Technische Universität München
TUM School of Management

Three Essays on Costing and Decision Making in the Digital Age

Marcus Richard Witter

Vollständiger Abdruck der von der TUM School of Management der Technischen Universität München zur Erlangung eines Doktors der Wirtschafts- und Sozialwissenschaften (Dr. rer. pol.) genehmigten Dissertation.

Vorsitz:	Prof. Dr. Jürgen Ernstberger
Prüfer der Dissertation:	1. Prof. Dr. Gunther Friedl
	2. Prof. Magne Jørgensen, Ph.D.

Die Dissertation wurde am 16.03.2023 bei der Technischen Universität München eingereicht und durch die TUM School of Management am 15.05.2023 angenommen.

Acknowledgements

I sincerely thank my doctoral supervisor Gunther Friedl for his mentoring guidance, constructive feedback, and continuous support throughout my doctoral studies and beyond. Sitting in my first university class in Cost Accounting in May 2011 at TUM, I could have never imagined where my academic trajectory would take me. Thank you for providing the platform to follow my research interests and encouraging me to participate in conferences and conduct a research stay abroad. I extend my gratitude to Magne Jørgensen and his team for welcoming me kindly to the Simula Research Laboratory. Discussing my essays with you and your team was enriching and very instructive.

I further thank my doctoral fellows and friends for mutual encouragement, fruitful discussions on various research topics, and, particularly, for creating a collaborative atmosphere at the chair. Thank you for countless trips, days full of laughter, and fun-filled evenings. I give special regards to Eline Schoonjans, Moritz Rombach, and Peter Schäfer for their detailed feedback and countless discussions on my essays, from which I have benefited a lot, and Alexander Schult, Lukas Schloter, and Yanis Gamarra for keeping the team spirit up. Further, I thank my co-author Michael Blumberg for collaborating constructively on my third essay. Moreover, I am grateful to Strategy&, particularly, Christian Foltz, Hans-Jörg Kutschera, and Henning Rennert for their generous support and counsel during my doctoral thesis.

My heartfelt gratitude belongs to my family, my parents Sally and Ronald, and my brother Sven for their relentless support throughout my life, teaching me perseverance and advising me to slow down and speed up when necessary. Finally, I am indebted to my fiancée and partner in life, Sabine, for her never-ending encouragement, valuable feedback, and help in prioritizing my ideas and work. Words cannot describe how thankful I am to share the journey through life together with you. This dissertation is dedicated to them.

Abstract

The digital transformation affects many industries and shifts firms' value creation towards digital products and services. In this dissertation, I use different research methods to investigate the implications of the digital transformation on firms' decision-making and cost-management practices. In the first essay, I run an experiment to investigate how accountability, i.e., the need to justify one's decision, affects decision-making quality under increasing levels of information load. I find that information overload reduces decision-making quality, and I provide evidence that managers facing information overload are more likely to make the optimal decision if they are held accountable than if they are not. In the second essay, I conduct a multiple-case study to assess how firms across industries estimate their future software project costs. I find that firms structure their cost estimation process into three stages, i.e., *Understand*, *Plan*, and *Implement*, and ten activities. I describe the activities of each stage and show how firms adapt their actions to their organizational and project-specific setting. I identify challenges that firms face during their cost estimation and triangulate these with extant literature to derive propositions that support managers in designing their estimation process. In the third essay, I propose a cost system design for software firms that accounts for the changes in cost structures. The cost system integrates different cost-management and modeling approaches and allows firms to evaluate past, calculate current, and manage future software product costs. The three essays contribute to the decision-making and cost-management literature and business practice from different perspectives. First, information providers such as controllers should be careful not to overload managers with information and, if necessary, consider holding them accountable for their decisions. Second, I provide a process model for estimating software costs that practitioners can use to standardize their estimation process. Third, I discuss how controllers can re-design firms' cost systems to meet digital product and process requirements.

Summary in German

Diese Dissertation befasst sich mit den Auswirkungen der digitalen Transformation auf die Entscheidungsfindung und das Kostenmanagement in Unternehmen. Im ersten Aufsatz untersuche ich experimentell, wie sich die Einführung einer Rechenschaftspflicht auf die Entscheidungsqualität bei ansteigender Informationslast auswirkt. Ich stelle fest, dass eine Informationsüberlastung die Entscheidungsqualität verringert. Darüber hinaus liefere ich Evidenz, dass Manager bei Informationsüberlastung häufiger die optimale Entscheidung treffen, wenn sie rechenschaftspflichtig sind. Die Ergebnisse implizieren, dass Informationsgeber wie Controller ihre Manager nicht mit Informationen überladen und die Rechenschaftspflicht als Controlling-Instrument zur Entscheidungsoptimierung prüfen sollen. Im zweiten Aufsatz führe ich eine Multiple-Case-Studie durch, um zu untersuchen, wie Unternehmen die Kosten ihrer Softwareprojekte schätzen. Ich stelle fest, dass Unternehmen den Kostenschätzungsprozess in drei Phasen und zehn Aktivitäten strukturieren und fasse die Ergebnisse in einem Prozessmodell zusammen. Ich beschreibe die Aktivitäten jeder Phase und zeige, wie Unternehmen ihre Handlungen an ihre organisatorischen und projektspezifischen Gegebenheiten anpassen. Ich identifiziere Herausforderungen, denen sich Unternehmen während der Kostenschätzung gegenübersehen, und trianguliere sie mit der wissenschaftlichen Literatur, um Empfehlungen abzuleiten, die Manager bei der Gestaltung ihres Schätzungsprozesses unterstützen. Im dritten Aufsatz schlage ich ein Kostensystemdesign für softwareproduzierende Unternehmen vor. Die Basis stellt eine Analyse bereits existierender Kostensysteme und der Eigenschaften von Softwareprodukten und -prozessen dar. Das Kostensystem integriert verschiedene Kostenmanagement- und -modellierungsansätze und ermöglicht es Unternehmen, vergangene, aktuelle und zukünftige Softwarekosten zu berechnen. Der Aufsatz leistet einen Beitrag zur Literatur über die Gestaltung von Kostensystemen, indem ich erörtere, wie Controller die Kostensysteme von softwareproduzierenden Unternehmen gestalten können, um die Anforderungen an digitale Produkte und Prozesse zu erfüllen.

Contents

List of figures **vi**

List of tables **viii**

Abbreviations **ix**

1 Introduction **1**

1.1 Motivation 1
1.2 Theoretical background and literature context 7
1.3 Methodologies and results 10
1.4 Contribution to academic and practical debates 16
1.5 Structure of the dissertation 19

2 Finding the Needle in the Haystack: How Information Load and Accountability Influence Decision Quality **22**

2.1 Introduction 23
2.2 Theory and hypotheses development 27
2.2.1 The effect of information load 27
2.2.2 The effect of accountability 28
2.3 Experimental method 30
2.3.1 Experimental setting 30
2.3.2 Experimental conditions and variables 33
2.4 Results 34
2.4.1 Participants and procedure 34
2.4.2 Test of hypotheses 36
2.5 Discussion and conclusion 40

3 Understand, Plan, and Implement: A Multiple-Case Study on How Firms Estimate Software Costs **43**

3.1 Introduction 44
3.2 Background and related work 47
3.2.1 Software cost estimation processes 47

3.2.2 Software cost estimation methods . . . 49
3.3 Research design . . . 53
3.3.1 Multiple-case study research approach . . . 53
3.3.2 Data sample . . . 54
3.3.3 Data sources and analysis . . . 55
3.4 Results . . . 57
3.4.1 Outline of the UPI process model for cost estimation . . . 57
3.4.2 Description of the UPI model for software cost estimation . . . 58
3.4.2.1 Understand . . . 58
3.4.2.2 Plan . . . 60
3.4.2.3 Implement . . . 69
3.5 Challenges and propositions . . . 72
3.6 Conclusion . . . 76

4 Divide and Conquer: Designing Cost Systems for Software Firms 79
4.1 Introduction . . . 80
4.2 Theory and research question . . . 84
4.2.1 Review of extant cost systems . . . 84
4.2.2 Accounting for information goods' cost structure . . . 87
4.3 Proposed software cost-management and modeling systems . . . 89
4.3.1 Life-cycle framework for software projects . . . 89
4.3.2 Inherent cost allocation model . . . 93
4.3.3 Dynamic target costing . . . 96
4.4 Integration of the cost-management and modeling systems . . . 98
4.5 Discussion and conclusion . . . 100

5 Conclusion 103
5.1 Summary of main results . . . 103
5.2 Limitations . . . 105
5.3 Avenues for future research . . . 107
5.4 Concluding remarks . . . 108

Appendix 110

Bibliography 124

List of figures

1 Introduction **1**

1.1 Schematic assignment of the three essays to the software life-cycle phases and organizational levels. 17

2 Finding the Needle in the Haystack: How Information Load and Accountability Influence Decision Quality **22**

2.1 Predicted effects. 30

2.2 Instructions in the Accountability—Present conditions. 31

2.3 BSC data in the Information Load—High and Accountability—Present condition. 32

2.4 Results (dependent variable = decision quality). 37

3 Understand, Plan, and Implement: A Multiple-Case Study on How Firms Estimate Software Costs **43**

3.1 Basic estimation process. 48

3.2 Categorization of software cost estimation methods. 50

3.3 Conceptual illustration of the UPI model. 58

4 Divide and Conquer: Designing Cost Systems for Software Firms **79**

4.1 Conceptual illustration of life-cycle costing in the context of software projects. . . 90

4.2 Typical cost collection and allocation mechanisms segmented by cost source for a software firm. 91

4.3 Inherent cost allocation based on the inherent cost drivers of a software project. . 94

4.4 Systematic framework outline. 98

Appendix **110**

A.1 Case-related instructions. 110

A.2 BSC data. 111

A.3 A briefing note. 112

A.4 Sequence of events in the experimental design. 113

A.5 Overview of the data structure. 116

A.6 Data structure of the *Understand* phase. 117
A.7 Data structure of the *Plan* phase (1/2). 118
A.8 Data structure of the *Plan* phase (2/2). 119
A.9 Data structure of the *Implement* phase. 120
A.10 Cost structure table for industrial goods and information goods. 121
A.11 Illustration of the dynamic target costing process. 122
A.12 Description of the dynamic target costing process. 123

List of tables

1 Introduction **1**

1.1 Overview of the three essays. 21

2 Finding the Needle in the Haystack: How Information Load and Accountability Influence Decision Quality **22**

2.1 Descriptive statistics for decision quality for each experimental condition. 36

2.2 Factorial ANOVA (dependent variable = decision quality). 37

2.3 Simple effects (dependent variable = decision quality). 38

2.4 Descriptive statistics for perfect decision quality for each experimental condition. 39

2.5 Logistic regression (dependent variable = the optimal decision). 39

2.6 Simple effects (dependent variable = the optimal decision). 40

3 Understand, Plan, and Implement: A Multiple-Case Study on How Firms Estimate Software Costs **43**

3.1 Sample overview. 55

3.2 Challenges and propositions. 73

4 Divide and Conquer: Designing Cost Systems for Software Firms **79**

4.1 Overview of the suggested cost-management and modeling systems. 83

Appendix **110**

A.1 Case study protocol. 114

A.2 Interview questions. 115

Abbreviations

ABC	Activity-based costing
AI	Artificial intelligence
BSC	Balanced scorecard
CAS	Complex adaptive systems
CEO	Chief executive officer
COCOMO	Constructive cost model
EBSPM	Evidence-based software portfolio management
ECU	Electronic control unit
EEPS	Early estimation and planning stages
ERP	Enterprise resource planning
ISBSG	International software benchmark standards group
IT	Information technology
KPI	Key performance indicator
LCC	Life-cycle costing
ML	Machine learning
MBO	Management-by-objectives
PERT	Program evaluation and review technique
PO	Product owner
R&D	Research and development
SLOC	Source lines of code
SVP	Senior vice president
UPI	Understand, plan, and implement

1 Introduction

1.1 Motivation

"We have to master the digital transformation, if we want to survive."

Herbert Diess, Former CEO of Volkswagen AG

(Afhüppe et al., 2018)

In an interview with the German business newspaper *Handelsblatt*, Herbert Diess refers to "the digital transformation as the central task of the automotive industry," a task that is even more vital than the electrification of vehicles (Afhüppe et al., 2018). The quote of the former CEO of Volkswagen AG highlights the importance and actuality of accommodating the opportunities and challenges of the digital transformation.

Digital technologies, viewed as combinations of information, computing, communication, and connectivity technologies (Bharadwaj et al., 2013), enable products to realize innovative functionalities. Scholars and practitioners refer to digital transformation as the process of utilizing digital technologies to induce organizational changes that generate new paths for value creation (Vial, 2019) and lead to profound shifts on both firm and industry levels (Scott and Orlikowski, 2022). Research on digital transformation has focused on how firms strategically shift processes, products, and services within and across firms, i.e., how firms create value (e.g., Yoo et al., 2010, Vial, 2019, Warner and Wäger, 2019).

The rise of digital technologies goes hand in hand with the process of digitization and digitalization. Digitization, i.e., the encoding of analog information into a digital format (Yoo et al., 2010), is induced by Moore's law and an increasingly cheap and easy-to-use digital infrastructure (Fichman et al., 2014). Hardware miniaturization, increasing microprocessor power, cheaper and

more reliable memory, broadband communication, and a more efficient power management foster firms to digitize functionalities and capabilities of physical products, including cars, phones, televisions, cameras, and books (Yoo et al., 2010). With embedded digital capabilities, products offer new functionalities, opportunities for customer use, and improved price/performance characteristics that transform their development and production processes (Yoo et al., 2010). The increasing digitization, software, and processing power facilitate firms' ability to acquire and process data. The increasing level of data creates new opportunities and challenges for firms (Bhimani and Willcocks, 2014, Gupta et al., 2018). On the one hand, firms benefit from automizing routine processes and introducing business intelligence and data analytics to improve decision making (Möller et al., 2020). On the other hand, firms must adapt their business activities to the new digital products and services.

Scholars and practitioners generally refer to the term digitalization as the use of digital technologies to change a business model and provide new revenue and value-producing opportunities—it is the process of moving to a digital business (Gartner glossary, 2020). Studies examine how digitalization influences industry standards (Scott and Orlikowski, 2022) and affects firms' strategic choices in developing and offering digital products and services (e.g., Yoo et al., 2010, Lusch and Nambisan, 2015), digital platforms and infrastructure (e.g., Hinings et al., 2018), new digital business strategies (e.g., Bharadwaj et al., 2013, Adner et al., 2019), and business models (e.g., Warner and Wäger, 2019).

The automotive industry displays an instructive example of the shift from physical to digital value creation (Riasanow et al., 2017). Car manufacturers digitize most subsystems of their cars and connect these through car-based software architectures, leading to cars acting as computer platforms on which firms across industries can develop and integrate new devices, networks, services, and content (Henfridsson and Lindgren, 2010, Yoo et al., 2010). Software was first combined with hardware to an embedded system in the late 1960s to improve engine controls. Over time, engineers connected the isolated functions to systems containing multiple functions that interact and depend on each other, realizing more complex tasks (Broy, 2006). Only until the mid of the 2000s, up to 40% of a car's production costs were related to electronics and software, whereas 50-70% of the embedded systems were software costs (Broy, 2006). The growing importance of software is further increasing based on industry trends like autonomous driving, connectivity, and electrification, for which software displays the functional enabler (Riasanow et al., 2017). Automotive embedded software is a multi-billion dollar market and is expected

to continuously grow 9% until 2030 (McKinsey & Company, 2021). However, the shift in value creation is not only limited to the Automotive industry. Firms in various industries, such as aerospace, industrial machinery, and consumer electronics, rely on embedded systems and direct their business focus increasingly towards embedded systems (McKinsey & Company, 2021).

The shift in value creation from hardware to software-oriented products entails various challenges. As software development costs increasingly dominate a vehicle's total costs, car manufacturers must adapt their traditional cost-by-part paradigm and production-centric cost models (Broy, 2006). Further, practitioners found that automotive project software complexity has grown 300% over the past decade. Thereby, the software complexity grows twice as fast as the growth in development productivity (McKinsey & Company, 2020), leading to launch delays, budget overruns, and quality issues. Aerospace firms experience similar trends. As engineers implement system functionalities with software, the source lines of code (SLOC) are doubling every four years (Aeresospace Vehicle Systems Institute, 2016). At the same time, development efforts and costs for the respective systems increase exponentially with the increase in SLOC (Aeresospace Vehicle Systems Institute, 2016). Thus, firms need to find ways of reducing development costs.

