17TH INTERNATIONAL DEPENDENCY AND STRUCTURE MODELING CONFERENCE, DSM 2015
FORT WORTH, TEXAS, USA, NOVEMBER 04 – 06, 2015

An Initial Metamodel to Evaluate Potentials for Graph-based Analyses of Product Development Projects

Nepomuk Chucholowski, Udo Lindemann

Institute of Product Development, Technische Universität München, Germany

Abstract: This paper presents an initial metamodel for product development projects. It enables to model different entities (nodes), relationships (edges) and their attributes, understanding product development projects as complex systems. The modeling approach shows potential to provide helpful input for project managers in situations when projects deviate from plan during their execution. The potential was evaluated in a simplified use case. First results are presented and paths for future development of the approach are derived.

Keywords: Project Management, Synchronization Management, Interdependencies, Project models, Graph Theory

1 Introduction

A project can be defined as a "temporary endeavor undertaken to create a unique product, service or result" (PMI, 2013). Product development projects (PD projects) shall particularly relate to projects with the objective to develop a technical product as a complex system. Managing such PD projects is challenged by a high degree of ambiguity and uncertainty both regarding project's objectives (e. g. Browning, 2014) and activities necessary to accomplish the project (e. g. Lévárdy and Browning, 2009). Further often discussed challenges in project management arise from multidisciplinary environments (Hellenbrand, 2013), multi-project environments (Browning and Yassine, 2015) and an increasing degree of development cooperation between companies along the value chain (ProSTEP iViP, 2010). Project management can be seen as the planning and controlling of a project. For this, especially the coordination of activities plays an important role. However, while there is an increasing complexity in products and in ways how organizations join to develop them, project management in practice still often uses only traditional methods such as Gantt-Charts (GPD, 2015).

Existing research efforts address this issue, aiming to support project management for the development of complex systems. Yet, what is missing so far is concrete methodological support for project managers when there is a derivation from plan during the execution of a project (e. g. schedule overrun or lack of human resources). Such situations in today's practice might be handled in smaller projects by project managers who decide what to do – based on their experience (gut-feeling) or project team discussions. In bigger projects, all the consequences of adaptions in project plans are hard to overlook due to product complexity, involved disciplines and, most of all, distributed development. It seems that then, the only considered measures in practice are to add work shifts or external human resources. This not only increases project costs but also error rates.

The objective of this paper is to present primal results of the evaluation whether and how graph-based analysis can support managers of PD projects. (Lévárdy and Browning, 2009) understand project control as a decision making process, which aims to maximize the value of the overall project results, considering its current state and environment. Our superordinate goal is to support these decisions by enabling project managers to capture the current project state (i. e.: what has already been done and what is still to do?). For this, we use a graph-based modeling approach and define a metamodel that integrates relevant aspects of different project systems (cf. Browning et al., 2006) and their interrelations.

The contents of this paper are based on preliminary literature analyses and on initial qualitative interviews with industry experts. Literature about (1) project management and project / process modeling in general, and (2) interdependencies in engineering projects and synchronization management was focused. In order to develop the metamodel, a rapid prototyping approach is chosen, whereas this paper presents the first draft. This procedure aims to allow early feedback from other experts on this topic.

2 Background

2.1 *Why needs* the controlling of PD projects to be supported and *how*?

Today's PD projects are often characterized by a high degree of complexity (Forsberg et al., 2005). This complexity stems from the rising complexity of the products being developed and the correlating complexity of necessary development processes, which include an increasing number of activities, project stakeholders and interdependencies within the project's environment (Lindemann et al., 2009). The complexity paired with uncertainty and ambiguity (being natural to projects) exacerbates project management and leads to project failures and significant cost and/or schedule overruns, as examples given in literature (e. g. Forsberg et al., 2005; Jackson, 2006) and from discussions with practitioners show.

On an abstract level, project management stands primarily for the planning and controlling (i. e. to control/measure and to react) activities in a project. Our preliminary literature study shows, that most methods presented in literature in the area of development project management focus on initial planning activities at the beginning of a project using process modeling, simulation and optimized standard processes. No method is found that aims to support the adaption of a project plan during project execution (controlling). The only guidance found in literature, e. g. for the management elements *project control* and *corrective actions* in (Forsberg et al., 2005), stays on a very abstract level. For example, recommendations on what to do when adaptions on the critical path are necessary, are: eliminate or shorten tasks on the critical path; increase the number of workhours; etc. Especially in the context of distributed product development, it is difficult to synchronize and coordinate development activities and such recommendations are useless. In order to manage interdependencies an integrated view on all project elements and their interrelations is necessary. This can be reached by understanding a PD project as a system.

