences during the ASDP simulation.

During risk/opportunity calculation, the probability of failing/exceeding a certain target (*e.g.,* final requirement value) is multiplied by the impact/benefit of failing/exceeding (where impact/benefit is the function of the difference between the actual and target values). To estimate project risk, stochastic process modeling methods like the *risk value method* simulate the process many times and record the probable outcomes of the process variables. These probable outcomes are then compared to the targets, and risk is calculated from the probabilities and impacts of the differences.

Risk/opportunity calculation in ASDP has more difficult requirements than conventional stochastic process modeling methods. Since risk/opportunity has to be calculated after every activity to support the selection of the following activity, the results of several simulation runs cannot be used normally, but the discrete values of the process parameters have to be applied. Further, ASDP employs different algorithms for technical performance and programmatic risk, because SD activities affect programmatic and technical process parameters differently.

This phenomenon is shown in *Figure K.3*. While project cost and duration is zero at project kick-off, and the cost and duration of every single activity are discrete values acquired through sampling in each simulation run, design technical performance is different in nature. Due to high technical uncertainty in the beginning of the SD project, the starting values of the TPMs inherit high variance, which is then systematically reduced through the execution of various SD activities. Hence, the starting values of the TPMs are PDFs, not discrete values. Furthermore, as discussed before, SD activities have twofold effects on the values of uncertain TPMs, *i.e.,* they improve the most likely value and/or reduce the dispersion of the possible outcomes.

Activity effects on technical performance are random parameters represented by symmetric PDFs in ASDP (*Figure K.4*). On the one hand, the mean values of the PDFs can be obtained statistically after the collection of historic project data and experts' opinions. On the other hand, empirical values are used for the determination of the variance of outcomes in the PDFs. The variance of experimentation results (*i.e.,* the precision of an experiment or test) is defined by the *coefficient of variation* (c_v). The coefficient of variation is a measure for the dispersion of the experimentation outcomes and calculated the following way:

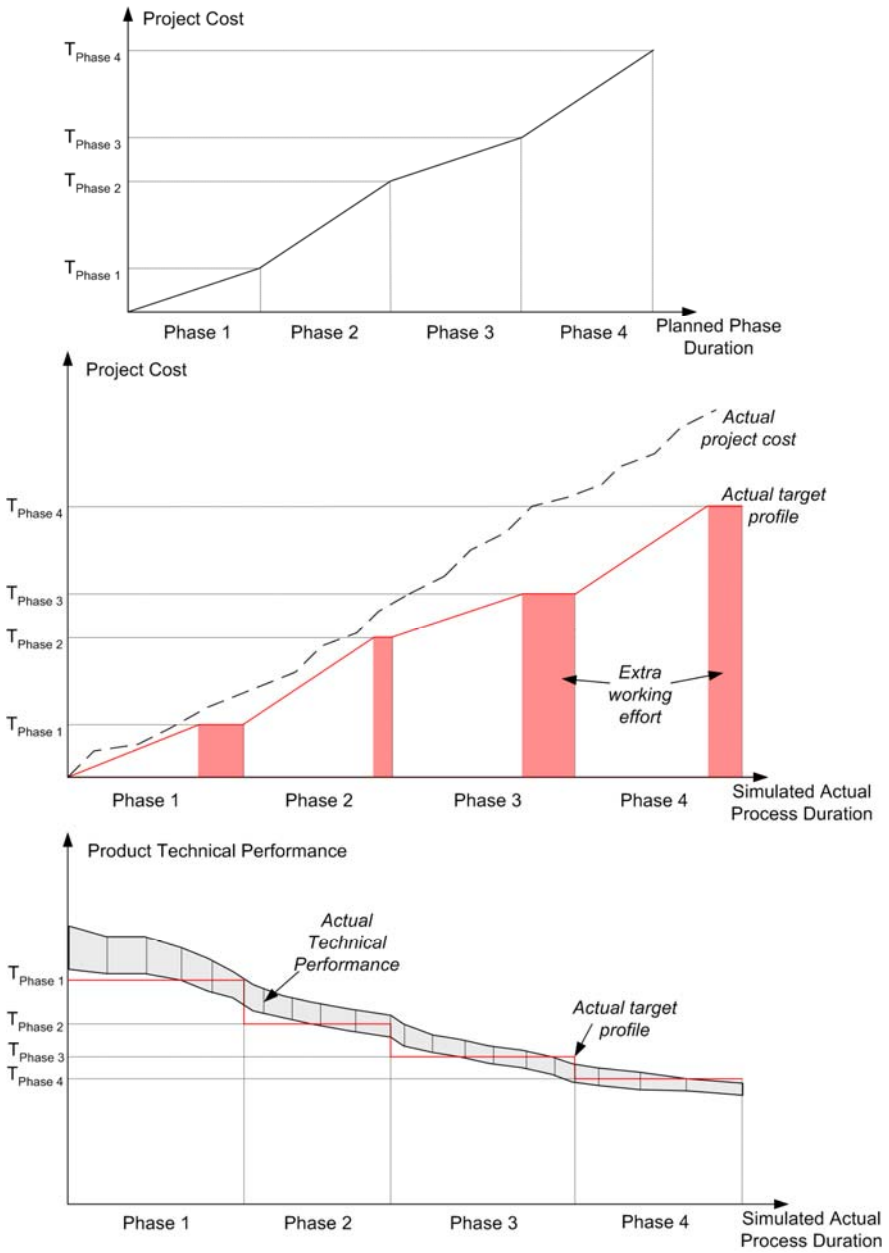


Figure K.5 Illustrative target profiles versus actual simulation data

$$c_v = \frac{\sigma}{\mu} \qquad \textbf{(K-1)}$$

Where σ is the standard deviation, and μ is the mean or most likely value. According to test experts, the value of c_v in real life experiments is 10-15% and 5% in laboratory tests[10]. Hence, in ASDP these values are applied for the PDFs for the dual activity effects on technical performance in *Figure K.4*.

The possible outcomes of the TPMs after each activity are PDFs enabling the calculation of technical performance risk using the *risk value method*. However, the same risk estimation technique cannot be directly used for programmatic risk calculation. Hence, it had to be slightly modified to fulfill the requirements of adaptive process modeling. The first modification aimed to allow for in-process risk calculation. That is, the actual project risk had to be calculated after each activity, not only at the project end. Thus, *target profiles* had to be used as a proxy for the

[10] The author appreciates this valuable information stemming from Carlo Leardi of TetraPak Carton Ambient.

single target values to permit the continuous evaluation of the state of the project. Since most companies record the actual cost and schedule profiles of their projects, and the theory of *earned value* is well known among project managers, this method was adapted for risk calculation in ASDP. A key goal of ASDP is to support project management with an effective method to calculate project risk by using the existing data collected during earned value measurement.

The application of target profiles for the comparison of actual and planned project cost and product technical performance are depicted in *Figure K.5*. The first graph shows the nominal cost profile defined by the phase target values. The second graph depicts the actual versus target cost values during an illustrative simulation run. As the graph shows, target profiles are used for risk calculation until the actual value is smaller than the target value, and afterwards, the phase targets are the reference values. The third graph depicts the evolution of TPM outcomes in the process, for a TPM where the lower is the better. Here, the target values are constants for each phase and gradually approach the final requirement value.

Another difficulty of in-process risk calculation is that the process variables are discrete values, not PDFs. While traditional process simulation methods organize the probable outcomes of numerous simulation runs into PDFs to determine the likelihood of each possible simulation result and calculate risk at the project end, ASDP applies a six-step approach to permit in-process risk calculation:

1. Calculate the actual, discrete values of the process variables based on the characteristics of the activities.

2. Compare them to the target profiles.

3. Determine the difference.

4. Estimate the probability of failing the final target given this difference.

5. Determine the impact of this difference.

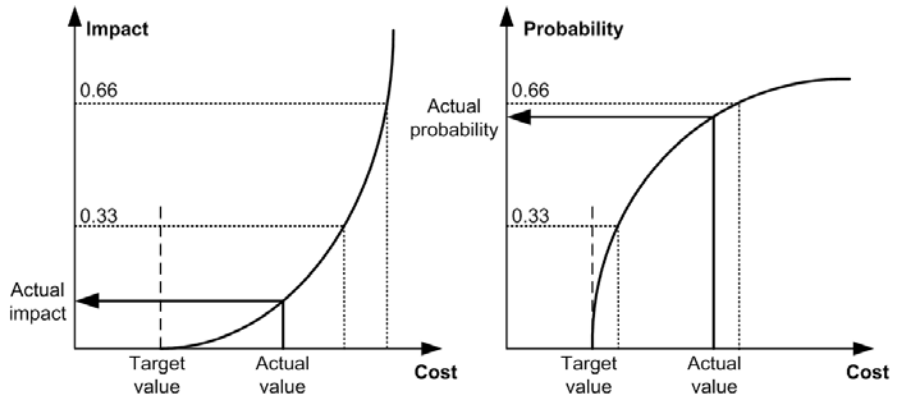
6. Calculate risk using the estimated probability and impact values.

For example, *Figure K.6* depicts the two functions we used for project cost. Both functions represent estimates of the effect of a difference between planned and actual cost at some point during a project on the final project outcome. In this example in *Figure K.6*, the actual cost is higher than expected at some point in the project, and this difference implies a moderate probability (0.33-0.66) that the budget will ultimately overrun, but such an outcome has a low impact on the project (0.00-0.33).

Using the risk estimations for each process variable, the overall project risk is calculated with the following formula:

$$R^A_{project} = w_S R^A_S + w_C R^A_C + w_{TP} R^A_{TP} \qquad \textbf{(K-2)}$$

where $R^A_{project}$ is the actual overall project risk, R^A_S is the actual schedule risk, R^A_C is the actual cost risk, R^{ASDP} is the actual technical performance risk, and w_S, w_C, and w_{TP} are the weightings for each variable. These weightings represent the relative importance of the project goals. Thus, these weightings are constants, defined in the beginning of the project, and they are normalized to sum to one. Again, there are more sophisticated functions that could be used in lieu of *Equation (K-2)*. Each of these has advantages and disadvantages. Here the weighted average is applied only because of its simplicity.



A) impact of failing the final target depending on the actual value at time point X

B) probability of failing the final target depending on the actual value at time point X

Figure K.6 ASDP risk calculation inputs for project cost and schedule

The outcome of the risk calculation is a number between 0 and 1, which can then be classified as low, moderate, or high risk for the project. These risk values are the main drivers of activity selection and iteration planning in ASDP.

K.4.3. Activity Value Determination

ASDP calculates the relative activity value, defined earlier, as a variable representing the activity's capability to increase the overall project value in the actual process state. As the ultimate goal of decision-making is to maximize project value, the ASDP simulation algorithm always selects the activities that most effectively reduce risk, utilize opportunity, and thus increase value in the process. For example, many companies have standard procedures to correct typical design failures, including activity modes with a special focus on failure correction. The value of such activities or procedures is usually low, until the special kind of failure is detected. If the failure is found, the application of this certain activity is the best choice, because this activity was developed to correct this particular failure, and thus it has the best effect on the overall project value.

The *first step* in calculating activity value is to determine the absolute activity value, *i.e.,* the expected effects of the activity on cost, schedule, and technical performance. To account for uncertainty, activity value is determined using the *expected values* of the stochastic activity attributes. Since the expected value is the average of the outcomes represented by a PDF, decisions are often based on this value (assuming a "risk-neutral" decision-maker).

To obtain the real value of the activity for the project in the changing process environment, the expected risk reduction achievable through the application of the activity is determined in the *second step* of activity value calculation. As relative activity value depends on the state of the project, ASDP applies dynamic weighting factors to emphasize the need to improve performance in the critical areas identified during the calculation of the actual risk in the project:

$$\begin{aligned} &R^{A}_{TPM1} = \text{high} \Rightarrow w_{TPM1} = 3x \\ &R^{A}_{TPM2} = \text{moderate} \Rightarrow w_{TPM2} = 2x \\ &R^{A}_{TPM3} = \text{low} \Rightarrow w_{TPM3} = x \\ &\sum w_{TPMi} = 1 \Rightarrow w_{TPM1} = 0.5; w_{TPM1} = 0.33; w_{TPM3} = 0.167 \end{aligned} \qquad \textbf{(K-3)}$$

These weighting factors are then used in the following formula to calculate the expected technical performance risk reduction to be provided by an activity:

$$R^E_{TP} = w_{TPM1}R^E_{TPM1} + w_{TPM2}R^E_{TPM2} + ... + w_{TPMn}R^E_{TPMn} \text{ where } \sum_{i=1}^{n} w_{TPMi} = 1 \qquad \textbf{(K-4)}$$

where R^E_{TP} is the expected overall technical performance risk after the activity, R^E_{TPMi} are the expected technical performance risk values in the key areas of technical performance represented by the TPMs of the project after the activity, and w_{TPMi} are the dynamic weighting factors obtained from the actual technical performance risk levels using *Equation K-3*.

The expected overall project risk after the activity can be calculated once the risks in all three performance areas have been determined using the following formula:

$$R^E_{project} = w_S R^E_S + w_C R^E_C + w_{TP} R^E_{TP} \qquad \textbf{(K-5)}$$

where $R^E_{project}$ is the expected overall project risk after the activity, R^E_S is the expected schedule risk after the activity, R^E_C is the expected cost risk after the activity, R^E_{TP} is the expected technical performance risk after the activity, and w_S, w_C, and w_{TP} are the weightings for the process variables used in *Equation K-3*.

The relative value of an activity for the project can now be calculated as the expected risk reduction provided by the activity:

$$V^E_{rel} = R^A_{project} - R^E_{project} \qquad \textbf{(K-6)}$$

where v^E_{rel} is the expected risk reduction, *i.e.,* the expected difference in overall project risk before and after the activity. Using the determined relative activity values, the activity selection in ASDP is performed during process simulation instead of before, as with traditional approaches.

K.4.4. Activity Selection

Process definition in ASDP is done starting from a *grand process space* that includes all the possible process states and all activity options relevant for the project. In this grand process space, all potential activities are included to facilitate the better handling of ambiguity during process planning. For example, if there are two modes for a certain activity with similar outcomes, but with different confidence levels, V&V methods, procedures, or coverage, and the planners cannot decide which mode to choose for the plan, both modes are included in the process for potential use. The DSM in *Figure K.7* depicts an illustrative grand process space. Diamonds in the DSM show contingent or conditional dependencies between activities, whereas the filled circles are the non-contingent relations. Contingency means that a certain activity or activity mode can be followed by various activity modes with similar goals, but different focus, input needs, or activity values. In case of contingent relations, the more suitable activity (*i.e.,* the better option) for the process is chosen at the point immediately before one of the contingent

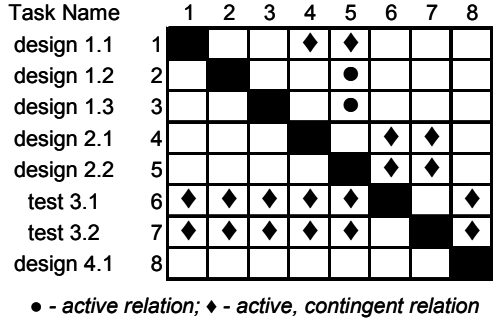


Figure K.7 ASDP grand process space

activities must begin.

Activity selection in ASDP is a two-staged procedure. *First*, the availability of all possible activities is determined by comparing their entry criteria to all available input information. All activities whose entry criteria are met are potential candidates. *Second*, the activity with the highest impact on the overall project value (the activity with the highest relative value) is chosen (activated) from the set of potentials.

K.4.5. Process-State-Based Iteration Modeling

Another key aspect of process modeling is the modeling of iteration cycles. Probabilistic iteration models, which were discussed in *Chapter I.2.1,* assumed that the risk of rework depends on the probability of change in the input of the activity multiplied by the impact of that change on the activity. However, in practice, these probabilities are difficult to determine a priori. Furthermore, probabilistic iteration models were designed to support work*flow*-driven project planning, and they fail to address the needs of work*state*-driven project management.

A key reason for design iteration is that the planned performance of the design is not achieved through the originally planned process. That is, technical performance risk incorporated in the design was not reduced to the planned level. Another characteristic of design iteration is that decisions on iteration usually affect the existing process architecture. That is, new activities with a better focus on the discovered problems are applied instead of just reattempting the activity that produced the unsatisfactory result. Consequently, risk reduction and rework effectiveness are key drivers of iteration planning, which have to be considered during process modeling.

When it is time to decide whether to iterate or not, the likelihood of project success is assessed considering the technical status of the project and availability of resources. This process-state-based decision on iteration is modeled in ASDP using the results of the risk calculations concerning the actual process variables (R^A_C; R^A_S; and R^{ASDP}). *Figure K.8* shows an example decision tree used in ASDP for iteration planning. As depicted, the satisfaction of the technical performance requirements is fundamental for the project success, so technical risk always has the highest priority during decisions on iteration (*i.e.,* $w_{TP} > w_S > w_C$). In case of high technical risk, rework is usually essential (unless the requirements can be eased). Conversely, lack of risk diminishes the need for rework. The branch in the middle, with moderate technical risk, is the most usual case in SD projects. This case often leads to a more specific analysis of the product and process characteristics to reach a good decision. In *Figure K.8*, schedule is more important than cost, so it is considered next, after technical performance.

A decision tree such as this one is dynamically formed at each step of the process to support implementation of the main aspects of design strategy in ASDP, leading to better decisions and more realistic process simulation. Alternatively, a *weighted average* or *multi-attribute utility function* could be used to find the best decision in a one-stage decision tree.

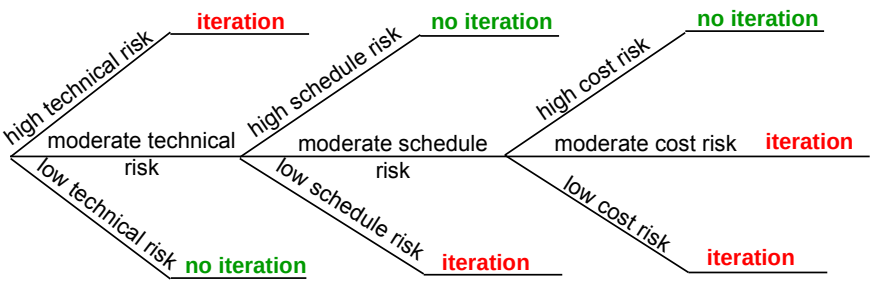


Figure K.8 Decision tree for risk-based iteration

K.4.6. Discrete Event Simulation Basics

The ASDP model applies discrete event simulation to compute the distributions of process duration, cost, and product technical performance based on the approach in [Browning & Eppinger 2002]. The ASDP simulation model improves with Browning & Eppinger's approach in three areas. *First*, it considers product technical performance in addition to process duration and cost. *Second*, it applies adaptive activity selection to allow for continuous process improvement and value maximization during simulation. *Third*, it employs a dynamic, risk- and opportunity-based decision on iteration, instead of using a priori probabilities and impacts.

The application of discrete steps during simulation allows the incorporation of project-state-based decisions after each activity. That is, during each state of the simulation, the performance of the deliverables is evaluated and suitable activities are selected to adapt the process architecture to the dynamically changing process states. This improves process performance and continuously maximizes the value added. The steps of the simulation algorithm are depicted in *Figure K.9*. The model simulates a series of state transitions in multiple paths using the previously determined activity durations as simulation steps. After each step (*i.e.,* after each finished activity) the same algorithm is performed in the model to determine the state of the project and to select the next activity.

Figure K.10 shows three states of an illustrative process to provide some insights into how the ASDP simulation algorithm works. The grand process in *Figure K.10* includes both "wide scope" activity modes, which can be considered as general design and test activities (activities with x.1 numbering; *e.g., "design 1.1"*) and rework modes focused on specific failure correction procedures (activities with x.2 or greater numbering; *e.g.*, "*design 1.2"* and "*design 1.3"*). Rework activity modes have lower fidelity (*e.g.,* they affect only 1-2 TPMs), cost, and duration due to their reduced scopes. Furthermore, the entry criteria for these activities are "higher" because they are special activities to correct failures; thus, a certain level of performance is required from the design to perform these modes.

The DSMs in *Figure K.10* use the following notation: *solid black circles* depict active activity relations, *empty circles* represent inactive relations (*i.e.,* paths not chosen), and *diamonds* show yet-to-be-determined contingent relations. Additionally, the boxes on the diagonal show the status of the activities during the activity selection procedure: *checks* (✓) depict previously performed activities, *question marks* show potential next activities, *stars* in the diagonal represent the activities selected for the next process step, and the *"X"* show the inactive activities not selected.

The three states (shown on the three rows) in *Figure K.10* demonstrate the evolution of the

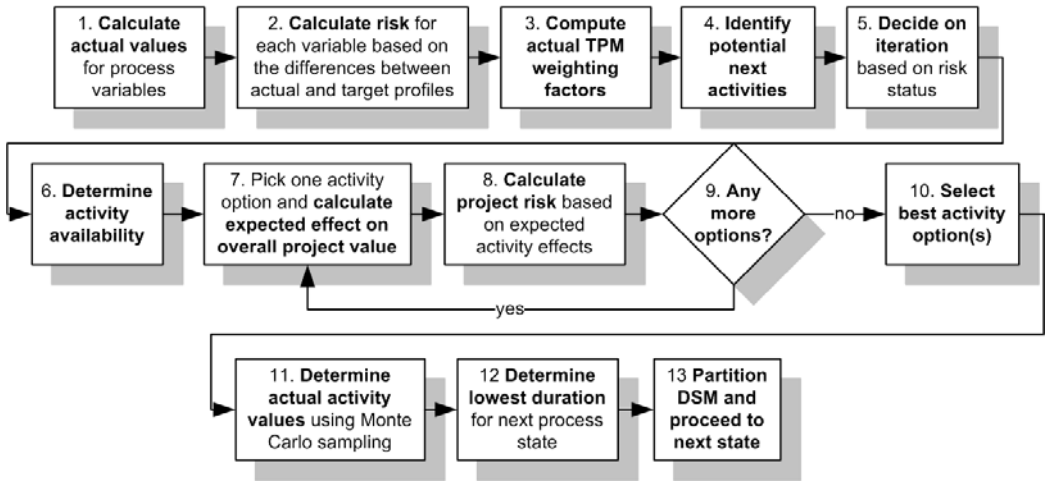
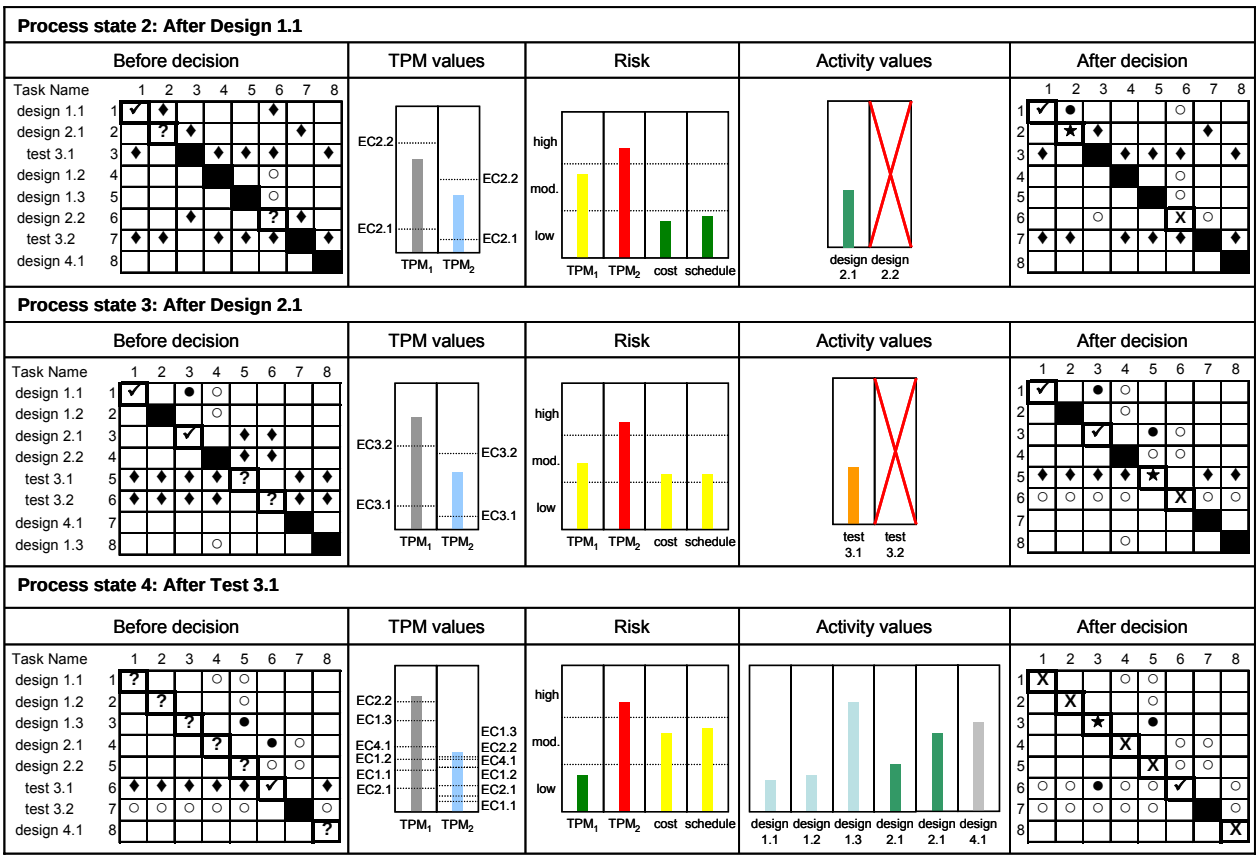


Figure K.9 ASDP simulation algorithm



● - *active relation;* ○ - *inactive relation;* ♦ - *contingent relation;* ✓*– finished activity;* ? *– potential activity;* ★ - *selected activity;* **X** *– inactive activity; EC – entry criteria, TPM – technical performance measure*

Figure K.10 Three states of the Discrete Event Simulation in the ASDP

process architecture during project simulation. Each row first contains a DSM depicting the starting point of the activity selection procedure. The activities with a question mark show the potential candidates for the next simulation step; these activities will be evaluated during the selection procedure. The second diagram in each row shows the evolving performance of the TPMs relative to each potential activity's entry criteria. Performance improvement is usually the result of design activities, while V&V activities reduce uncertainty in the TPM values.