Research has shown that technological improvement is only one part of the puzzle for firms to stay competitive in a digital world (Vial, 2019). Firms must adjust their strategy (Bharadwaj et al., 2013), management practices (Matt et al., 2015), organizational structure (Selander and Jarvenpaa, 2016), processes (Carlo et al., 2012), and culture (Karimi and Walter, 2015) to successfully follow new paths of value creation (Svahn et al., 2017). Many car manufacturers and suppliers have announced strategic initiatives to rebuild their organizations to adapt to software's rising importance. A case in point is the German car manufacturer Volkswagen, which founded the *CARIAD SE* to strengthen its software development efforts (CARIAD SE, 2021). The firm's objective is to increase the in-house share of car software development from less than 10% in 2020 to more than 60% by 2025. Therefore, Volkswagen is willing to invest more than 7 billion EUR and aims to expand the organization from 3,000 employees to more than 10,000 digital experts by hiring new employees, entering into new partnerships, and acquiring companies until 2025 (Volkswagen Group, 2019).

Digitalization also entails implications for various business activities, supply chains, and support functions (Möller et al., 2020). For example, the digital transformation affects the finance function, controllers' tasks, and practices (Bhimani and Willcocks, 2014, Brands and Holtzblatt,

2015).[1] Organizational cost structures shift and are characterized by rising fixed costs that strongly exceed variable cost elements (Afuah and Tucci, 2001, Bhimani and Willcocks, 2014). Although investments in technology and hardware for use in production always had an impact on cost structures, the effect is particularly strong for investments in digital technology (Bhimani and Willcocks, 2014). Further, traditional capital budgeting and investment control approaches might not account for the characteristics of digital products and their potential for exponential growth fueled by platform strategies and network economies (Möller et al., 2020).

Firms must react to the organizational cost mix changes and adapt traditional cost-management practices as they move towards digitally enabled businesses (Bhimani and Willcocks, 2014). Typical cost-management approaches to calculating costs per item produced or delivered do not hold for virtual firms (Bhimani and Willcocks, 2014). Controllers must find models beyond the cost or market-based traditional approaches, which account for product- and value-co-creation (Bhimani and Bromwich, 2009). Costing methods, such as functional or activity-based costing, are only applicable to a limited degree as the cost variability in digitized operations does not relate to traditional volume- and non-volume-based cost drivers (Bhimani and Bromwich, 2010). Revenue sources, such as key customers, may differ from the actual resource consumers. Costs occur when developing and providing consumer platforms, but the revenue is generated from the volume instead of the activities or services consumed by the users. Examples include advertising, market experiments, or customer data derived from Big Data analysis (Bhimani and Bromwich, 2010). As the revenue generators must not cause costs, internet-based firms need to operate on different commercial models than traditional firms producing tangible goods and require their accounting system designs to incorporate some financial intelligence to capture the dynamics of cost and revenue sources (Bhimani and Willcocks, 2014). The differentiation between products consumed by users and customers who generate revenues also necessitates firms to rethink their pricing strategy and the calculation of quality, target, or product life cycle costs (Castells, 2010).

The shift in value creation necessitates extended job roles and skills for the finance function within firms and, in particular, for providing management accounting information (Bhimani and Willcocks, 2014, Oesterreich et al., 2019). Individuals in finance functions may need to develop new competencies and build up expertise in technology and analytics (Möller et al., 2020). The traditional image of a "bean counter" and number-centric "information provider" shifts towards job roles like a "business partner" or a "change agent" who internally consults and

[1] I use the term "finance function" instead of "management accounting and control function". Similarly, I refer to the term "controller" rather than "management accountant".

provides services to the firm's management (Oesterreich et al., 2019). In the past, controllers relied on internal data sources to provide information for making decisions, e.g., concerning sales, expenses, or product costs (Brands and Holtzblatt, 2015). Today, controllers can access rapidly growing data volumes inside and outside the organization, enabling future-oriented analytics, e.g., predictive analytics, to identify, understand, and react to market trends and customer behavior when developing products and making strategic decisions (Bhimani and Willcocks, 2014). Brynjolfsson et al. (2011) and Brynjolfsson and McElheran (2016) find that firms that apply data and analytics for decision making achieve higher productivity and better outputs. In this context, firms must evaluate how to provide accounting information to managers (Shields, 1995). Managers need to understand and analyze the altered nature of data, including structured and unstructured data from various sources, activities, processes, and devices. Managers must learn to process this information into formatted reports to allow for more effective and efficient decision making (Bhimani and Willcocks, 2014).

This dissertation contributes to the academic literature on costing and decision making in the digital age from the perspective of an accounting researcher. In doing so, I derive three research questions at the interface of management accounting and software engineering and aim to build a bridge between both disciplines. In three essays, I investigate (1) whether practitioners can mitigate the detrimental effects of information overload on decision-making quality by introducing accountability as a control mechanism, (2) how firms estimate their software costs, and (3) propose a cost system design for software firms. I apply experimental, qualitative, and conceptual research methods to answer the research questions soundly.

The dissertation's first research question (cf. **Essay I** in Chapter 2) experimentally investigates whether and how accountability affects decision-making quality when managers face different levels of information load. The analysis addresses the issue that the increasing volume of available information, driven by artificial intelligence (AI) and big data, supports firms in acquiring and processing information but also implies the risk of information overload (Gupta et al., 2018). I consider managers as boundedly rational agents who encounter limits in solving complex problems and processing information (Simon, 1955) and making suboptimal decisions or avoid decision making when they face a number of choices between five and nine (Iyengar, 2010). The role of controllers plays a decisive role when investigating issues of information overload. On the one hand, controllers are significant information providers to firms' decision-makers and, to

some extent, responsible for designing usable information systems and decision-making frameworks. On the other hand, controllers make crucial decisions and are therefore subject to being overloaded with information (Schick et al., 1990).

While Essay I focuses on decision making in the digital age, Essay II and Essay III emphasize the costing practices of software firms. In **Essay II** (cf. Chapter 3), I conduct a multiple-case study to provide a process model that shows how firms estimate their software costs. While delivering software projects on time and within budget gains increasing importance in the digital economy (Rahmati et al., 2021), software projects exceed their budget on average by 30% percent (Halkjelsvik and Jørgensen, 2012). The successful delivery of projects strongly depends on meeting time and cost estimates (Chow and Cao, 2008). In this context, improving estimation accuracy allows firms to more effectively plan and control the project budget, reduce costs and delays, and improve customer satisfaction (Heemstra, 1992, Jørgensen and Carelius, 2004, Huang et al., 2008). However, estimating costs for software projects is challenging due to their complex development environment, including various technical and social factors (Kula et al., 2022). In the past, scholars have focused on developing statistical models for estimating software costs (Moløkken-Østvold and Jørgensen, 2005, Menzies et al., 2017). Today, scholars call for research involving practitioners in estimation studies to produce relevant findings for business practice (Eduardo Carbonera et al., 2020) and understand the "why" and "how" factors (Hannay et al., 2007) to incorporate the human aspects of software development (Hannay et al., 2007). This essay aims to create transparency by describing how firms comprehensively estimate software costs. Further, I outline common estimation challenges and derive propositions for managers to increase process consistency and enhance project control by triangulating the results of the qualitative data analysis with literature on software cost estimation.

Essay III (cf. Chapter 4) proposes a cost system design for software firms that can serve multiple purposes in the context of computing software's product costs. Introducing guidelines for designing cost systems for software products is important because the digital transformation creates new paths for value creation (e.g., Yoo et al., 2010, Vial, 2019, Warner and Wäger, 2019), shifting the focus from hardware products to digital technologies. The rise of digital technologies creates new opportunities for firms but also involves challenges for existing business activities and support functions (Vial, 2019, Möller et al., 2020), such as the finance function (Bhimani and Willcocks, 2014). Firms must adapt traditional cost-management practices to account for the organizational cost mix changes, characterized by rising fixed costs that strongly exceed the

variable costs (Bhimani and Willcocks, 2014). Extant literature on designing cost systems has focused on manufacturing and commercializing tangible goods (Schweitzer et al., 2015). Astromskis et al. (2014) find low dissemination of cost systems in software firms, which they relate to the lack of cost systems adapted to the requirements for software's product and process peculiarities and low expected benefit of applying these systems. Thus, we follow the call for research to adapt management accounting systems to industry- and firm-specific factors (Messner, 2016) by analyzing differences in the development, manufacturing, and commercialization of traditional industrial goods and intangible goods such as software and outline the benefits of implementing such systems.[2]

1.2 Theoretical background and literature context

The three essays address different issues for controllers in the digital age. Consequently, I refer to different theories and literature streams when deriving my research questions and hypotheses.

Essay I grounds on the information load and accountability theories in management accounting literature. Controllers need to recognize three variables when providing managers with reports to enable efficient decision making: (1) the uncertainty of the managers, (2) the information load, and (3) the data load (Iselin, 1993). Information and data load differ as information load only includes information cues relevant to the decision, while data load also contains irrelevant cues. Scholars differ between two types of data load. First, information cues that are not relevant to the decision at any time. Second, cues that may predict the main criterion but are correlated at the same time with another cue that predicts the criterion equally well or better. The latter is part of information load (Iselin, 1993) and used in my experiment.

Information overload describes a condition in which the processing requirements of individuals exceed their processing capacity (Schneider, 1987). Information overload can originate from cognitive restrictions in processing information (Simon and Newell, 1971), little motivation (Muller, 1984), or too much choice of different options (Iyengar, 2010). An increase in the amount of information requires individuals to process more information, making it difficult to filter the relevant cues for decision making. The filtering process can be error-prone and reduce individuals' cognitive capacity, leading to lower decision quality (Iselin, 1993). Further, information overload

[2]Essay III in Chapter 4 is based on a joint research project. I refer to my co-author and me when using plural pronouns in the context of this essay.

can emerge from high processing requirements related to complex tasks (Tushman and Nadler, 1978), time pressure (Schick et al., 1990), or budget restrictions (Roetzel, 2019). Lastly, scholars agree that information overload harms the performance of individuals (Eppler and Mengis, 2004). Drawing on information load literature, I state my first hypothesis: *Information overload reduces decision quality.*

I introduce accountability to mitigate the adverse effects of information overload on decision quality when no decision aids are available to managers. Accountability is a crucial design feature of management control systems (e.g., Ahrens, 1996, Merchant and Otley, 2006, Birnberg et al., 2008, Fehrenbacher et al., 2020) and "refers to the implicit or explicit expectation that one may be called on to justify one's beliefs, feelings, and actions to others" (Lerner and Tetlock, 1999, p. 255). Scholars show that holding individuals accountable can improve judgment and decision quality (e.g., Siegel-Jacobs and Yates, 1996, Libby et al., 2004, Chang et al., 2013, Dalla Via et al., 2019, Fehrenbacher et al., 2020) by stimulating critical thinking and anticipating objections in one's argumentation (Tetlock, 1983, Ahrens, 1996), increasing their mental effort to justify their position (Tetlock et al., 1989) and search effort (Schneider, 1987). Referring to accountability theory, I expect a positive influence on decision quality. However, I hypothesize that the effect of accountability on filtering of information under a low information load is smaller than under information overload. Thus, I study their interactive effects and state my second hypothesis: *Increasing information load raises the positive impact of accountability on decision quality.*

Essay II picks up two literature streams for conducting the multiple-case study of how firms estimate software costs. First, I review extant cost estimation processes in the software engineering literature. I describe the basic principles of the process (Trendowicz and Jeffery, 2014) and outline three exemplary procedures. First, the software planning and control framework by Boehm and Papaccio (1988) suggests improving software costs by performing management-by-objectives (MBO) control loops and organizing the development strategy around project predictability and control. Second, the early estimation and planning stages (EEPS) model by Edwards and Moores (1994) differentiates on a high level between the estimation of software costs before project launch (top-down) and the control of costs during the project (bottom-up). Third, the two-stage estimation process for large-scale distributed agile projects by Usman et al. (2018) distinguishes between the high-level quotation stage and the detailed analysis stage for a product customization task.

Second, I analyze the literature on effort and cost estimation in the software engineering domain. Scholars have developed and tested different estimation methods since the 1980s (e.g., Benbasat and Vessey, 1980) while two categories of methods have proven to be dominant. One category focuses on developing statistical estimation models for estimating software costs (Moløkken-Østvold and Jørgensen, 2005, Menzies et al., 2017). The other category focuses on improving expert judgments (Basten and Mellis, 2011), which is the dominant method applied in practice (Trendowicz et al., 2011, Jørgensen, 2014, Usman et al., 2015). However, scholars argue that there is no "one-size-fits-all" approach for estimating software costs of all domains and applications (Resmi and Vijayalakshmi, 2019). Additionally, scholars find that the research on software cost estimation does not respond to the needs of practitioners (Basten and Mellis, 2011) and call for research to involve professionals in estimation studies (Eduardo Carbonera et al., 2020) to better understand the "why" and "how" factors (Hannay et al., 2007).

In **Essay III**, we evaluate the product and process characteristics of intangible goods such as software. Then, we review extant cost systems which support managers to compute product costs (Balakrishnan et al., 2012), determine product prices (Banker et al., 1994), and plan long-term resource capacities (Balakrishnan et al., 2011). We discuss the basics of cost system theory, e.g., selecting a cost object dependent on the decision situation and determining costs by collecting data and assigning direct and, in particular, overhead costs to a cost object. We examine three concepts in more detail: activity-based costing (ABC), life-cycle, and target costing. Cooper and Kaplan (1988) introduced the ABC concept to respond to the increasing share of indirect costs across industries. Their idea was to determine costs for a cost object based on activity consumption (Noreen, 1991). Therefore, they suggest defining multiple indirect cost pools which share similar activities (Noreen, 1991). Life-cycle costing defines the entire product life cycle within the scope of cost observation, integrating design, development, and post-commercialization phases (Riezler, 1996, Schweitzer et al., 2015) During cost observation, managers can choose between a planning perspective to make decisions early in the product life cycle and optimize costs or an as-is perspective to realize adjustments of actually incurred costs (Ewert and Wagenhofer, 2014). The concept of target costing focuses on influencing costs early during the product design and product development phases (Kato, 1993, Ewert and Ernst, 1999) to generate new ideas for product development and levers for cost reduction (Tani, 1995). In doing so, managers set targets using market information and behavioral control aspects (Hiromoto, 1988).

1.3 Methodologies and results

In **Essay I**, I test my hypotheses in an experiment with 198 participants from the laboratory for experimental research in economics of a leading Western European university. In the experiment, participants decide on the optimal investment amount for a follow-up efficiency improvement project based on balanced scorecard (BSC) data (Dalla Via et al., 2019). I base the BSC measures on Humphreys et al. (2016), including financial, customer, internal business process, and learning and growth project data. The participants need to analyze the BSC data and derive the nonlinear relation between previous investment amounts and financial performance data (Ittner and Larcker, 1998b) while ignoring the irrelevant non-financial information cues.

I manipulate information load using a BSC for three reasons. First, the design of a BSC enables participants to measure decision quality as the dependent variable (Dalla Via et al., 2019). In this setting, I measure decision quality based on the accuracy of participants' investment decision, which is optimal when the amount invested in the project leads to the highest net profit (Dalla Via et al., 2019). Second, a BSC contains financial and non-financial metrics, allowing participants to identify interrelations among the given performance measures (Banker et al., 2000). Introducing accountability as a control measure may influence individual's motivation and sharpens their information-processing capabilities, which leads to better decision making (Dalla Via et al., 2019). Third, a BSC allows a flexible format for firms to present various amounts of information. Scholars contend that the complexity of a BSC leads to information overload, as it illustrates multiple perspectives, including 16 measures (Ding and Beaulieu, 2011).

I employ a 2 x 2 between-subjects design in which I vary information load as low or high (Chewning and Harrell, 1990, Swain and Haka, 2000) and accountability as absent or present (Fehrenbacher et al., 2020). In the low information load condition, participants are presented with four BSC measures, one for each BSC perspective, as four measures display only a few options (Iyengar, 2010). Scholars commonly use 16 measures in a BSC, each perspective containing four measures (Lipe and Salterio, 2000, Ding and Beaulieu, 2011, Humphreys et al., 2016), serving as a proxy for the high information load, i.e., information overload, condition (Ding and Beaulieu, 2011). I manipulate accountability as to whether participants are held accountable for their decision (Fehrenbacher et al., 2020). When held accountable, participants are required to justify their decision by explaining how they arrived at it. When not held accountable, participants' decision is treated confidentially and anonymously (Siegel-Jacobs and Yates, 1996).

I test my hypotheses with an ANOVA analysis. I find a significant main effect of information load on decision quality, supporting *Hypothesis 1* that information overload reduces average decision quality. However, the results reveal no main effect of accountability, and there is no significant interaction of information load and accountability on decision quality, rejecting *Hypothesis 2*. Possible reasons why accountability does not improve the average decision quality can be found in the literature. Potentially, among participants in my experiment, the irrelevant information cues have evoked an effect of dilution that can offset the benefits from accountability (Tetlock and Boettger, 1989, Tetlock et al., 1989, Siegel-Jacobs and Yates, 1996, Bartlett et al., 2014).