2.2 Systems perspective on PD Projects

A product development project can be divided into the five subsystems depicted in Figure 1 [(Browning et al., 2006) based on (Negele et al., 1997) and (Ropohl, 1975)].

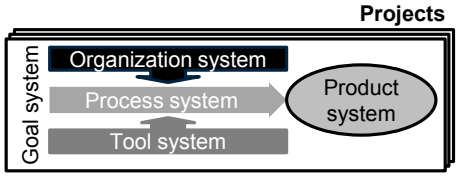

Figure 1. The five systems of a project as presented by (Browning et al., 2006).

The Process Architecture Framework most recently presented in (Browning, 2013) enables to model the process system as a structure of different activities and deliverables that are interconnected. The framework aims to provide a comprehensive basis for many different views on a process, e. g. Gantt-Charts or network diagrams. The other systems of a project are not directly addressed, even though partial interfaces to other systems can be included as attributes of activities or deliverables (e. g. roles that execute an activity). Other frameworks exist that enable to model particular aspects of the other project systems (e. g. functional models, organization charts, resource plans etc.). However, according to (Browning, 2013) it might be beneficial to integrate aspects from all five project systems to enable integrated views on product development projects.

Following his line of argumentation, we aim to develop a metamodel that integrates all relevant aspects of a develop project into a model of different entities and relationships. A metamodel defines a modeling language (Favre, 2005) and hence enables to generate models as instances of the metamodel.

(Forsberg et al., 2005) argue that visualization is essential for humans to comprehend complex issues. That is why we aim to support project control by providing visual models as graphs of relevant project data. This allows to model several entities of the different project systems as nodes and their interrelations as edges. A big advantage of using graphs is that attributes cannot only be allocated to nodes but also to edges. Hence, qualitative and quantitative information describing interrelations such as interdependencies between development tasks can be stored and visualized in graphs. Further, graphs enable computational representation and transformation of data (Helms, 2012) and thus provide extensive possibilities to analyze data and visualize the results. Additionally, graphs can serve as the foundation for almost any formal modeling language such as SysML diagrams, function models or geometric models (Helms, 2012). Hence, they enable to transfer different perspectives on integrated data of a development project into other models.

3 An Initial Metamodel for Product Development Project Graphs

3.1 Elaboration of the metamodel

So far, only few suggestions on which elements of the project systems should be included in an integrated approach are found in literature. For example, (Browning et al., 2006) suggest to extend their generalized process framework with further objects from the five project systems as follows: *Organizational units* (person, team, company, etc.); *Tools* (facility, template, computer system, software application, etc.); *Product elements* (sub-system, component, etc.); *Goals* (requirement, objective, policy, etc.)

Here, no relationships between entities are addressed. (Kreimeyer and Lindemann, 2011) present a multiple-domain matrix (MDM) for engineering design processes, where *tasks* (= activities), *artifacts* (= deliverables), *events*, *organizational units*, *resources*, *time* and *product attributes* are considered as possible entities (i. e. domains in the MDM). Additional to the domains, the MDM specifies different types of relationships between elements (cf. Table 1). The MDM represents a metamodel as a comprehensive aggregation of elements and interdependencies that are modeled in conventional project plans. Yet, the product system is only considered by the domain *product attributes*. The product architecture, for example, is not included in the model. The correlation between elements from the process system and product system is investigated by (Hellenbrand, 2013). His MDM includes *functions* and *components* (product system), *process results*, *process steps* and *milestones* (process system), and *individuals* (organization system). Still, as stated by (Kreimeyer and Lindemann, 2011), a shortcoming of these matrix-based approaches is the limitation in assigning attributes to edges.

Table 1. Extract from the MDM for design processes by (Kreimeyer and Lindemann, 2011).

	Task	Artifact	Event	Organizational unit
Task	• **precedes temporally** • precedes logically	• **has output of** • processes • controls • changes	• **generates** • processes • ends in	• **belongs to**
Artifact	• **is input for** • supports • controls • starts	• **transits into** • is used to create • is in conflict with	• **sends** • is created at • is needed at	• **belongs to**
Event	• **starts** • controls	• **produces**		• **occurs in**
Org. unit	• **is responsible for**	• **has responsibility for**		• **communicates wi**

A new metamodel is developed based on these findings and based on thoughts about which partial models in or between the different project systems exist and are commonly known (Figure 2). The metamodel addresses aspects and relations from all five project systems by defining node classes and edge classes, which can be instantiated and specified with attributes in graph models.