The third diagram in each row shows the risk status at that point in time. In the beginning, the performance measures are immature and thus represent high risk, whereas the programmatic parameters (cost and schedule) are not. As the process unfolds, technical performance uncertainty and risk diminish (for *"TPM_1"*), but programmatic risks increase (as the allocated resources are spent). The fourth diagram in each row shows the expected change in value expected from each of the potential next steps in the process and helps to determine which activity should be chosen. Finally, the DSM on the far right of each row shows the resulting process after the decision.

On the one hand, the number of potential activities grows as the performance level of the design increases. On the other hand, the relative value of each activity also changes as the risk changes in the project. As depicted in the last row of *Figure K.10*, the first design cycle did not achieve its targets; *"TPM_2"* is still at high risk. This means iteration for the process with activities that have a high potential to improve the value of TPM2. *Figure K.10* shows that *"design 1.3"* is the best option for the next process state, so the process continues with this activity. Iterations are continued until technical risk is reduced to an acceptable level, or until the resource expenditures increase the cost and schedule risk problematically.

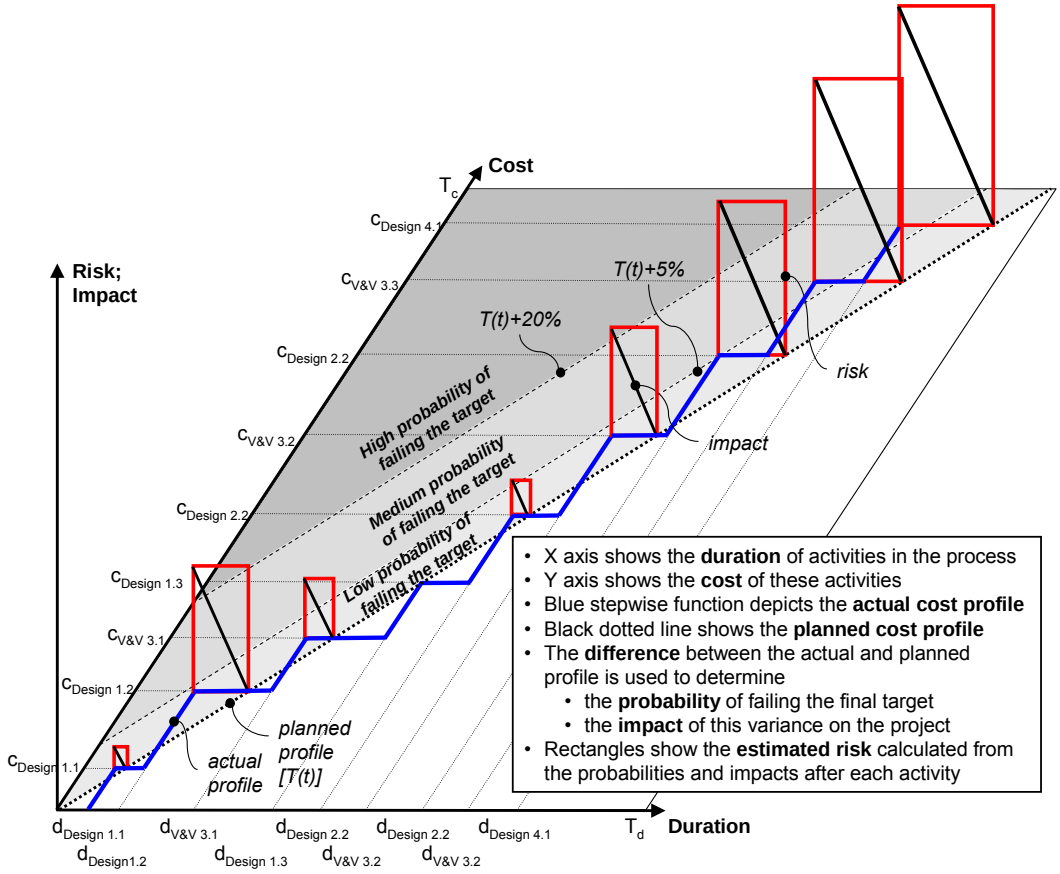


Figure K.11 Changing risk values during ASDP simulation

The DSMs on the far right show the results of each simulation state, the activities with the stars that were selected for the next step, and the ones with the "X" that were not selected. The two DSMs in the third state illustrate the dynamic behavior of the process elements during simulation. Though the activities *"design 1.1"* and *"design 2.1"* were already selected for the process previously, the decision on iteration changed their status. That is, these activities became potential activities for the next process step (iteration) and are evaluated together with the other candidates. Thus, the *"✓"* in the diagonal transformed into *"?"* (first DSM) and then into *"X"* (second DSM) after the decision on the next step was made.

Figure K.11 shows the changing cost risk status during the discrete event simulation. The area of the rectangles in the z-axis dimension depicts the cost risk. Note how it grows towards the end of the project. Since cost has the lowest priority among the three main project objectives, iterations were done to reduce technical performance risk, in spite of the dangerous increase in cost risk.

K.4.7. Adaptation of the Target Profiles

In the example in the previous section, the assumption was made that both the programmatic constraints and the technical performance targets are known before the simulation starts. However, one major goal of process modeling during project planning is to determine realistic cost and schedule targets that enable the maximal fulfillment of the technical project goals. Hence, ASDP provides the possibility to apply discrete event simulation to obtain all probable outcomes for the process variables. These outcomes define a stochastic design space for the activity network (*i.e.,* a process space).

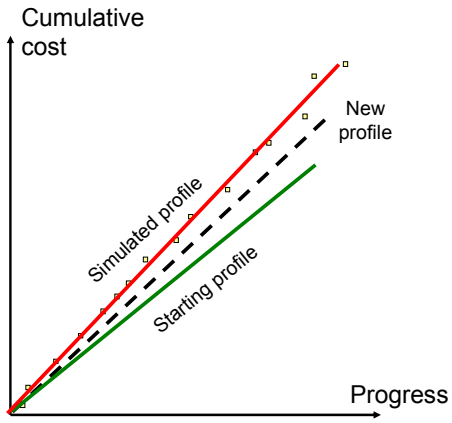


Figure K.12 Adaptation of the starting profile to the simulation results

At the beginning of the ASDP simulation, the project manager arbitrarily (or perhaps with customer or marketing guidance) defines the project's cost and schedule targets and their probability functions (*e.g., Figure K.6B*). For example, he or she calculates the total duration of the regular activities (first modes) on the critical path and adds 20% buffer for iterations. Additionally, the probability functions for the risk calculations have to be defined similarly to the ones in *Figure K.6B*. These values are then considered as starting values for the simulation.

Even though *Figure K.12* shows the target profile as one linear, in reality, it is rare that target values of different phases fit on one linear function. Hence, the adaptation of the starting profile in the ASDP tool is done separately for each phase, and the result is then not a linear, but a piecewise linear function. This is an important aspect of the ASDP simulation algorithm, since the application of one single linear target profile could lead to wrong simulation results.

Figure K.13 depicts the results of the ASDP simulation for project cost. The TriPDFs depict the probable values of the cost profiles recorded during the discrete event simulation. These data are valuable information for the project manger about the possible and likely cost trajectories of the project. With this information, project planning can define realistic management reserves that provide the project with the required resource flexibility.

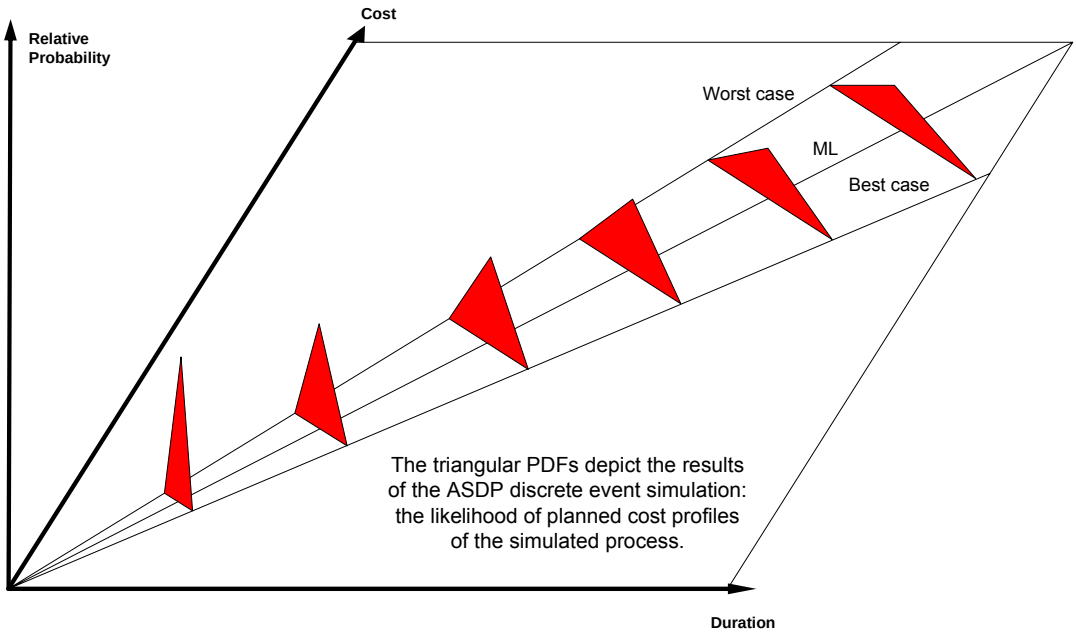


Figure K.13 Results of the initialization of the ASDP simulation

Task Name		1	2	3	4	5	6	7	8
design 1.1	1	3			0.4	0.1			
design 1.2	2		2			0.35			
design 1.3	3			1		0.15			
design 2.1	4				3		0.35	0.05	
design 2.2	5					2	0.09	0.51	
V&V 1	6	0.03	0.1	0.15	0.05	0.01	3		0.05
V&V 2	7	0.02	0.2	0.2	0.04	0.05		2	0.1
design 4.1	8								3

Activity priority: 3 – high; 2 – medium; 1 – low
Relation frequency: 0 – 1

Figure K.14 Illustrative ASDP output: Activity relation probabilities

K.4.8. Model Outputs

The goal of ASDP is to identify a core process, a network of activities that add the most overall value to the project. The core process is the one at which the highest design performance is achieved for the lowest process cost and duration. Since the output of each simulation run can be a different process, during the evaluation of the simulation results, those activities that constitute the most frequent building blocks of the processes with the highest value can be identified. That is, the output of the ASDP simulation is a network of activities, where each activity is prioritized based on its contribution to the overall process value.

Figure K.14 shows one simulation output, a DSM. The boxes in the diagonal depict the priority of the activity—*i.e.*, how often was that activity present in the high-value processes? The off-diagonal cells show the frequencies of the relations used during the simulation runs. The activities that add most value in most situations (those with the highest priority) build the core process. Using the identified core process, the expected performance of the single activities and the whole core process can be calculated and compared to the project requirements and constraints to assess the feasibility of the project. The off-diagonal frequencies provide potential inputs to probability-based iteration models such as [Browning & Eppinger, 2002]; such inputs may be superior to the probabilities elicited from project participants and managers.

Another ASDP output is depicted in *Figure K.15*. The estimated target profiles (see also) are main sources of valuable information on the project in the planning phase. The variance of the final parameter values, the diversity of possible performance profiles, the risk incorporated in each process option, and the expected target values and profiles are pieces of information that significantly improve the project manager's knowledge of the project, reduce uncertainty, and support the consideration of process failure modes during planning.

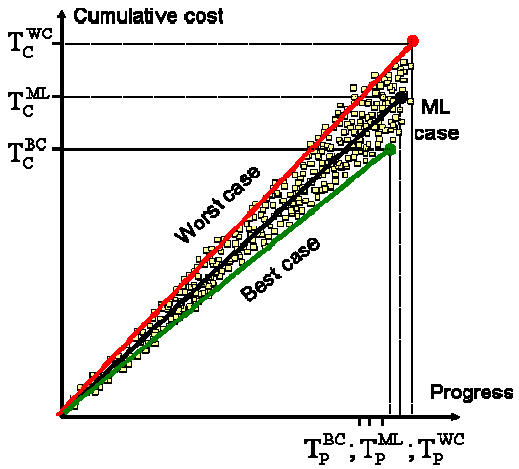


Figure K.15 Estimated target profiles

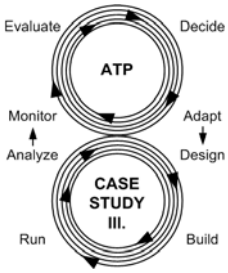
K.5. CHAPTER SUMMARY

The theory of the work*state*-driven process modeling method *"Adaptive System Development Process (ASDP)"* was introduced in this chapter. ASDP models the steps of both loops of the adaptive SD framework, and thus it simulates the process of controlled learning in the adaptive SD enterprise. During the multiple discrete event simulation runs, the flexible process space is explored and all possible process states and scenarios are considered for given project objectives. The resulting process alternatives (*i.e.,* alternative paths in the flexible process space) provide the planners with solutions of the expected and unanticipated process failure modes and thus, support both project planning and adaptation. Further, the possible outcomes for the target profiles define the range of resource needs in the project that helps adjust the buffer sizes for the lifecycle phases.

L. CASE STUDY III – ADAPTIVE PROCESS MODELING AT TETRAPAK CARTON AMBIENT

L.1. CHAPTER ABSTRACT

The validation of the *Adaptive System Development Process (ASDP)* method and software tool is the main topic of this chapter. Hence, the goal here is to prove the validity and applicability of the ASDP concept using real industry data from the pilot projects at TetraPak Carton Ambient. The results of *Pilot Project IIc* described here show how a flexible process plan could be generated from alterative process plans at TetraPak, and demonstrate the value of this adaptive process modeling technique for work*state*-driven project management.



L.2. PILOT PROJECT IIC – IMPLEMENTATION AND VALIDATION OF THE ASDP METHOD

L.2.1. Pilot Project IIc Description

The ASDP method described in the previous chapter was implemented in a software tool that is based on the VVTPM software environment. This enables the transfer of process models from the VVTPM to the ASDP tool, and facilitates the comparability of the simulation results from the two different tools.

As mentioned before, *Pilot Project IIc* was a "virtual" pilot project. That is, the validation of the ASDP theory and software tool was conducted using the planning data from *Pilot Project IIa* and *IIb* to prove the validity of ASDP as an alternative planning technique for the VVTPM.

The structure of decision points and the evaluation criteria, including the lifecycle phase targets for all dimensions of process performance, as well as the impact functions were the main inputs for ASDP process modeling from *Pilot Project IIa*. This information could be imported from the VVTPM tool to avoid unnecessary programming and planning effort. The resulting target profiles for project cost and product technical performance as functions of planned project duration are depicted in *Figure L.1*.

There is one main difference between the two software tools concerning process evaluation, *i.e.,* the way risk is calculated in the tools. While in the VVTPM tool, risk is a measure of estimated profit loss in Euro (€), risk values in ASDP are between 0.00 and 1.00 to allow for the determination of risk levels during process adaptation (0.00-0.33 low; 0.33-0.66-medium; 0.66-

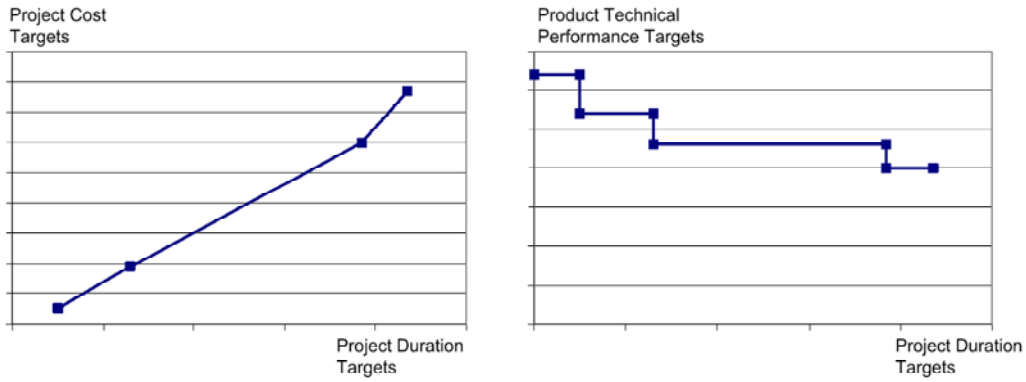


Figure L.1 Cost and technical performance profiles in Pilot Project IIc

1.00 high). Hence, the impact functions defined in the VVTPM tool using monetary units (€) had to be normalized and the impact values had to be converted into numbers between 0.00 and 1.00. During this procedure, the highest impact values among the monetary impact functions were sought and set to 1.00. Then numbers between 0.00 and 1.00 were assigned to the representative values of he impact functions without changing the basic form of the function.

Besides the evaluation criteria, the activity network had to be defined in the ASDP tool. As the previous section proposed, ASDP models adaptive project management, where the leader of the project applies a *flexible process space* instead of strict plan to guide the SD work. This flexible

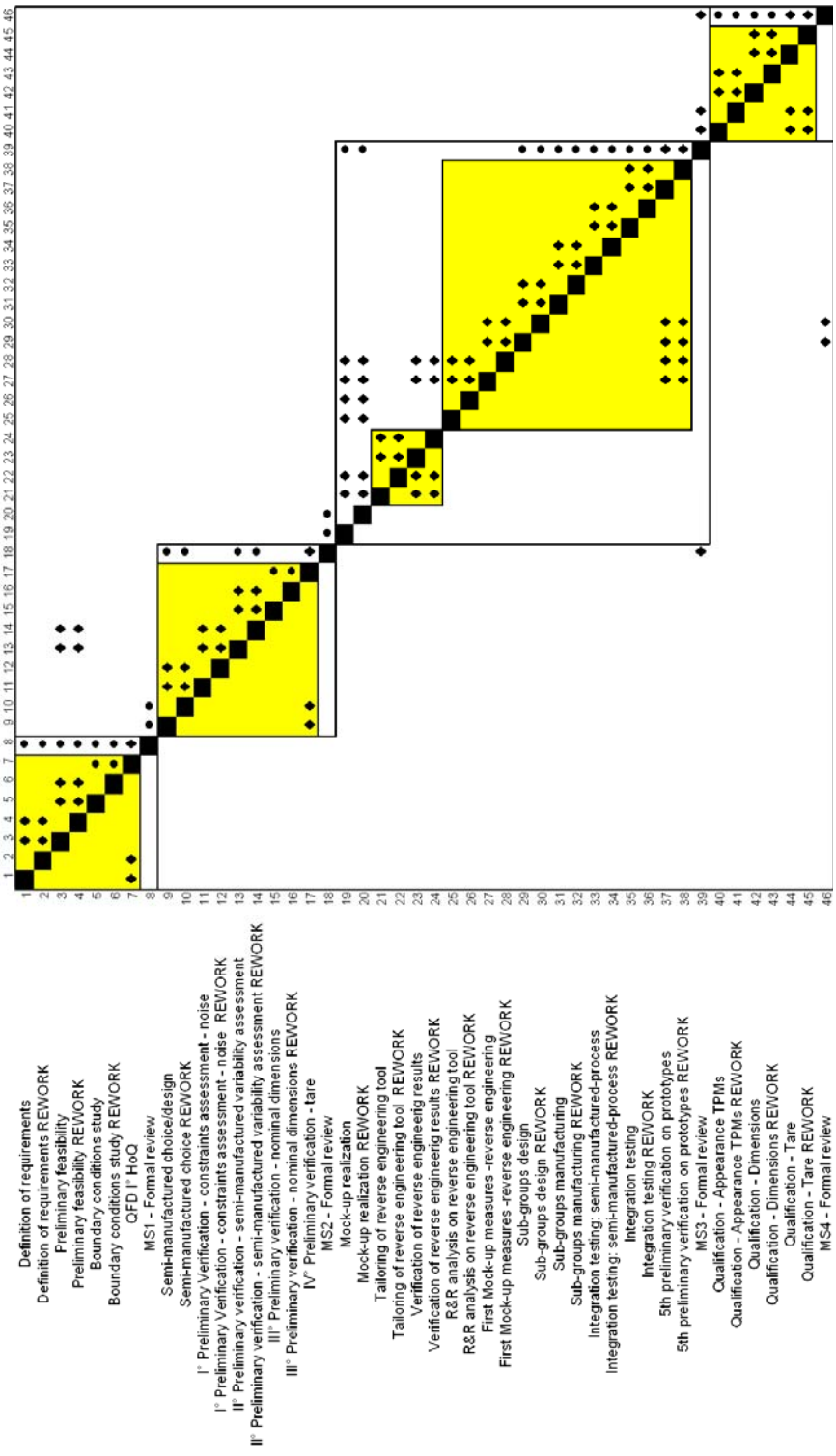


Figure L.2 Grand process space in Pilot Project IIc

process plan drives the project towards maximal stakeholder value by enabling and even forcing the manager to select the best activities for the places in the SD process, where the planning team defined flexibility on the plan, because they could not decide on the best solution due to high uncertainty or ambiguity.

Although the project management system at TetraPak has been moving towards adaptiveness and the *decision support framework* implemented in *Pilot Project IIa* allows effective project adaptation, Project Planning could not be convinced to use the ASDP tool to experiment with it. Hence, the process variants defined in the VVTPM in *Pilot Project IIb* were applied to validate the ASDP tool and the decision to build the ASDP tool on the VVTPM software environment instead of developing a new, standalone solution proved to be a good decision from the validation point of view.