Next, decision quality is examined in more detail. In particular, I investigate the frequency of participants making the optimal decision across conditions, i.e., when participants reach a 100% accurate decision. I estimate a logistic regression with a binary variable that indicates whether the participant made the optimal decision. I do not find a main effect of information overload and no significant interaction effect of information load and accountability on decision quality. However, participants facing information overload are significantly more likely to make the optimal decision if they are held accountable than if they are not.

In **Essay II**, I analyze the following research question: *"How do firms estimate software project costs?"* To answer this question, I must identify and analyze practitioners' different estimation approaches. To do so, I conduct a multiple-case study according to Eisenhardt (1989) and Corbin and Strauss (1990). A multiple-case study design displays a suitable research method to address the research question of how firms estimate software costs for many reasons. First, it enables the analysis of contemporary phenomena in a rich, real-world context (Eisenhardt and Graebner, 2007) and relies on multiple data sources to triangulate results. Second, multiple cases enhance the generalizability of the results (Eisenhardt and Graebner, 2007, Yin, 2018) by applying a replication logic (Eisenhardt, 1989) that confirms or disapproves the inference from the past cases.

The essay aims to develop a model that shows how firms estimate software costs depending on their organizational and project-specific setting. To gain diverse perspectives, I follow a diverse sampling strategy. The data sample comprises nine software-producing firms from five industries, including computer software and cloud solutions, financial services, electric/electronics, automotive, and professional services. Besides differences in industry, the variation accounts for different project environments, e.g., varying levels of time constraints or requirements uncertainty and the varying perspectives of contractors and suppliers. The unit of analysis displays

an individual software project from the start of planning to the first release, excluding updates and maintenance (Austin and Devin, 2009).

My main source of analysis displays 14 semi-structured interviews with software executives, project managers, engineers, and other subject matter experts. I stopped conducting interviews after the additional findings had decreased strongly (Eisenhardt, 1989). Besides conducting the interviews, I received and analyzed project documents of two firms, including process flows and project records. The written data displays an additional source of documentary evidence and enables a more straightforward interview process, a better understanding of project content and context, and data triangulation. I followed the coding approaches by Miles et al. (2019) and visualized the emerging structures according to Gioia et al. (2012). I arrived at 1,268 codes (1st-order concepts) that I grouped into 31 2nd-order themes, into ten aggregated dimensions, and into three highest dimensions.

I conduct a cross-case analysis to develop the UPI process model that shows how firms estimate their software costs. I structure the process into three levels: project stages, activities, and activity features. On the highest abstraction level, firms follow three generic project stages to estimate software costs: Understand, Plan, and Implement. On the intermediate abstraction level, the three stages include ten activities. On the lowest abstraction level, firms implement features of these activities. Firms have a homogeneous perspective on following the project stages and activities. However, the activity features are heterogeneous depending on the firm and project-specific setting. I describe the activity features within each stage in detail and outline why firms select different activity features and how they proceed in their development context.

During the stage *Understand*, firms focus on understanding the characteristics of the project environment and the requirements. The project environment analysis follows three criteria: product uncertainty, time dependency, and customer-supplier involvement. Based on these criteria, firms decide on further activities in the *Plan* and *Implement* stages. Afterward, the development team tries to understand the project requirements as comprehensively and granularly as possible. Firms decompose the requirement largely, logically, and hierarchically into functional blocks, epics, features, stories, or tasks depending on the degree of completeness.

In the *Plan* stage, firms select the estimation and data strategy, the estimation method, the project governance, and, if necessary, the contract type between customer and supplier. The

estimation strategy can follow a top-down or bottom-up approach depending on the project environment categorization and requirements analysis results. For example, firms estimate development costs top-down if the product uncertainty is low and analogs are available. In contrast, firms choose a bottom-up strategy for short-term estimations, which occur continuously before sprints within the agile development process. Firms can also combine both strategies by estimating costs top-down during initial planning and bottom-up during ongoing development.

Firms define the data strategy to build a standardized data repository. To increase data-driven estimations, firms set the criteria and outline the process for collecting data throughout the entire development process.

Subsequently, firms select the estimation method in alignment with the project governance. Firms primarily rely on group-based expert estimations, regardless of the project environment. Within group-based expert estimations, they only differentiate the aggregation level of the estimates depending on the product uncertainty and the current development phase. Examples of group-based expert estimations are the Planning Poker method, Delphi method, Dice game, or T-shirt size estimations. For projects where relevant past data is available, firms selectively apply statistical approaches as an add-on to expert estimations.

Firms decide on the project governance by implementing different control mechanisms for the development team. Controls are decisive in reacting to deviations and to the different stakeholders' political motives for over- or underestimation. The development teams focus on informal controls like clan control, which are enforced by the estimation method and are based on finding consent between all team members. Among others, the controlling team is responsible for setting the right incentives, defining the Key Performance Indicators (KPIs), and implementing a governance board with formal controls, e.g., when the target-actual comparison of estimates deviates strongly.

The selection of the contract type between contractor and supplier impacts the estimation process and incorporates benefits and drawbacks for each party. Generally, firms can choose between three contract types: time and material, fixed price, and agile fixed price. The *time-and-material* contract refers to the total development time invested by the supplier and upfront agreed cost rates. Due to its flexibility concerning the project scope, firms often agree on this contract type in early development phases and when the requirements are hard to specify. A *fixed-price* contract comprises a fixed fee for delivering a specified software (Jørgensen et al., 2017). The

agile fixed-price contract displays a hybrid solution with a fixed project size but a variable scope for which partial volumes are agreed on iteratively. This contract type fosters communication between the stakeholders and partially eliminates the disadvantages of the other contract types.

The *Implement* stage is based on the previous two stages and focuses on implementing the decomposed requirements of *Understand*. During the *Implement* stage, firms consider project and team factors when estimating the development costs. The project factors include categorizing the decomposed requirements by requirements and system complexity and novelty. Further, they comprise within-project, cross-project, and cross-firm lessons learned to re-calibrate their estimates. The team factor evaluates the individual team members' capabilities and availability. Finally, firms define rules for implementing the factors, e.g., as a defined number of story points or as an adjustment to the number of iterations within the development process.

Besides setting up the UPI model descriptively, I identify procedural estimation challenges and suggest five propositions that should support managers in designing their estimation process, reduce situational or human biases, and increase the consistency of their estimation results. To derive the propositions, I evaluate the identified estimation challenges and triangulate these with the insights gained during the literature review and with theories in information systems. First, firms should conduct a specification workshop with all stakeholders to increase the clarity of the project requirements, reduce the risk of neglecting tasks, and improve information sharing between stakeholders (Vidgen and Wang, 2009). Second, managers should systematically train estimators based on past projects and scientific literature. Managers can support the learning process and increase productivity by giving estimators checklists to support the estimation process (Jørgensen and Molokken, 2003, Usman et al., 2018). Third, firms should build a firm-specific database to realize data-driven decision making. Systematic data collection would also enable firms to apply machine learning (ML)-based methods. Fourth, estimators should recognize team characteristics and the interests of stakeholders during estimation. Team characteristics include varying capabilities, capacities, locations, cultures, and political motives. Fifth, managers should adapt controls to react to the project teams' need for control and flexibility. I suggest extending the controls by introducing emergent outcome controls (Harris et al., 2009) to agile development settings. This form of control comprises two control mechanisms: scope boundaries and ongoing feedback. Scope boundaries like a product vision, feature specifications, or technical constraints limit the range of possible solutions without specifying outcomes. At the same time, ongoing feedback is decisive when the scope boundaries need to be set tighter.

In **Essay III**, we examine the following research question: *"How can controllers design cost systems for software firms to encompass software's product and process peculiarities?"* Therefore, we analyze software's product and process characteristics to derive requirements for designing cost systems for software firms. We identify differences between industrial goods and information goods in cost occurrence, cost composition, and decision-making relevance. Information goods' costs incur during the development while the production and distribution are related to virtually zero costs (Shapiro and Varian, 1999, Jones and Mendelson, 2011). Contrary, industrial goods' costs arise mainly during production and distribution (Jones and Mendelson, 2011). The cost occurrence highlight differences in value creation between information and industrial goods. While information goods often require creative development effort and flexibility (Harris et al., 2009), industrial goods follow a production process that can be physically observed. Next, we compare the differences with the underlying mechanisms in extant management accounting research on cost system designs for tangible products.

Based on this review, we conceptually develop a cost system design that supports software firms in making informed decisions on software product costing. The cost system takes multiple perspectives on software costs by following the divide-and-conquer principle and integrating three cost-management and modeling systems. We divide the larger problem of product costing into smaller partial problems ("Divide") if a problem cannot be directly solved. Then the partial problems can be solved and combined into an overall solution ("Conquer"). Thus, each cost-management and modeling system can realize its capabilities while establishing synergies across systems. In doing so, we follow the idea of (Kaplan, 1988) that "One cost system Isn't Enough."

First, we recommend that firms take a life-cycle perspective and define the software development project as the cost object to collect, measure, and allocate costs. The life-cycle perspective allows managers to monitor and plan software costs of a complete economic life cycle beyond the core development phase (Bradley and Dawson, 1999, Zarnekow and Brenner, 2005). Considering total software costs is important because the recurring costs for production (e.g., operations, support, and maintenance) (Zarnekow and Brenner, 2005) make up a high share of total software costs (Boehm, 1981, Nguyen, 2010). Zarnekow and Brenner (2005) state that firms do not collect and evaluate data of actual recurring costs because they lack awareness regarding their significance. In this context, we also recommend shifting the focus from the software product to the software project as a cost object independent of the life-cycle phases. The total project costs comprise

project—internal costs, shared reuse costs, firm—internal support costs, and external service provider costs. We suggest how to allocate indirect costs to the specified cost sources.

Second, we propose a regression-based cost model that allows firms to examine the effect of project parameters (independent variables) on project costs (dependent variable). We describe the model as an inherent cost allocation process, i.e., how to allocate the total project costs to virtual cost drivers such as functional size or project duration. The sum of the intercept, the independent variables, and the residual represent the total costs. We structure the model in four steps: (1) Setting up a firm-specific database for project cost data and characteristics. (2) Selecting firm and project-specific parameters as cost drivers for the model. (3) Developing a regression model based on their idiosyncratic data and variables. (4) Defining a firm-specific inherent cost allocation scheme to divide the total project costs into their virtual cost components. In doing so, we apply a linear regression analysis.

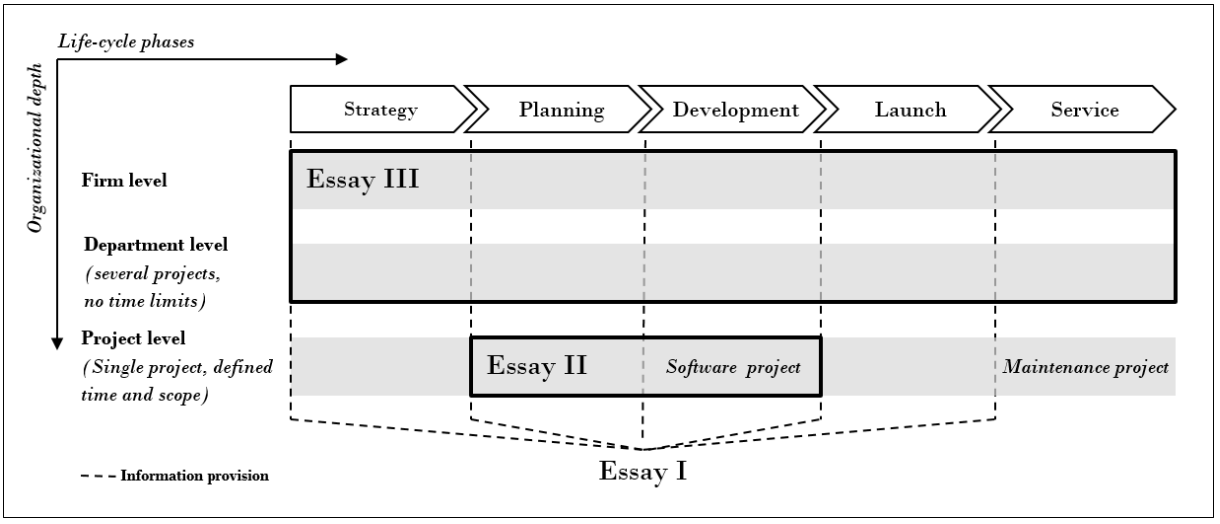
Third, we illustrate a dynamic target costing approach to plan and steer costs during software design and early development stages. We realign the traditional target costing approach to account for the virtual nature of software products and the dynamic requirements. Specifically, we initiate the dynamic target costing process after analyzing the functional and non-functional requirements. We suggest shifting the focus towards the development costs and defining target costs aggregated for multiple versions of one product. In doing so, we aim to dissolve the relatively static definition of functions and associated components to account for flexible software versioning strategies. Additionally, we recommend dissolving the static link between customer preferences and component cost weights because customer preferences cannot be directly related to the importance of functions due to the separate consideration of application and solution domain (Brügge and Dutoit, 2014). Further, we suggest defining target costs not only on a product level but also on a component and function level.

1.4 Contribution to academic and practical debates

The three essays contribute to the literature on management accounting, software engineering, and their intersection. Together, the essays present evidence on decision making and costing from multiple perspectives. In **Essay I**, I analyze how information providers should provide information to firms' decision-makers to mitigate the negative effects of overloading managers with information induced by the digital transformation. In doing so, I take the perspective of

a controller who designs information systems, decision-making frameworks, reports, or financial ratios for managers on different organizational levels, e.g., firm, department, or project. In **Essay II**, I analyze the software cost estimation process on a granular project level, from understanding the project requirements and environment to implementing the actual estimation. Here, I take the perspective of a project controller or project manager who focuses on estimating the costs of their responsible project. In contrast, **Essay III** outlines why firms must adapt their extant cost systems to the requirements of intangible products on a firm or departmental level. Thereby, I take the perspective of a chief controlling officer or head of controlling who is responsible for defining congruent cost-management and modeling systems that allow firms to gain multiple perspectives on product costing and, thus, improve decision making. Figure 1.1 displays a schematic assignment of the three essays to the software life-cycle phases and organizational levels (Ebert, 2007).

FIGURE 1.1: Schematic assignment of the three essays to the software life-cycle phases and organizational levels.



Notes: The life-cycle phases and organizational levels are adapted from Ebert (2007).

Essay I contributes to the information load and accountability literature. First, I complement the literature about the negative effects of information overload on decision quality (e.g., Iselin, 1988, Chewning and Harrell, 1990, Roetzel, 2019). Managers need information to make good decisions, but if overloaded with information, they might make worse decisions. The results support evidence that organizations must consider individuals experiencing information overload when designing decision-making frameworks.

Second, I pick up recent calls for research (Gupta et al., 2018, Brynjolfsson et al., 2021) and investigate accountability as an additional measure to alleviate negative consequences of information overload in the context of the digital transformation (e.g., Eppler and Mengis, 2004, Brown-Liburd et al., 2015, Kelton and Murthy, 2016). Individuals and firms increasingly face this challenge because AI and big data support acquiring and processing of information, but at the same time, enhance the risk of the brain suffering from information overload (Gupta et al., 2018). Previous research found that financial incentives (Ding and Beaulieu, 2011) or decision models and decision aids reduce the adverse effects of information overload (Paul and Nazareth, 2010, Ding and Beaulieu, 2011, Dalla Via et al., 2019). I add that accountable individuals are significantly more likely to make the optimal decision under information overload than non-accountable individuals.

Essay II makes three contributions. First, I add to the literature on software cost estimation by reflecting on the current state of practice (Jørgensen and Shepperd, 2007, Eduardo Carbonera et al., 2020). I strengthen scholars' and practitioners' understanding of the software cost estimation process by enlarging their perspective on software cost estimation beyond describing individual estimation methods and comprising the entire process from understanding the project environment to estimating the costs. I integrate the peculiarities of selected industries, value chain positions, and organizational roles to understand different facets of the estimation process and increase the external validity of my results. Thereby, I add to Boehm and Papaccio (1988) who describe a software planning and control framework, Edwards and Moores (1994) who introduce the EEPS model, and Jørgensen and Molokken (2003) and Usman et al. (2018) who suggest using checklists structured along the software cost estimation process. Firms can compare their current estimation process with the UPI model to adapt or extend their process and increase the consistency of their estimation results. Further, the results allow researchers to recognize gaps between research and practice and base their future research efforts on real-life findings (Eisenhardt and Graebner, 2007).

Second, I show that group-based expert judgment is still the dominant method across industries due to the high degree of implicit expert knowledge and the high effort in setting up a customized formal model. Differences only occur at the aggregation level of the estimates, e.g., in epics, stories, or story points. The results are in line with the literature on effort estimation, which highlights that accurate estimation results depend on the previous experience from a similar task (Jørgensen, 2004b, Haugen, 2006, Idri et al., 2015).