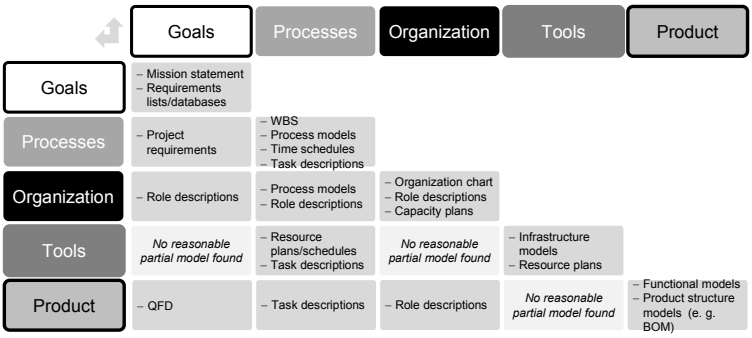

Figure 2. Overview of existing partial models that describe elements (on diagonal) and relationships (on lower triangular matrix) of the five project systems. Abbreviations: WBS - work breakdown structure; QFD - quality function deployment; BOM - bill of materials.

3.2 Description of the metamodel

The suggested metamodel contains entities and relations regarding all five project systems and is illustrated in Figure 3. The relations between the entities are depicted simplified for the sake of clearness. Table 2 shows a complete list of all relations assumed as relevant for the metamodel. A lot more relations are imaginable, which are often also modeled in existing project models. They are considered as indirect relations that can be conducted from other direct relations.

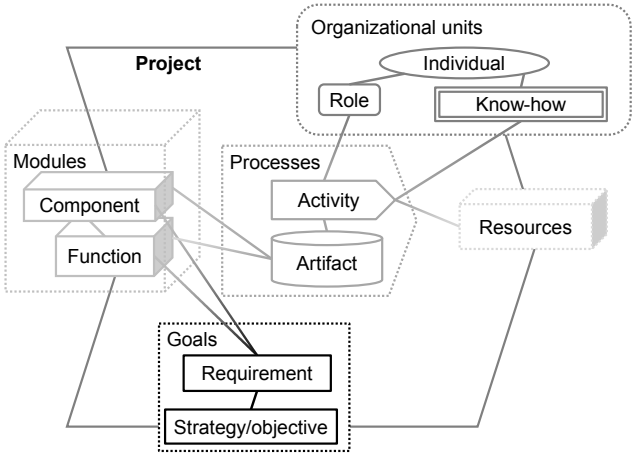

Figure 3. Visualization of the proposed metamodel.

The process system is incorporated by *activities* and *artifacts*, as suggested by (Browning et al., 2006). Activities have different artifacts as inputs and transform them into (or generate new) outputs. Further, artifacts can be related to an activity even though they are not "processed", e. g. when they only provide necessary information to that activity. In order to realize different decomposition levels, activities and artifacts can be assigned to

superior activities and artifacts, respectively. Compositions of activities and artifacts build *processes*. A process, however, is not included as a node class that can be modeled as an instance; it rather serves as a term to qualitatively describe certain groups of activities and artifacts during model analysis. In order to instantiate a group of activities as one element into the model, they can be assigned to a superordinate activity.

The product system is described by product *components* and *functions*. Different relationships between components (e. g. geometrical) or functions (e. g. two functions are realized by the same part) are possible. Analog to activities and artifacts, components and functions can further have a hierarchical structure. Moreover, they can be grouped into *modules*, whereas again like processes, modules cannot be instantiated in a model but facilitate model interpretation.

Typical goals of PD projects are to develop new technical products in a certain time frame and to a defined budget. Hence, the goal system consists of project-related *requirements* (of the project and of the product to be developed). Additionally, superordinate goals such as company-wide *strategies* and *objectives* are part of the goal system and have to be respected in every project.

Roles, *individuals* and *know-how* are entities within the organization system. Roles define the organizational structure in a project and are impersonal descriptions of tasks and responsibilities. Each individual in a company can be assigned to roles and has different know-how. The term ‘know-how’ relates to skills, knowledge or experience, which shall not be distinguished here.

The tool system provides *resources* to the project, whereas here only material resources such as IT-systems, software, machinery or material that are needed in order to execute an activity are meant. In contrast, human resources (e. g. work hour capacity per person) are part of the organization system (individuals) and are modeled as attributes.