During process definition in the ASDP tool, a *grand process space* is defined involving all activities and activity modes relevant for the project. At TetraPak, two of the four process variants in Pilot Project IIb were selected as input for process definition. One of the two selected process plans, *"Strategy 3",* included additional SD efforts in reverse engineering and analysis, *i.e.,* two activities forming an iteration loop to be conducted sequentially in the process. In the other process model, in *"Strategy 4"*, an analysis task was included on repeatability and reproducibility (R&R) instead of the iteration loop.

The resulting grand process space is depicted in the DSM in *Figure L.2*. This flexible process includes the following activities not part of the basic process models described in the VVTPM case study (see *Figure J.2*):

- *Rework modes* for each activity with effect on the TPM "package width". During the definition of these activity modes, activity sizes were reduced by the learning effects defined in the VVTPM (*i.e.,* 0.3 in Phase 1 and 2, 0.5 in Phase 3, and 0.6 in Phase 4). The new activity modes were included in the grand process architecture as phantoms of the original ones, *i.e.,* with the same relations to other activities.

- *Two activities with rework modes in Phase 3*. As mentioned before, the *Strategies "3"* and *"4"* differed in the activity networks of Phase 3. Hence, during the definition of the grand process space for the project, the additional activities were included in the original process model (see *Figure J.2*). As *Figure L.3* shows, three new activities and three rework modes were incorporated in the process space after the activity *"Mock-up Realization"*. So, if the discrete event simulation arrives at the state after *"Mock-up Realization",* it has to decide which of the two alternative process paths have more improvement potential for the project, and choose the best of the four respective activity modes.

The grand process space generated this way includes all feasible activity options that were considered during planning. This process could now be analyzed using the ASDP simulation algorithm.

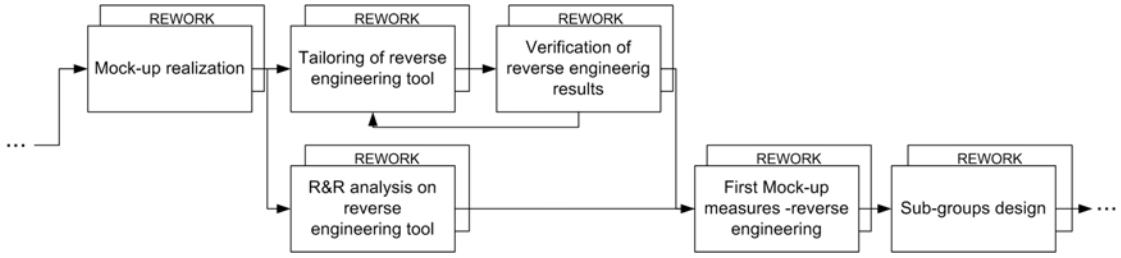


Figure L.3 Two alternative process paths in the ASDP grand process space

L.2.2. Overall Simulation Results

To analyze the behavior of the grand process space defined in the ASDP tool, 1000 discrete event simulation runs were conducted. In each run, the activity characteristics are different including the effects on all relevant process parameters (project cost, duration, and TPMs).

The overall results of the ASDP simulation regarding the three process attributes are depicted in *Figure L.4*. As the charts present, the results do not show any positive or negative trend, and thus the whole range of the simulation runs can be considered as feasible for the evaluation. The charts in *Figure L.4* also highlight an interesting characteristic of the SD process, *i.e.,* the form of project cost and duration are similar and thus they correlate. This is not necessarily true for technical performance, even if design iterations in ASDP are driven by the actual technical risk status.

During the analysis of the resulting process architectures after the ASDP simulation, some astonishing results were found. In the TetraPak grand process space, 731 different process schedules were detected due to the high number of activity options. Since conventional workflow-driven process modeling works with only one process option, this result alone underlines the demand for better project scheduling methods that have the capability of simulating changes in the process design during process analysis.

The simulation outcomes for project cost and duration in the last diagrams in *Figure L.4* show that process changes and unplanned design iteration in the SD project due to inadequate quality are quite likely during the project and these will lead to project cost and duration higher than the specified targets. Even though the simulation outcomes include many quite pessimistic process scenarios, the cost and duration of the majority of the generated process schedules exceed the preliminarily set targets that should be considered during planning. That is, the project plan intrinsically includes considerable programmatic risk that can result in higher resource consumption than planned.

The next representative simulation output depicts the frequencies of activities and likelihoods relations after the simulation in the TetraPak grand process space. As *Figure L.5* presents, while some activities were very "popular" during simulation, others were picked quite rarely by the ASDP algorithm. For example, in the first lifecycle phase, rework was conducted more often through regular activity modes than through rework modes. This information that calls for management attention concerning the sizing of conventional and rework modes can be derived from the activity frequencies in the diagonal of the DSM.

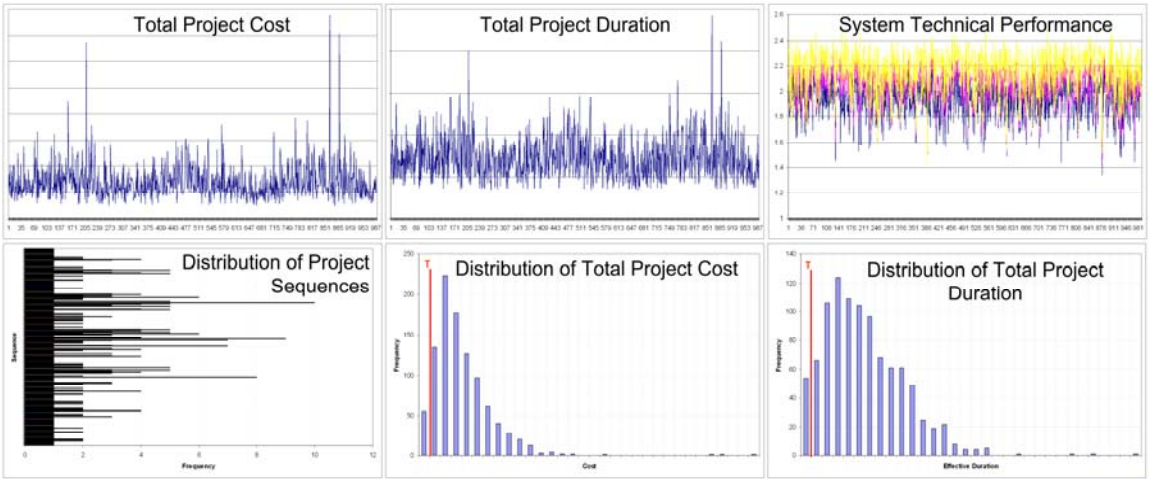


Figure L.4 Simulation results for the three process attributes

Activity	Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45
Definition of requirements	0	990		0.26	0.26				0.26																																					
Definition of requirements REWORK	1		253	0.07	0.07				0.07																																					
Preliminary feasibility	2			1,094		0.28	0.28		0.28					0.28	0.28																															
Preliminary feasibility REWORK	3				149	0.04	0.04		0.04					0.04	0.04																															
Boundary conditions study	4					1,106		0.29	0.29																																					
Boundary conditions study REWORK	5						137	0.04	0.04																																					
QFD I° HoQ	6	0.07	0.07					1,243	0.26																																					
MS1 - Formal review	7								990	0.26	0.26																																			
Semi-manufactured choice/design	8								210			0.05	0.05																																	
Semi-manufactured choice/design REWORK	9									2.691		0.70	0.70						0.05																											
I° Preliminary Verification - constraints assessment - noise	10										1,918			0.50	0.50				0.70																											
I° Preliminary Verification - constraints assessment - noise REWORK	11											988		0.26	0.26																															
II° Preliminary verification - semi-manufactured variability assessment	12												2,344		0.61	0.61		0.61																											
II° Preliminary verification - semi-manufactured variability assessment REWORK	13													567	0.15	0.15		0.15																											
III° Preliminary verification - nominal dimensions	14														2,632		269	0.07																											
III° Preliminary verification - nominal dimensions REWORK	15																	2,901	0.26																											
IV° Preliminary verification - tare	16									0.50	0.50								1,798	0.26	0.26																										
MS2 - Formal review	17																			852		0.22	0.22			0.22	0.22																			
Mock-up realization	18																				139	0.04	0.04		1.00	0.04	0.04																			
Mock-up realization REWORK	19																					3,840	595	0.14	1.00																					
Tailoring of reverse engineering tool	20																					0.72	0.72	3,491	0.14																					
Tailoring of reverse engineering tool REWORK	21																					0.17	0.17		904																					
Verification of reverse engineering results	22																									10																				
Verification of reverse engineering results REWORK	23																										2																			
R&R analysis on reverse engineering tool	24																											946																		
R&R analysis on reverse engineering tool REWORK	25																												89	0.02	0.02														
First Mock-up measures - reverse engineering	26																													611	0.02	0.02													
First Mock-up measures - reverse engineering REWORK	27																														718	0.11	0.11												
Sub-groups design	28																															108													
Sub-groups design REWORK	29																																927	0.24	0.24										
Sub-groups manufacturing	30																																	621		0.16	0.16								
Sub-groups manufacturing REWORK	31																																		414	0.11	0.11								
Integration testing: semi-manufactured-process	32																																			30		0.01	0.01	0.01					
Integration testing: semi-manufactured-process REWORK	33																																				1,005	0.26	0.26	0.26					
Integration testing	34																																					763	272	0.19						
Integration testing REWORK	35																																						272	0.07						
5th preliminary verification on prototypes	36																																							991	0.26	0.26				
5th preliminary verification on prototypes REWORK	37																																								179		0.05	0.05			
MS3 - Formal review	38																		0.21																								1,236	0.32	0.32			
Qualification - Appearance TPMs	39																																									1,328		0.35	0.35		
Qualification - Appearance TPMs REWORK	40																																										89	0.02	0.02		
Qualification - Dimensions	41																																											1,143	0.20		
Qualification - Dimensions REWORK	42																																											272	0.06		
Qualification - Tare	43																																												990		
Qualification - Tare REWORK	44																																													
MS4 - Formal review	45																																													

Figure L.5 Frequencies of activities and likelihoods relations in the ASDP simulation results

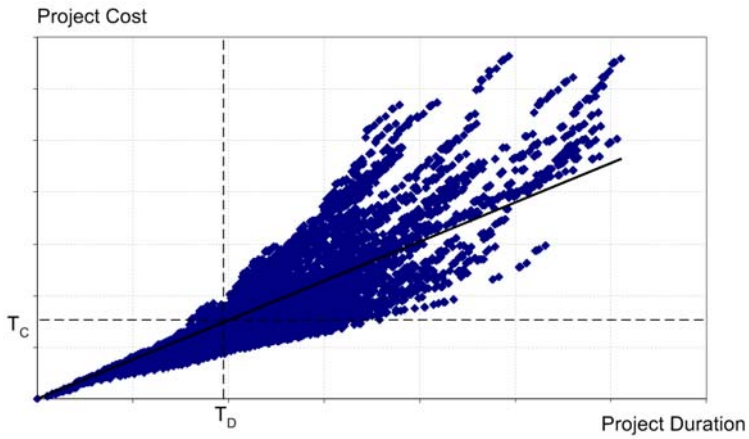


Figure L.6 Actual project cost profiles after simulation

It is also important to note that the *rework modes* were not only active during rework simulation, but they could be selected also for the first iteration loop. Though the entry criteria were set higher than for regular activity modes, many times they were conducted in the first loop. On the other hand, regular activity modes were sometimes accomplished as rework activities. The reason for this phenomenon can be found in the characteristics of the process adaptation algorithm. The algorithm always picks the activities with the highest effect on the overall project value considering the actual risk status. Thus, on the one hand, in process states with high technical risk it selects regular modes for rework, and on the other hand, it will decide for lower required effort even in the first iteration loop if technical risk is low or programmatic risk is high in the project.

Another interesting result that can be obtained from *Figure L.5* refers to the success of the two process branches in *Phase 3*. According to the activity frequencies, the iteration loop with reverse engineering activities (incl. regular and rework modes) outscored the R&R activities, since

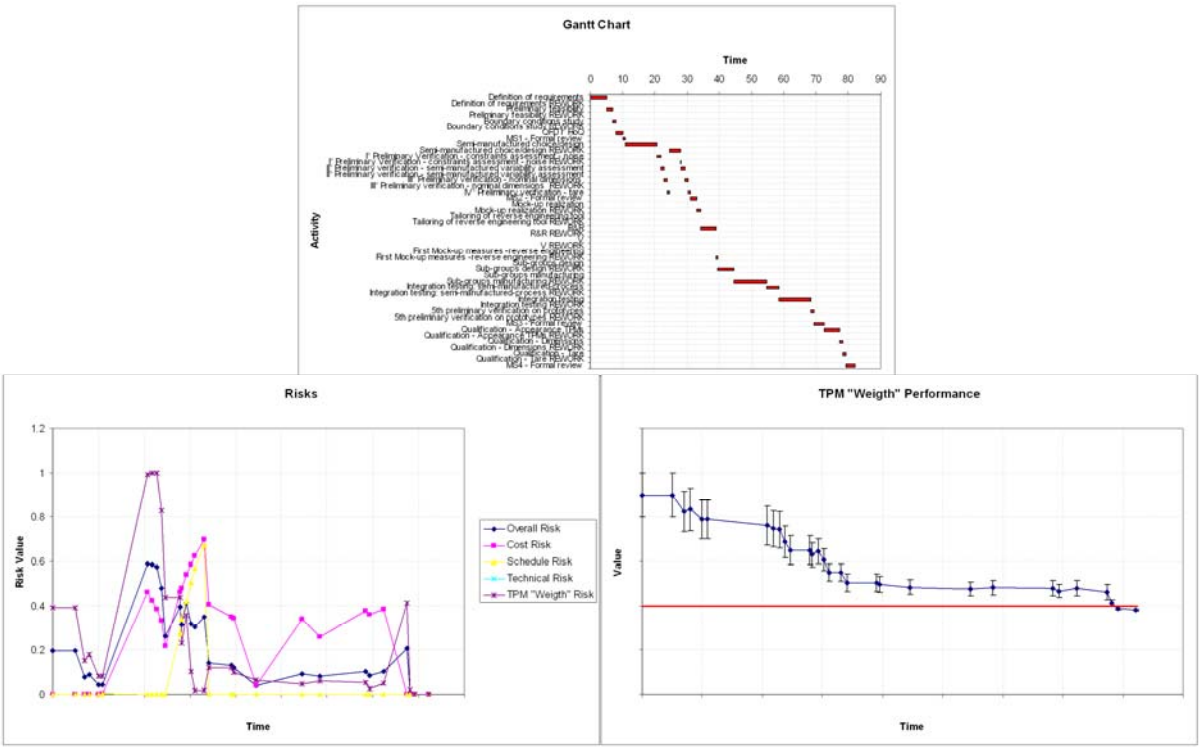


Figure L.7 Characteristics of the best process architecture (with lowest overall risk)

they were conducted 4395 times and the R&R activities only 12 times during the 1000 simulation runs.

Furthermore, the high number shows that the iteration cycle concerning reverse engineering activities were conducted at an average of 4.1 times in each simulation run. The other rework loops were repeated with a maximum of three (*i.e.,* an average of 1.25 iterations in the first phase, app. 3 iterations in the second phase, an average of 1.65 iterations in the main loop of phase 3, and app. 1 iteration in the last phase). These numbers show that the computer-based design and analysis activities in the second and third phases have a significant contribution to the overall project performance. Furthermore, design iterations in the second and third phases resolve most of the technical uncertainty in the SD project and thus guarantee a smooth qualification in the last phase of the project.

The last result concerning the behavior of the grand process space is depicted in *Figure L.6*. The actual cost profiles here show the same result as the diagrams in *Figure L.4*, *i.e.,* the planned project budget will quite likely be too low for the project. Though cost has the lowest priority among all three dimensions of process performance, it has considerable effects on the profitability and thus stakeholder value of the project. *Figure L.5* also shows the probable outcomes of project duration. This aspect is more alarming than the project cost, since delivery deadlines are main quality criteria at TetraPak. As the chart shows, the durations of a large number of process schedule overdrafts go beyond the original targets.

The actual profiles show that work*state*-driven process simulation provides a larger variety of outputs than conventional work*flow*-driven simulation. On the one hand, many of the generated process architectures are not feasible for the project requirements describing process failure modes with low likelihood. On the other hand, the overall simulation results show that the results of conventional process simulation are often too optimistic. The best process architecture found during the VVTPM simulation (*"Strategy 1"*) did not include the additional activities in *Figure L.2*, and presented a "good enough" way to accomplish the project goals. However, as the overall ASDP simulation results depict, iterations are more likely in the SD project than estimated

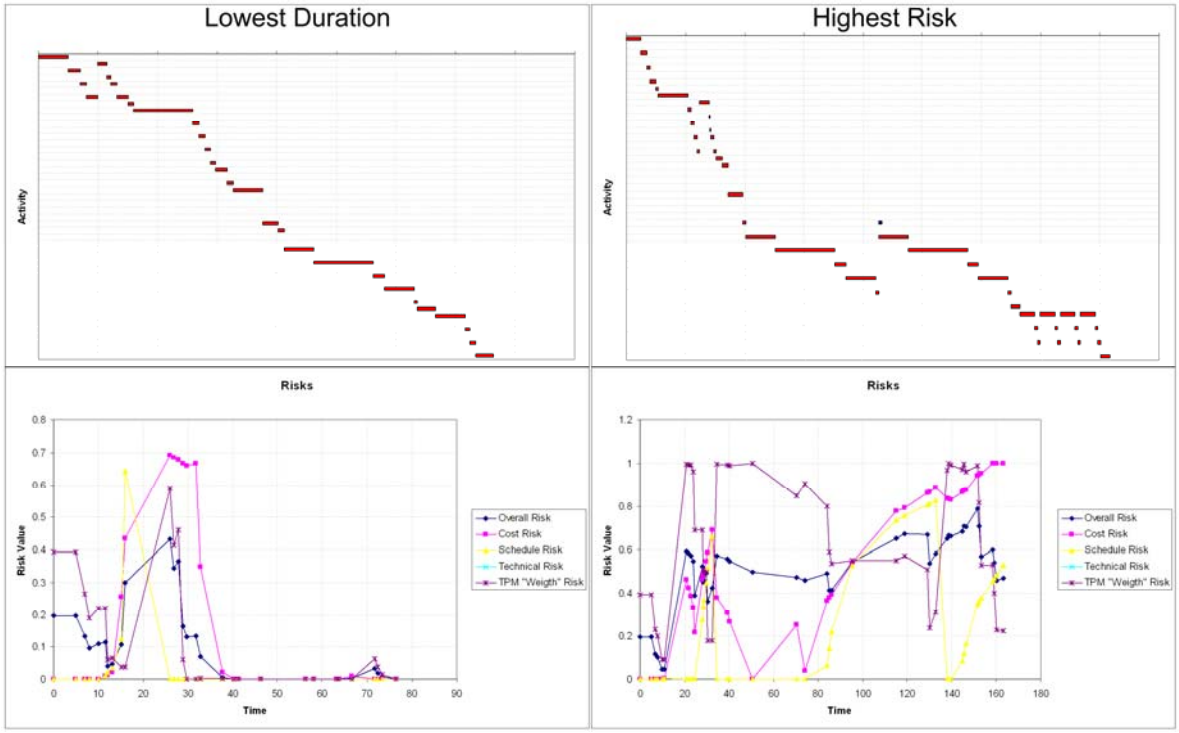


Figure L.8 Process schedule with the lowest duration and the highest overall risk

before, which contributes to considerably higher project cost and duration. Hence, it is recommended to allocate more resources to the project to deal with the technical challenges and use effective project control to identify the best points to switch to activities that are more effective in solving the actual SD problems than the ones on the basic plan (*"Strategy 1"*).

L.2.3. Representative Process Outcomes

Besides the aspects that describe the overall behavior of the grand process space, the ASDP simulation generates outputs that specify representative process schedules identified during simulation. One such process schedule is the best one, *i.e.,* the one that incorporates the lowest overall risk. As *Figure L.7* depicts, the best schedule involves only one iteration in phase 2, all the other design problems could be solved correctly at the first time. This is the main reason for low or moderate programmatic risk throughout the whole project. Though technical performance risk represented by the risk value in the TPM *"package weight"* are low at the end, it can be noticed that the effects of activities in phase 3 are rather low on the technical aspects.

Two more process schedules are depicted in *Figure L.8*. The first schedule is the one with the lowest project duration. In this process, iterations were frontloaded in the SD, resulting in early uncertainty and thus risk reduction. The second process schedule shows the opposite strategy, where long iterations were done in the product qualification phase, where SD work is most expensive. This is the reason for the rather high overall risk values at the end of the process.

L.3. CONCLUSIONS ON ASDP AND CHAPTER SUMMARY

The case study in this chapter showed that adaptive, work*state*-driven process modeling provides a much higher variety of simulation results than conventional process modeling. The high diversity of process architectures and process scenarios explored during ASDP simulation provides different inputs to project planning than project managers are used to. On the one hand, ASDP simulation generates an unexpectedly high number of different process plans that can be analyzed during project planning. On the other hand, this huge number of possible process architectures and schedules for a project is reduced by identifying the more frequent activities and the best schedules for different project objectives.

Hence, ASDP is not a substitute for conventional process modeling techniques, but a tool that provides a large variety of inputs to the project management about the possible future states of the SD and proposes actions to deal with these situations. Thus, ASDP reduces ambiguity about the project during planning and helps define a project plan with increased capability to fulfill its mission.

M. CASE STUDY EVALUATION AND SUMMARY

M.1. CHAPTER ABSTRACT

This final chapter of the thesis presents the evaluation of Case Study I and II from the industry point of view. The goal of the evaluation is to determine the overall *feasibility* of two methods implemented at TetraPak Carton Ambient: the *decision framework for system adaptation* developed to support the procedure of adaptive project control, and the *VVT Process Modeling Tool* for project planning. These two methods are now in daily use at TetraPak and provide the basis for a possible future implementation of the *Adaptive System Development Process* method discussed in the previous two chapters.

The evaluation process is supported by a generic metrics system that breaks down the feasibility of any methodology, method, or tool into criteria and measures that describe feasibility and support the detailed assessment of the implementation success of novel SD techniques in a company. The case study evaluation was carried out with Carlo Leardi, the leader and main supporter of the implementation and validation of the new methods at TetraPak Carton Ambient.