Third, I suggest propositions that support managers in designing their estimation process. The propositions aim at supporting managers in overcoming identified challenges in software cost estimation, reduce situational and human biases, and increase estimation accuracy.

Essay III contributes to the cost system design literature by conceptualizing how firms can systematically model and manage software costs. We add to Otley (1980), who suggests setting up cost-management systems contingent on organizational circumstances, e.g., production technology and organizational structure. Further, we pick up the calls for research by Messner (2016), who recommends scholars to tailor management accounting systems to organization-specific practices, e.g., research and development, and industry-specific characteristics, e.g., regulations.

Further, our study adds to the research on measuring and allocating product and service costs (e.g., Cooper and Kaplan, 1988, Datar and Gupta, 1994, Labro and Vanhoucke, 2007, Balakrishnan et al., 2011, 2012) by suggesting guidelines for designing a cost system for software projects. We contribute to the research gap identified by Astromskis et al. (2014), who observe a need for a framework that defines principles for a cost system design for software firms.

Altogether, this dissertation contributes to the academic literature and practical debates on costing and decision making in the digital age. I take the perspective of an accounting researcher and derive insights at the intersection of management accounting and software engineering. I suggest how controllers should provide decision-relevant information to other managers amidst the rising information volumes induced by big data. I describe how software-producing firms estimate software costs and derive propositions for their estimation process. And finally, I propose how to adapt traditional cost-management practices by discussing guidelines for designing cost systems for software firms to account for the organizational cost mix changes.

Table 1.1 provides an overview of the three essays, including the research questions, methodologies, unit of analyses, findings, and contributions.

1.5 Structure of the dissertation

The remainder of the dissertation is organized along with my three essays. All essays represent separate research projects. Thus, there might be some overlapping content. However, readers of this dissertation can understand the three essays independently of each other.

Chapter 2 contains **Essay I** with the title "Finding the Needle in the Haystack: How Information Load and Accountability Influence Decision Quality". In this essay, I experimentally investigate whether and how accountability affects decision-making quality when managers face different levels of information load. In Chapter 3, I present **Essay II**, "Understand, Plan, and Implement: A Multiple-Case Study on How Firms Estimate Software Costs". This essay provides a process model that shows how firms estimate their software costs. Chapter 4 contains **Essay III** "Divide and Conquer: Designing Cost Systems for Software Firms" and proposes a cost system design for software firms. Finally, Chapter 5 summarizes the key findings of the three essays, discusses theoretical and managerial implications, limitations, and provides overall concluding remarks. The appendix includes additional information about the three essays, such as the case-related instructions, the BSC manipulation, the briefing note for the experiment, and the flow of the experimental design for Essay I, the case study protocol, the interview questions, the overview of the data structure, and the actual data structure of the UPI model for Essay II, and exemplary cost structure differences between industrial and information goods, and the illustration and description of a selected cost-management system for Essay III.

Table 1.1: Overview of the three essays.

Essay characteristics	**Essay I (cf. Chapter 2)**	**Essay II (cf. Chapter 3)**	**Essay III (cf. Chapter 4)**
Research question	Does accountability affect decision-making quality under varying levels of information load?	How do firms estimate software project costs?	How can controllers design cost systems for software-producing firms?
Research approach	Experimental	Qualitative	Conceptual
Methodology	Laboratory experiment	Multiple-case study	Previous research
Unit of analysis	∼200 participants making an investment decision.	Individual software projects of nine case firms.	Extant cost system designs and software characteristics.
Findings	Managers facing information overload are more likely to make the optimal decision if they are held accountable than if they are not. Information overload reduces the average decision quality but does not interact with accountability.	Firms structure their cost estimation process into three stages and ten activities while adapting their actions to changing environments. I identify estimation challenges and suggest propositions for designing the estimation process.	We provide a cost system design for software firms by integrating three cost-management and modeling approaches. We conceptualize how firms can calculate, evaluate, and manage software product costs from multiple perspectives.
Contributions	Firms should consider information overload and accountability when designing their decision-making frameworks.	I provide the UPI process model and enlarge the perspective of cost estimation beyond selecting individual methods.	We define guidelines for software-producing firms for designing their cost system to make informed decisions on product costing.

Notes: This table summarizes the three essays within this dissertation. The table provides an overview of the essays' research questions, methodologies, unit of analyses, findings and contributions.

2 | Finding the Needle in the Haystack: How Information Load and Accountability Influence Decision Quality

Abstract

This study investigates how information load and accountability influence decision-making quality. In today's digital age, the volume of available information increases faster than the attention and processing capabilities, leading to information overload. Drawing on information load literature, I hypothesize that decision quality declines under information overload. I introduce accountability (i.e., justifying one's decision) as a mitigation measure and expect a positive influence on selecting the relevant cues for making the right decision. In detail, I hypothesize that the effect of accountability on the filtering of information increases with rising information load. I test these hypotheses in an experiment in which nearly 200 students decide on the optimal investment amount for an efficiency improvement project. I apply a Balanced Scorecard setting and vary information load at two levels—low and high—and manipulate accountability at two levels—absent and present. The results are partially consistent with my hypotheses. Information overload reduces the average decision quality, yet accountability does not improve the average decision quality, and accountability and information load do not interact. However, accountability increases the likelihood of making the optimal decision under information overload. The different results could indicate a dilution effect that shifts the attention during complex tasks towards irrelevant information. I discuss the implications of my findings for management accounting research and practice.

Author: Marcus Witter

Status: Working Paper[3]

[3]This essay was presented at the 44th Annual Congress of the European Accounting Association (EAA) in Bergen, Norway, the 19th Annual Conference for Management Accounting Research (ACMAR) in Vallendar, Germany, and the 38th Eurasian Business and Economics Society (EBES) Conference in Warsaw, Poland (online). I thank the participants for their valuable comments. Further, I thank Stefanie Baumgartner, a former master's degree student at TUM, who supported me in reviewing the literature. Our discussions greatly benefited the work.

2.1 Introduction

I investigate whether and how accountability affects decision-making quality when managers face an increasing information load. This analysis is important because the growing abundance of information, driven by artificial intelligence and big data supporting organizations and their members in acquiring information, provides opportunities for making better decisions (Brynjolfsson et al., 2021). Yet, the wealth of available information also implies the risk of exceeding their attention and processing capabilities (Gupta et al., 2018). Scholars describe this condition as information overload, and question how organizations should adopt decision-making frameworks, design reports (van Knippenberg et al., 2015), financial ratios or statements (Iselin, 1993), and organize usable information system interfaces to provide information to decision-makers (Schick et al., 1990). The latter is particularly relevant for controllers who are significant information providers to decision-makers across firms and are decision-makers and, thus, users of information themselves (Schick et al., 1990).

In today's digital age, managers can easily acquire information to increase the likelihood of making the right decisions. However, when overloaded with information, they tend to make suboptimal decisions or avoid decision making (Gupta et al., 2018). Individuals are considered as boundedly rational agents who encounter limits in solving complex problems (Simon, 1955), can be distracted from relevant information (van Knippenberg et al., 2015), and make suboptimal decisions or avoid decision making when they face a number of choices between five and nine (Iyengar, 2010). When designing reports for supporting managers making a decision, information providers must consider that three variables affect the decision quality: 1) the uncertainty experienced by the managers, 2) the information load, i.e., the quantity of information cues that are relevant for making a decision, and 3) the data load, i.e., the quantity of cues that are irrelevant for the decision (Iselin, 1993). Scholars differentiate two types of data load. The first type of data load contains redundant and irrelevant cues for the decision at any time. The second type of data load consists of cues that may predict the main criterion but are correlated with another cue that predictions the main criterion equally well or better. We adhere to the categorization of Iselin (1993) and consider the second type as part of information load.

According to the human information processing approach by Schroder et al. (1967), decision quality is optimal at a certain level of information load. As soon as the information load becomes higher or lower, decision quality decreases. Specifically, an increase in the amount of information

requires more of the limited information processing capacity, complicating the identification of relevant information cues by increasing the demand to filter out irrelevant data, leading to errors and deteriorating decision quality (Iselin, 1993). Extant literature on information load deals with mitigating the adverse effects of information overload (e.g., Ding and Beaulieu, 2011, Brown-Liburd et al., 2015, Kelton and Murthy, 2016). On the one hand, Chewning and Harrell (1990) suggest mitigating information overload by equipping decision-makers with a decision model adapted to the situation. On the other hand, Paul and Nazareth (2010) find that incorporating a decision aid into group decision processes enables groups to process a larger amount and more complex information.

I suggest that accountability mitigates the detrimental effects of information overload when no decision aids are available to the manager. Accountability exhibits a pivotal design feature of management control systems (e.g., Ahrens, 1996, Merchant and Otley, 2006, Birnberg et al., 2008, Fehrenbacher et al., 2020) and "refers to the implicit or explicit expectation that one may be called on to justify one's beliefs, feelings, and actions to others" (Lerner and Tetlock, 1999, p. 255). In this study, accountability refers to whether individuals are accountable for explaining how they arrived at their decision (Fehrenbacher et al., 2020). A large body of literature in this area shows that holding managers accountable can significantly affect judgment and decision quality (e.g., Siegel-Jacobs and Yates, 1996, Libby et al., 2004, Chang et al., 2013, Dalla Via et al., 2019, Fehrenbacher et al., 2020). Accountable individuals tend to think more critically and are more likely to make the right decision (Tetlock, 1983, Ahrens, 1996). However, the positive effect does not hold for all individuals, as accountability can trigger stress (Hall et al., 2017) and inherits the risk of focusing not only on relevant but also irrelevant information, evoking a dilution effect that compromises decision quality (Siegel-Jacobs and Yates, 1996, Tetlock and Boettger, 1989). As information overload and accountability potentially affect individuals' decision-making quality, I study their interactive effects.

Building on information load literature and accountability theory, I assume that differences in cognitive processes induced by experimental conditions affect individuals' decision quality. I suspect that a high amount of information necessitates an increase in the information search effort, thus, making it more difficult for individuals to identify decision-relevant information cues. Hence, I predict that information overload leads to lower decision quality. I assume that accountable individuals increase their mental effort to justify their position (Tetlock et al., 1989)

and identify causality between decision-relevant information cues. Thus, I predict that accountability increases decision quality. In particular, I expect that being accountable is advantageous to decision quality under information overload because individuals are motivated to increase their search effort and mitigate the detrimental effects of information overload.

I test my hypotheses with an experiment. I adapt the experimental setting from Dalla Via et al. (2019) and employ an investment decision task in which participants are asked to decide on the optimal amount for a follow-up efficiency project. Participants need to derive the nonlinear relation between previous investment amounts and financial performance data from a Balanced Scorecard (BSC). In order to make the optimal decision, participants need to identify and process the relevant performance measures in the BSC and ignore irrelevant information cues. I employ a 2 x 2 between-subjects design in which I manipulate information load at two levels—low and high—and manipulate accountability at two levels—absent and present.

Manipulating the information load through a BSC is well suited to test my hypotheses for three reasons. First, a BSC setting, by design, enables the measurement of decision quality as the dependent variable (Dalla Via et al., 2019).[4] Second, a BSC provides decision-makers with financial and non-financial performance data, allowing the identification of interrelations among given performance dimensions (Banker et al., 2000). Hence, accountability may influence individuals' motivation and sharpen how they process information, resulting in better decision quality (Dalla Via et al., 2019). Third, a BSC provides a versatile format for organizations to present different amounts of information. Ding and Beaulieu (2011) investigate the effect of different levels of information load in the context of performance evaluations and contend that the BSC represents information overload, as it includes multiple perspectives and up to 16 measures.

My results are partially consistent with my predictions. Information overload reduces the average decision quality. However, the effect of accountability depends on the variable used to capture the decision quality. Accountability does not counteract the detrimental effects of information overload on average decision quality, and information load and accountability do not interact. The missing effect of accountability on decision quality could indicate a dilution effect that comprises decision quality (Tetlock and Boettger, 1989, Siegel-Jacobs and Yates, 1996), shifting individuals' focus towards irrelevant information (Bartlett et al., 2014). Further, being

[4]Following Dalla Via et al. (2019), I measure decision quality by the accuracy of participants' investment decision. Decision quality is highest when the amount invested in the project leads to the best financial performance, in my case, the highest net profit.

accountable can stress individuals (Hall et al., 2006) and reduce the consistency of behavior as soon as individuals do not know how to solve the problem (Siegel-Jacobs and Yates, 1996). However, the frequency of making the optimal decision is significantly higher for accountable individuals than non-accountable individuals under information overload.

My study contributes to two streams of literature. First, they complement the literature about the effects of information overload (e.g., Iselin, 1988, Chewning and Harrell, 1990, Roetzel, 2019). Consistent with previous results, a high amount of information reduces decision quality. The results support existing evidence that information overload is a decisive factor for designing decision-making frameworks.

Second, I contribute to the literature that studies measures to alleviate the negative consequences of information overload (e.g., Eppler and Mengis, 2004, Brown-Liburd et al., 2015, Kelton and Murthy, 2016), picking up recent calls for research (Gupta et al., 2018, Brynjolfsson et al., 2021). The issue gains increasing importance because technological advancements increase the available information for most decision-makers. Previous research in information systems and accounting identified financial incentives (Ding and Beaulieu, 2011) or decision models and decision aids to reduce the negative effects of information load (Paul and Nazareth, 2010, Ding and Beaulieu, 2011, Dalla Via et al., 2019). I add that accountability can increase the probability that managers who are overloaded with information will still make the optimal decision. This finding also enriches the management accounting literature that investigates other effects of accountability on decision making (e.g., Iselin, 1988, Chewning and Harrell, 1990, Roetzel, 2019). Assuming that accountability as a control mechanism is in place in most, if not all, organizations, the linkage of information overload and control mechanisms in organizations displays a valuable setting to understand the role that accountability plays in improving decision making under information overload.

The remainder of this study is organized as follows. Chapter 2.2 elaborates on the theoretical background of my hypotheses, Chapter 2.3 outlines the experimental design, Chapter 2.4 shows my results, and, finally, Chapter 2.5 concludes.

2.2 Theory and hypotheses development

2.2.1 The effect of information load

Information overload describes a condition in which managers or organizations cannot adequately process information, i.e., the processing requirements exceed the processing capacity (Schneider, 1987). On the one hand, from a cognitive point of view, managers are boundedly rational agents who experience limits in processing information, then stop acquiring information and make a decision based on the limited information they have (Simon, 1955). Also, other personal factors such as managers' motivation (Muller, 1984) can influence the decision-making process. On the other hand, the characteristics of a task can increase information processing requirements, e.g., when individuals face noticeably complex tasks (Tushman and Nadler, 1978), time pressure (Schick et al., 1990), or budget restrictions (Roetzel, 2019). Further, information complexity (Schneider, 1987) and the presence of irrelevant information cues for making an optimal decision increase the information processing requirements (Iselin, 1993, Hartmann and Weißenberger, 2020). Recent studies indicate that technological advancements reinforce the increase in information processing requirements, potentially leading to the brain suffering from information overload (van Knippenberg et al., 2015, Gupta et al., 2018). Individuals and organizations acquire information readily with low costs (Levitin, 2014), while information systems can process and store an increasing amount of data (Roetzel, 2019).

The manifold causes of information overload lead to a variety of symptoms. First, individuals stop searching for additional information necessary for making a decision (Simon, 1955, Schroder et al., 1967, Gupta et al., 2018) and tend to disregard information cues before all information is used (Roetzel, 2019). Second, the existence of irrelevant information creates overconfidence in the decision accuracy of individuals and dilutes judgment (Fleisig, 2011). Third, information overload triggers the loss of priorities and goal orientation (Schneider, 1987), and a general lack of perspective, leading to stress (Schick et al., 1990). Fourth, information overload increases mental discomfort and raises emotions when making decisions (Swar et al., 2017), leading to irrational behavior (Schneider, 1987).

Scholars find consensus that the symptoms of information overload harm the performance of an individual (Eppler and Mengis, 2004). Focusing on the amount of information provided for making decisions, an individual's decision-making performance correlates positively with the

amount of information up to a certain point in which information processing requirements equal information processing capacity. Beyond this tipping point, decision quality declines as individuals receive more information than they can process, resulting in an inverted u-shaped function (Schroder et al., 1967, Driver and Streufert, 1969). Drawing on information load literature, I hypothesize that information overload leads to lower decision quality.