The relationships listed in Table 2 are supplemented by an indication whether a relation can point from one [1] or more than one [n] source node(s) to exactly one [1] or more [n] target nodes. For reasons of simplification, the relation *activity requires know-how* is also used when special *know-how is required for* the usage of a *resource* (e. g. how to use a CAD tool), since the relationship only exists when there is an activity actually using the resource. Moreover, the relation *role supports activity* is included in the relation r*ole is responsible for activity* (RACI classification). For some edges attributes are defined. For instance, since artifacts are generated and evolve during the process, it is interesting to document the degree of completion or maturity with these edges.

Table 2. Detailing of the relations included in the metamodel.

Relations	Explanation
Activity [n] *is part of* Activity [1]	Decomposition of activities.
Artifact [n] *is part of* Artifact [1]	Decomposition of artifacts.
Activity [1] *produces* Artifact [n]	A new artifact is generated by an activity.
Activity [n] *edits* Artifact [n]	An already existing artifact is input for and gets changed by an activity, what makes it also an output. **Attribute**: Degree of change
Activity [n] *requires* Artifact [n]	An artifact is used within an activity, without changing the artifact. **Attribute**: Required maturity of artifact
Artifact [n] *instantiates* Component [n]	An artifact describes/defines/specifies a component. **Attribute**: Degree of completion
Artifact [n] *instantiates* Function [n]	An artifact describes/defines/specifies a function. **Attribute**: Degree of completion
Component [n] *is part of* Component [1]	Decomposition of components (part structure).
Function [n] *is part of* Function [1]	Decomposition of functions (functional structure).
Component [n] *realizes* Function [n]	A component takes part in the realization of a technical function.
Component [n] *fulfills* Product Requirement [n]	A component takes part in the fulfillment of a product requirement.
Function [n] *fulfills* Product Requirement [n]	A function takes part in the fulfillment of a functional requirement.
Activity [n] *requires* Resource [n]	A resource is required for a certain activity **Attribute**: Required quantity (e. g. hours)
Activity [n] *requires* Know-how [n]	Know-how is required to execute an activity. **Attribute**: Required know-how level
Individual [n] *is assigned to* Role [n]	Individuals can be assigned to roles which describe their function in an organization.
Individual [n] *has* Know-how [n]	Individuals have know-how. **Attribute**: Know-how level
Role [n] *is responsible for* Activity [n]	A role is disciplinary responsible for the execution of an activity.
Role [1] *is accountable for* Activity [n]	A role is legally/economically accountable for the activity.
Role [n] *is consulted regarding* Activity [n]	A role should be consulted during the execution of an activity in order to get relevant information.
Role [n] *is informed regarding* Activity [n]	A role that has the right to be informed about the activity and its results.

3.3 Exemplary Use Case

In order to preliminary evaluate the feasibility of the pursued approach and the suitability of the developed metamodel for PD projects, the theoretical metamodel is partially described as a modeling language (cf. Figure 4). The modeling language defines different node classes, edge classes and their attributes. Classes can be decomposed into subclasses, whereas subclasses inherit all attributes from their superior classes. Abstract classes are

used to pool a group of attributes that are common to several subclasses. Abstract classes do not appear as an entity in any instance model.

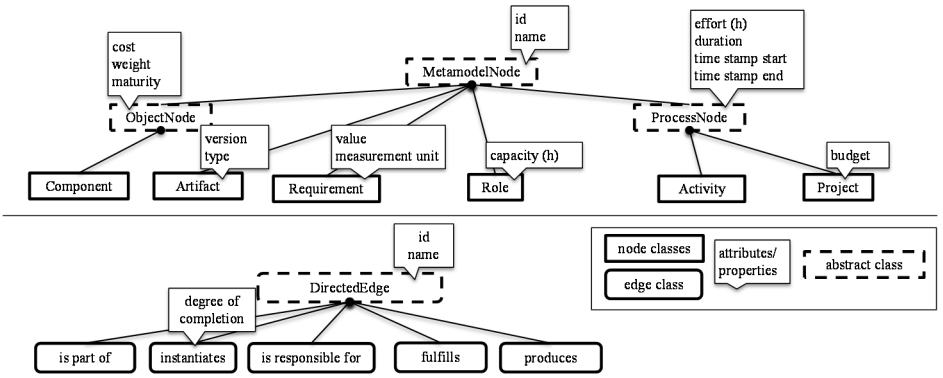

Figure 4. Typification of selected node classes and edge classes for a preliminary implementation of the metamodel for PD projects.