M.2. EVALUATION SYSTEM

To assess the results of the two pilot projects in the TetraPak case study, a generic assessment system was set up based on existing systems engineering evaluation frameworks [*e.g.,* INCOSE 1995, Fricke 1998]. *Figure M.1* depicts this assessment system that provided the fundament of the evaluation process. The various levels of the hierarchic tree structure classify the most important drivers of method feasibility and thus the case study success into four main categories. These four main criteria are then broken down into lower hierarchy level measures that provide the basis for the quantification of feasibility. To support the assessment, a questionnaire was

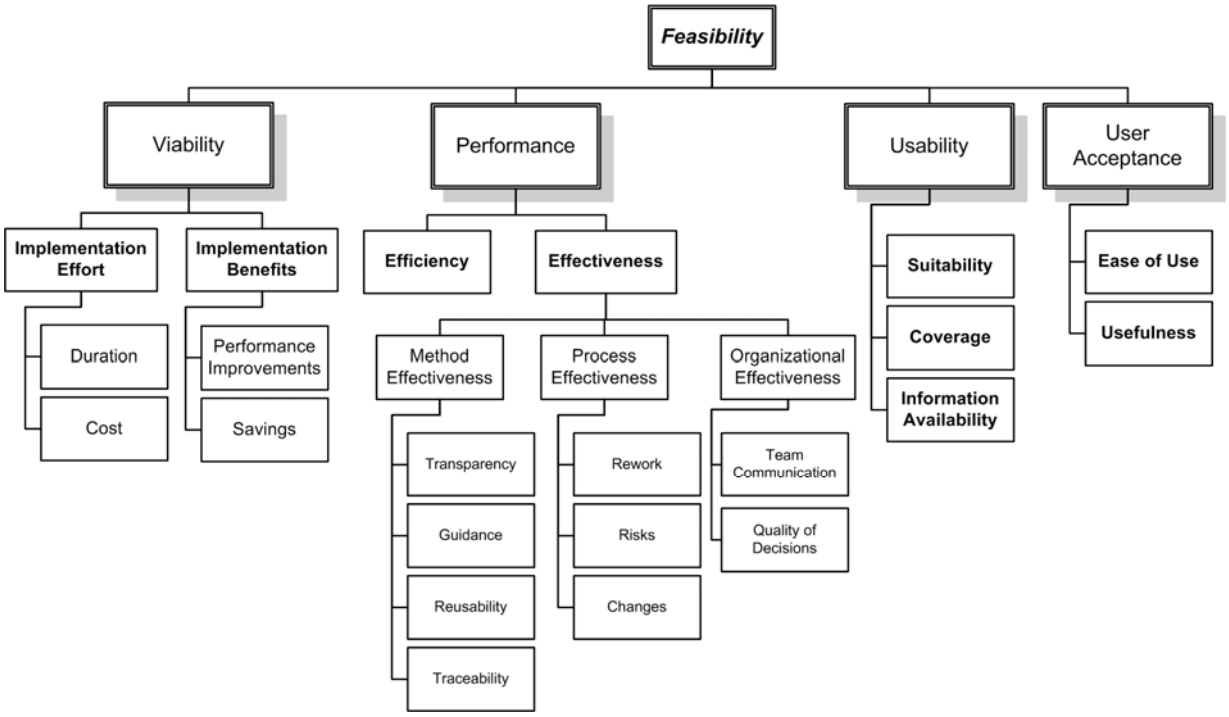


Figure M.1 Case study assessment system

generated including questions in each aspect of feasibility. The questionnaire gave the interviewee three main tasks:

- First, the level of importance of the factors (*i.e.,* criteria and measures) for the company had to be defined (first chart in *Figure M.2*). These importance levels were then used to determine parameter weightings for the computation of overall method feasibility as the weighted average of the four criteria.

- After the weightings were set, the second task of the interviewee was to choose the answers to each question from a five-level scale (second chart in *Figure M.2*) and thus indicate the level of satisfaction of a certain evaluation aspect. The assessment questionnaire included focused on questions for each specific measure to support the evaluation of the case study results. It was not possible to work with real company data due to high confidentiality at TetraPak.

- As the assessment system depicts, the main criteria are broken down into three hierarchic levels of measures. Whilst the measure at the highest level received weightings from the interviewee, measures at the two lowest levels were of equal importance. During evaluation, the values for each hierarchy level were computed from the lower level measures using the weightings and the values of the measures.

- Finally, the interviewee had the possibility to explain his or her decision in form of plain text. These comments were useful to understand the evaluation results and get feedback on the quality of both the questionnaire and assessment system.

The hierarchic structure of the assessment system in *Figure M.1* was defined on the basis of the *Goal-Question-Metric approach* of INCOSE [1995]. At the highest hierarchy level of the assessment system, the main goal of the pilot project can be found. The fulfilment rate of this high-level project goal shows the overall success of the pilot project. In case of a method introduction at an organization or in a project, the *feasibility* of the method to solve the critical problems in the intended environment is evaluated.

The second level in the assessment system depicts, the main criteria showing the fulfilment of the case study goal(s). To determine the feasibility of a method, the following four generic parameters were defined: (1) *viability*; (2) effects on *performance*; (3) *usability*; and (4) *acceptance* of the method in the case study. These four main metrics of the assessment system were identified after a review of relevant systems engineering literature on measurement and assessment [*e.g.,* INCOSE 1995, Fricke 1998, DoD & US Army 2000, *etc.*] and the comparison of their findings with the case study goals.

The third level of the assessment system contains the measures whose values are defined by the answers in the questionnaire. These measures represent the most important quantitative and qualitative aspects of feasibility. The questionnaire includes questions to determine the status of these measures and derive information on the overall success of the method introduction in the pilot project.

Points	Levels of Importance
3	High
2	Medium
1	Low
0	Zero

Levels of Satisfaction	Points
Very high	5
High	4
Medium	3
Low	2
Very low	1
Zero	0

Figure M.2 Levels used in the evaluation questionnaire

It is important to note that the assessment system describes feasibility in general. Since the author of this thesis was interested to see which areas are affected by the two implemented methods, the whole assessment system including every measure was applied in the evaluation process. The reason for this is that the purpose of the evaluation was not the mere verification of the feasibility, but the exploration of the overall value of the methods in a certain company environment. As the results of the evaluation will show, some factors in the assessment system might be conflicting, *e.g.,* a method that improves the effectiveness (*i.e.,* better outcomes) of decision-making might contribute to reductions in decision-making efficiency (*i.e.,* time to make a decision). Furthermore, a new method is usually inefficient until the users get acquainted with it.

Therefore, if the assessment system is applied in the industry for project evaluation and verification, it is important to tailor the measures to the given project goals and eliminate the aspects of feasibility not relevant for the project. This way, it can be guaranteed that project evaluation measures the fulfilment of the *desired* project goals and not the satisfaction of general metrics on feasibility independent of the actual project environment. Thus, the basic rule of measurement is also valid here: measure only what you are to achieve, because everything else only reduces the quality of the measurement process and its outcomes.

M.3.CASE STUDY EVALUATION RESULTS

This section presents the evaluation results in a bottom-up approach. First, the results in the four main areas defined by the measurement criteria are described and then the evaluation outcomes for the overall feasibility are shown. Further, evaluation results for the two methods are presented in parallel, not sequentially, to see the difference between their values for SD projects.

To describe the situation of the methods at TetraPak, it must be noted that previous versions of the VVTPM tool had already been implemented during the SysTest project the year before the case studies. Thus, the developers at TetraPak had already been familiar with the VVTPM software environment. In addition, steps of the decision framework were in use before in the SD projects of TetraPak; however, the current form of the decision-making procedure described in this thesis had never been used before. Thus, none of the methods were completely new for the developers at TetraPak, but the versions used in the pilot projects had not been known before.

M.3.1. Viability

The first criterion for method feasibility is viability, defined as *having a reasonable chance of succeeding* or *something is financially sustainable* [Merriam-Webster Online Website]. In the context of the case study, viability means that the introduced method brings more benefits to the company than the cost of introduction, operation, and maintenance, *i.e.,* it is viable or profitable for the organization to apply the method in a project. Hence, the two measures contributing to the viability of the methods are the (1) *implementation effort,* meaning the amount of required resources to implement the method in a project; and (2) *implementation benefit,* expressing the achievable benefits through the introduction of the method. The possible benefits can be broken down into the estimated value of *performance improvements* achievable through the method in the SD system (product, process, or organization) and the *savings* concerning the better utilization of resources during operation.

Figure M.3 shows that both the cost and benefit side of the implementation of the systematic decision framework for adaptive project control had positive effects at TetraPak. While the cost of implementation was *medium*, its duration was *highly* satisfying (*i.e.,* the duration was low). Furthermore, the decision framework was estimated to have considerable (*high*) effect on project performance improvement, which means *moderate* benefits for the project.

To compute the values of the measures *implementation effort* and *benefits*, the average of the equally important third-level measures were used. Hence, for the decision framework for adaptive SD, both measures received *0.7* meaning *high* user satisfaction. As the interviewee decided that both implementation effort and benefits are of the same importance for the case study, the overall viability of the decision framework could be calculated to *0.7* meaning *high satisfaction*.

The implementation effort related to the VVTPM tool in the pilot project was low contributing to *high* user satisfaction. There was also an agreement that the application of the VVTPM tool fostered the improvement of project performance significantly, leading to *moderate* benefits for the project.

Using the low-level factors the main measures for viability could be calculated. These values showed that the application of the VVTPM tool required lower effort (note that the developers had already been familiar with the tool), but the tool brought the same implementation benefits as the decision-making framework. Due to the required lower implementation effort, user satisfaction with the VVTPM tool was slightly higher than with the decision framework*: 0.75* representing *high* level of satisfaction.

M.3.2. Method Effects on SD Performance

The second main criterion of feasibility, the *effects of the method on SD performance,* explores how the implemented methods contributed to the technical characteristics of the project. *Performance* describes those operational and support characteristics of the system (or method) that allow it to effectively and efficiently perform its assigned mission over time. The support characteristics of the system include both supportability aspects of the design and the support elements necessary for system operation [DoD 2001b]. Based on this definition, performance improvements provided by a method for an SD system were classified into two aspects: contributions to the *effectiveness* and *efficiency* of the SD system.

Efficiency means the amount of output produced relative to the amount of resources (time and money) that go into the production [Wikipedia Website]. The concepts of efficiency and productivity are related to each other. Hence, efficiency is the measure of the resources required to achieve an output with a certain level of quality.

Effectiveness is the extent to which the goals of the system (or method) are attained, or the degree to which a system can be elected to achieve a set of specific mission requirements [DoD 2001b]. As system effectiveness is the extent to which the goals of the system are attained, it is broken down into the effectiveness of three major SD subsystem types in the assessment system, *i.e., method* (or tool), *process,* and *organizational effectiveness*. These measures describe the effectiveness

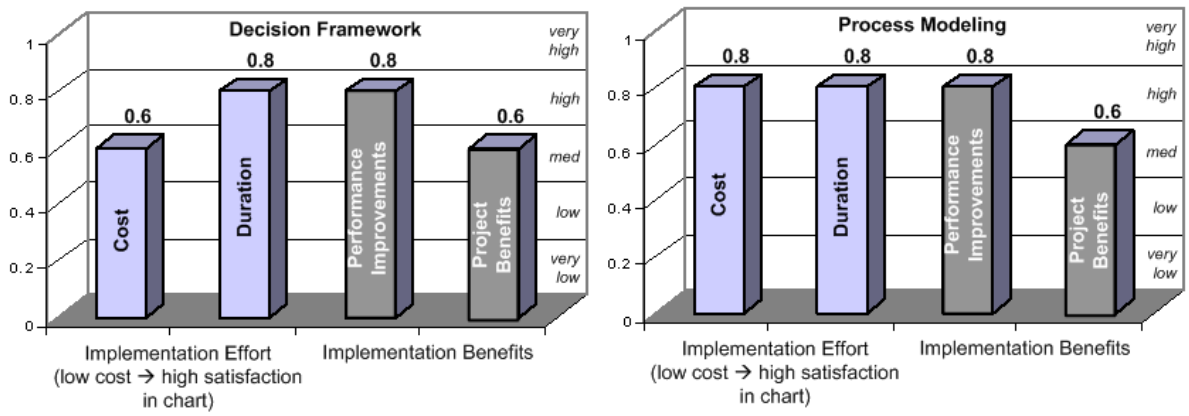


Figure M.3 Satisfaction of the measures for the criteria method viability

of the overall system. Thus, a new method implemented at a company contributes to one or more of these areas.

First, four basic lower-level measures constitute the measure *method effectiveness*, namely the *improved transparency* and *guidance* provided by the method in the project. Additionally, the *reusability* of the results of the method is a key aspect of the effectiveness of systems engineering and modeling methods. Reusability fosters the collection of project experience and lesson learned that support the continuous system improvement efforts. The *traceability* of the results to the original goals is the fourth measure that describes effectiveness.

Second, *process effectiveness* describes the *capability of the process*. The result of deliberate project planning is a process with the capability to deal with predicted and unpredictable situations in the project. The higher the capability of the process, the lower are the effects of undesired *rework*, *technical risk,* and *changes* on the final process output. Process capability can be increased through better planning and thorough control in the project.

Third, some methods foster the *effectiveness of the communication* among team members that facilitates the productivity of the teamwork. A special kind of teamwork is decision-making done at various hierarchy levels of the project, with different scopes and including diverse team members in the project. The *quality of decision-making* has a major contribution to project effectiveness and the overall success of an organization.

M.3.2.1. Effectiveness

After the measures of the contribution of the implemented method to SD performance were introduced, now the results of the evaluation are presented. First, the contributions of the methods to effectiveness are discussed.

The first measure to be evaluated was *method effectiveness* in the questionnaire. The overall value of *method effectiveness* determined by the four low-level measures (transparency, guidance, reusability, and traceability) was considered to be moderately or highly satisfying for both methods (see *Figure M.4*). The decision framework was particularly effective in providing *guidance* for the developers during decision-making. Thus, it contributed to better decisions in the pilot project. Furthermore, the *reusability* of the framework was *high,* which supported the broad application of the decision-making procedure and the results in the pilot project.

The second measure was the contribution of the decision-framework to *process effectiveness*. While the decision framework was effective in making decisions on rework and reducing technical risk through the better identification of the critical areas, it failed to reduce the number

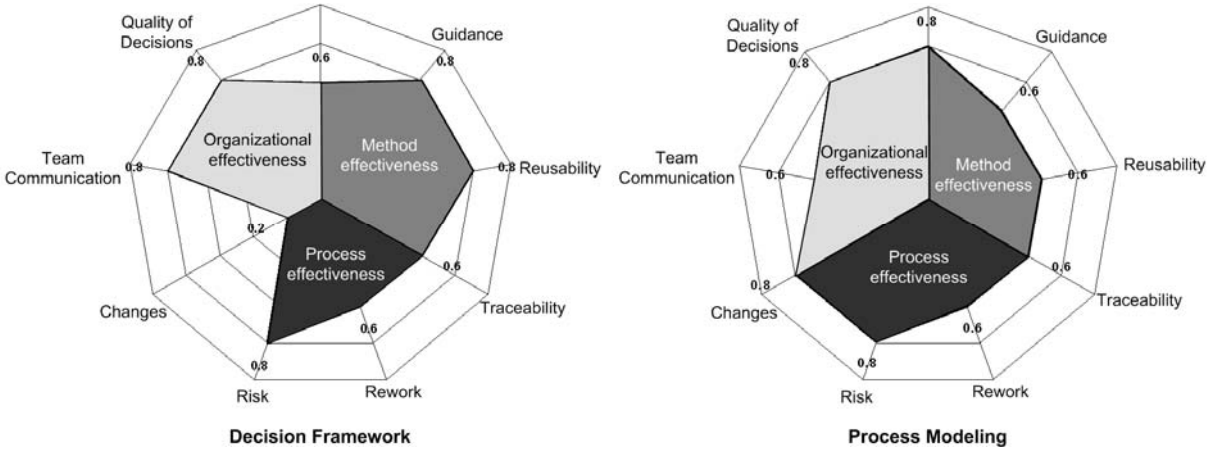


Figure M.4 Measures of the contribution of the methods to SD effectiveness

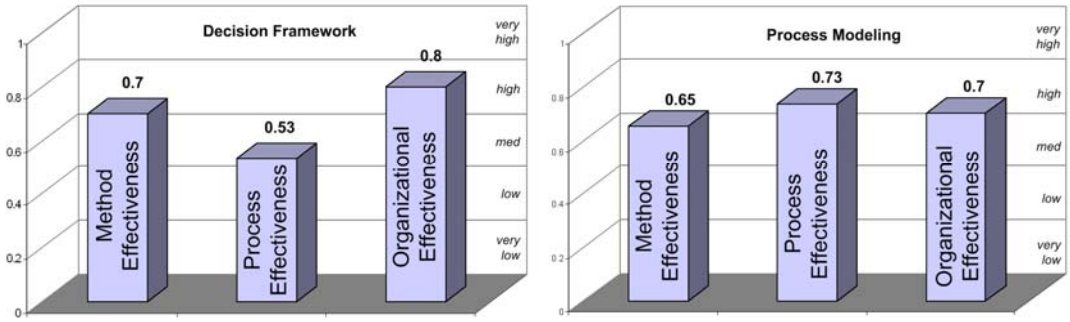


Figure M.5 Effects of the two methods on SD effectiveness

of changes in the project. This is an important lesson learned from the evaluation of the case study questionnaire, *i.e.,* effective decision-making involving the right stakeholders contributes to many small, but effective process changes. These small changes capture the failures at an early stage, and thus they do improve process effectiveness. Thus, many changes, if small and effective, are much better than few major process changes. Consequently, the decision framework did increase process effectiveness significantly, even if the number of effective change actions increased.

The third category of effectiveness was the contribution of the decision framework to *organizational effectiveness*. In this category, the decision framework was particularly successful, because it received the score *high* for improving *team communication* and *quality of decisions*. As the comments showed, the reason for this success was that the *right persons* could be invited to the meetings, who were able to make the right decisions.

The second method evaluated in the pilot projects was the VVTPM. The questions aimed to assess the effectiveness of the process modeling tool during decision-making and the selection of the right failure correction actions. A main benefit of the application of a parameter-based process modeling tool during decision-making is its contribution to the evaluation and selection of the best process option at a certain SD state. Using the VVTPM tool, it was possible to quickly define improvement options and analyze their effects on risk reduction. The other three measures (*guidance*, *reusability,* and *traceability*) received the grade *medium – 0.6* in the assessment.

The second measure of effectiveness was the effect of the VVTPM tool on *process effectiveness*. The results show that process modeling fosters decision-making by supporting the selection of the right rework actions and thus reduces undesired rework. Furthermore, change actions aiming at reducing technical risk were especially successful by identifying the right activities and stakeholders through process simulation. Finally, the VVTPM increased the effectiveness of meetings and the quality of decisions due to the same reasons.

Figure M.5 depicts the effects of the decision framework and the VVTPM tool on the three

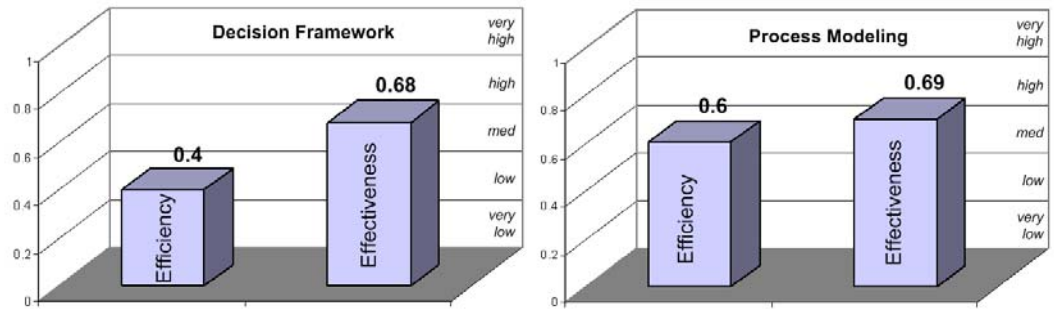


Figure M.6 Overall results for the two main measures of SD performance

main measures of SD effectiveness. Satisfaction is *high* in every aspect except for the contribution of the decision framework to process effectiveness that was lower due to the false assumption of the author about the way the number of changes affect process effectiveness.

M.3.2.2. Efficiency

Efficiency is the second measure defining the contribution of the methods to SD performance improvements. This part of the questionnaire brought the second lesson for the author. While the decision framework was quite successful at improving the effectiveness of the SD decisions (*i.e.,* the outcomes of decision-making were of higher quality), it had only *low* effects on the *efficiency* of decision-making (*Figure M.6*). Now, after the case studies, the reason became obvious: *good decisions require time.* That is, the application of the decision-making framework led to better results, because the procedure of decision-making was more thorough and thus additional effort was necessary. Hence, the application of the decision framework requires a tradeoff between effectiveness and efficiency; however, the resulting effectiveness is much higher than the required extra resources.

The VVTPM tool on the other hand is a method that increases both the effectiveness and efficiency of decision-making in the SD. The explanation for this was provided by a comment in the questionnaire, *i.e.,* the VVTPM supports preventive risk assessment during the project and thus helps allocating resources to the critical areas before a negative event happens. Thus, timely meetings can be organized and informed decisions on risk prevention can be made more efficiently than decisions on failure corrections could be made.

To summarize, both methods were highly satisfying for the improvement of SD performance. Further, the application of the adaptive decision-making framework requires more resources than conventional decision-making due to its higher information needs. On the contrary, the VVTPM was principally successful at improving both SD performance aspects.

M.3.3. Usability of the Methods

Usability is the third criterion of method feasibility in the assessment system. Usability implies *the ease with which a human user can obtain a required service from a system (or method)* [INCOSE 1998a]. The criterion *usability* is comprised of three main measures: *suitability*, *coverage,* and *information availability*.

Suitability is a measure of the degree to which a system (or method) is appropriate for its intended use with respect to non-operational factors such as man-machine interface, training, safety, documentation, producibility, testability, transportability, maintainability, manpower availability, supportability, and disposability [INCOSE 1998a]. In case of a method, suitability is the degree to which the method fulfils its purpose in the intended project environment.

Coverage means the degree to which the solution a method provides covers the problem area. The usual requirement for a new method is to have at least the same coverage as the previous one and improve its predecessor in other aspects (*e.g.,* effectiveness, efficiency, suitability, *etc.*).

Information availability describes the rate of available and required information needed to use a method effectively. If this rate is low, *i.e.,* a large amount of new information has to be collected for a new method, usability decreases due to the required extra effort during implementation.

As *Figure M.7* depicts, *usability* is a main weakness of the decision framework. The first measure *suitability* received *0.4* points, which means *low* satisfaction on the five-point scale. The reason is that the novel method attempts to standardize decision-making procedures and

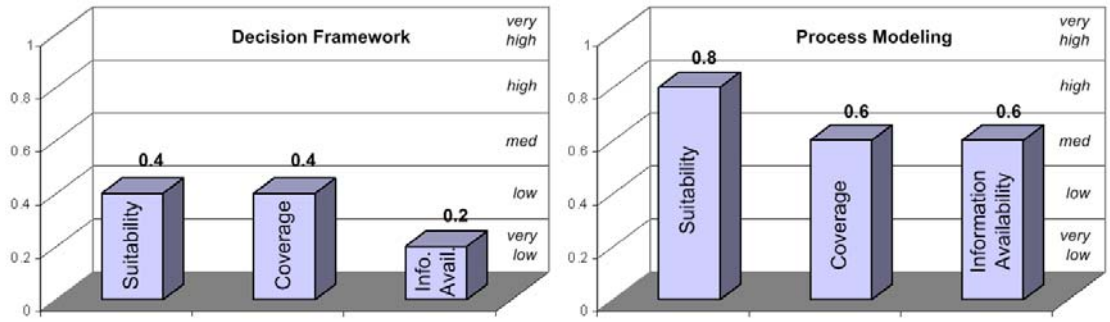


Figure M.7 Three aspects of method usability

standard procedures are often rejected by humans. This was the case in the case studies, too. That is, decision-makers preferred to apply subjective analysis to the standard procedure.

The second aspect of usability, the *coverage* of the new decision framework was measured in relation with existing methods. This is the reason for the relatively bad result (*0.4 - low*). The decision framework did not attempt to increase the coverage of decision-making methods and procedures, but the objective was to provide a systematic procedure for the decisions.

Information availability for the effective use of the method was the third usability aspect. Here, the user was asked to estimate the degree of information already available for the decision framework. Due to the novelty of the method and the parameter-based decision-making methodology, information for the risk- and opportunity-driven decisions was relatively hard to find in the pilot project. Additionally, the collected information had to be processed to acquire the necessary input data for the procedure. Thus, the *very low* satisfaction with information availability relative to methodologies currently in use at TetraPak is a natural trait of introduction processes of novel systems engineering methods. This aspect of usability will definitely improve with time.

This assumption is further underlined by the evaluation results regarding the *usability* of the VVTPM tool. Since, the process modeling tool had already been introduced a year ago, the sources of input information and the methods for processing had already been revealed by the developers. This is the reason for the *medium* satisfaction concerning the information availability for the VVTPM tool.

The *suitability* of the VVTPM tool to solve the pilot project problem was valued *high,* because the key stakeholders of the project were highly interested in experimenting with risk-based process modeling as a decision support method. Additionally, the coverage of the process modeling tool was graded *medium*, since the VVTPM is still in the introduction phase at TetraPak, and the overall goal is to cover 95% of all similar cases with the tool.

To sum up, the novelty of the decision framework contributed to low grades in all three areas of usability. However, this was expected for the introduction of new methods. The VVTPM received much better marks, but the information channels had already been discovered before; and the developers were familiar with the tool and more interested in the results.