H1: *Information overload reduces decision quality.*

2.2.2 The effect of accountability

Management accounting scholars consider accountability as a pivotal design feature of management control systems (e.g., Ahrens, 1996, Merchant and Otley, 2006, Birnberg et al., 2008, Fehrenbacher et al., 2020). Further, research in psychology considers accountability as a relevant factor for influencing decision-makers (Lerner and Tetlock, 1999, Chang et al., 2013). Lerner and Tetlock (1999) argue that being accountable evokes three cognitive processes, i.e., conformity, preemptive self-criticism, and defensive bolstering. First, accountable individuals are confronted with social anxiety and seek conforming behavior. Second, accountability enforces self-critical and inclusive behavior to take different points of view and avoid confrontations. Third, accountable individuals turn to defensive bolstering, i.e., to seek as many reasons as possible to prove critics wrong. The involvement of such cognitive processes sharpens the actions of individuals by itself and does not depend on the presence of an auditor (Roberts, 1991). Being accountable stimulates critical thinking and the anticipation of objections in one's argumentation (Tetlock, 1983, Ahrens, 1996), enhancing the likelihood of making right judgments (Tetlock, 1983) and providing more rational reasons for a decision (Moser et al., 2013), ultimately improving decision quality (Siegel-Jacobs and Yates, 1996, Libby et al., 2004, Jermias, 2006, Dalla Via et al., 2019, Fehrenbacher et al., 2020). Accountable individuals increase their mental effort to justify their position (Tetlock et al., 1989), acknowledge more value tradeoffs, and are more tolerant in evaluating the advantages and disadvantages of a decision to an unfamiliar audience (Tetlock, 1983).

Being accountable can be implemented in various forms. In general, management accounting literature distinguishes two essential types of accountability, e.g., outcome accountability and process accountability (e.g., Siegel-Jacobs and Yates, 1996, Libby et al., 2004, Chang et al., 2013,

Moser et al., 2013, Patil et al., 2014, Dalla Via et al., 2019, Schulz-Hardt et al., 2021). Outcome-accountable individuals are solely responsible for the outcomes of their decision or judgment, regardless of the underlying cognitive process. In contrast, process-accountable individuals are responsible for the degree to which they can explain and justify their decision (Lerner and Tetlock, 1999). In general, scholars consider process accountability to motivate information search efforts (Dalla Via et al., 2019, Siegel-Jacobs and Yates, 1996), reduce the variability in judgment (Siegel-Jacobs and Yates, 1996), increase the complexity of thinking, and have a more powerful effect on decision quality (Siegel-Jacobs and Yates, 1996, Libby et al., 2004, Chang et al., 2013, Patil et al., 2014, Dalla Via et al., 2019). In my research, I refer to Fehrenbacher et al. (2020) and summarize accountability to whether participants are held accountable for "how they arrived at their decision," potentially capturing both the decision process and the decision outcome.

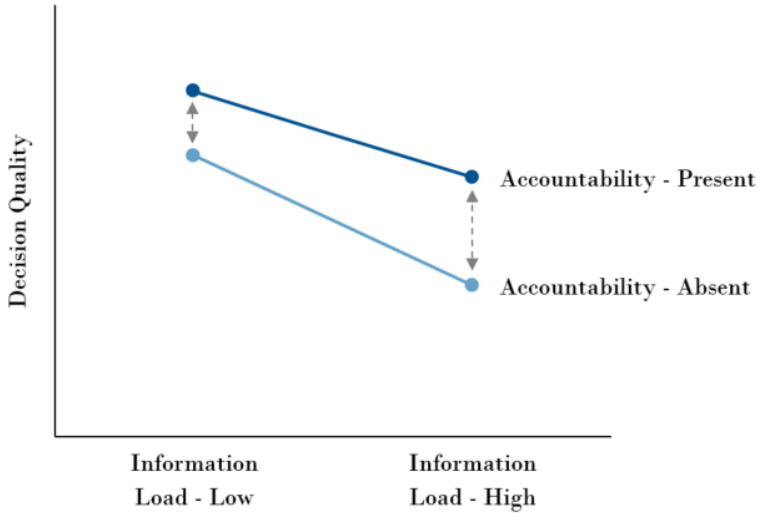
Various implications of accountability have been investigated in the past two decades. Fehrenbacher et al. (2020) study the effect of accountability on positive and negative affective reactions in capital budgeting decisions. They find that accountability reduces the tendency to select an economically non-preferred project which is suggested by a manager triggering a positive affective reaction. Fehrenbacher et al. (2020) argue that accountable individuals are more inclined to pre-emptive self-criticism and, hence, are more likely to anticipate objections, leading to an increase in decision quality. In contrast, accountability does not reduce the tendency to disregard an economically preferred project, which is suggested by a manager triggering negative reactions. Libby et al. (2004) apply process accountability to a BSC setting to increase the cognitive effort involved in performance evaluations and, thus, ensure the use of not only common, but also unique BSC measures. The study illustrates that accountability makes participants more self-critical, reducing the overweighting of general decision-making measures.

Drawing on accountability theory, I expect that the advantages of accountability outweigh the disadvantages. I conjecture that accountability reduces the adverse effects of information overload by motivating individuals to acquire and process information more diligently and infer interrelations among decision-relevant information cues better, thus, improving decision quality. Under low information load, the effect of accountability depends on the complexity of the given information and task and, lastly, how successful individuals are in ascertaining the interrelations without being accountable. If the available information and task to be solved are straightforward to establish causality, there will be no effect of accountability on decision quality. However,

being accountable can still enable individuals to increase decision quality— yet not as much as under information overload. Therefore, I expect to find an increased difference in decision quality between accountable and non-accountable individuals under information overload. Figure 2.1 illustrates my formally stated hypotheses 1 and 2 graphically.

H2: *Increasing information load raises the positive impact of accountability on decision quality.*

FIGURE 2.1: Predicted effects.



Notes: This figure shows the predicted patterns for decision quality across the information load conditions (Information Load—Low and Information Load—High) and accountability type conditions (Accountability—Absent and Accountability—Present).

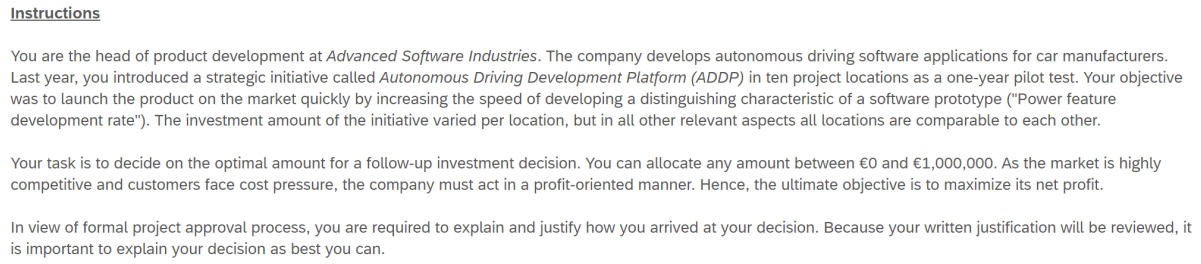
2.3 Experimental method

2.3.1 Experimental setting

I adjust the experimental setting from Dalla Via et al. (2019) and Humphreys et al. (2016) and ask participants to assume the role of the head of product development of a software company for which they receive BSC performance data. The company introduces a development efficiency initiative called "Autonomous Driving Development Platform" in ten development locations as a one-year pilot test. The objective was to quickly launch future product generations on the market by increasing the speed of developing a distinguishing characteristic of a software

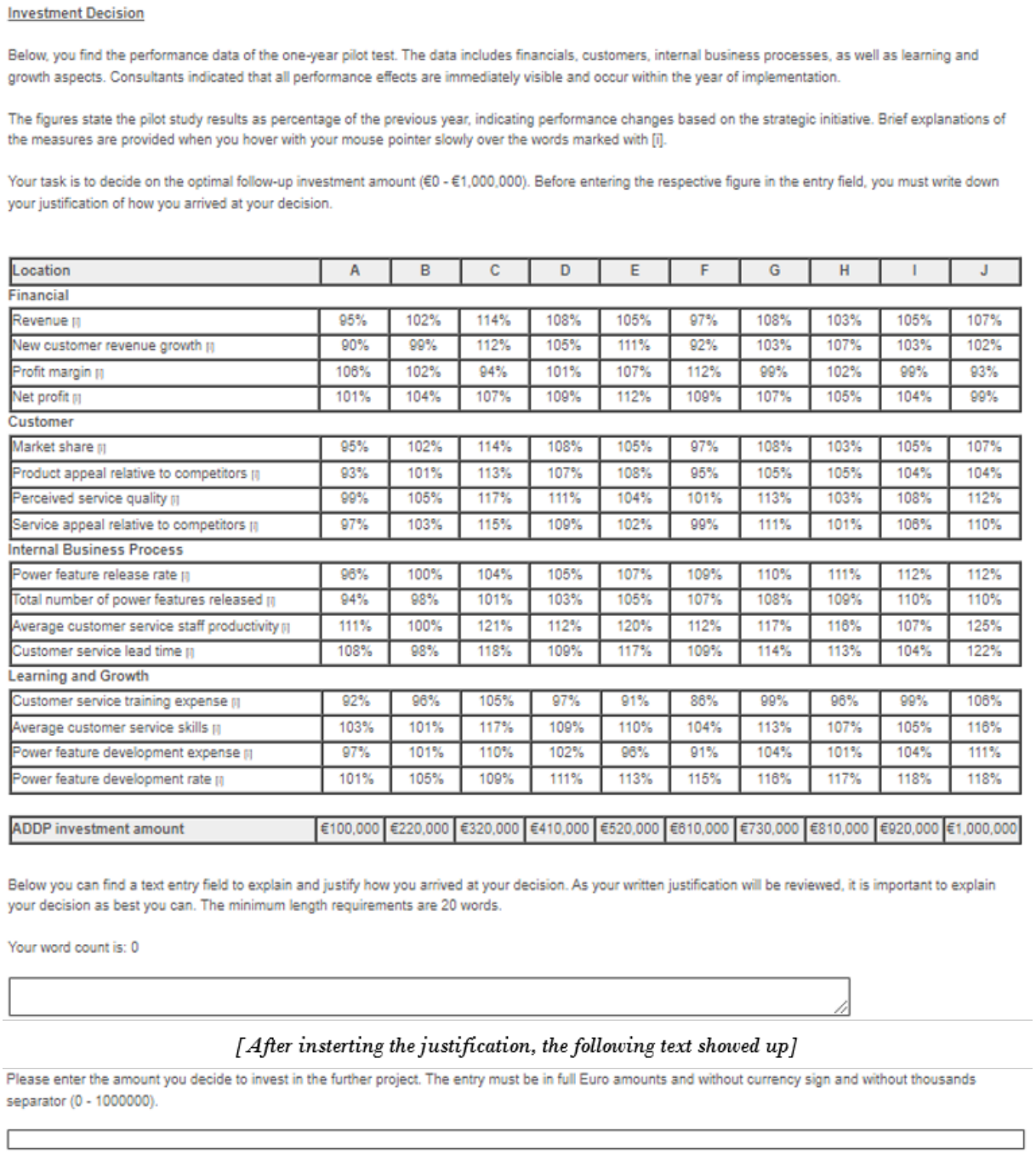
prototype ("Power feature development rate"). The participants learn that the market is highly competitive and customers face cost pressure, and, hence, that the company's ultimate objective is to maximize its net profit (see Figure 2.2).

FIGURE 2.2: Instructions in the Accountability—Present conditions.



Participants received the pilot test data as percentage results of the previous year, indicating performance changes based on the development efficiency initiative, and were asked to make a follow-up investment decision. To measure the effectiveness of the development efficiency initiative, the investment amount varied between 100,000 EUR to 1,000,000 EUR across the different development locations. In all other relevant aspects, all locations are comparable to each other. Following Dalla Via et al. (2019), the power feature development rate data increases linearly with the investment amount data. In contrast, the relationship between changes in financial performance data, i.e., the net profit, and the investment amount data is nonlinear. This reflects that leading financial performance indicators frequently change with individual decisions in a nonlinear manner (Ittner and Larcker, 1998b). To make the optimal decision, participants were required to ignore potential inclinations to maximize the power feature development rate by investing maximum amounts. Instead, participants had to solely focus on maximizing the net profit, for which the optimal amount was 520,000 EUR (see Figure 2.3). Figure A.1 in the appendix shows the case-related instructions in the different accountability conditions, and Figure A.2 in the appendix displays the different BSC conditions.

FIGURE 2.3: BSC data in the Information Load—High and Accountability—Present condition.



Investment Decision

Below, you find the performance data of the one-year pilot test. The data includes financials, customers, internal business processes, as well as learning and growth aspects. Consultants indicated that all performance effects are immediately visible and occur within the year of implementation.

The figures state the pilot study results as percentage of the previous year, indicating performance changes based on the strategic initiative. Brief explanations of the measures are provided when you hover with your mouse pointer slowly over the words marked with [i].

Your task is to decide on the optimal follow-up investment amount (€0 - €1,000,000). Before entering the respective figure in the entry field, you must write down your justification of how you arrived at your decision.

Location	A	B	C	D	E	F	G	H	I	J
Financial										
Revenue [i]	95%	102%	114%	108%	105%	97%	108%	103%	105%	107%
New customer revenue growth [i]	90%	99%	112%	105%	111%	92%	103%	107%	103%	102%
Profit margin [i]	106%	102%	94%	101%	107%	112%	99%	102%	99%	93%
Net profit [i]	101%	104%	107%	109%	112%	109%	107%	105%	104%	99%
Customer										
Market share [i]	95%	102%	114%	108%	105%	97%	108%	103%	105%	107%
Product appeal relative to competitors [i]	93%	101%	113%	107%	108%	95%	105%	105%	104%	104%
Perceived service quality [i]	99%	105%	117%	111%	104%	101%	113%	103%	108%	112%
Service appeal relative to competitors [i]	97%	103%	115%	109%	102%	99%	111%	101%	106%	110%
Internal Business Process										
Power feature release rate [i]	98%	100%	104%	105%	107%	109%	110%	111%	112%	112%
Total number of power features released [i]	94%	98%	101%	103%	105%	107%	108%	109%	110%	110%
Average customer service staff productivity [i]	111%	100%	121%	112%	120%	112%	117%	116%	107%	125%
Customer service lead time [i]	108%	98%	118%	109%	117%	109%	114%	113%	104%	122%
Learning and Growth										
Customer service training expense [i]	92%	96%	105%	97%	91%	86%	99%	96%	99%	106%
Average customer service skills [i]	103%	101%	117%	109%	110%	104%	113%	107%	105%	116%
Power feature development expense [i]	97%	101%	110%	102%	96%	91%	104%	101%	104%	111%
Power feature development rate [i]	101%	105%	109%	111%	113%	115%	116%	117%	118%	118%

ADDP investment amount	€100,000	€220,000	€320,000	€410,000	€520,000	€610,000	€730,000	€810,000	€920,000	€1,000,000

Below you can find a text entry field to explain and justify how you arrived at your decision. As your written justification will be reviewed, it is important to explain your decision as best you can. The minimum length requirements are 20 words.

Your word count is: 0

[After insterting the justification, the following text showed up]

Please enter the amount you decide to invest in the further project. The entry must be in full Euro amounts and without currency sign and without thousands separator (0 - 1000000).

Notes: Participants in this treatment group saw a brief repetition of the task instructions and the BSC data. In addition to the BSC data, explanations of the measures were provided when participants hovered with their mouse pointer over the respective measures.

I choose a BSC to manipulate the information load due to three reasons. First, following Dalla Via et al. (2019), a BSC setting enables the measurement of decision quality, which,

in my case, is the highest for the investment amount, which leads to the highest net profit. Second, a BSC supports strategic planning and management decisions (Kaplan and Norton, 1996a,b, 2001a,b), yet it is also applied in accounting research for control reasons such as performance evaluations (Ittner and Larcker, 1998a, Lipe and Salterio, 2000, 2002, Banker et al., 2004). A BSC provides decision-makers with financial and non-financial performance data, enabling the identification of causal effects among performance dimensions (Banker et al., 2000). Thus, being accountable may influence participants' motivation and sharpen the way they process information, resulting in better decision quality (Dalla Via et al., 2019). Third, a BSC allows organizations to present different amounts of information. Ding and Beaulieu (2011) investigate the levels of information load in the context of performance evaluations, e.g., using two, eight, and 16 BSC measures, and contend that a BSC with 16 measures represents information overload itself.

Further, I choose an investment decision setting for information systems projects for three reasons. First, scholars call for experimental research in the context of information since technological advancements like AI or big data can lead the brain to suffer from information overload (Gupta et al., 2018). I base the BSC measures on Humphreys et al. (2016), whereas the data includes financials, customers, internal business processes, as well as learning and growth aspects of software projects. Second, uncertainty originates from software's intangible nature, incorporating less transparently observable tasks and input materials. Third, investment decisions by definition involve uncertain outcomes that are important to a firm's long-term survival and about which complete information is not available (Maritan, 2001). Thus, an investment decision for information systems projects is of particular practical importance as it displays an extreme example, and the results can be transferred to tangible products of various industries.