In order to apply the developed metamodel, a fictitious example of a project for the development of an electric drill is used. So far, 17 *components*, 44 *artifacts*, 7 *requirements*, 4 *roles* and 57 *activities* are instantiated; including different decomposition levels to some extent. The nodes are connected via 55 *IsPartOf*-Nodes, 44 *Instantiates*-Nodes, 44 *IsResponsibleFor*-Nodes, 14 *Fulfills*-Nodes and 44 *Produces*-Nodes. All nodes and edges in one graph are depicted in Figure 5. In the next steps, rules have to be defined that enable different perspectives on the underlying project data and workflow sequences have to be developed that allow useful analysis with the help of graph transformation.

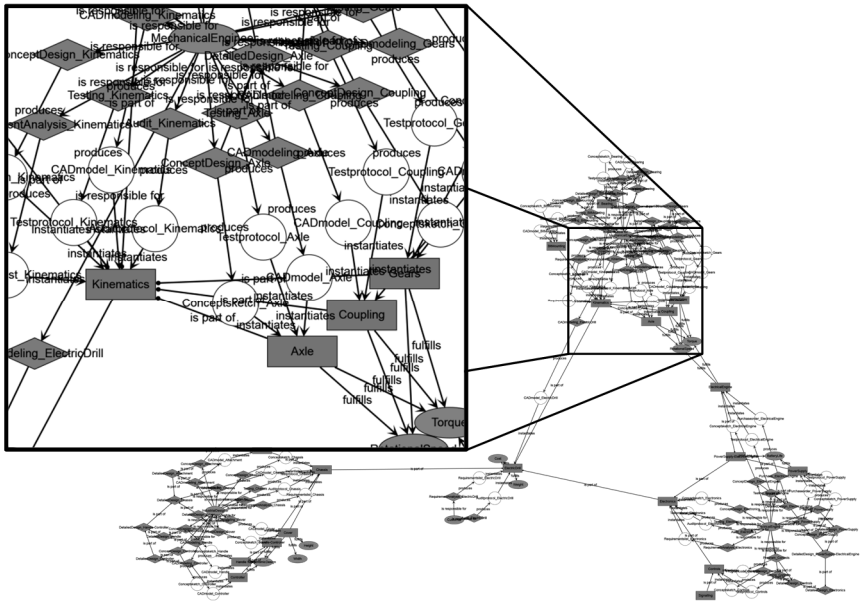

Figure 5. Project graph for the development of an electric drill (created with Soley® Studio, www.soley-technology.com).

5 Conclusion and Outlook

5.1 Advantages

The partial implementation of the proposed metamodel shows that it is feasible to model project data in a graph with nodes and edges. The biggest potential is seen in being able to actually *model* interdependencies between different elements in project systems as objects (edges) with own attributes. A graph-based model further allows computational analyses of structures, using attribute values for nodes and edges at the same time. Moreover, analysis results are visualized as graphs and hence facilitate the comprehension of complex issues. It would further be possible to derive different standard visualizations showing user-specific perspectives on the project systems.

The project data modeled in the graph can be imported from different existing plans, models and datasets. Are the workflows for the data import and analysis procedures defined once, they can be executed again with no extra effort. Thus, the modeling effort, which is an often stated problematic issue in the application of matrix-based methods, is reduced to a minimum.

5.2 Limitations

In the current version of the metamodel, only basic project system elements and direct relationships are included. Important indirect relations, for example, have to be defined also in the metamodel in order to use them in a graph representation or analysis. For the exemplary use case, no iteration of activities was considered and artifacts were just modeled as outputs from the activities (no required inputs). In addition, all nodes and edges were considered as statically given. In order to support project controlling, both descriptive data of what already happened and prescriptive data of what is planned to happen in future, is necessary. These facets are unconsidered, yet.

Further limitations regarding the frontiers between project systems have to be noted. Artifacts (as process results) are part of the process system, but can be seen as instances of the product system. The same is true for requirements. They are abstract entities that describe how the future product should look like. However, the description of requirements in a requirements list is an artifact. Another limitation is that project requirements and company-wide goals cannot directly be assigned to any other entity of the project systems, since they are related to the project as a whole. For example, the overall effort of working hours put into a project is related to the sum of all activities in a project. All these issues are pending to be resolved.