M.3.4. User Acceptance

User acceptance is the final, fourth criterion that describes the feasibility of the implemented methods. The definition of *user acceptance* is derived from the *Technology Acceptance Model (TAM)* [Davis *et al.* 1989, Bagozzi *et al.* 1992], which is an information systems theory that models how users come to accept and use a technology (or in this cases a method). The model suggests that

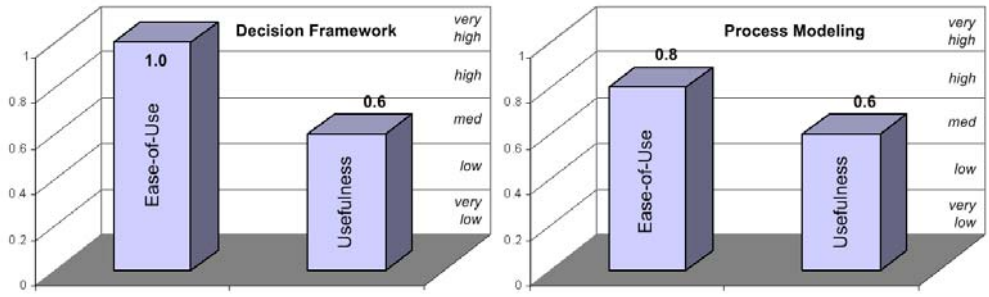


Figure M.8 Acceptance of the methods

when users are presented with a new software package (or a method, or a tool), a number of factors influence their decision about how and when they will use it, notably:

- *Usefulness,* implying the degree to which a person believes that using a particular system (or method) *would enhance his or her job performance* [Davis 1989]

- *Ease-of-use,* meaning the degree to which a person believes that using a particular system *would be free from effort* [Davis 1989].

These two aspects of user acceptance were selected as main measures in the assessment system, as well. The assessment of the decision framework concerning user acceptance provided the best results among all four aspects of feasibility (*Figure M.8*). The measure *ease-of-use* received the highest possible grade meaning *maximal user satisfaction*. The grade *very high* means in this case that the users had absolutely no problems with following the decision-making procedure, because both the sequence and meaning of the steps are clear and easy to understand. The other measure, the usefulness of the method, brought about medium satisfaction that can be explained again by the natural human aversion against standardization.

Even though the decision framework outscored the VVTPM tool in this aspect, the process modeling software received good notes, too. The application of the VVTPM tool was very easy (*high satisfaction*) in the case study, and the usefulness was at *medium* level. The reason for the average usefulness was the rather high pressure on the developers that reduced their motivation for process modeling requiring extra effort.

To summarize, the decision framework contributed to *very high* and the VVTPM to *high* user satisfaction in the case study, which is an important basis for further exploitation of the methods in the company.

M.3.5. Evaluation Summary

During the assessment process, the feasibility of two methods developed and presented in

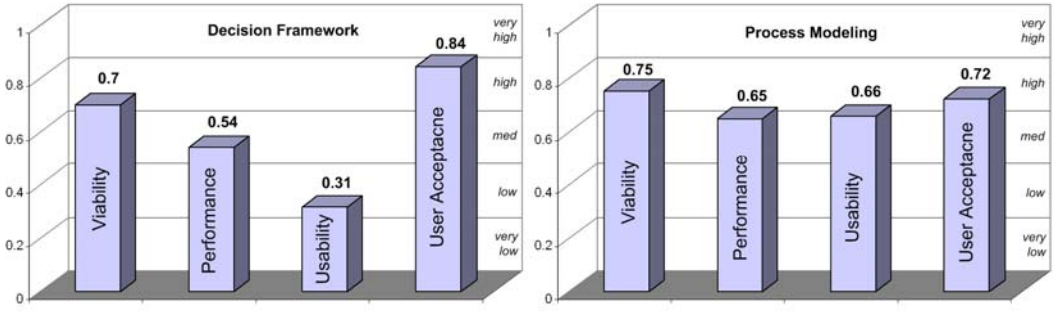


Figure M.9 Four main criteria of method feasibility

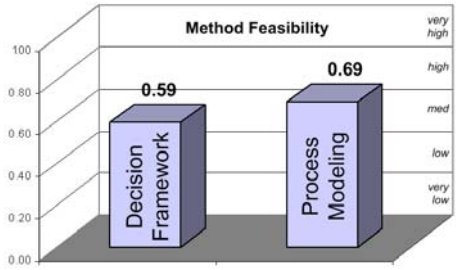


Figure M.10 Feasibility of the two methods

this thesis were assessed using a systems engineering metrics system. The users of the two methods evaluated the importance of the assessment aspects for the case study, provided quantified answers on a five-point scale to describe their satisfaction with the methods, and commented their decision.

The values of the four main criteria calculated from the answers in the assessment questionnaire are depicted in *Figure M.109*. The results in one aspect are dissatisfactory, *i.e.,* the usability of the decision framework was graded *low*. Otherwise, the two methods received medium to very high notes.

Now, the overall objective of the assessment process can be fulfilled, *i.e.,* a statement for the overall feasibility of the two methods implemented at TetraPak can be obtained. Using the criteria weightings defined by the user (viability – 0.2; performance – 0.2; usability – 0.3; acceptance – 0.3), the overall value of method feasibility could be calculated. These values are depicted in *Figure M.10*.

The decision framework received 59 out of 100 points implying a medium to high user satisfaction with the novel decision-support procedure. Even if the first difficulties of the developers with data collection and standardization of procedures are weaknesses of the method, the overall result is promising.

It was proven that the VVTPM is an excellent assistance in project planning and decision-making. The scores in all areas reached the second best level indicating high user satisfaction with the method. While the VVTPM procedure is a stochastic, parameter-based approach that supports risk-driven decision-making, it can be considered as the first, main step towards work*state*-driven project management and adaptive process modeling.

M.4. CHAPTER SUMMARY

The outstanding feedback on the VVTPM tool and the satisfaction with the adaptive, risk- and opportunity-driven decision-making framework signalize that TetraPak is on the right way towards the broad implementation of an adaptive SD philosophy in the corporate culture. Their efforts to implement systems engineering company-wide and improve the effectiveness of decision-making by translating the mere technical information into transparent measures understandable for everyone, will contribute to SD performance improvements and long-term market success.

N. THESIS SUMMARY

N.1. THESIS WRITING – AN ADAPTIVE SD PROJECT

The development process of this thesis was a wonderful journey through the worlds of Systems Engineering, Project Management, Organizational Theory, and System Design and Development. It was a *true adaptive SD project*, where the system architecture (*i.e.,* architecture of the thesis document) and the exact goals of the SD work grew together with the developer's (*i.e.,* the author's) knowledge. In the iterative learning process of identifying research problems, exploring existing solutions in relevant literature, tailoring these solutions to the thesis research context, and verifying these resulting new methods, the thesis concepts were continuously refined and evolved towards a novel systems engineering framework for the adaptive development of engineering systems.

This adaptive SD project of thesis writing involved a thorough literature review and interviews with experts from both industry and academic domains to *understand the SD problem* and *develop requirements* for the thesis work. The thesis development project also contained the *development of concepts* for both the architecture and contents of this document. These concepts were *verified* and *improved* through analyses and reviews with renowned experts from academia and industry and demonstration at relevant technical conferences (*e.g.,* INCOSE Annual Symposia, ASME-DETC Conferences, TMCE Conference, DSM Workshops, *etc.*). The concepts were then *validated* in the industry, at TetraPak Carton Ambient to prove the feasibility and applicability of the proposed theory in the European industry.

Finally, the thesis research project involved a software development task to support the validation and verification of the concepts generated in the scope of the thesis. This part of the thesis project provided the author with the particular experience of assisting the implementation of his own ideas and concepts in a real, working product. The knowledge acquired during software implementation, which resulted in the continuous correction and adjustment of the original concept, make the proposed adaptive SD framework so valuable and the results of the thesis remarkable. That is, *the methods proposed in this thesis do work in reality.*

N.2. CONTRIBUTIONS TO THE SYSTEMS ENGINEERING BODY OF KNOWLEDGE

Systems engineers of many decades grew up considering the SD project as a sequential, stage-gate process organized according to the *waterfall SD philosophy*. Besides many benefits of this kind of SD paradigm, it has main weaknesses that make the waterfall model inappropriate for the design of SD projects under highly dynamic circumstances. While there are many proposed alternative concepts to heal the illnesses of sequential SD, there is still much to contribute here for academic researchers.

One goal of this thesis was to define a *mental model*, a simple framework for SD that accounts for frequent changes in the external and thus in the internal environments of modern SD projects. This framework proposed that *controlled learning* is in the heart of *sense-and-respond SD systems*. Sensing is a key element for an adaptive system, because the fitness of system behavior is defined by the needs of its environment. Learning is also vital, since it is the iterative process of generating adequate response to the sensed needs.

The main benefit of the model-based *adaptive SD framework* for systems engineers and managers is that it supports the generation of an *emergent SD strategy* that is guided and bounded

by the growing knowledge of the developers, and the feedback of the system stakeholders on the achievements of the SD process. Thus, instead of attempting to precisely plan the project from the beginning to the end at the outset of the project, the philosophy of the adaptive SD framework is to start with a flexible project structure that is then continuously refined and detailed according to the actual project performance.

However, the definition of a new systems engineering philosophy is useless without methods that guide the implementation of it in the industry. This thesis proposed model-based methods to support the two main functions of adaptive systems engineering: (1) the planning of project control structures for system adaptation; and (2) the planning of the iterative learning processes with high capability of innovation.

The *Procedure of Adaptive Project Control* and its tailored version, the *TetraPak Decision Support Framework,* implement risk- and opportunity-driven decision-making on project adaptation in the SD project. These methods take the parameter-based measurement data from Systems Engineering Measurement and derive estimations for the possible project success (in terms of possible gains and losses) from it. This information is then applied to determine both the team structure required for effective decision-making, and to identify and select risk reduction and opportunity capturing strategies to achieve maximal system lifecycle value.

Furthermore, the dissertation introduced two parameter-based, stochastic process modeling methods to support adaptive SD project planning. The work*flow*-driven *VVT Process Modeling (VVTPM)* procedure integrates Project Management with Systems Engineering Management by facilitating the consideration of both the programmatic and technical aspects of the project in one planning tool. Hence, using the VVTPM tool, project plans can be design iteratively to maximize the effectiveness and efficiency of the SD process in a certain project environment.

The *Adaptive System Development Process (ASDP)* method is an improvement of the VVTPM to account for work*state*-driven project planning. During the discrete event simulation steps in the ASDP tool, the adaptive SD framework is simulated including the SD process of learning and innovation, and the process-state-based decisions on project adaptation. This novel process modeling technique revolutionizes project planning by supporting the selection of the best plan from a grand process space including all activity options relevant for a given project environment.

N.3. CONCLUSION

The adaptive systems engineering methodology in this thesis presents an SD model of the future and proposes methods to implement it in the present. While the adaptiveness of SD projects require a paradigm shift in engineering thinking, it will take a while until concepts like the adaptive SD framework can be directly implemented in the industry. However, as the case studies showed, there is high potential for the companies to move towards adaptiveness and increase the value of the developed products through fast response to shifts in the customer's needs. Adaptiveness is currently a "delighter requirement" and main source of competitive advantage in many industry segments; however, it will be a "must need" in the long term.

The value of this thesis is that it investigates the systems engineering aspects of adaptive SD and proposes validated methods to move towards adaptiveness. Hence, the findings of this dissertation can be considered as a small step in the long way towards truly adaptive SD enterprises.

O. APPENDIX I – CASE STUDY II DATA IN THE VVTPM TOOL

O.1. STRATEGY INPUTS

O.1.1. DSMs

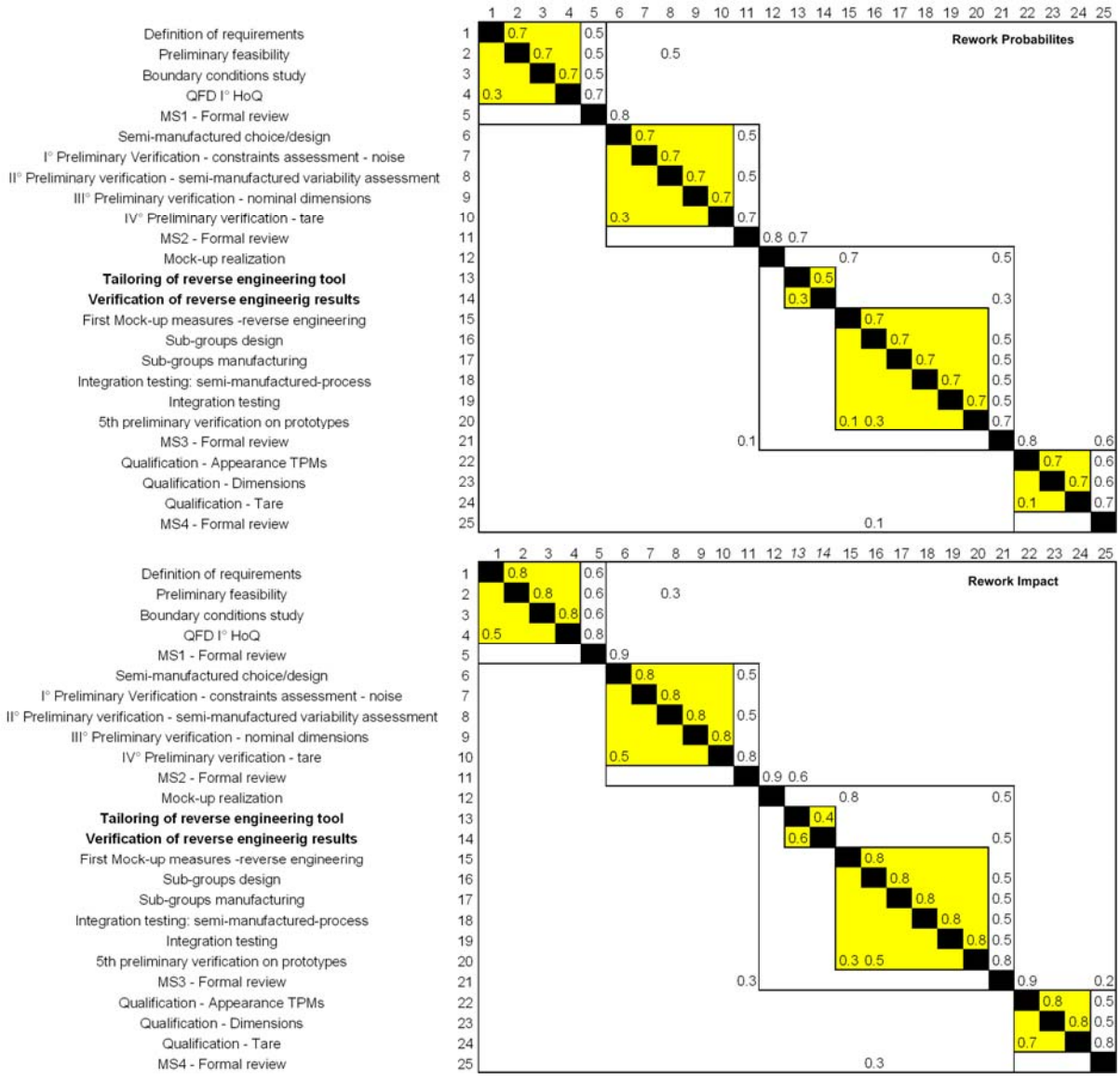


Figure O.1 Rework probabilities and impacts in Strategy 2

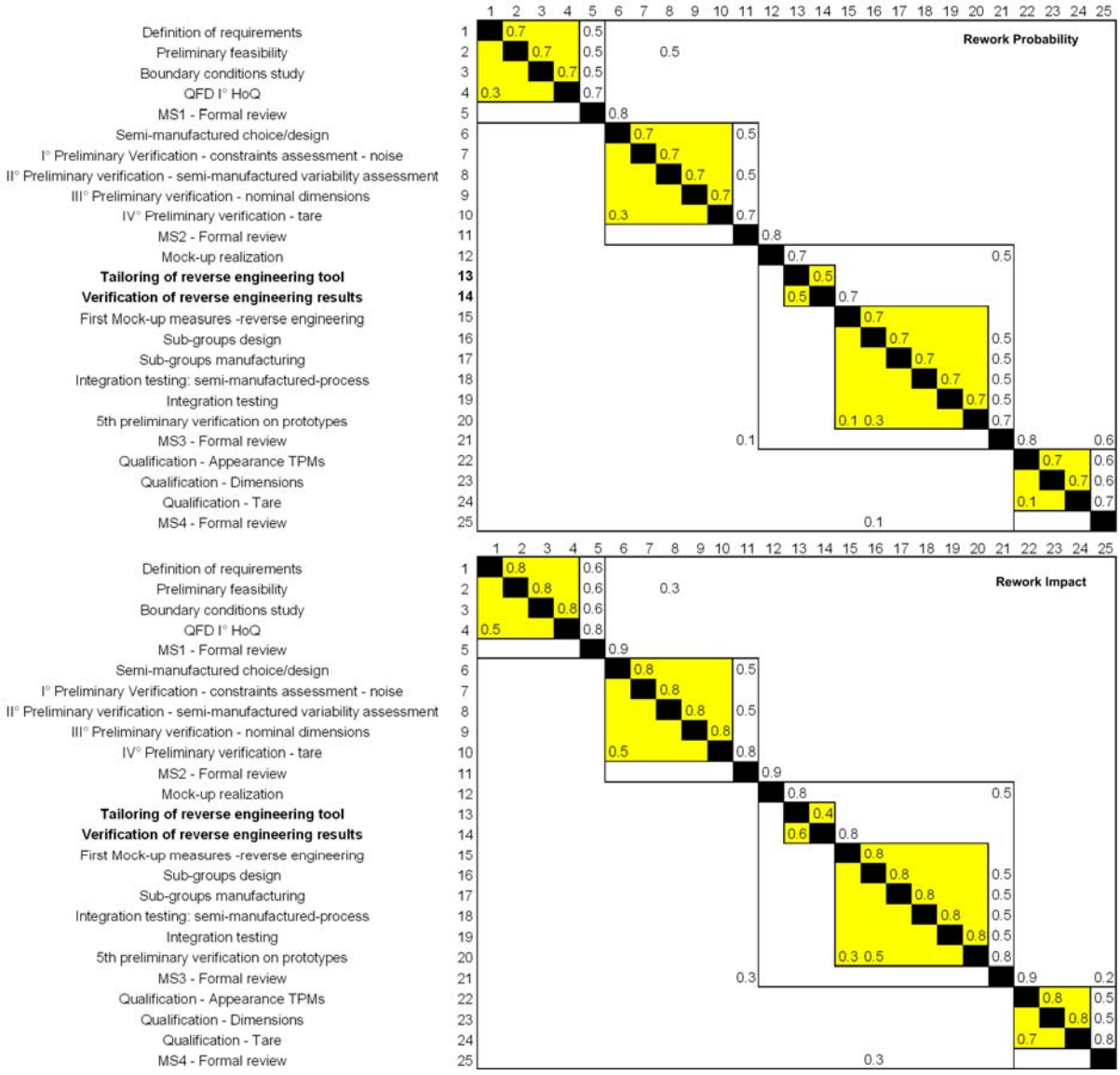


Figure O.2 Rework probabilities and impacts in Strategy 3

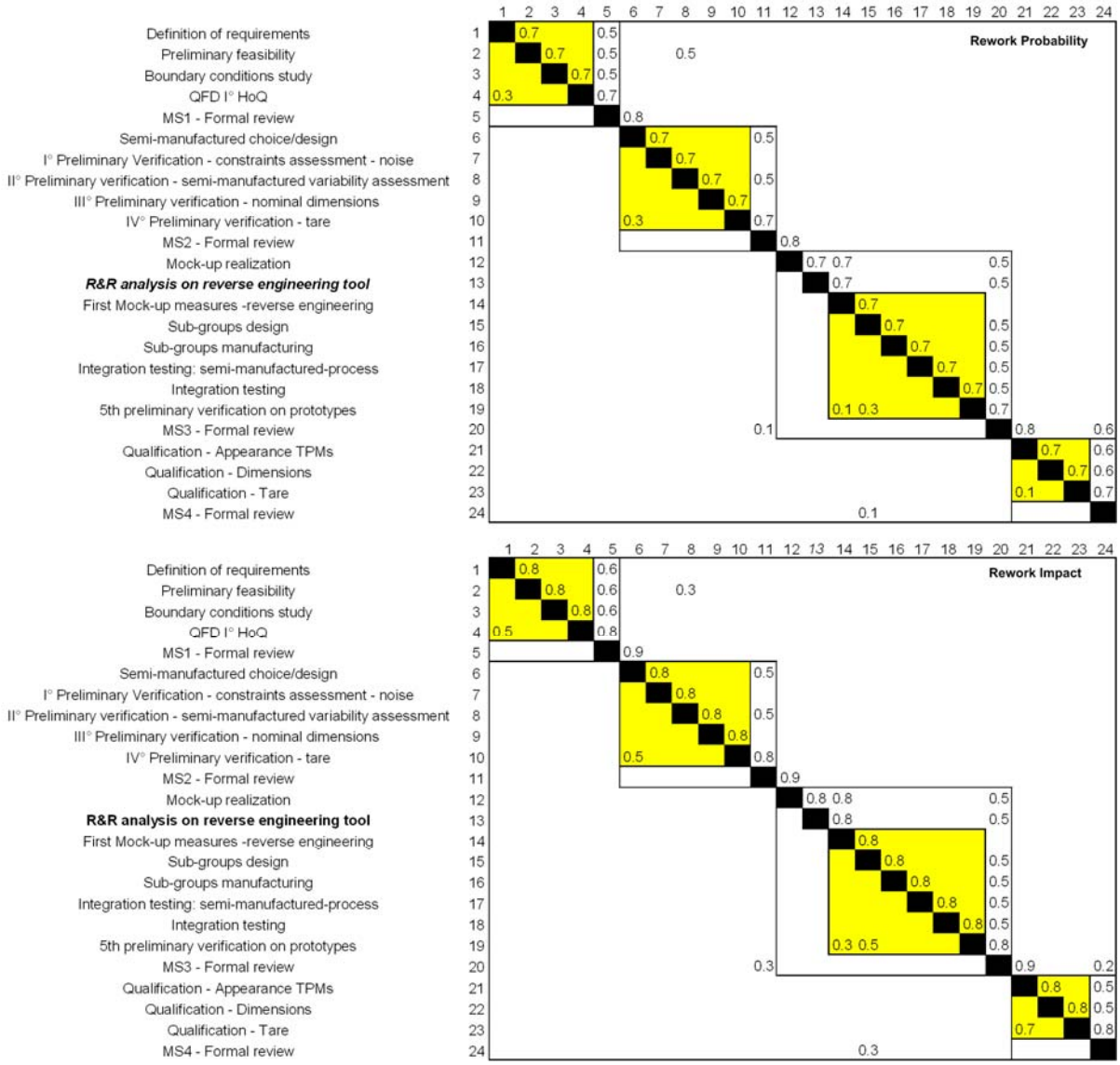


Figure O.3 Rework probabilities and impacts in Strategy 4

O.1.2. Activity Values

Activity	Performance Level	width	
		Most Likely	Uncertainty
Definition of requirements	MEDIUM	MEDIUM	VERY LOW
Preliminary feasibility	LOW	VERY LOW	LOW
Boundary conditions study	MEDIUM	LOW	VERY LOW
QFD I° HoQ	LOW	NONE	NONE
MS1 - Formal review	MEDIUM	LOW	NONE
Semi-manufactured choice/design	MEDIUM	LOW	VERY LOW
I° Preliminary Verification - constraints assessment - noise	VERY LOW	VERY LOW	NONE
II° Preliminary verification - semi-manufactured variability assessment	MEDIUM	MEDIUM	LOW
III° Preliminary verification - nominal dimensions	MEDIUM	MEDIUM	LOW
IV° Preliminary verification - tare	MEDIUM	NONE	NONE
MS2 - Formal review	MEDIUM	LOW	NONE
Mock-up realization	LOW	LOW	VERY LOW
Tailoring of reverse engineering tool	MEDIUM	MEDIUM	MEDIUM
Verification of reverse engineerig results	MEDIUM	LOW	LOW
First Mock-up measures -reverse engineering	MEDIUM	HIGH	LOW
Sub-groups design	MEDIUM	LOW	VERY LOW
Sub-groups manufacturing	LOW	LOW	VERY LOW
Integration testing: semi-manufactured-process	MEDIUM	MEDIUM	LOW
Integration testing	MEDIUM	MEDIUM	LOW
5th preliminary verification on prototypes	MEDIUM	HIGH	MEDIUM
MS3 - Formal review	MEDIUM	LOW	NONE
Qualification - Appearance TPMs	HIGH	MEDIUM	MEDIUM
Qualification - Dimensions	HIGH	HIGH	LOW
Qualification - Tare	LOW	LOW	LOW
MS4 - Formal review	HIGH	MEDIUM	LOW