2.3.2 Experimental conditions and variables

I employ a 2x2 between-subject experiment in which I manipulate information load and accountability. I manipulate information load as high or low (Chewning and Harrell, 1990, Swain and Haka, 2000) by presenting participants a BSC with either 16 or four measures. Most recent literature commonly illustrates the BSC with 16 measures in four perspectives (Lipe and Salterio, 2000, Ding and Beaulieu, 2011, Humphreys et al., 2016), which represents an approximation for the high information load condition (Ding and Beaulieu, 2011). I choose four BSC measures for

the low information load condition, one per BSC perspective, as four measures do not exceed participants' limits in processing information (Iyengar, 2010).

I manipulate whether or not participants are held accountable for their decision (Fehrenbacher et al., 2020). Researchers operationalize accountability by requiring decision-makers to justify their decision-making process or outcome before making a final decision (Libby et al., 2004). When held accountable, participants must justify their decision and explain how they arrived at it. When not held accountable, participants are reminded that their decision is confidential and anonymous (Siegel-Jacobs and Yates, 1996).

Following Dalla Via et al. (2019), decision quality, the first dependent variable for my analyses, is computed as follows:

$$\text{Decision Quality} = \frac{P^* - |P_a - P^*|}{5,200} \tag{2.1}$$

Decision quality measures the investment decision quality. The numerator captures how far the investment amount that the participant allocated to the project P_a deviates from the optimal amount of 520,000 (P^*). The denominator scales the dependent variable to equal 100 when the optimal amount is invested to the project. By considering the absolute value of the deviation, the formula guarantees that identical deviations relate to equal values, regardless of whether they are above or below the optimum. For example, both 220,000 and 820,000 deviate 300,000 from the optimum (of 520,000) and yield the same value (of 42) for decision quality. The absolute value of the deviation is subtracted from the optimal value to guarantee that higher values relate to higher decision quality, i.e., 100 corresponds to 100% decision quality.

Second, to explore decision quality in more detail, I additionally examine how many participants invested the optimal amount of 520,000 EUR and, thus, reached a decision quality of 100%.

2.4 Results

2.4.1 Participants and procedure

In total, 198 participants from the laboratory for experimental research in economics of a leading Western European university participated in my experiment. Students display suitable

subjects to address theories of cognitive and decision-making processes (Libby et al., 2002). The participants are part of the university's experimental subject pool and self-registered for the experiment online. I implemented the experiment using Qualtrics. I compensated participants with a fixed fee of 7.00 EUR. Following Dalla Via et al. (2019), I chose a fixed fee compensation for two reasons. First, I want to avoid confounding the accountability manipulation with a performance-based payment. Second, I would have had to compensate process-accountable participants based on their written justification, but non-accountable participants based on their decision accuracy, which would have been difficult to implement.

I randomly assigned the participants in every experimental session to one of the four treatment conditions. At the beginning of every session, all participants attended an introductory virtual meeting in which I read the general instructions of the experiment aloud to the participants (see Figure A.3 in the appendix for the exact wording of the briefing note). After concluding the briefing, I shared the link and password for conducting the experiment. Participants read two introductory pages of general instructions and compensation information on-screen. Then they learned about the company's background, the task to be performed, and the managerial role to be assumed. To ensure that participants understood their task and introduced accountability manipulation, participants had to pass a brief comprehension and manipulation check of which amount they were allowed to allocate to the project and whether they were held accountable. Afterward, participants received the description of the business case and its performance data in the form of a BSC. As a next step, participants in the accountability condition were asked to justify their decision before entering the amount they decided to invest. In the end, participants were asked to complete a post-experimental questionnaire and insert their payment details. Figure A.4 in the appendix illustrates the sequence of events.

Taking insufficient effort responding into consideration (Huang et al., 2012), I excluded 9 participants from the analysis, leading to a usable sample size of 189.[5]

Participants took, on average, 11.1 minutes to complete the experiment. 49.2% of the participants were male, 47.1% female, 1.6% non-binary, and 2.1% preferred not to mention their gender.

[5]Qualtrics predicts a total processing time of in average 11.9 min for non-accountable participants and 12.5 min for accountable participants. I assume a shortened response time as an indicator of insufficient effort responding, describing responses which are provided by unmotivated students who conduct the survey too fast to cognitively process the provided information in an adequate manner (Huang et al., 2012). I perceive it unlikely for participants to complete the survey with reasonable cognitive effort in less than 3.5 minutes, which displays the mark for the 5% percentile. I suggest the 5% percentile as a sufficient yet conservative cutoff value, as it remains below one-third of the predicted processing time estimated by Qualtrics. In total, 9 participants completed the experiment in less than 3.5 minutes.

31.2% of the participants studied management, 30.1% engineering, 12.2% economics, and 26.5% other subjects. On average, participants were 24.6 years old, with 51.9% being in a Master's, 39.2% in a Bachelor's, and 8.9% in another program. Further, 74.1% have up to three years of work experience and 25.9% more than three years of work experience.

2.4.2 Test of hypotheses

Table 2.1 summarizes the decision quality, as defined in 2.3, by experimental condition. On average, decision quality under low information load (mean = 72.01) is higher than under high information load (mean = 63.46). On average across information load conditions, the mean decision quality is similar when participants are accountable and when they are not (67.74 vs. 67.69).

TABLE 2.1: Descriptive statistics for decision quality for each experimental condition.

	Accountability		Overall
Information Load	Accountability Absent	Accountability Present	
Low Information Load	73.76 (28.81) N=45	70.41 (34.88) N=49	72.01 (31.99) N=94
High Information Load	61.62 (35.35) N=45	65.12 (35.06) N=50	63.46 (35.05) N=95
Overall	67.69 (32.64) N=90	67.74 (34.89) N=99	67.71 (33.75) N=189

Notes: Descriptive statistics include: mean (standard deviation) and number of observations.

To test my hypotheses, I perform an ANOVA analysis. Table 2.2 presents the results. They reveal a significant main effect of information load on decision quality (F = 3.15, p = 0.0775), supporting Hypothesis 1.[6] I checked for the information load manipulation in the post-experimental questionnaire. Participants indicated their level of agreement on a seven-point Likert scale, ranging from (1) strongly disagree to (7) strongly agree. The information load manipulation was verified with the following item: "I perceived the amount of information of the pilot test as very high". Following the human processing approach, the perceived amount of information should be higher in the high information load conditions than in the low information load conditions. Consistent with my manipulation, the mean score on this question was significantly higher in the high information load conditions than in the low information load conditions (5.20 versus 4.21; t = -4.380, $p < 0.001$). Hence, I conclude that participants perceived, according to

[6]All reported p-values in this study are two-tailed.

the treatment group, a different amount of information, which allows testing for the differential effects of information load.

The results, however, reveal no main effect of accountability (F = 0.00, p = 0.9887). Further, against Hypothesis 2, there is no significant interaction effect of information overload and accountability on decision quality (F = 0.49, p = 0.4866).

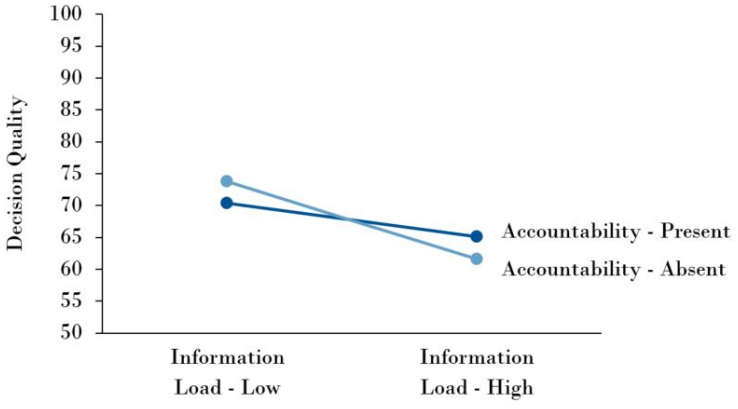
Table 2.2: Factorial ANOVA (dependent variable = decision quality).

Source	Sum of Squares	dF	Mean Square	F	p-value
Accountability - Present	0.23	1	0.28	0.00	0.989
Information Load - High	3,580.19	1	3,580.19	3.15	0.078
Accountability x Information Load	552.02	1	552.02	0.49	0.487
Error	210,103.09	185	1,135.69		
Total	214,110.70	188	1,138.89		

Notes: All reported p-values are two-tailed.

Figure 2.4 provides a graphical overview of the effects on decision quality. Comparing the experimental conditions reveals that the decision quality among participants with low information load is higher if they are not held accountable (mean = 73.76) than if they are held accountable (mean = 70.41). Yet, the decision quality among participants with information overload is higher if they are held accountable (mean = 65.12) in comparison to if they are not (mean = 61.62).

Figure 2.4: Results (dependent variable = decision quality).



Notes: This figure shows the average decision quality across the information load conditions (Information Load—Low and Information Load—High) and accountability type conditions (Accountability—Absent and Accountability—Present). Decision quality measures the quality of the investment decision, with higher scores referring to better decisions (i.e., a score of 100 corresponds to a decision quality of 100%).

Table 2.3 displays the corresponding simple effects. When accountability is absent, participants' decision quality is significantly higher under low information load than high information load (effect size = +0.376, t = 1.785, p = 0.078). However, when accountability is present, no difference occurs across accountability types (effect size = +0.151, t = 0.753, p = 0.453). Further, results indicate that accountability increases decision quality neither in the low information load condition (effect size = +0.104, t = 0.506, p = 0.614), nor in the high information load condition (effect size = -0.099, t = -0.483, p = 0.630).

Potentially, among participants in my experiment, accountability does not improve decision quality because the negative effects of dilution might offset benefits from accountability. Although the benefits of process accountability prevail in various accounting studies (Glover, 1997, Hoffman and Patton, 1997, Libby et al., 2004), studies comparing process accountability to non-accountability identify the dilution effect as dominant (Tetlock et al., 1989, Siegel-Jacobs and Yates, 1996, Bartlett et al., 2014). My experimental setting, including a BSC with 16 measures of ten project locations, can exceed participants' cognitive complexity, increasing the likelihood of dilution.

Table 2.3: Simple effects (dependent variable = decision quality).

	Effect Size	t-stat (p-value)
Low vs. High Information Load in Accountability Present condition	+0.151	0.753 (0.453)
Low vs. High Information Load in Accountability Absent condition	+0.376	1.785 (0.078)
Accountability Absent vs. Present in Low Information Load condition	+0.104	0.506 (0.614)
Accountability Absent vs. Present in High Information Load condition	-0.099	-0.483 (0.630)

Notes: Effect sizes refer to Cohen's d.

Next, I examine the likelihood of making the optimal decision across conditions, i.e., reaching a 100% decision quality.[7] In total 59 participants (31.2%) invested the optimal amount, of whom 18 were in both conditions in which participants are held accountable. The frequency of investing the optimal amount among participants who are not accountable was 14 in the low information load condition and 9 in the high information load condition. Comparing the frequency patterns of the different conditions shows that the results do not reflect the continuous variable decision quality results. In particular, the results show that accountable participants chose the optimal

[7] I performed an exploratory analysis of the written justifications of the participants in the accountability conditions. In these conditions, 36 participants chose the optimal amount of 520,000 EUR. I identified that 33 participants have consciously chosen the optimal investment amount by, for instance, describing the net profit maximization as the ultimate objective (e.g., "as the main aim of the company is to increase profit", "biggest percentage increase on the net profit").

decision more often (36 accountable participants and 23 non-accountable participants). Table 2.4 summarizes the optimal decision by experimental condition.

Table 2.4: Descriptive statistics for perfect decision quality for each experimental condition.

	Accountability		Overall
Information Load	Accountability Absent	Accountability Present	
Low Information Load	14 (31.1%) N=45	18 (36.7%) N=49	32 (34-0%) N=94
High Information Load	9 (20.0%) N=45	18 (36.0%) N=50	27 (28.4%) N=95
Overall	23 (25.6%) N=90	36 (36.4%) N=99	59 (31.2%) N=189

Notes: Descriptive statistics include: frequency (percentage) and number of observations.

To test this difference for statistical significance, I estimate a logistic regression with a binary variable that indicates whether the participant made the optimal decision. The results, however, reveal no main effect of information overload (z = -1.20, p = 0.230) and no significant interaction effect of information overload and accountability on decision quality (z = 0.87, p = 0.386). Table 2.5 reports the results.

Table 2.5: Logistic regression (dependent variable = the optimal decision).

Source	Coefficient	Std. error	z	p-value
Accountability Present	0.251	0.438	0.57	0.566
High Information Load	-0.591	0.493	-1.20	0.230
Accountability - Present x High Information Load	0.560	0.650	0.87	0.386
Cons	-0.795	0.322		

Notes: All reported p-values are two-tailed.

When analyzing the simple effects, participants with high information load are significantly more likely to make the optimal decision if they are held accountable than if they are not (z = 1.71, p = 0.088). However, accountability does not improve the optimal decision under low information load (z = 0.57, p = 0.566). This result does not mimic the result for the continuous variable decision quality, for which I did not find a significant effect of accountability on decision quality. Table 2.6 shows the corresponding simple effects.

Table 2.6: Simple effects (dependent variable = the optimal decision).

	Coefficient	Std. err.	z	p-value
Accountability Absent vs. Present (High Information Load)	0.811	0.475	1.71	0.088
Low. vs High Information Load (Accountability Present)	-0.032	0.418	-0.08	0.939

Notes: All reported p-values are two-tailed.

2.5 Discussion and conclusion

In this study, I examine how information load and accountability affect decision making. The results contribute to various literature streams and contain implications for business practice. Consistent with findings from information load literature (e.g., Iselin, 1988, Chewning and Harrell, 1990, Roetzel, 2019), I observe that information overload impairs decision quality. I observe that this happens even if the information is presented in a BSC, which is considered as a lucid tool to present high amount of information in several dimensions.

The results imply that organizations must tailor decision-making frameworks to individuals' requirements and acknowledge that a high amount of available information does not necessarily entail better decisions. Chewning and Harrell (1990) suggest two approaches to prevent information overload and increase decision quality. On the one hand, organizations must recognize individuals' limits in solving complex problems and reduce information cues to a considerable amount while only considering decision-relevant information. However, this requires that organizations deal with the decision preventively and consider previous lessons learned when designing decision-making frameworks for individual business situations. On the other hand, organizations may equip individuals with decision models or causal chains, which reduce cognitive complexity and direct individuals' focus to relevant information cues for decision making (Humphreys et al., 2016, Dalla Via et al., 2019), creating a competitive advantage for organizations that use and analyze information effectively and efficiently.

I extend this stream of literature by studying whether accountability mitigates the adverse effects of information overload in the absence of decision models or decision aids. I find that, under information overload, accountable participants are more likely to make the optimal decision. However, while the average decision quality is higher when participants under information overload are held accountable than when they are not, the difference is not statistically significant.

The findings complement the management accounting literature that studies effects of process accountability on decision making (e.g., Iselin, 1988, Chewning and Harrell, 1990, Roetzel, 2019).

The missing effect of accountability on the average decision quality could indicate a dominant dilution effect in my setting, increasing individuals' attention to irrelevant information and, thus, comprising decision quality (Tetlock and Boettger, 1989, Siegel-Jacobs and Yates, 1996). Bartlett et al. (2014) identify that process accountability implies additional cognitive information processing, which prompts a consideration of not only relevant but also irrelevant information, triggering a dilution effect and leading to an increased risk of judgment bias. Further, being accountable can increase the stress level of individuals (Hall et al., 2006). Depending on the stress level caused, accountability can lead to differentiated decision quality. While a medium level of stress may lead to a state of excitement and increased interest in the task to be accomplished, a high amount of stress causes tension and emotional exhaustion, which inversely leads to a high stress level, thus, decreasing decision quality (Hall et al., 2017). Also, the effect of accountability depends on the situation and its implementation, as being accountable only motivates individuals to a limited extent. As soon as individuals do not know how to improve the decision, there is a risk of reducing the overall consistency of behavior and disrupting the performance (Siegel-Jacobs and Yates, 1996). Further, Lerner and Tetlock (1999) highlight that the implications of accountability depend on the timing of its manipulation. Pre-decision accountability leads to preemptive self-criticism and individuals carefully preparing their decision, whereas post-decision accountability can trigger defensive bolstering and individuals processing information in a biased manner.

My results are subject to the limitations associated with experimental studies. First, I cannot exclude the possibility that my results do not generalize to other information load settings. I manipulate information overload by the use of a BSC with 16 measures according to Ding and Beaulieu (2011) which represents an abstraction of the technological progress fostered, for instance, by AI and big data. Second, it is unclear whether findings from how I implement accountability in the experiments generalize to other settings as I apply only a single decision rule (Dalla Via et al., 2019). Researchers could analyze how accountability influences decision quality under multiple decision rules in the next step. Further, being accountable in a BSC setting triggers the recognition of unique measures in performance evaluations (Libby et al., 2004). In my experimental task, participants had to focus solely on maximizing the net profit,

whereas Libby et al. (2004) argue that accountable individuals tend to consider not only financial measures for their decision, but also measures from other perspectives.