5.3 Outlook

During the elaboration of the metamodel it became clear, that the five project systems cannot be seen as delimited to one project. When there are several projects, many entities only exist once within a surrounding multi-projects system (Figure 6). For example, strategies, employees, test facilities or even components in a product-platform often are related to several projects. Considering this in future, graph-based analytics bear even more potential when the management of multiple projects is supported.

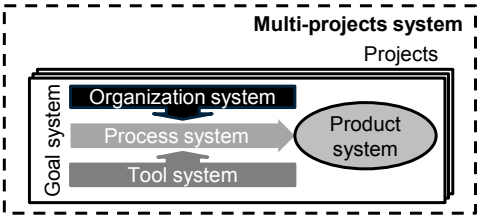

Figure 6. Projects as part of a multi-projects system, where certain entities have to be considered in several projects comprehensively.

In future, the metamodel will be refined and workflows for graph analyses will be developed. Other models and simulation approaches will be included. Further, graphical representations of the results will be elaborated.

References

Browning, T.R., 2013. Managing Complex Project Process Models with a Process Architecture Framework. International Journal of Project Management 32 (2), 229–241.

Browning, T.R., 2014. A Quantitative Framework for Managing Project Value, Risk, and Opportunity. IEEE Transactions on Engineering Management 61 (4), 583–598.

Browning, T.R., Fricke, E., Negele, H., 2006. Key concepts in modeling product development processes. Systems Engineering 9 (2), 104–128.

Browning, T.R., Yassine, A.A., 2015. Managing a Portfolio of Product Development Projects under Resource Constraints. Deision Sciences 46 (5).

Favre, J.-M., 2005. Foundations of Meta-Pyramids: Languages vs. Metamodels - Episode II: Story of Thotus the Baboon, in: Bezivin, J., Heckel, R. (Eds.), Language Engineering for Model-Driven Software Development. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.

Forsberg, K., Mooz, H., Cotterman, H., 2005. Visualizing Project Management : Models and Frameworks for Mastering Complex Systems, 3rd ed. John Wiley & Sons, Hoboken, New Jersey.

GPD, 2015. Plenty of tools, too much data, unending meetings, and plans difficult to believe in. URL: <http://www.gpdesign.com/learn/papers/pdfs/GPD_DS_TooMuchData.pdf>. Accessed 9 April 2015.

Hellenbrand, D., 2013. Transdisziplinäre Planung und Synchronisation mechatronischer Produktentwicklungsprozesse, 1st ed. Dr. Hut, München.

Helms, B., 2012. Object-Oriented Graph Grammars for Computational Design Synthesis. PhD Thesis, München.

Jackson, C.K., 2006. The Mechatronics System Design Benchmark Report: Coordinating Engineering Disciplines. Aberdeen Group, 30 pp. Accessed 17 April 2015.

Kreimeyer, M., Lindemann, U., 2011. Complexity metrics in engineering design: Managing the structure of design processes. Springer, New York.

Lévárdy, V., Browning, T.R., 2009. An Adaptive Process Model to Support Product Development Project Management. IEEE Transactions on Engineering Management 56 (4), 600–620. 10.1109/TEM.2009.2033144.

Lindemann, U., Maurer, M., Braun, T., 2009. Structural Complexity Management: An Approach for the Field of Product Design. Springer, Berlin.

Negele, H., Fricke, E., Igenbergs, E., 1997. ZOPH - A Systemic Approach to the Modeling of Product Development Systems. INCOSE International Symposium 7 (1), 266–273. 10.1002/j.2334-5837.1997.tb02181.x.

PMI, 2013. A guide to the project management body of knowledge (PMBOK guide).

ProSTEP iViP, 2010. Collaborative Project Management PSI 1-1: Recommendation: Reference Model; Version 3.0. ProSTEP iViP, 78 pp.

Ropohl, G., 1975. Systemtechnik - Grundlagen und Anwendung. Hanser, München.

Acknowledgements

We thank the German Research Foundation (Deutsche Forschungsgemeinschaft – DFG) for funding this project as part of the collaborative research center ‘Sonderforschungsbereich 768 – Managing cycles in innovation processes – Integrated development of product-service-systems based on technical products’. We also thank Kevin Burger for his support in elaborating the example and graph models. Last, we gratefully thank Soley®-Team for providing Soley® Studio (www.soley-technology.com/en/pr-soley-studio/) and their precious knowledge in graph modeling.

Contact: N. Chucholowski, Institute of Product Development, Technische Universität München, Boltzmannstraße 15, 85748 Garching bei München, Germany, +49 89 289 151 36, chucholowski@pe.mw.tum.de