Figure O.4 Activity effects on TPM package weight in Strategy 2 an Strategy 3

Activity	Performance Level	width	
		Most Likely	Uncertainty
Definition of requirements	MEDIUM	MEDIUM	VERY LOW
Preliminary feasibility	LOW	VERY LOW	LOW
Boundary conditions study	MEDIUM	LOW	VERY LOW
QFD I° HoQ	LOW	NONE	NONE
MS1 - Formal review	MEDIUM	LOW	
Semi-manufactured choice/design	MEDIUM	LOW	VERY LOW
I° Preliminary Verification - constraints assessment - noise	VERY LOW	VERY LOW	NONE
II° Preliminary verification - semi-manufactured variability assessment	MEDIUM	MEDIUM	LOW
III° Preliminary verification - nominal dimensions	MEDIUM	MEDIUM	LOW
IV° Preliminary verification - tare	MEDIUM	NONE	NONE
MS2 - Formal review	MEDIUM	LOW	
Mock-up realization	LOW	LOW	VERY LOW
R&R analysis on reverse engineering tool	MEDIUM	LOW	MEDIUM
First Mock-up measures -reverse engineering	MEDIUM	HIGH	LOW
Sub-groups design	MEDIUM	LOW	VERY LOW
Sub-groups manufacturing	LOW	LOW	VERY LOW
Integration testing: semi-manufactured-process	MEDIUM	MEDIUM	LOW
Integration testing	MEDIUM	MEDIUM	LOW
5th preliminary verification on prototypes	MEDIUM	HIGH	MEDIUM
MS3 - Formal review	MEDIUM	LOW	NONE
Qualification - Appearance TPMs	HIGH	MEDIUM	MEDIUM
Qualification - Dimensions	HIGH	HIGH	LOW
Qualification - Tare	LOW	LOW	LOW
MS4 - Formal review	HIGH	MEDIUM	LOW

Figure O.5 Activity effects on TPM package weight in Strategy 4

P. GLOSSARY OF TERMS

A	
Acceptance	see *Technology Acceptance Model*
Activity modes	Activity modes are particular "versions" of an activity with a similar purpose but different characteristics and performance levels (*e.g.*, rework mode for an activity).
Agility	Agility refers to the ability of an organization to thrive in a continuously changing, unpredictable business environment [Rigby *et al.* 2000].
Ambiguity	A form of uncertainty is unforeseen uncertainty or ambiguity referring to the unknown unknowns in the project, meaning the absence of knowledge of the planning team about what will need to be done and when. Ambiguity is more dangerous than foreseen uncertainty, because it is more difficult to plan for it. The planning team is either unaware of an event's possibility or considers it unlikely and does not bother to create contingencies (or preventive actions) for it [*e.g.*, De Meyer *et al.* 2002].
C	
Clustering	Integration analysis or clustering is a method for manipulating the structure of the system to recognize functionally related elements that are highly dependent on each other. These functionally related elements of a physical system are called clusters or chunks in literature [*e.g.*, Steward 1981a, Pimmler & Eppinger 1994, Browning 2001]. The foremost objective of clustering is to maximize interactions between elements within clusters (chunks) while minimizing interactions between clusters [Rechtin 1991, Baldwin & Clark 2000]. Thus, the main goal of clustering is to maximize hidden information in the design and thus the independence of the design modules.
Complexity	In systems engineering and systems theory, definitions classify complexity into two main categories: structural complexity concerning the complicatedness of the order of the elements in the system and behavioral complexity regarding the behavior of the system over time [Sussman 2003].
Complexity	Complexity is a feature describing the structure of the system that depends on the variety (diversity of elements) and the connectivity (diversity of relations) of the system. Furthermore, variability (changeability) has a main effect on system complexity [*e.g.*, Patzak 1982, Negele 1998, Igenbergs 2000, Wenzel 2003].
Configuration Management	Configuration Management is a discipline applying technical and administrative direction and surveillance to: (1) identify and document the functional and physical characteristics of configuration items; (2) control changes to configuration items and their related documentation; and (3) record and report change processing and implementation status. [MIL-STD-480B].
Coverage	The coverage of a method is the degree to which the solution covers the problem area.
Critical Path	In project management, a critical path is the sequence of project network terminal elements with the longest overall duration, determining the shortest time to complete the project. [Wikipedia Website].
Cybernetics	Cybernetics is a theory of the communication and control of regulatory feedback [Wiener 1948].
D	
Decision points	See *Review*
Design of Experiments (DoE)	DoE is a structured, organized method for determining the relationship between factors (Xs) affecting a process and the output of that process (Y) [I-Six Sigma Online Website].

E

Ease-of-use	Ease-of-is is the degree to which a person believes that using a particular system would be free from effort [Davis 1989].
Effectiveness	Effectiveness means the extent to which the goals of the system are attained, or the degree to which a system can be elected to achieve a set of specific mission requirements [DoD 2001b].
Efficiency	Efficiency implies the amount of output produced relative to the amount of resources (time and money) that go into the system [Wikipedia Website].
Emergence	Emergence refers to the macro-level patterns arising in systems of interacting agents. Emergent phenomena cannot be deduced from knowledge of behavior of individual parts and is not reducible to the parts alone. Emergent complexity is driven by a few simple patterns that combine to create infinite variety [Wikipedia Website].
Engineering System	Engineering systems are systems designed by humans having some purpose and are composed of interacting parts [Moses 2004].

F

Feasibility	Feasibility is the degree to which the requirements, design, or plans for a system or component can be implemented under existing constraints [INCOSE 1998a].
Flexibility	Flexibility is the ability to respond effectively to changing circumstances [Nilsson & Nordahl 1995].
Flexibility	Flexibility is the ability to change with little penalty in time, effort, cost or performance [Upton 1994].
Flexibility	Systems that have the ability to be modified to do jobs not originally included in the requirements definition are called flexible systems [McManus & Hastings 2005].
Flexibility	Flexibility can be defined as the degree of responsiveness or adaptability for any future change in a product design [Rajan *et al.* 2004].
Foreseen uncertainty	Foreseen uncertainty means that there is a variation in the possible values of the performance characteristics of the end product and the cost and duration of the SD. Hence, foreseen uncertainty is often referred to as representing the known unknowns in the SD [*e.g.*, De Meyer *et al.* 2002].
Frontloading	Frontloading is a strategy that seeks to improve development performance by shifting the identification and solving of [design] problems to earlier phases of a product development process [Thomke & Fujimoto 2000].
Functions (of a system / model element function)	Functions describe the dependencies between the inputs, properties, and outputs of system elements. The results of the functions, (*i.e.*, the element outputs) depend on how the inputs influence the properties of the element. Thus, functions describe how the element transforms inputs into outputs. Negele [1998] (see also [Patzak 1982] and [Ehrlenspiel 1995]) classifies functions as primary and secondary functions. While primary functions mainly contribute to the main purpose of the system or element, secondary functions are *e.g.*, supporting, disturbing, or irrelevant functions.

H

Hierarchy	Hierarchy is an ordered network of concepts or objects in which some are subordinate to others [INCOSE 1998a]. Hierarchy in a system relates to the vertical relationship among system elements that is the result of system decomposition.

I

Implementation benefits	Implementation benefits are a measure that expresses the achievable benefits through the introduction of a new method, tool, or methodology in a project.
Implementation effort	Implementation effort is a measure that describes the estimated amount of the required resources to implement a new method, tool, or methodology in a project.

Information availability	Information availability describes the rate of available required information needed to effectively use a method (tool or methodology). If this rate is low, *i.e.*, a large amount of information has to be collected for a new method, the usability of the new method decreases due to the required extra effort during implementation.
Inputs and outputs (I/O)	I/Os are interfaces between the elements and their environment [*e.g.*, Negele 1998]. Other system-internal or –external elements influence the system element by providing the element with the outputs they produced, *i.e.*, the results of their behavior. This way, elements of a system can exchange their products (*e.g.*, information, material, energy). The behavior and the outputs of a system as a whole depend on the interaction between system elements, *i.e.*, how certain output products of the single elements influence the output products of other elements.
Input-Process-Output (IPO) method	A generic approach for the modeling of systems and their components is the IPO approach [*e.g.*, Negele 1998] describing the main characteristics of a system element operating in an (internal or external) system environment according to the above definitions. IPO, an abbreviation of input-(process product, person or purpose)-output, is a generic, object oriented modeling technique applicable to any kinds of systems.
Interface	Interfaces are the functional and physical characteristics required to exist at a common boundary or connection between persons, or between systems, or between persons and systems [INCOSE 1998a].
Iteration	Iteration is the repetition of tasks to improve an evolving SD process [Eppinger *et al.* 1997]. Iteration means work on previous activities to improve the initial design and correct the defects found (*i.e.*, redesign) [Ford & Sterman 1998].
L	
Lifecycle (of a product, system, project)	Projects are usually divided into project phases with clear objectives and deliverables to provide this process with a logical structure, better management control, and appropriate links to the ongoing operations of the performing organization. Collectively, the project phases are known as the project life cycle [PMI 1996].
M	
Model	A model is an abstraction of the reality, a representation of a system with a purpose [*e.g.*, Igenbergs 2000].
Modeling and simulation	Modeling and simulation provide virtual duplication of products and processes, and represent those products or processes in readily available and operationally valid environments. Use of models and simulations can reduce the cost and risk of life cycle activities [DoD 2001a].
Modularity	Modularity refers to the architecture type where closely related elements of a system are grouped in clusters or chunks based on their horizontal interactions. Thus, product architectures with chunks or modules that (1) implement one or a few functional elements in their entirety and (2) comprise two or more components that strongly interact with each other within the cluster, and have few, well-defined relations to elements in other clusters are called modular architectures.
P	
Partitioning	System partitioning using a design structure matrix (DSM) helps optimize the sequence of activities in a process by manipulating the process structure. During partitioning, the goal is to get the DSM in an upper-triangular form to the extent possible, with a minimum number of sub-diagonal marks pulled as close to the diagonal as possible and grouped in blocks [Browning 2001]
Performance	Performance is defined as those operational and support characteristics of the system (or method) that allow it to effectively and efficiently perform its assigned mission over time [DoD 2001b]
Problem	Problem is the difference or gap between the actual system state and the targets or

	the system objectives describing the future system state [Haberfellner *et al.* 2002].
Process	Process is a set of activities performed to achieve a given purpose (SEI) [INCOSE 1998a].
Process	Process is a set of interrelated resources and activities which transform inputs into outputs [ISO 8402].
Process	A process is a network of customer-supplier relationships and commitments that drive activities to produce results of value [Pall 2000].
Process Capability	Process capability refers to the ability of a process to produce a defect-free product or service in a controlled manner of production or service environment. Various indicators are used, some address overall performance, some potential performance [I-Six Sigma Online Website].
Project	A project is a temporary endeavor undertaken to create a unique product or service [PMI 1996].
Project management	Project management is the application of knowledge, skills, tools, and techniques to project activities in order to meet or exceed stakeholder needs and expectations from a project [PMI 1996]
Properties (of a system / model element)	The states, effects, and behavior of the elements are described through their properties. Properties can be descriptive, quantitative, or qualitative. Based on the actual values of the properties, the characteristics of an element can be determined. Negele [1998] classifies element properties as additive – not additive; constant – variable; dependent – independent; discrete – continuous; deterministic – not deterministic.
Q	
Quality	Quality is the totality of features and characteristics of a product or service that bear on its ability to satisfy stated or implied needs [ISO 8402].
Quality	Quality means conformance to requirements derived from the customer's needs [Crosby 1979].
Quality Function Deployment (QFD)	QFD is a process for systematically translating customer requirements into appropriate technical requirements during all stages of product development from the earliest stages of product design through production [INCOSE 1998a].
R	
Real options	The elements of a system that provide "rights, not obligations" to achieve some goal or activity. Generally speaking, all elements of a system that provide flexibility can be considered as real options [De Neufville 2003a].
Relations (of a system / model element)	Relations are the interrelations or dependencies among system elements. Relations between system elements form a network of causes and effects that describe how the system operates [Sterman 2000]. Hence, the network of relations shows the underlying logic of a system, *i.e.*, the characteristics of the causal network of elements in a system. Relations can be classified by type, importance, or strength [Negele 1998]
Repeatability	Repeatability is the variation in measurements obtained when one person measures the same unit with the same measuring equipment [I-Six Sigma Online Website].
Requirement	A requirement is an essential condition that a system has to satisfy [ISO 2382-20])
Requirements engineering	Requirements engineering comprises the process of elicitation, analysis, specification, validation/verification, and management of requirements [DoD 2001a].
Review	Systems engineering or project management activity by which the technical and programmatic progress of the project is assessed relative to the project requirements and contstraints. Conducted at logical transition points in the PD effort to reduce risk /

	capture opportunitites by adapting the project to the changed characteristics. Reviews are usually conducted at every hierarchy level of the project.
Rework	Rework or refinement implies returning to previously worked activities to account for changes. It can stem from new information and/or failure to meet design objectives [Smith & Eppinger 1997b]
Risk	Risk is a measure of the potential inability to achieve overall program objectives within defined cost, schedule, and technical constraints. Risk has two components: (1) the probability or likelihood of failing to achieve a particular outcome; and (2) the consequences or impacts of failing to achieve that outcome [DoD 2002].
Risk Management	In the context of industrial systems engineering, risk management is the recognition, assessment, and control of uncertainties that may result in schedule delays, cost overruns, performance problems, adverse environmental impacts, or other undesired consequences [INCOSE 2002].
Robustness	Robustness characterizes a system's ability to be insensitive towards changing environments [Fricke & Schulz 2005]. Robust systems deliver their intended function under varying conditions without being changed [Taguchi & Clausing 1990, Taguchi 1993, Clausing 1994].
Robustness	Robustness is the ability of the system to do its basic job in unexpectedly adverse environments [McManus & Hastings 2005].
Rule of ten	The rule of ten states that costs of failure generally increase tenfold at each phase in the system lifecycle where the failure is detected [Ehrlenspiel 1995].
S	
Schedule	A schedule is the conversion of a project plan into an operating timetable [Meredith & Mantel 2003].
Sequencing	See *Partitioning*.
Six Sigma	Six sigma is a statistical method for quantifying the degree of deviation permitted by parts, products, and processes that guarantees that failure will typically occur less than three times in a million opportunities [INCOSE 1998a].
Stakeholders	Stakeholders are the persons who are somehow affected by the development, manufacturing, operation, maintenance, and disposal of the system. Stakeholders generally comprise the gamut of customer acquirers, end users, consumers, partners, supplier, unions, the corporation, the shareholders, and the society [Murman *et al.* 2002].
Statistical process control (SPC)	SPC is the application of statistical methods to identify and control the special cause of variation in a process [I-Six Sigma Online Website].
Suitability	Suitability is a measure of the degree to which a system (or method) is appropriate for its intended use with respect to non-operational factors such as man-machine interface, training, safety, documentation, producibility, testability, transportability, maintainability, manpower availability, supportability, and disposability [INCOSE 1998a].
System	A system is an interacting combination of elements to accomplish a defined objective. These include hardware, software, firmware, people, information, techniques, facilities, services, and other support elements [INCOSE 2002].
System	A system is a set of interrelated components which interact with one another in an organized fashion toward a common purpose. The components of a system may be quite diverse, consisting of persons, organizations, procedures, software, equipment, and facilities [NASA 1995].
System	A system is a composite of equipment, subsystems, skills, and techniques capable of

	performing or supporting an operational role. [MIL-STD-499A].
System	A system is any process that converts inputs to outputs [INCOSE 1998a]
System	The notion of a system may be seen as simply a more self-conscious and generic term for the dynamic interrelatedness of components [von Bertalanffy 1968]
System	The four basic characteristics that describe a system: (1) it consists of elements; (2) the elements have attributes; (3) the interaction between elements is described by relations; and (4) an element can be a system [Igenbergs 2000]
System architecture	The system architecture is the arrangement of elements and subsystems and the allocation of functions to them to meet system requirements [INCOSE 1998a].
System architecture	The architecture of a system is the arrangement of functional elements into physical chunks that become the building blocks for a product or family of products [Ulrich & Eppinger 2004].
System architecture	The architecture of a system shows how the system is built up by interacting subsystems and components whose individual functions and behaviors yield the performance of the original complex system [Yassine & Braha 2003].
System structure	If the natural order of the system elements is modified on a certain purpose, the resulting new order will not be the natural order any more. This new artificial order is then called the system structure [Schulz 2003].
System structure	The system structure shows which interactions or interdependencies exist between its subsystems without accounting for the functions or attributes that describe the individual characteristics of the subsystems.
Systems approach	The systems approach distinguishes itself from the more traditional analytic approach by emphasizing the interactions and connectedness of the different components of a system [Heylighen *et al.* 1999].
Systems engineering	Systems engineering is a process employed in the evolution of systems from the point when a need is identified through production and/or construction and ultimate deployment of that system for consumer use [Blanchard & Fabrycky 1990].
Systems engineering	Systems engineering is the treatment of engineering design as a decision-making process [Hazelrigg 1996].
Systems engineering	Systems engineering is a logical sequence of activities and decisions that transforms an operational need into a description of system performance parameters and a preferred system configuration [MIL-STD-499A].
Systems engineering	Systems engineering is an interdisciplinary approach that encompasses the entire technical effort, and evolves into and verifies an integrated and life cycle balanced set of system people, products, and process solutions that satisfy customer needs [EIA/IS-632].
Systems engineering	Systems engineering is a mechanism for proceeding from interpretation of the customer's requirements to an optimized product by steadily applying a wide–ranging attention to product requirements, extending to all details of the user's needs, producibility constraints, and life cycle aspects, essentially through an organized concurrent engineering practice. This takes place through iteration and nesting of a proper routine of analysis/synthesis, typical of a comprehensive system approach, within and across all levels of integration and all phases of a system life cycle [ECSS–E–10–01].
Systems engineering	Systems engineering is an interdisciplinary engineering management process that evolves and verifies an integrated, life-cycle balanced set of system solutions that satisfy customer needs [DoD 2001a].
Systems engineering	Systems engineering is a robust approach to the design, creation, and operation of

	systems. In simple terms, the approach consists of identification and quantification of system goals, creation of alternative system design concepts, performance of design trades, selection and implementation of the best design, verification that the design is properly built and integrated, and post-implementation assessment of how well the system meets (or met) the goals. The approach is usually applied repeatedly and recursively, with several increases in the resolution of the system baselines [NASA 1995].
Systems engineering measurement	Systems engineering measurement is the process of assigning numerical values to process, product, or project attributes according to defined criteria. This process can be based on estimation or direct measurement. Estimation results in planned or expected measures. Direct measurement results in actual measures [INCOSE 1998b]. Measurement is a control function of system engineering that collects and analyzes the actual status of the major requirements (both on the technical and managerial levels) and provides a clear picture of the risk status in the project.
System Development (SD)	SD is a process of gradually building up a body of information, until it eventually provides a complete formula for manufacturing a new product [Smith & Reinertsen 1998]
System Development (SD)	SD is a search for something unknown, and the result of SD is a description of a thing to be made, including instructions about how to make it [Baldwin & Clark 2000].
System Development (SD)	SD is the process between defining a market opportunity based on the actual and predicted customer's needs and the beginning of the production [Browning 2003].
T	
Technical Performance Measure (TPM)	A TPM may be any function, physical characteristic, design goal, or parameter of the project that has been defined by the requirements of the program. TPMs are usually the key cost drivers of the project; they reside on the critical path schedule and represent high risk to the program [DoD 2001a].
Technical reviews and audits	The systems engineer measures design progress and maturity by assessing its development at key event-driven points in the development schedule. The design is compared to pre-established exit criteria for the particular event to determine if the appropriate level of maturity has been achieved. These key events are generally known as Technical Reviews and Audits [DoD 2001a].
Technology Acceptance Model	The Technology Acceptance Model suggests that when users are presented with a new software package (or a method or a tool), a number of factors influence their decision about how and when they will use it, notably: usefulness and ease-of-use.
Testing	The purpose of testing is to verify technical performance, operational effectiveness, and suitability and provide essential information to support the decision-making [DoD 2001a].
Total Quality Management (TQM)	TQM is a management approach of an organization, participation of all its members and aiming at long term success through customer satisfaction, and benefits to all members of the organization and to society [ISO 8402].
U	
Uncertainty	Uncertainty is a condition, event, outcome, or circumstance of which the extent, value, or consequence is not predictable [INCOSE 1998a]. Uncertainty usually comes in two forms: foreseen and unforeseen uncertainty (or ambiguity).
Unforeseen uncertainty	See *Ambiguity*.
Usability	Usability is the ease with which a human user can obtain a required service from a system (or method) [INCOSE 1998a].
Usefulness	Usefulness is the degree to which a person believes that using a particular system (or method) would enhance his or her job performance [Davis 1989].
V	

Validation	The purpose of validation is to ensure that the right system is being built in the SD project, *i.e.*, writing specifications and checking performance to make sure that the system does what it is supposed to do.
Value	Value is a capability provided to a customer at the right time at an appropriate price, as defined in each case by the customer [Womack & Jones 1996].
Verification	The purpose of verification is to ensure that the system is being built right in the SD project, *i.e.*, ensuring that the system correctly implements the specifications. Verification methods comprise analysis, inspection, and demonstration and test [DoD 2001a] with the common goal to determine the quality of the system, *i.e.*, if the design complies with the defined requirements assuming that the requirements are valid.
Verification and Validation (V&V)	Verification and validation (V&V) represents the intersection of systems engineering and testing [DoD 2001a].
Viability	Viability refers to having a reasonable chance of succeeding or to being financially sustainable [Merriam-Webster Online Website].

Z

ZOPH	ZOPH is an abbreviation built up from the first letters of the German terms for the four system types in the model, (*i.e.*, Zielsystem – goal system; Objektsystem – product system; Prozesssystem – process system; and Handlungssystem – organization or agent system). Additionally, the four systems of the ZOPH model are interacting with the system environment through the system boundary [Negele 1998].
ZOPH+T	This modified ZOPH system (or ZOPH+T to distinguish from Negele's original ZOPH model) is a grand model of the SD enterprise that develops diverse products in a multi-project environment. All the different projects represented by the process system are run by the persons involved in the organization system of the enterprise and use technologies (methods, tools, equipment, etc.) included in the technology system. The SD processes deliver various products (enabling and end products of the SD) included in the product system. All four subsystems operate according to the organizational and strategic goals of the company represented by the goal system.

Q. REFERENCES

Agile Manifesto Website (Manifesto for Agile Software Development): www.agilemanifesto.org

Ahmadi, R., Wang, H., 1999, "Managing Development Risk in Product Design Processes," *Operations Research*, 47(2): 235–246.

Alexander, C., 1964, *Notes on the Synthesis of Form*, Harvard University Press, Cambridge, MA.

Allen T. J., 1997, "Organizational Structure for Product Development," MIT Sloan, Working Paper # 166-97.