My study reveals further interesting avenues for research. On the one hand, it seems valuable to investigate additional control mechanisms as a basis for decision-making frameworks to mitigate the adverse effects of information overload on decision quality. On the other hand, future research could examine if digital working environments and the accompanying increasing amount of virtual meetings induce information overload and influence decision making.

3 | Understand, Plan, and Implement: A Multiple-Case Study on How Firms Estimate Software Costs

Abstract

This paper provides a process model that shows how firms estimate their software costs. I find that firms follow three stages and develop the UPI model—*Understand*, *Plan*, and *Implement*—for software cost estimation. I describe the activities of the UPI model and show how firms adapt their actions to their organizational and project-related setting. To derive the model, I conduct a multiple-case study and investigate the estimation process of software-producing firms from five industries. Further, I triangulate my results with theories from the information system literature to derive five propositions that support managers in designing their estimation process, e.g., finding the right balance between control and flexibility in different estimation stages. I contribute to the academic literature by describing the entire estimation process along three stages and outlining that software cost estimation goes beyond developing and testing estimation methods. Firms can apply the UPI model as a whole or partially to standardize their estimation process and follow the propositions to enhance the consistency of their estimation results.

Author: Marcus Witter

Status: Working Paper

3.1 Introduction

How do firms estimate software costs? Firms develop software in heterogeneous organizational and project-specific settings. However, one common pitfall is that the cost estimates of software projects tend to be too low (Moløkken-Østvold et al., 2004, Yang et al., 2008), leading to delayed software deliveries, budget overruns, and frustrated customers (Jørgensen, 2013). For example, Halkjelsvik and Jørgensen (2012) report that software projects exceed their budget on average by 30%. Simultaneously, successful software projects gain importance with the world economy transforming into an information economy (Shapiro and Varian, 1999). Fuelled by the rising demand for software products across industries, IT and software companies are situated on fertile ground for business growth. In contrast, industrial companies must manage the shift in value creation from a tangible, hardware-oriented approach to an intangible, software-oriented one. In this paper, I present a process model that I derive from a multiple-case study and that outlines how firms estimate their software costs. Further, I outline procedural estimation challenges and suggest propositions to support managers in designing the process by triangulating the results with theories in information systems literature.

The success of software projects strongly depends on meeting time and cost estimates (van Genuchten, 1991, Chow and Cao, 2008). Failure to do so arises from reasons such as staff turnover (Abdel-Hamid, 1989), alignment issues (Vermerris et al., 2014), poor requirements understanding (Lederer and Prasad, 1995) or underestimation of effort during project planning (Shmueli et al., 2016). Further, firms frequently lack accurate data at early project stages, causing wrong assumptions and making accurate estimates challenging (Angelis and Stamelos, 2000). Estimating effort is especially challenging for software products due to their complex development environment, which is affected by various social and technical factors (Kula et al., 2022).[8]

This study aims to reduce project overruns by creating transparency on how firms practically design and control their estimation process (Moløkken-Østvold and Jørgensen, 2005). Estimating software costs is a typical project management process that requires inputs and obtains

[8]The terms “effort" and “cost" are interchangeable and mainly applied synonymously. The working effort usually displays the central cost driver within software development projects and is considered a proxy for actual costs (Jørgensen and Shepperd, 2007, Huijgens et al., 2017). The effort drives the compensation of the development team and is recorded as direct variable costs of the project, positively associated with the project size and duration (Maltzman and Epstein, 2013, Chellappa and Mehra, 2018).

outputs by using resources (Trendowicz and Jeffery, 2014). Boehm and Papaccio (1988) introduce a software project planning and control framework in which managers estimate costs by project phase, activity, and product components to determine "should-cost" targets. Further, they suggest implementing MBO control loops to improve the control of software costs and outline situations when managers should favor project control over predictability (Boehm and Papaccio, 1988). Edwards and Moores (1994) outline the EEPS model, which differentiates between estimating software costs before project launch (top-down) and controlling software costs during the project (bottom-up). The model is structured in five sequential steps and describes a client's involvement from a developer's perspective in the negotiation process. In a similar vein, Usman et al. (2018) distinguish a product customization task into a high-level quotation and detailed analysis stage. However, they focus on a large-scale distributed agile setting. The quotation stage generates an efficient cost indication, whereas the detailed analysis stage focuses on refining the estimate.

Despite the availability of various estimation methods and guidelines (e.g., Boehm and Papaccio, 1988, Trendowicz et al., 2008), accurate cost estimation remains a critical challenge in the industry. Trendowicz and Jeffery (2014) distinguish between data-driven, expert-based and hybrid methods. Data-driven methods are based on the quantitative analysis of historical data and project the relationship between project costs and project characteristics of past projects onto new projects. In contrast, expert-based methods range from "gut feeling" to "structured estimation" , i.e., expert judgment supported by guidelines or checklists (Jørgensen, 2004b). Hybrid methods combine the advantages of different individual methods (e.g., Kocaguneli et al., 2012, Malgonde and Chari, 2019). The differentiation between quantitative and qualitative methods for cost estimation is also common for physical products in other disciplines (Niazi et al., 2006).

The question of how to estimate software costs has been the subject of research for several decades (e.g., Benbasat and Vessey, 1980). Extant literature on software cost estimation has focused on developing data-driven estimation models for measuring project size and duration (Moløkken-Østvold and Jørgensen, 2005, Menzies et al., 2017) and has analyzed estimation models from a technical point of view (Jørgensen and Shepperd, 2007). However, how to address specific cost estimation problems in practice is still underrepresented in academic literature (Eduardo Carbonera et al., 2020) because we do not know sufficiently well how the industry applies cost estimation models and what practitioners require. Thus, scholars recommend focusing on non-technical factors such as the estimation process and organizational issues when many stakeholders

are involved in the estimation process (Jørgensen and Shepperd, 2007, Jørgensen, 2014). Scholars call for research involving professionals in estimation studies to produce findings relevant to business practice (Eduardo Carbonera et al., 2020) and understand the "why" and "how" factors (Hannay et al., 2007) to account for the complex and human-behavior-driven phenomenon of software development (Austin and Devin, 2009). I address these shortcomings by analyzing why firms choose their estimation strategy and, ultimately, how they estimate software costs (Hannay et al., 2007). Motivated by extant literature, I state my research question: How do firms estimate software costs for individual projects?

I conduct a multiple-case study to record, analyze, and synthesize nine companies' software effort estimation processes. I apply the principles of diverse sampling to obtain results that can be generalized and to build strong theory (Eisenhardt, 1989). I focus my research on five different industries: software and cloud solutions, financial services, electronics, automotive, and professional services. The industry selection allows for controlling environmental variation, while the focus on individual software projects constrains variation in development types among the organizations. This specification reduces external variation and clarifies the domain of the findings as individual software projects in specific types of environments (Eisenhardt, 1989). After analyzing the cases individually, I apply a replication logic by comparing the cases with each other with a single case confirming or disproving the gained insights from the previous case (Eisenhardt, 1989).

I develop the UPI process model for cost estimation. The process model synthesizes the recorded single-case processes and differentiates between three structural levels. First, I find that firms follow three generic stages: Understand, Plan, and Implement. Second, I identify that the three project stages include ten activities. Third, I describe the features of the activities. In doing so, I identify that firms have a homogeneous perspective on the project stages and follow similar activities. Thus, the first two levels describe what firms consider when estimating software costs. However, the activity features at the third level are heterogeneous depending on the firm and project-specific setting. Based on the findings, I follow a contingency-based approach (Austin and Devin, 2009) by describing why firms select different activity features and how they proceed in their development context.

Further, I evaluate the identified estimation challenges and triangulate these with the results of previous research studies on software cost estimation to define propositions of how firms can standardize their process and enhance the consistency of their estimation results. I recommend

conducting workshops to increase the clarity of requirements, recognize team characteristics and the interests of different stakeholders, train estimators, adapt controls to the development situation, and build a database for data-driven decision-making.

This study aims to enhance the understanding of the software cost estimation process. It enlarges the perspective on estimating software costs beyond individual estimation methods and integrates the peculiarities of selected industries, value chain positions, and organizational roles. My findings suggest that practitioners should not rely on a unique estimation method but rather follow conceptual stages to meet context-specific requirements when estimating software effort. Further, this study enables firms to standardize their estimation process to increase the consistency of their estimation results.

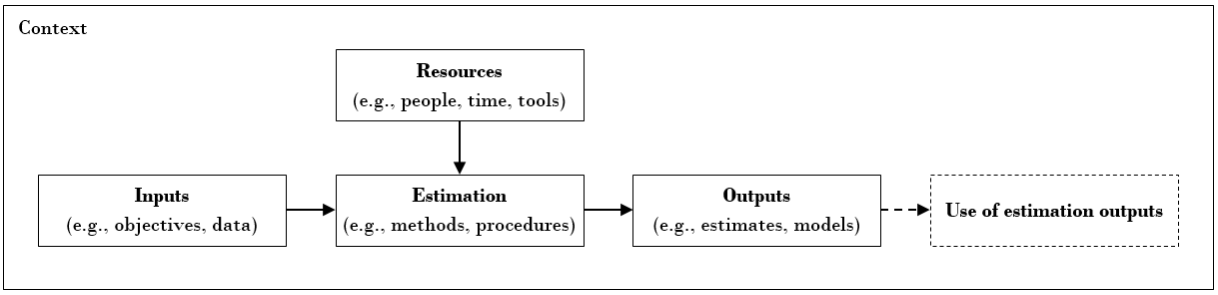
The remainder of this study is organized as follows. Chapter 3.2 reviews the software effort estimation literature. Chapter 3.3 outlines the multiple-case study approach. Chapter 3.4 describes the UPI model. Chapter 3.5 states the challenges and propositions, and Chapter 3.6 concludes.

3.2 Background and related work

3.2.1 Software cost estimation processes

Estimating software costs is a typical project management process requiring inputs and obtaining outputs using resources (Trendowicz and Jeffery, 2014). Inputs include the project objectives and qualitative and quantitative data from historical projects. The estimation can be carried out by various methods or procedures, which firms adjust to the project context and objectives. Resources consist of people, invested time, and applied tools for effort estimation. Similar to the actual estimation, the people and tools should adapt to evolving development contexts such as new software technologies or development paradigms. Based on the previous elements of the estimation process, firms derive the output. The output can represent an estimation model or an estimate (Trendowicz and Jeffery, 2014). Figure 3.1 illustrates the basic estimation process.

FIGURE 3.1: Basic estimation process.



Notes: This figure is based on Trendowicz and Jeffery (2014).

Boehm and Papaccio (1988) introduce a software project planning and control framework. They suggest two mechanisms for improving the control of software costs: First, by implementing MBO control loops. Second, by optimizing the software development strategy for project predictability and control. Managers estimate costs and schedules by phase, activity, and product components to create program evaluation and review technique (PERT) charts, work breakdown structures, and personnel plans. They use these resource allocation approaches to determine "should-cost" targets. As the project advances, they suggest different instruments to compare the progress and expenditures with the plan to create status reports that flag areas of MBO attention if necessary. Additionally, firms must balance their objectives on predictability and control of software costs. For example, firms prioritize control over predictability when synchronizing their software development with other developments, such as complex embedded system product launches. In doing so, firms can invest additional time at the project start to identify and eliminate sources of project risk. This risk-driven approach stands in contrast to a success-oriented approach which focuses on efficiency but is very costly if the projects' assumptions turn out differently (Boehm and Papaccio, 1988).

The EEPS model by Edwards and Moores (1994) outlines a survey-based theory, which differentiates between the estimation of software costs before project launch (top-down) and the control of software costs during the project (bottom-up), resulting in a plan-to-actual cost comparison. Further, the EEPS model describes how to involve the client from a developer's perspective in the functional and cost negotiation process. The EEPS model is structured in five steps and

refers to a waterfall development model, describing a linear, sequential software development approach. First, the client outlines the requirements to the developers. Based on the requirements, the developers discuss their plan to generate an initial rough estimate in a second step. Third, the client agrees on the initial estimate or negotiates a more detailed specification. Fourth, based on the agreement, the developers detail the project plan and iterate between the plan and estimate to ensure that the plan lies within the budget. In the last step, the client agrees to the budget, and the developers can start developing the system or software. Edwards and Moores (1994) recommend that managers differentiate between the top-down estimate based on an outlined plan and the bottom-up estimate based on a detailed project plan to accommodate for the amount of available information, the number of people involved, and the objective to be fulfilled by the information.

Usman et al. (2018) investigate how firms in large-scale distributed agile projects design their cost estimation process. They identify a two-stage estimation process for a product customization task by differentiating between the high-level quotation and the detailed analysis stage. During the first stage, firms prepare the quotation estimate in seven steps when they receive a request to initiate the customization task. The objective of the quotation stage is to efficiently generate the best cost indication as the basis for deciding whether to start a project or move on to the next development phase. However, firms have yet to determine which team will conduct the development. During the second stage, firms refine an analysis estimate and propose a solution in eight steps (Usman et al., 2018). Usman et al. (2018) outline how different roles work together to perform the estimation based on an expert-based approach in both stages. Further, they focus on describing tasks sequentially and elaborating on the collaboration between different project stakeholders.

3.2.2 Software cost estimation methods

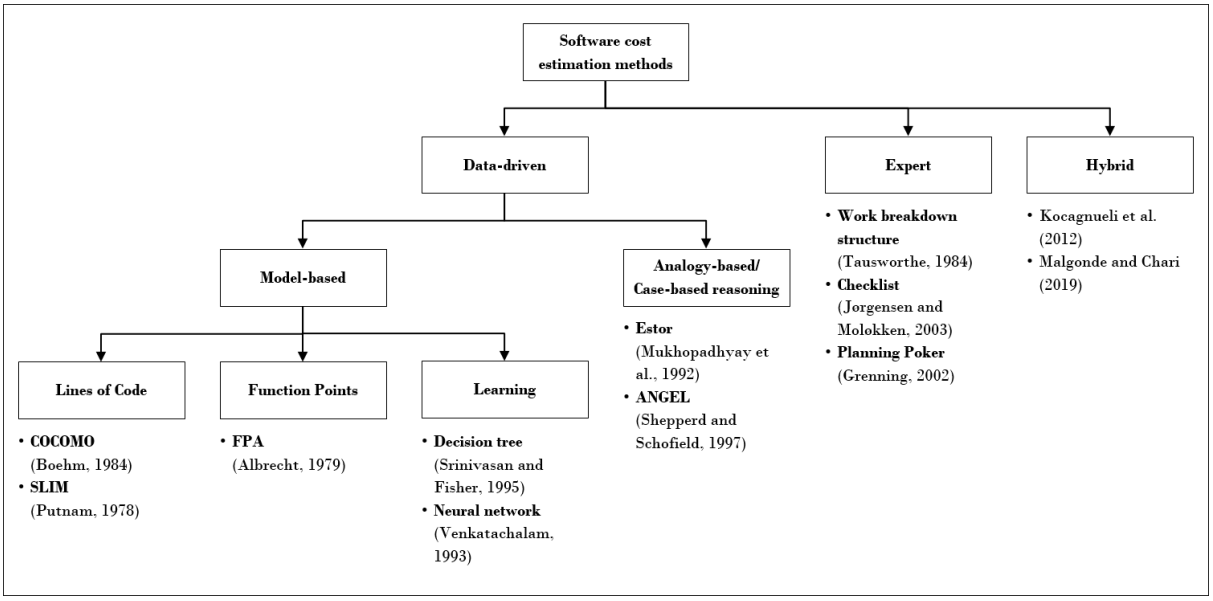
Scholars have proposed numerous methods for estimating software costs in the past decades. This study categorizes estimation methods into three overarching categories, i.e., data-driven, expert-based, and hybrid methods (Trendowicz and Jeffery, 2014). Data-driven methods predict project costs based on the quantitative analysis of past data. In doing so, they transfer the relationship between project costs and project characteristics of historical projects onto the new project (Trendowicz and Jeffery, 2014). Scholars differentiate between two sub-categories of data-driven methods: (1) model-based estimation and (2) analogy-based estimation. Model-based methods

apply algorithms to predict the costs of new projects based on past data (Menzies et al., 2006). They include (1) LOC-based models such as the Constructive Cost Model (COCOMO) (Boehm, 1981) and Software Life Cycle Model (SLIM) (Putnam, 1978), (2) function point-based methods such as the Function Point Analysis (FPA) (Albrecht, 1979), and (3) learning-based models such as decision trees (Srinivasan and Fisher, 1995) and neural networks (Venkatachalam, 1993). Analogy-based methods do not apply algorithms but explicitly leverage past project data for new estimations. Examples are case-based reasoning methods such as Estor (Mukhopadhyay et al., 1992) and ANGEL (Shepperd and Schofield, 1997).