Altshuller, G., 1984, *Creativity as an Exact Science*, Gordon & Breach Science Publishers, New York.

Amram M., Kulatilaka N., 1999, *Real Options: Managing Strategic Investments in an Uncertain World*, Harvard business School Press.

Andersson, J., Pohl, J., Eppinger, S.D., 1998, "A Design Process Modeling Approach Incorporating Nonlinear Elements," in Proceedings of ASME Design Engineering Technical Conferences, (DETC98–5663).

Antonsson, E.K., Otto, K.N., 1995, "Imprecision in Engineering Design," *Journal of Mechanical Design*, 117(B): 25-32.

Argyris, C., 1977, "Double-Loop Learning in Organizations," *Harvard Business Review*, 55 (Sept-Oct): 115-125.

Argyris, C., Schön, D., 1978, *Organizational Learning: A theory of action perspective*, Addison Wesley, Reading, MA.

Ashby, W.R., 1956, *Introduction to Cybernetics*, Methuen, London.

Assmann, G. 1998, *Vermeidung und Vorverlagerung von Änderungen, in Integriertes Änderungsmanagement*, (U. Lindemann and R. Reichwald, eds.), Springer-Verlag, Berlin.

Bagozzi, R.P., Davis, F.D., Warshaw, P.R., 1992, "Development and Test of a Theory of Technological Learning and Usage," *Human Relations*, 45(7), 660-686.

Balagan Website: www.balagan.org.uk

Baldwin, C.Y., Clark, K.B., 2000, *Design Rules: The Power of Modularity*, MIT Press, Cambridge vol. 1.

Beck, K., 2000, *Extreme Programming Explained: Embrace Change*, Addison Wesley.

Beedle, M., Devos, M., Sharon, Y., Schwaber, K., Sutherland, J., 2000, "SCRUM: A Pattern Language for Hyperproductive Software Development," in *Pattern Languages of Program Design 4*. (Neil Harrison, Brian Foote, and Hans Rohnert eds.), Addison-Wesley.

Belhe U., Kusiak, A., 1996, "Modeling Relationships among Design Activities," *Journal of Mechanical Design*, 118(4): 454–460.

von Bertalanffy, L., 1968, *General System Theory*, George Braziller, New York.

Black, T.A., Fine, C.F., Sachs, E.M., 1990, "A Method for Systems Design Using Precedence Relationships: An Application to Automotive Brake Systems," MIT Sloan School of Management, Cambridge, MA, 3208.

Blackburn, J.D. (ed.), 1991, *Time-based Competition: The Next Battleground in American Manufacturing*, Business One, Homewood, IL.

Blanchard, B.S., Fabrycky, W.J., 1990, *Systems Engineering and Analysis,* 2nd Ed., Prentice-Hall. Englewood Cliffs, NJ.

Boehm, B., 1981, *Software Engineering Economics*, Prentice Hall, Englewood Cliffs.

Boehm, B., 1986, "A Spiral Model of Software Development and Enhancement," *ACM SIGSOFT Software Engineering Notes*, August 1986.

Browning, T.R., 1998a, "Modeling and Analyzing Cost, Schedule, and Performance in Complex System Product Development," PhD Thesis, Massachusetts Institute of Technology, December, 1998, http://lean.mit.edu/

Browning, T.R., 1998b, "Integrative Mechanisms for Multiteam Integration: Findings from Five Case Studies," *Systems Engineering*, 1: 95–112.

Browning, T.R., 1999a, "Sources of Performance Risk in Complex System Development," in 9th Annual International Symposium of INCOSE, Brighton, U.K., 1999, pp. 711–718.

Browning, T.R., 1999b, "Sources of Schedule Risk in Complex System Development," *Systems Engineering*, 3: 129-142.

Browning, T.R., 1999c, "Designing System Development Projects for Organizational Integration," *Systems Engineering*, 2: 217–225.

Browning, T.R., 2001, "Applying the Design Structure Matrix to System Decomposition and Integration Problems: A Review and New Directions", *IEEE Transactions on Engineering Management,* (48)3: 292-306.

Browning, T.R., 2002, "Process Integration Using the Design Structure Matrix," *Systems Engineering*, 5(3): 180-193.

Browning, T.R., 2003, "On Customer Value and Improvement in Product Development Processes," *Systems Engineering*, 6(1): 49-61.

Browning, T.R., Deyst, Jr., J.J., Eppinger, S.D., Whitney, D.E., 2002 "Adding value in product development by creating information and reducing risk," *IEEE Transactions on Engineering Management,* 49(4): 443-458.

Browning, T.R., Eppinger, S.D., 2002, "Modeling impacts of process architecture on cost and schedule risk in product development," *IEEE Transactions on Engineering Management,* 49(4): 428-442.

Browning, T.R., Fricke, E., Negele, H., 2006, "Key Concepts in Modeling Product Development Processes," *Systems Engineering,* 9(2): 104-128.

Browning, T.R., Hillson, D.A., 2003, "A Quantitative Framework for Multi-Dimensional Risk and Opportunity Management," TCU M.J. Neeley School of Business Working Paper.

Browning, T.R., Honour, C.E., 2005, "Measuring the Lifecycle Value of a System," Proceedings of the 15th Annual International Symposium of INCOSE, Rochester, USA.

Browning, T.R., Ramasesh, R.V., 2005, "Modeling the Product Development Process: A Survey of the Literature," TCU M.J. Neeley School of Business Working Paper, Apr.

Carlsson, B., 1989, "Flexibility and the Theory of the Firm, "*International Journal of Industrial Organization*, 7(1989): 179-203.

Chao, L.P., Ishii, K., 2003, "Design Process Error-Proofing: Failure Modes and Effects Analysis of the Design Process," Proceedings of ASME 2003 Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Chicago, (DETC2003/DFM-48146).

Cho, S.H., 2001, "An Integrated Method for Managing Complex Engineering Projects Using the Design Structure Matrix and Advanced Simulation," SM Thesis, Dept. Mechanical Engineering, MIT, Boston.

Cho, S.H., Eppinger, S.D., 2005, "A Simulation-Based Process Model for Managing Complex Design Projects," *IEEE Transactions on Engineering Management*, 52(3): 316-328.

Cisek, R., Habicht, C., Neise, P., 2002, "Gestaltung wandlungsfähiger Produktionssysteme," *ZWF,* 97 (9): 441-445.

Clark, K.B., Fujimoto, T., 1989, "Overlapping Problem Solving in Product Development," In *Managing International Manufacturing*, K. Ferdows (Ed.), Elsevier Science, Amsterdam.

Clark, K.B., Fujimoto, T., 1991, *Product Development Performance: Strategy, Organization and Management in the World Automobile Industry*, Harvard Business School Press, Cambridge, MA.

Clark, K.B., Wheelwright, S.C., 1993, *Managing New Product and Process Development*, The Free Press, New York, NY.

Clarkson, P.J., Hamilton, J.R., 2000, "'Signposting', A Parameter-Driven Task-Based Model of the Design Process," *Research in Engineering Design*, 12: 18-38.

Clausing, D., 1994, *Total Quality Development: A Step-by-Step Guide to World-Class Concurrent Engineering*, ASME Press, New York, NY.

Cochran, W.G., Cox, G.M., 1957, *Experimental Designs*, 2nd Ed. Wiley, New York.

Cockburn, A., 2005, "Crystal (Clear)," http://alistair.cockburn.us.

Cordero, R., 1991, "Managing for Speed to Avoid Product Obsolescence: A Survey of Techniques," *Journal of Product Innovation Management*, 8: 289-294.

Crawley, E., De Weck, O., Eppinger, S.D., Magee, C., Moses, J., Seering, W., Schindall, J., Wallace, D., Whitney, D,. 2004, "The Influence of Architecture in Engineering Systems," MIT Engineering Systems Monograph.

Crosby, P.B., 1979, *Quality is Free: The Art of Making Quality Certain*, McGraw-Hill, New York.

Cusumano, M.A., 1992, "Shifting Economics: From Craft Production to Flexible Systems and Software Factories," *Research Policy*, 21(1992): 453-480.

Cusumano, M.A., Selby, R.W., 1995, *Microsoft Secrets: How the World's Most Powerful Software Company Creates Technology, Shapes Markets, and Manages People*, The Free Press, New York, NY.

Daft, R.L., Lengel, R.H., 1986, "Organizational Information Requirements, Media Richness and Structural Design," *Management Science*, 32(5): 554-570.

Daft, R.L., Macintosh, N.B., 1981, "A Tentative Explanation into the Amount and Equivocality of Information Processing in Organizational Work Units," *Administrative Science Quarterly*, 26: 207-224.

Dahan, E., Hauser, J.R., 2000, "Managing a Dispersed Product Development Process," Center for e-Business @ MIT Working Paper Nr. 103, http://ebusiness.mit.edu

Davis, F.D., 1989, "Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology," *MIS Quarterly*, 13(3), 319-340.

Davis, F.D., Bagozzi, R.P., Warshaw, P. R., 1989, "User Acceptance of Computer Technology: A Comparison af Two Theoretical Models," *Management Science*, 35: 982-1003.

Davis, S.M., 1987, *Future Perfect*, Addison-Wesley, Reading, MA.

De Groote, X., 1994, "The Flexibility of Production Processes: A General Framework," *Management Science*, 39(4): 395-409.

De Laurentis, D.A., Mavris, D.N., 2000, "Uncertainty Modeling and Management in Multidisciplinary Analysis and Synthesis," 38th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV.

De Meyer, A., Loch, C.H., Pich, M.T., 2002, "Managing Project Uncertainty: From Variation to Chaos," *Sloan Management Review*, 43(2): 60-67.

De Meyer, A., Nakane, J., Miller, J., Ferdows, K., 1989, "Flexibility: The Next Competitive Battle," *Strategic Management Journal* 10: 135-144.

Denker, S., Steward, D.V., Browning, T.R., 2001, "Planning Concurrency and Managing Iteration in Projects," *Project Management Journal*, 32(3): 31-38.

De Neufville, R., 2003a, "Real Options: Dealing With Uncertainty In Systems Planning And Design," Special Issue, *Integrated Assessment,* 4(1): 26-34.

De Neufville, R., 2003b, "Architecting/Designing Engineering Systems Using Real Options," MIT Engineering Systems Division, ESD-WP-2003-01.09-ESD Internal Symposium.

De Neufville, R., 2004, "Uncertainty Management for Engineering Systems Planning and Design," MIT Engineering Systems Monograph.

Department of Defense (DoD), 2001a, *Systems Engineering Fundamentals*, The Defense Acquisition University Press, Fort Belvoir, VA.

Department of Defense (DoD), 2001b, *Glossary - Defense Acquisition Acronyms and Terms*, Tenth Edition, Defense Acquisition University Press, Fort Belvoir, VA.

Department of Defense (DoD), 2002, *Risk Management Guide for DoD Acquisition*, The Defence Acquisition University Press, Fort Belvoir, VA.

Department of Defense (DoD) and US Army, 2000, *Practical Software and Systems Measurement – A Foundation for Objective Project Management*, version 4.0b http://www.psmsc.com

De Weck O., De Neufville, R., Chaize, M, 2004, "Staged Deployment of Communications Satellite Constellations in Low Earth Orbit," Journal of Aerospace Computing, Information, and Communication, Vol. 1, March 2004.

Dove, R., 2001, *Response Ability – The Language, Structure and Culture of the Agile Enterprise*, Wiley & Sons, New Your, NY.

ECSS–E–10–01: "Systems Engineering," European Space Agency (ESA), 19 April 1996.

EIA/IS 632: "Processes for Engineering a System," Electronics Industries Alliance (EIA), January 1999

Einhorn, H.J., Hogarth, R.M., 1986, "Decision-Making under Ambiguity," *Journal of Business*, 59(4): 225-250.

Eisenhardt, K.M., 1989, "Making Fast Strategic Decisions in High Velocity Environments," *Academy of Management Journal,* 32(3) 543–576.

Eisenhardt, K.M., Brown, S.L., 1998, "Time Pacing: Competing in Markets that Won't Stand Still", *Harvard Business Review,* 76(March-April):59-69.

Eisenhardt, K.M., Tabrizi, B.N., 1995, "Accelerating Adaptive Processes: Product innovation in the Global Computer Industry," *Administrative Science Quarterly*, 40(1995): 84-110.

Eppinger, S.D., 1991, "Model-based Approaches to Managing Concurrent Engineering," *Journal of Engineering Design,* 2(4): 283-290.

Eppinger,, S.D., Nukala, M., Whitney, D., 1997, "Generalized Models of Design Iteration Using Signal Flow Graphs," *Research in Engineering Design*, 9(2): 112–123.

Eppinger, S.D., Whitney, D.E., Smith, R.P., Gebala, D.A., 1994, "A Model Based Method for Organizing Tasks in Product Development," *Research in Engineering Design,* 6(1): 1–13.

Ehrlenspiel, K., 1995, *Intergierte Produktentwicklung – Methoden für Prozessorganization, Produktdarstellung und Konstruktion*, Carl Hanser Verlag, München.

Erixon, G., 1998, "Modular Function Deployment—A Method for Product Modularization," Doctoral Thesis, Department of Manufacturing Systems, Assembly Systems Division, The Royal Institute of Technology, Stockholm, Sweden.

Faulkner, T., 1996, "Applying Options Thinking to R&D Valuation," *Research-Technology Management*, 39(3): 50-56.

Fernandez, C., 1998, "Integration Analysis of Product Architecture to Support Effective Team Co-location," SM thesis, MIT, Cambridge, MA.

Fine, C., Whitney, D., 1999, "Is the Make-Buy Decision a Core Competence?" Moreno Muffatto and Kulwant Pawar (eds.), *Logistics in the Information Age*, Servizi Grafici Editoriali, Padova, Italy, pp. 31-63.

Finkel, S., Burazanis, M., 2004, "A Transparent Way for Process Guidance in Satellite Design," Proceedings of the 14th Annual Symposium of INCOSE, Toulouse.

Finkel, S., Wilke, M., Metzger, H., Wahnfried, M., 2002, "Design Centers – Transfering Experience from Astronautics to Aeronautics," Proceedings of the 12th Annual Symposium of INCOSE, Las Vegas.

Fishburn, P.C., 1970, *Utility Theory for Decision-making*, John Wiley & Sons, New York.

Ford, D.N., Sterman, J.D., 1998, "Dynamic Modeling of Product Development Processes," *System Dynamics Review*, 14(1): 31-68.

Ford, D.N., Sterman, J.D., 2003, "The Liar's Club: Concealing Rework in Concurrent Development," *Concurrent Engineering: Research & Applications*, 11(3): 211-219.

Forrester, J.W., 1961, *Industrial Dynamics*, MIT Press, Cambridge MA.

Fowler, T.C., 1990, *Value Analysis in Design*, Van Nostrand, New York.

Fricke, E., 1998, „Der Änderungsprozess als Grundlage einer nutzerorientierten Systementwicklung," PhD Thesis, Technische Universität München, Utz Verlag, München, Germany.

Fricke, E., Gebhard, B., Negele, H., Igenbergs, E., 2000, "Coping with Changes - Causes, Findings, and Strategies," *Systems Engineering*, 3(4): 169-179.

Fricke, E., Schulz, A.P., 2005, "Design for Changeability – Principles to Enable Changes in System throughout their Entire Lifecycle," *Systems Engineering*, 8(4): 342-359.

Garvin, D., 1988, *Managing Quality*, Free Press New York.

Gebala, D.A., Eppinger, S.D., 1991, "Methods for Analyzing Design Procedures," in Proc. ASME 3rd Int. Conf. on Design Theory and Methodology, pp. 227–233.

Gersick, C.J.G., 1988, "Marking Time: Predictable Transactions in Group Tasks," *Academy of Management Journal*, 31: 9-41.

Gersick, C.J.G., 1994, "Pacing of Strategic Change: The Case of a New Venture," *Academy of Management Journal*, 37: 9-45.

Gerwin, D., 1987, "An Agenda for Research on the Flexibility of Manufacturing Processes," *International Journal on Operations and Production Management*, 7(1): 38-40.

Ghemawat, P., 1991, *Commitment: The Dynamic of Strategy*, Macmillan, New York.

Gilb, K., 2004, *Evolutionary Project Management & Product Development (Evo),* www.gilb.com

Gladden, G. R., 1982, "Stop the Life Cycle - I Want to Get Off," *Software Engineering Notes*, 7(2): 35-39.

Grady, J.O., 1994, *System Integration*, CRC Press, Boca Raton, FL.

Ha, A.Y., Porteus, E.L., 1995, "Optimal Timing of Reviews in Concurrent Design for Manufacturability," *Management Science*, 41(9): 1431-1447.

Haberfellner, R., De Weck, O., 2005, "Agile SYSTEMS ENGINEEIRNG versus AGILE SYSTEMS engineering," Proceedings of the 15th Annual International Symposium of INCOSE, Rochester, USA.

Haberfellner, R., Nagel, P., Becker, M., Büchel, A., von Massow, H., 2002, *Systems Engineering – Methodik und Praxis*, 11th Edition, Industrielle Organisation, Zürich.

Haeckel, S.H., 1999, *Adaptive Enterprise – Creating and Leading Sense-and-Respond Organizations*, HBS Press, Boston, MA.

Hartigan, J.A., 1975, *Clustering Algorithms*, New York: John Wiley & Sons.

Hauser, J., Clausing, D., 1988, "The House of Quality," *Harvard Business Review*, 66(3): 63-73.

Hazelrigg, G.A., 1996, *Systems Engineering: An Approach to Information-Based Design*, Prentice Hall, Upper Saddle River, NJ.

Hazelrigg, G.A., 1998, "A Framework for Decision-Based Engineering Design," *Journal of Mechanical Design*, 120(4): 653-658.

Heylighen F., Bollen J, Riegler A., 1999, *The Evolution of Complexity*, Kluwer Academic, Dordrecht.

Highsmith, J.A., 2000, *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems*, New York: Dorset House.

von Hippel, E., 1990, "Task Partitioning: An Innovation Process Variable," *Research Policy*, 19: 407–418.

Holland, J., 1995, *Hidden Order: How Adaptation Builds Complexity,* Perseus, Cambridge, MA.

Holland, J., 1999, *Emergence: From Chaos to Order*, Perseus, Cambridge, MA.

Holmqvist, T.K.P., Persson M.L., 2003, "Analysis and Improvement of Product Modularization Methods: Their Ability to Deal with Complex Products," *Systems Engineering*, 6(3): 195–209.

Höltta K., De Weck O. L., 2005, "Trade-off between Modularity and Performance for Engineered Systems and Products," ICED 2005: The 15th International Conference on Engineering Design, Melbourne, Australia.

Hoppe, M., Lévárdy, V., Haskins, C., Bogomolni, I., Igenbergs, E., 2004a, "Generic Verification, Validation and Testing Methodology," Tutorial at the 14th Annual International Symposium of INCOSE, Toulouse, France, June 20-24, 2004.

Hoppe, M., Lévárdy, V., Leardi, C, Mendikoa, I., De Abajo, N., 2004b, "Application Experiences of the VVT Process Modeling Procedure at the Verification and Validation Planning," Proceedings of the 14th Annual International Symposium of INCOSE, 20-24, June, 2004, Toulouse, France.

Hoppe, M, Lévárdy, V., Vollerthun, A, Wenzel, S., 2003 "Interfacing a Verification, Validation and Testing Process Model with Product Development Methods," Proceedings of the 13th Annual International Symposium of INCOSE, Washington.

Huchzermeier, A., Loch, C.H., 2001, "Project Management under Risk: Using the Real Options Approach to Evaluate Flexibility in R&D," *Management Science*, 47(1): 85–101.

Iansiti, M., 1995a, "Shooting the Rapids: Managing Product Development in Turbulent Environments," *California Management Review*, 38(1): 37-58.

Iansiti, M., 1995b, "Technology Development and Integration: An Empirical Study of the Interaction between Applied Science and Product Development," *IEEE Transactions on Engineering Management*, 42(3): 259-269.

Iansiti, M., 1995c, "Technology Integration: Managing Technological Evolution in a Complex Environment," *Research Policy*, 24(2): 521-542.

Iansiti, M., MacCormack, 1997, "Developing Products in Internet Time," *Harvard Business Review*, 75(Sept-Oct): 108-117.

Iansiti, M. West. J., 1997, "Technology Integration: Turning Great Research into Great Products," *Harvard Business Review* , 75(May-June): 69-78.

IEEE P1220: "Standard for Application and Management of the Systems Engineering Process," of Electrical and Electronics Engineers, Inc, 26 September 1994.

Igenbergs E., 2000, *Grundlagen der Systemtechnik*, Systems Engineering lecture notes, Technische Universität München, Institute of Astronautics, München, Germany.

Imai, K., Nonaka, I., Takeuchi, H., 1985, *Managing the New Product Development Process: How the Japanese Companies Learn and Unlearn*, Clark, K.B., Hayes, R.H., Lorenz, C. (eds.), *The Uneasy Alliance*. Harvard Business School Press, Boston, MA.

International Council on Systems Engineering (INCOSE) Website: www.incose.org

International Council on Systems Engineering (INCOSE), 1995, *Metrics Guidebook for Integrated Systems and Product Development*, www.incose.org

International Council on Systems Engineering (INCOSE), 1998a, *INCOSE SE Terms Glossary*, Version 0, October 1998, www.incose.org

International Council on Systems Engineering (INCOSE), 1998b, *Systems Engineering Measurement Primer*, version 1.0, March 1998, www.incose.org

International Council on Systems Engineering (INCOSE), 2002, *Systems Engineering Guidebook, A How to Guide for all Engineers*, Version 2, www.incose.org

ISO 2382-20: Information technology – Vocabulary – Part 20: System development, International Organization for Standardisation (ISO), 1990

ISO 8402: Quality management and quality assurance –Vocabulary, International Organization for Standardisation (ISO), 1994

I-Six Sigma Online Website, www.isixsigma.com

Jones, R.A., Ostroy, J.M., 1984, "Flexibility and Uncertainty," *Review of Economic Studies*, 13-32.

Kahmeyer, M., Warnecke, H.J., Sheider, W.D., 1994, "Fractal Product Design: Design for Assembly and Disassembly in Fractal Factory," DFMA Conf, 1994, pp. 1–9.

Kehat, E., Shacham, M., 1973, "Chemical Process Simulation Programs-2: Partitioning and Tearing of System Flowsheets," *Process Technology International*, 18(3): 115-118.

Kleim, R., Ludin, I., 1997, *Reducing Project Risk*, Brookfield, Gower, Vermont.

Kreichgauer, O., 1995, *Quantitatives dynamisches Modell zur Simulation von Systembelastungen in Luftverkehrsabläufen*, PhD Thesis, Technische Univrsität München

Krishnan, V., Eppinger, S.D., Whitney, D.E., 1995, "Accelerating Product Development by the Exchange of Preliminary Product Design Information," *Journal of Mechanical Design*, 117(December 1995): 491-498.

Krishnan, V., Eppinger, S.D., Whitney, D.E., 1997, "A Model-Based Framework to Overlap Product Development Activities," *Management Science*, 43(4):437-451.

Kusiak, A., 1999, *Engineering Design: Products, Processes, and Systems*, Academic Press, San Diego, CA.

Kusiak, A., Huang, C., 1996, "Development of modular products," *IEEE Trans Components, Packaging Manuf Technol* PartA 19(4): 523–538.

Kusiak, A., Wang, J., 1993, "Efficient Organizing of Design Activities," *International Journal of Production Research*, 31: 753–769.

Kusiak, A., Wang, J., 1995, "Dependency Analysis in Constraint Negotiation," *IEEE Transactions on Systems, Man, and Cybernetics*, 25(9): 1301-1313.

Lévárdy, V., Browning, T., 2005, "Adaptive System Development Process – Designing a Test Process that Adapts to the State of a Project," Proceedings of the 15th Annual International Symposium of INCOSE, Rochester, USA.

Lévárdy, V., Browning, T., Hoppe, M., 2004a, "Adaptive System Development Process - An Integrated Modeling Approach for Test and Design Activities in the Product Development Process," Proceedings of the ASME 2004 Design Engineering Technical Conferences and Computers and Information in Engineering Conference (DETC2004/57390), Salt Lake City.

Lévárdy, V., Hoppe, M., 2004, "Verification, Validation and Testing Strategy Planning Supported by a Process Model," Proceedings of the Tools and Methods of Competitive Engineering (TMCE) 2004 Conference, April 12-16, 2004, Lausanne, Switzerland.

Lévárdy, V. Hoppe, M., Honour, E., 2004b, "Verification, Validation & Testing Strategy and Planning Procedure," Proceedings of the 14th Annual International Symposium of the INCOSE, Toulouse, France.

Lévárdy V., Hoppe, M, Wenzel S., Vollerthun, A., 2003, "Process Modeling Procedure for Verification, Validation and Testing Planning," Proceedings of ASME 2003 Design Engineering Technical Conferences and Computers and Information in Engineering Conference (DETC2003/DTM-48682) Chicago.

Loch, C.H., Terwiesch, C., 1998, "Communication and Uncertainty in Concurrent Engineering," *Management Science*, 44(8): 1032–1048.

Loch, C.H., Terwiesch, C., Thomke, S, 2001, "Parallel and Sequential Testing of Design Alternatives," *Management Science,* 45(5): 663-678.

Lowe, P., 1995, *The Management of Technology – Perception and Opportunities*, Chapman & Hall.

MacCormack, A., Verganti, R., 2003, "Managing the Sources of Uncertainty: Matching Process and Context in Software Development" *Journal of Product Innovation Management*, 2003; 20: 217-232.

MacCormack, A., Verganti, R., Iansiti, M., 2001, "Developing Products on Internet Time: The Anatomy of Flexible Development Process," *Management Science,* 47(1): 133-150.

Mansfield, E., Raporport, J., Schnee, J., Wagner, S., Hamburger, M., 1972, *Research and Innovation in the Modern Corporation,* Macmillan, London.

McCord K.R., Eppinger, S.D., 1993, "Managing the Integration Problem in Concurrent Engineering," MIT Sloan School of Management, Cambridge, MA, Working Paper No. 3594.

McCracken, D.D., Jackson, M.A., 1982, "Life-Cycle Concept Considered Harmful," *ACM Software Engineering Notes*, Apr. 1982: 29-32.

McCulley, C., Bloebaum, C.L., 1996, "A Genetic Tool for Optimal Design Sequencing in Complex Engineering Systems," *Structural Optimization*, 12: 186–201.

McKee, D., 1992, "An Organization Learning Approach to Product Innovation," *Product Innovation Management*, 9: 232-245.

McManus, H., Hastings, D., 2005, "A Framework for Understanding Uncertainty and its Mitigation and Exploitation in Complex Systems," Proceedings of the 15th Annual International Symposium of INCOSE, Rochester, USA.

Meier, C., 2005, "Optimization of Activity-Based Design Structure Matrices Using Genetic Algorithms," Diploma Thesis, Technische Universität München.

Meier, C., Hoppe, H., Lévárdy, V., 2004, "Process Modeling Tool for Verification & Validation Strategy Planning," 2nd Cambridge Design Structure Matrix (DSM) Workshop, 12th – 14th September 2004, University of Cambridge, UK.

Meredith, J.R., Mantel, S.J., 2003, *Project Management: A Managerial Approach*, 5th Edition, Wiley, New York.

Meredith, S., Francis, D., 2000, "Journey towards Agility: The Agile Wheel Explored," *The TQM Magazine*, 12(2):137-143.

Merriam-Webster Online Website: www.m-w.com

Metz, P., 1996, "Integrating Technology Planning with Business Planning," *IEEE Engineering Management Review*, 24(4): 118-120.

Meyer, M.M., Utterback, J.R., 1995, "Product Development Cycle Time and Commercial Success," *IEEE Transactions on Engineering Management*, 42(4): 297-304.

MIL-STD-480B: "Configuration Control - Engineering Changes, Deviations and Waivers," Department of Defense (DoD), 15 July 1988.

MIL-STD-499A: "Engineering Management," Department of Defense (DoD), 1 May 1974.

Moore, G., 1999, *Crossing the Chasm: Marketing and Selling High-Tech Products to Mainstream Customers*, Harper Business, New York.

Moses, J., 2004, "Foundational Issues in Engineering Systems: A Framing Paper," Engineering Systems Monograph, MIT, Cambridge MA: 16.

Murman, E. *et al.*, 2002, *Lean Enterprise Value*, Hampshire, Palgrave, UK.

National Aeronautics and Space Administration (NASA), 1995, *NASA Systems Engineering Handbook*, SP-6105, ldcm.nasa.gov/library/Systems_Engineering_Handbook.pdf

Negele, H., 1998, *Systemtechnische Methodik zur ganzheitlichen Modellierung am Beispiel der integrierten Produktentwicklung*, PhD Thesis, Technische Universität München, Utz Verlag, München, Germany.

von Neumann, J., Morgenstern, O., 1944, *Theory of Games and Economic Behaviour*, Princeton Univ. Press, Princeton NJ., USA.

Neumann, K., 1990, *Stochastic Project Networks: Temporal Analysis, Scheduling and Cost Minimization*, Springer-Verlag, Berlin, Germany.

Nilsson, C.H., Nordahl, H., 1995, "Making manufacturing flexibility operational – part 1: a framework," *Integrated Manufacturing Systems*, 6(1): 5-11.

Nishinaga, N., Ogawa, Y. *et al.*, 2003, "SOFTSAT: Reconfigurable Communication Satellite System," 21st International Communication Satellite Systems Conference and Exhibit, Yokohama, Japan, AIAA 2003-2420.

O'Donovan, B., Eckert, C., Clarkson, J., 2004, "Simulating Design Processes to Assist Design Process Planning," Proceedings of ASME 2004 Design Engineering Technical Conferences and Computers and Information in Engineering Conference (DETC2004-57612) Salt Lake City.

Otto, K., Wood, C., 2001, *Product Design: Techniques in Reverse Engineering and New Product Development*, Prentice Hall, Upper Saddle River, NJ.

Pahl, G., Beitz, W., 1999, *Engineering Design: A Systematic Approach*, Springer-Verlag, New York.

Pall, G.A., 2000, *The Process Centered Enterprise – The Power of Commitments*, CRC Press LCC, Boca Raton, FL.

Pande, P.S., Neuman, R.P., Cavanagh, R.R., 2000, *The Six Sigma Way*, McGraw-Hill.

Paquin, J.P., Couillard, J., Ferrand, D.J., 2000, "Assessing and Controlling the Quality of a Project End Product: The Earned Quality Method," *IEEE Transactions on Engineering Management,* 47(1): 88-97.

Patzak, G., 1982, *Systemtechnik – Planung komplexer, innovativer Systeme*, Springer Verlag, Berlin, Germany.

Pich, M.T., Loch, C.H., De Meyer, A., 2002, "Uncertainty, Ambiguity, and Complexity in Project Management," *Management Science*, 48(8): 1008–1023.

Pimmler, T.U., Eppinger, S.D., 1994, "Integration Analysis of Product Decompositions," in Proc. ASME 6th Int. Conf. on Design Theory and Methodology, Minneapolis, MN.

Pisano, N.D. CDR, 1995, "Technical Performance Measurement, Earned Value, and Risk Management: An Integrated Diagnostic Tool for Program Management," http://www.acq.osd.mil/pm/tpm/nickpaso.htm

Pritsker, A.A.B., Sigal, C.E., 1983, *Management Decision-making: A Network Simulation Approach,* Englewood Cliffs, NJ: Prentice-Hall.

Project Management Institute (PMI), 1996, *A guide to the project management body of knowledge*, www.pmi.org

Rajan, P., Van Wie, M., Wood, K., Otto, K., Campbell, M., 2004, "Empirical Study on Product Flexibility," Proceedings of ASME 2004 Design Engineering Technical Conferences and Computers and Information in Engineering Conference, (DETC2004/DTM-57389).

Rask, I., Sunnersjö, S., 1998, "Design Structure Matrices for the Planning of Rule-Based Engineering Systems," in Proc. Eur. Conf. on Integration in Manufacturing, Göteborg, Sweden.

Rechtin, E., 1991, *Systems Architecting: Creating* & *Building Complex Systems*, Englewood Cliffs, NJ: Prentice-Hall.

Rechtin, E., Maier, M., 1997, *The Art of Systems Architecting*, Boca Raton, CRC Press.

Riehle, D., 2005, "A Comparison of the Value Systems of Adaptive Software Development and Extreme Programming: How Methodologies May Learn from Each Other," SKYVA International, www.skyva.com

Rigby, C., Day, M., Forrester, P., Burnett, J., 2000, "Agile Supply: Rethinking, Systems Thinking, Systems Practice," *International Journal of Agile Management Systems*, 2(3): 178-186.

Roemer, T.A., Ahmadi, R., 2004, "Concurrent Crashing and Overlapping in Product Development," *Operations Research*, 52(4): 606-622.

Roemer, T.A., Ahmadi, R, Wang, R., 2000, "Time-Cost Tradeoffs in Overlapped Product Development," *Operations Research*, 48(6): 858–865.

Rogers, J.L., 1989, "A Knowledge-Based Tool for Multilevel Decomposition of a Complex Design Problem," NASA, Hampton, VA, TP-2903.

Rogers, J.L., 1996, "Integrating a Genetic Algorithm into a Knowledge-Based System for Ordering Complex Design Processes," NASA, Hampton, VA, TM-110 247.

Rosenau, M.D., 1988, "Speeding your Product to Market," *Journal of Consumer Marketing,* 5: 23-40.

Rosenau, M.D., 1990, *Faster New Product Development*, AMACOM, New York.

Royce, W.W., 1970, "Managing the Development of Large-Scale Software: Concepts and Techniques," Proceedings, Wescon, August 1970.

Rushton, G. J. Zakarian, A., 2000, "Modular Vehicle Architectures: A Systems Approach," in 10th Annual International Symposium of INCOSE, Minneapolis, MN, pp. 29–35.

Sabbagh, K., 1996, *Twenty-First Century Jet*, Scribner, New York.

Sanches, R., Mahoney, J.T., 1996, "Modularity, Flexibility and Knowledge Management in Product and Organization Design," *Strategic Management Journal*, 17(winter special issue): 63-76.

Sarbacker, S.D., Ishii, K., 1997, "A Framework for Evaluating Risk in Innovative Product Development," Proceedings of the ASME 1997 Design Engineering Technical Conference (DFM-97-238), Sacramento, CA.

Schilling, M.A., 2000, "Toward a General Modular Systems Theory and its Application to Interfirm Product Modularity," *Academy of Management Review*, 25(2): 312-334.

Schulz, A.P., 2003, "Systemtechnische Gestaltung der Informationsarchitektur im Entwicklungsprozess," PhD Thesis, Technische Universität München, Utz Verlag, München, Germany.

Schulz, A.P., Clausing, D.P., Fricke, E., Negele, H., 2000, "Development and Integration of Winning Technologies as Key to Competitive Advantage," *Systems Engineering*, 3(4): 180-211.

Schulz, A.P., Fricke, E., 1999, "Incorporating Flexibility, Agility, Robustness, and Adaptability within the Design of Integrated Systems - Key to Success?" Proceedings of 18th Digital Avionics Systems Conference (DASC), St. Louis, USA.

Scott, J.A., 1998, "A Strategy for Modelling the Design-Development Phase of a Product," Ph.D. Thesis, Department of Marine Technology, University of Newcastle upon Tyne, UK.

Senge, P.M., 1990, *The Fifth Discipline – The Art and Practice of the Learning Organization*, Random House UK.

Sharifi, H., Zhang, Z., 1999, "A Methodology for Achieving Agility in Manufacturing Organizations: An Introduction," *International Journal of Production Economics*, 62(1999): 7-22.

Sharman, D.M., 2002*, Valuing architecture for strategic purposes*, SM Thesis (SDM), MIT Cambridge, MA.

Sharman, D.M., Yassine, A.A., 2003, "Characterizing Complex Product Architectures," *Systems Engineering*, (7)1: 35-60.

Shiba, S., Graham A., Walden, D., 1993, *A New American TQM: Four Practical Revolutions in Management*, Productivity Press, Cambridge.

Shishko, R., 2000, "The Proliferation of PDC-Type Environments in Industry and Universities", Proceedings of the 2nd EUSEC, Munich.

Shishko, R., Ebbeler, D.H., Fox, G., 2004, "NASA Technology Assessment Using Real Options Valuation," *Systems Engineering*, 7(1): 1-12.

Siddiqi, A, De Weck, O., Hoffman, J., 2005, "Sustainability in System Architectures through Reconfigurability: A Case Study of Planetary Surface Vehicles," IAF 2005, Fukuoka, Japan.

Siddique, Z., Rosen, D., 1999, "Product Platform Design: A Graph Grammar Approach," Proc ASME Des Eng Tech Conf, Las Vegas, NV, (DETC99/DTM-8762).

Smith, P.G., Reinertsen, D.G., 1998, *Developing Products in Half the Time: New Rules, New Tools*, John Wiley & Sons

Smith, R.P., Eppinger S.D., 1997a, "Identifying Controlling Features of Engineering Design Iteration," *Management Science* 43(3): 276-293.

Smith, R.P., Eppinger S.D., 1997b, "A Predictive Model of Sequential Iteration in Engineering Design," *Management Science* 43(8): 1104-1120.

Smith, R.P., Eppinger S.D., 1998, "Deciding Between Sequential and Parallel Tasks in Engineering Design," *Concurrent Engineering: Research and Applications* 6(1): 15-25.

Son, S.Y, Olsen, T.L, Yip-Hoi, D., 2000, "Economic Benefits of Reconfigurable Manufacturing Systems," 2000JUSFA-13193, Proceedings of the 2000 Japan-USA Flexible Automation Conference, July 23-26, 2000, Ann Arbor, Michigan.

Stalk, G., Hout, T.M., 1990, *Competing Against Time: How Time-Based Competition is Reshaping Global Markets*, Free Press, New York, NY.

Stanke A, Ulbricht B., 1997, *Modelle und Methoden der Neuproduktplanung. Forschungs- und Entwicklungsmanagement: Simultaneous Engineering, Projektmanagement, Produktplanung, Rapid Product Development*, (eds. Bullinger HJ, Warschat J), B.G. Teubner, Stuttgart.

Sterman, J.D., 2000, *Business Dynamics – Systems Thinking and Modeling for a Complex World*, McGraw-Hill Higher Education.

Steward, D.V., 1981a, *Systems Analysis and Management: Structure, Strategy, and Design*, New York: PBI.

Steward, D.V., 1981b, "The Design structure system: A method for managing the design of complex systems," *IEEE Transactions on. Engineering Management*, 28: 71–74.

Suarez, F.F., Cusumano, M.A., Fine, C.H., 1991, "Flexibility and Performance: A Literature Critique and Strategic Framework," Working Paper 50-91, International Center for Research in the Management of Technology, MIT, Boston.

Suh, N., 1990, *The Principles of Design*, Oxford University Press, New York.

Sussman, J.M., 2003, "Collected Views on Complexity in Systems" MIT Working Paper Series, ESD-WP-2003-01.06-ESD Internal Symposium.

Swift, J.A., Ross, R.E., Omachonu, V.K., 1998, *Principles of Total Quality*, St. Lucie Press, Boca Raton, FL.

Taguchi, G., 1993, *Taguchi on Robust Technology Development: Bringing Quality Engineering Upstream*, ASME Press, New York.

Taguchi, G., Clausing, D., 1990, "Robust Quality," *Harvard Business Review*, 1990 (Jan-Feb): 65-75.

Taguchi, G, Wu, J., 1980, *Introduction to Off-Line Quality Control*, Central Japan Quality Association, Nagoya, Japan.

Takeuchi, H., Nonaka, I., 1986, "The New New Product Development Game," *Harvard Business Review*, 1986(January-February): 137-146.

Tang, D., Zheng, L., Li, Z., Li, D., Zhang, S., 2000, "Re-engineering of the Design Process for Concurrent Engineering," *Computers & Industrial Engineering*, 38: 479–491.

Terwiesch, C., De Meyer, A., Loch, C.H., 2002, "Exchanging Preliminary Information in Concurrent Engineering: Alternative Coordination Strategies", *Organization Science*, 13(4): 402-419.

Terwiesch, C., Loch, C.H., 1999, "Measuring the Effectiveness of Overlapping Development Activities", *Management Science*, 45(4): 455-465.

Thebeau, R.E., 2001, Knowledge *Management of System Interfaces and Interactions for Product Development Processes*, Master's Thesis (Eng. & Mgmt.), MIT, Cambridge, MA.

Thomke, S., 1997, "The Role of Flexibility in the Development of New Products: An Empirical Study," *Research Policy*, 26(1997): 105-119.

Thomke, S., 1998, "Managing Experimentations in the Design of New Products," *Management Science*, 44(6): 743-762.

Thomke, S., 2001, "Enlightened Experimentation: The New Imperative for Innovation," *Harvard Business Review*, 2001(February): 66-75.

Thomke, S., 2003, *Experimentation Matters – Unlocking the potential of new technologies for innovation*, Harvard Business School Press, Boston.

Thomke, S., Bell, D.E., 2001, "Sequential Testing in Product Development," *Management Science*, 47(2): 308-323.

Thomke, S., Fujimoto, T., 2000, "The Effect of "Front-Loading" Problem-Solving on Product Development Performance," *Journal of Product Innovation Management,* 17(2000): 128–142.

Thomke, S., Reinertsen, D., 1998, "Agile Product Development: Managing Development Flexibility in Uncertain Environments," *California Management Review*, 41(1): 8-30.

Thornton, A.C., 2001, "Optimism vs. Pessimism: Design Decisions in the Face of Process Capability Uncertainty," *Journal of Mechanical Design*, 123: 313-321.

Thurston, D.L., Locascio, A., 1994, "Decision theory for design economics," *The Engineering Economist*, 40(1): 41-72.

Toffler, A. 1971, *Future Shock*, Bantam Books, New York.

Triantis A., Hodder J., 1990, "Valuing Flexibility as a Complex Option." *The Journal of Finance,* 45(2): 549-565.

Trigeorgis L., Mason J., 1987, "Valuing Managerial Flexibility." *Midland Corporate Finance Journal*, 5: 14-21.

Tseng, M.M., Jiao, J., 1996, "Design for Mass Customization," *Annals of the CIRP,* 45(1): 153-156.

Ulrich, K.T., 1995, "The Role of Product Architecture in the Manufacturing Firm," *Research Policy*, 24(1995): 419-440.

Ulrich, K.T., Eppinger, S.D., 2004, *Product Design and Development*, 3rd ed. New York: McGraw-Hill.

Upton, D.M., 1994, "The Management of Manufacturing Flexibility," *California Management Review*, 36(2): 72-89.

Urban, G., Hauser, J., 1980, *Design and Marketing of New Products*, Prentice Hall, Englewood Cliffs NJ.

Vesey, J.T., 1991, "The New Competitors: They Think in Terms of Speed to Market," *Academy of Management Executive*, 5(2): 23-33.

V-Model XT Website – Release 1.2, (Bundesministerium des Innern): www.v-modell-xt.de

Vollerthun, A., 2001, *Integration von Konzeptentwurf und Marketing*, PhD Thesis, Technische Universität München, Utz Verlag, München, Germany.

Vollerthun, A., 2002, "Design-to-Market: Integrating Conceptual Design and Marketing," *Systems Engineering*, 5(4): 315–326.

Wadhwa, S., Rao, K.S., 2003, "Flexibility and Agility for Enterprise Synchronization: Knowledge and Innovation Management Towards Flexagility," *Studies in Informatics and Control*, 12(2): 111-128.

Walther, C., 1994, *Systemtechnische Zusammenhänge zwischen Eigenschaften und Funktionen großer Systeme – Methoden zur Darstellung von Änderungsauswirkungen*, PhD Thesis, Technische Universität München, Utz Verlag, München, Germany.

Warfield, J., 1973, "Binary Matrices in System Modeling," *IEEE Transactions on Systems, Man, and Cybernetics*, 3: 441-449.

Weick, K.E., 1993, "The Collapse of Sensemaking in Organizations: The Mann Gulch Disaster," *Academic Science Quarterly*, 38: 628-652.

Weil, R.L., Kettler, P.C., 1971, "Rearranging Matrices to Block-Angular Form for Decomposition (and Other) Algorithms," *Management Science,* 18: 98–108.

Wenzel, S., 2003, *Organisation und Methodenauswahl in der Produktentwicklung*, PhD Thesis, Technische Universität München, Utz Verlag, München, Germany.

Wheelwright, S.C., Clark, K.B., 1992, *Revolutionizing Product Development*, The Free Press, New York.

Whitfield, R.I., Duffy, A.H.B., Coates, G., Hills, W., 2003, "Efficient Process Optimization," *Concurrent Engineering: Research and Applications*, 11(2): 83-92.

Whitfield, R.I., Smith, J., Duffy, A.H.B., 2002, "Identifying Component Modules," Seventh International Conference on Artificial Intelligence in Design AID'02, Cambridge, UK, 15-17 July 2002.

Whitney, D.E., 1990, "Designing the Design Process," *Research in Engineering Design*, 2: 3–13, 1990.

Whitney, D.E., 2003, "Physical Limits to Modularity", MIT, Engineering Systems Division Working Paper Series, ESD-WP-2003-01.03 ESD Internal Symposium.

Wiener, N, 1948, *Cybernetics or Control and Communication in the Animal and the Machine*, MIT Press, Cambridge.

Wikipedia Website – The Free Enyclopedia: en.wikipedia.org

Wilke M., 2003, *Integrierte modellbasierte Satellitenentwicklung*, PhD Thesis, Technische Universität München, Utz Verlag, München, Germany.

Wilke M., Vollerthun A., Schiffner M., Zeyen B., Igenbergs, E., 2000, "Das Space System Concept Center – Eine innovative Lehrumgebung für den integrierten Systementwurf," Proceedings of Deutscher Luft- und Raumfahrtkongress 2000, Leipzig.

Womack, J.P., Jones, D.T., 1996, *Lean Thinking: Banish Waste and Create Wealth in your Corporation*, Simon & Schuster, New York.

Wöhe, G, 1996, "Einführung in die Allgemeine Betriebswirtschaftslehre," 19. Edition, Franz Vahlen, Munich

Yassine, A.A., Braha, D., 2003, "Complex Concurrent Engineering and the Design Structure Matrix Method," *Concurrent Engineering: Research and Applications*, 11(3): 165-176.

Yassine, A.A, Joglekar, N., Braha, D., Eppinger, S.D., Whitney, D, 2003, "Information Hiding in Product Development: The Design Churn Effect," *Research in Engineering Design,* 14(2003): 145–161.

Yu, T.L., Yassine, A.A., Goldberg, D.E., 2003, "An Information Theoretic Method for Developing Modular Architectures Using Genetic Algorithms," Proceedings of ASME 2003 Design Engineering Technical Conferences and Computers and Information in Engineering Conference (DETC2003/DTM-48647) Chicago, USA.

Zakarian, A., Rushton, G., 2001, "Development of modular electrical systems," *IEEE Transactions on Mechatronics,* 6(4): 507–520.

Zäh, M.F., Müller, N., Prasch, M., Sudhoff, W., 2004, "Methodik zur Erhöhung der Wandlungsfähigkeit von Produktionssystemen," *ZWF*, 99(4): 173-177.

Zehnder T., 1997, *Kompetenzbasierte Technologieplanung*, Gabler, Wiesbaden.

Zelenovic, D.M., 1982, "Flexibility – A Condition for Effective Production Systems," *International Journal of Production Research*, 20(3): 319-337.

Zhuang, M., Yassine, A.A., 2004, "Task Scheduling of Parallel Development Projects Using Genetic Algorithms," Proceedings of DETC '04, ASME 2004 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Salt Lake City, Utah USA, September 28-October 2, 2004, (DETC2004/DAC-57159).