I refer to a broad definition of expert-based methods, including the range from unaided intuition, i.e., "gut feeling", to expert judgment supported by past data, guidelines, or checklists, i.e., "structured estimation" (Jørgensen, 2004b). Further, firms can apply work breakdown structures (Tausworthe, 1979) or methods such as Planning Poker (Usman et al., 2015). Hybrid methods aim to reduce the weaknesses of individual methods by combining methods (e.g., Kocaguneli et al., 2012, Malgonde and Chari, 2019).

Figure 3.2 categorizes the estimation methods and shows typical methods including exemplary authors.

FIGURE 3.2: Categorization of software cost estimation methods.



Other disciplines categorize cost estimation methods for physical products similarly. For example, Niazi et al. (2006) differentiate between qualitative and quantitative product cost estimation. Qualitative methods are subdivided into intuitive and analogical cost estimations, whereas quantitative methods are split into parametric and analytical cost estimations. Controllers focus on quantitative estimation methods by defining cost functions that describe the cause-and-effect relationship between cost drivers and costs (Friedl et al., 2017). In doing so, they differentiate between analytical and statistical methods (Friedl et al., 2017). Analytical methods examine the cause-and-effect relationships between outputs and inputs by quantifying the resource consumption of processes. Controllers can leverage resources such as bills of material, work schedules, or technical documentation. Statistical methods such as regression analysis leverage the costs of past periods to estimate the cost function and forecast the costs of future periods (Friedl et al., 2017).

Recent software cost estimation literature observes that no "one-size-fits-all" approach is suitable for estimating effort in software development projects of all domains and applications (Resmi and Vijayalakshmi, 2019). Basten and Mellis (2011) find that the focal points of research do not reflect actual industry applications. While scholars focus on data-driven methods and size measures from a technical viewpoint (Menzies et al., 2017), e.g., regression-based estimation models (Jørgensen and Shepperd, 2007), practitioners prefer applying methods such as expert judgment, analogy-based, and work-breakdown-based effort estimation (Basten and Mellis, 2011). Trendowicz et al. (2011) report that most survey participants apply expert-based methods, and only 20% of the surveyed companies use data-driven models like COCOMO. Usman et al. (2015) find that agile teams rely on expert-based effort estimation methods due to the high degree of implicit development knowledge and focus on people and their interactions. However, firms often combine several expert-based methods because stand-alone techniques can only partially address some difficulties of cost estimation (Usman et al., 2015).

In expert estimation, one expert or an entire team of experts estimates the effort of development tasks based on their experience and without any underlying model. Instead, experts rely on their intuition, which displays a non-explicit and non-recoverable reasoning process (Jørgensen, 2004b). Expert estimations can follow a top-down or bottom-up strategy. For top-down strategies, experts estimate the total project effort without dividing the project into individual activities by comparing the project characteristics with completed, similar projects. Afterward, experts distribute the total effort estimate over the different project activities. For bottom-up

strategies, experts decompose the project into its individual activities, estimate each activity's effort, and add the estimates to a total estimate (Jørgensen, 2004a). Popular examples are the work breakdown structure, checklists, or group-based estimations like Planning Poker. The methods are more flexible with respect to the required input and time spent to generate the estimates (Jørgensen, 2004a).

The work breakdown structure displays a typical planning tool, which links objectives with activities and resources to a logical framework (Tausworthe, 1979). It organizes all work activities in a hierarchical order of detail, allowing managers to derive manageable tasks with quantifiable inputs, outputs, schedules, and responsibilities (Jørgensen, 2004b).

Scholars propose to apply checklists, which provide guidelines or process frameworks to standardize the estimation process and enhance the consistency of results (Jørgensen and Molokken, 2003, Jørgensen, 2004b, Usman et al., 2018). Jørgensen (2004b) argues in favor of checklists as they aggregate previous estimation experience and are easier to leverage than building up databases. He provides guidelines and recommendations for estimating software costs, e.g., to combine estimates from different experts and estimation strategies, ask the estimator to justify their estimates, or use documented data from previous development tasks but filter irrelevant and unreliable estimation information (Jørgensen, 2004b).

Development teams frequently apply Planning Poker as a single method (Usman et al., 2015). This method enables face-to-face interactions and discussions between team members to reach a consensus for the effort estimate (Moløkken-Østvold et al., 2008). It aims at reducing the estimation time and involving all relevant stakeholders.

Most experts rely on formal or informal analogies for estimating the project effort (Hihn and Habib-Agahi, 1991). Formal analogies are supported by historical data, process guidelines, or checklists, whereas informal analogies are based on unaided expert judgment such as gut feeling (Jørgensen, 2004b). However, analogy-based effort estimation methods are limited to the availability and quality of past project data (Li et al., 2007) and assume that projects with similar features entail a similar project effort, which is frequently wrong (Keung et al., 2008).

Why does research emphasize data-driven models while practice relies on expert-based methods? First, scholars argue that researchers and practitioners have not yet developed a data-driven model that estimates effort more accurately than other methods for widespread adoption in business practice (Vicinanza et al., 1991, Menzies and Shepperd, 2012). Data-driven models can

hardly be adapted to individual needs and organizational environments and ignore tacit expert knowledge. Further, firms have little understanding of the relationship between cost drivers and costs in complex parametric models (Vicinanza et al., 1991).

Second, researchers strive to develop a broadly applicable data-driven estimation model.[9] Researchers frequently rely on publicly available data sets like the ISBSG[10] or EBSPM[11] repositories to build their models (Rastogi et al., 2014, Huijgens et al., 2017). However, the characteristics of the data sets are only transferable to organizations to a limited extent, threatening the external validity of the developed models. Instead, practitioners prefer to apply models that specifically suit the requirements and characteristics of their projects.

3.3 Research design

3.3.1 Multiple-case study research approach

Following the calls for qualitative research in software cost estimation (Hannay et al., 2007, Eduardo Carbonera et al., 2020), I conduct a theory-building, multiple-case study according to Eisenhardt (1989) and Corbin and Strauss (1990). A case study design helps to address the research question of how firms estimate software costs for individual projects for two reasons. On the one hand, case study research displays an empirical inquiry that analyzes contemporary phenomena in a real-life context (Eisenhardt and Graebner, 2007), i.e., how practitioners solve the challenge of software cost estimation. On the other hand, case study research depends on multiple data sources enabling the triangulation of results.

I investigate multiple cases for three reasons. First, a theory built from multiple cases can usually be better generalized and is more accurate than a theory from single cases (Eisenhardt and Graebner, 2007, Yin, 2018). Second, a multiple-case study design enables the analysis within and across cases (Eisenhardt, 1989). My first interview partners confirmed the appropriateness of the multiple-case study approach by indicating that there is no best practice estimation process. Thus, comparing the process is of particular interest. During my analyses, I employ a

[9] I refer, for example, to Saeed et al. (2018) who suggest developing a data-driven model which achieves superior accuracy in comparison to existing models independent of the used data set.

[10] The International Software Benchmark Standards Group repository contains roughly 6000 projects from various industries and a wide range of countries.

[11] The Evidence-Based Software Portfolio Management repository entails data from roughly 500 software projects within three companies in the banking and telecom sector in The Netherlands and Belgium.

replication logic (Eisenhardt, 1989). Following Yin (2018), I consider the cases as a series of experiments with a single case confirming or disproving the drawn inference from the previous cases. Third, multiple-case studies allow the analysis of phenomena in a rich, real-world context (Eisenhardt and Graebner, 2007), resulting in a better understanding of the "why" and "how" factors (Hannay et al., 2007) and the complex and human-behavior-driven phenomenon of software development.

3.3.2 Data sample

I follow a diverse sampling strategy to develop a contingency-based process model that guides firms in estimating costs depending on their organizational and project-specific setting. The case variation is decisive in gaining manifold perspectives and considers a broad range of potential estimation strategies. Further, the diverse sampling enhances the generalizability of the results. The sampling strategy specifies the domain of the findings to software development environments. The unit of analysis is the individual software project from the start of project planning to the first release, excluding further updates and product maintenance (Austin and Devin, 2009).

Five criteria guided the selection of the cases: (1) The firms had to be active in different industries. Thus, I select nine case companies from five industries: computer software and cloud solutions, financial services, electronics, automotive, and professional services. (2) The firms had to develop different software products or services. I consider cases with firms developing standard software and firms developing customized solutions. While all firms have significant software development efforts, some develop software as a byproduct, while others consider software their main product. (3) The sample must include varying development environments, e.g., concerning time constraints and requirements uncertainty. Taking the example of the auto industry, the rising demand for technologies like autonomous driving fosters the progress of software-based functionalities and, at the same time, increases its development complexity at an unprecedented speed (Broy, 2006). Uncertainty originates in software's intangibility and inscrutability characteristics. It is reinforced during early development stages, following rather unclear or unknown requirements with necessary project information being often inaccurate or unavailable (Boehm and Papaccio, 1988). (4) The firms had to take different positions in the value chain and follow different business models, e.g., developing software as a supplier, purchasing software as a client, or offering software solutions as a service to end customers themselves. (5) The sample must include firms of varying sizes and organizational structures, e.g., small firms with up to 20

employees and no dedicated controlling or purchasing unit, medium-sized firms with up to 250 employees with a controlling and purchasing unit covering all software projects, and large firms with more than 100,000 employees with controlling and purchasing units organized by projects or customers. Table 3.1 gives an overview of the case sample.

Table 3.1: Sample overview.

#	Industry	Product/Service	Roles	Number Interviewed
1	Software	Enterprise resource planning	Chief Controlling Officer, Head of Controlling, Controlling Manager, Financial Planning Associate	4
2	Software	Customer relationship management	Chief Revenue Officer	1
3	Software	Cloud-based crowd sensor technology	Founder & CEO	1
4	Financial services	People-to-people platform	Head of Engineering	1
5	Electric/Electronics	Automated test equipment	SVP Software R&D	1
6	Electric/Electronics	Electronic manufacturing services	Director Software Engineering, Director IT Process Management	2
7	Automotive	Car manufacturer	Engineer	1
8	Automotive	Electronics	Director Hardware Technology	1
9	Professional services	IT consulting services	Partner, Director	2

3.3.3 Data sources and analysis

I gather data from two sources to ensure the validity and reliability of my results. First, I conducted semi-structured interviews with 14 employees of the described sample. The list of interviewees consists of software executives, project leaders, engineers, and individuals from the project audience, including subject matter experts of the various involved departments, e.g., accountants and hardware engineers. The semi-structured interviews enforce a systematic coverage of findings across interviews but still allow for improvisation and exploration of emerging issues. Second, I review various organizational documents or archival records, e.g., workflows, work breakdown structures, or project reports. The written data provides an additional source of documentary evidence to gain contextual information and to process past events. These additional sources enable the triangulation of data (Yin, 2018).

I validated the interview objective and guidelines independently by two research colleagues from other disciplines and one practitioner in the field of software development to ensure the clarity of its structure and the selected questions. Further, I interviewed an expert in the professional services industry to validate the initial interview results. After analyzing the data, I conducted an expert workshop with software developers, purchasers, and cost analysts of a firm in the automotive industry to validate the case companies' statements and the aggregated results.

The interviews are the main source for analysis. I designed the questions in a problem-centric way and included narrative and exemplary elements to ensure that the interviewee understood the question. During the interviews, I occasionally used selected illustrations of software engineering and cost estimation process flows to evoke past experiences and create a basis for the interviews. I structured the interviews in several parts. After starting with a brief introduction and personal-related questions, I discussed the software effort estimation process, including the rationale behind the process choice, mechanisms to control the process, and how the organizations leverage data to estimate software effort. I let the interviewee evaluate their estimation results and discussed reasons for effort under- or overestimation and their suggestions for improvement. I concluded the interviews with an outlook on new software cost estimation trends. Table A.1 in the appendix illustrates the case study protocol and Table A.2 in the appendix states the interview questions. I conducted the interviews in November and December 2021. All interviewees preferred interviews via video conference.

An interview took, on average, one hour and nine minutes, resulting in a total of 16 interview hours and more than 414 pages of transcript. Following Eisenhardt (1989), I stopped conducting further interviews after the additional findings had decreased significantly. After creating word-by-word transcripts of each interview, I coded the interviews in MAXQDA.[12] I followed the suggested coding approaches by Miles et al. (2019) and visualized the emerging structures according to Gioia et al. (2012). In total, I arrived at 1,268 codes (1st order concepts) that I grouped into 31 2nd order themes, into ten aggregated dimensions, and into three highest dimensions (see the overview of the data structure in Figure A.5 in the appendix, the data structure of the *Understand* phase in Figure A.6 in the appendix, the data structure of the *Plan* phase in Figure A.7 and Figure A.8 in the appendix, and the data structure of *Implementation* phase in Figure A.9 in the appendix). I collected and analyzed data in an iterative, case-by-case approach while developing a case narrative for each case to facilitate the development of an inside view. I focus on the cost estimation process, including identifying key events, activities, and cost-influencing factors. Following an initial within-case analysis, I conducted a cross-case analysis to derive the process model.

[12]MAXQDA is a software for qualitative and mixed methods research, supporting the researcher to code and visualize the data.

3.4 Results

3.4.1 Outline of the UPI process model for cost estimation

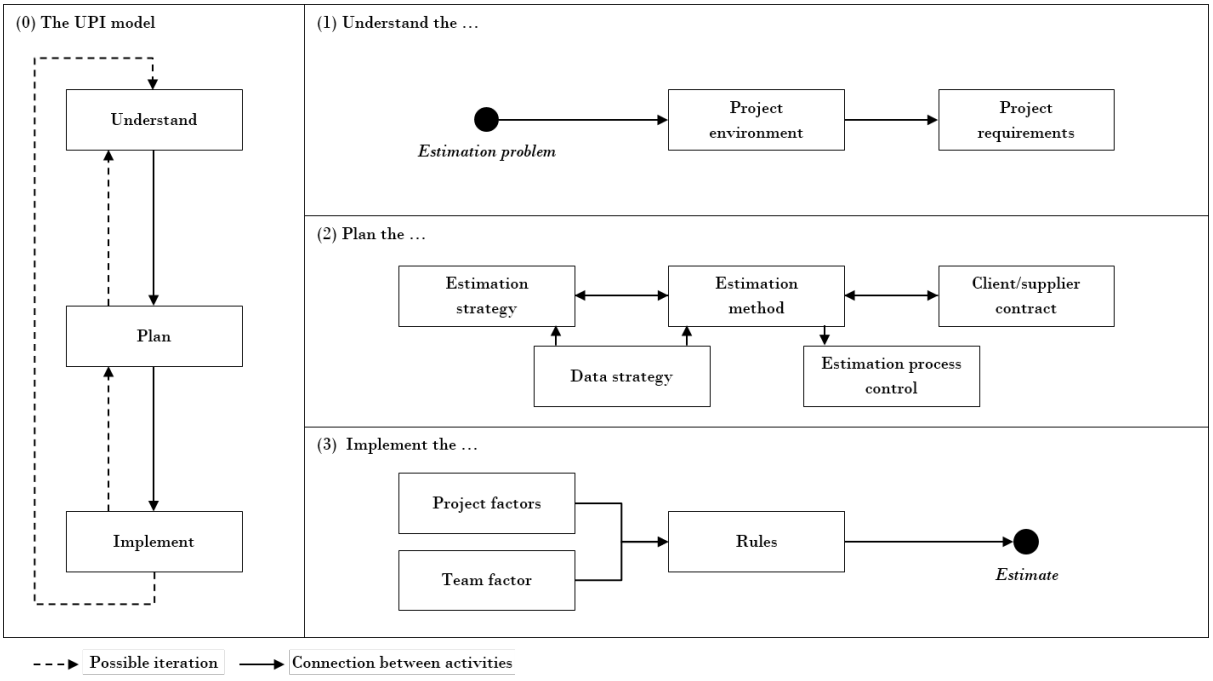
I develop a process model that synthesizes the findings from the nine cases. The interviews reveal that firms take three stages in estimating software costs: Understand, Plan, and Implement. The stages characterize different tasks, objectives, and outputs.

The stage *Understand* consists of comprehending and analyzing the characteristics of the project environment and requirements. Managers categorize the project environment according to product uncertainty, time dependency, and client-supplier involvement. Afterward, they try to understand the project requirements comprehensively and granularly. Based on the results, managers decide how to proceed in the other two stages.

The stage *Plan* entails selecting the estimation and data strategy, the estimation method, the project governance, and, if applicable, the contract between client and supplier. Managers decide on the estimation strategy, e.g., to estimate the effort bottom-up or top-down, depending on the results of categorizing the project environment. Further, managers select the estimation method in alignment with the project control mechanisms and strategy to build or extend a project database. Managers leverage the results of *Plan* to implement the estimation.

The stage *Implement* funnels the results of *Understand* and *Plan*. In tasks at this stage, the development team estimates software costs based on project and team factors. These factors are considered calibration factors when estimating software costs. The project factor comprises a novelty and complexity factor, whereas the team factor accounts for team members' competence and availability. Further, managers define implementation rules for the project and team factors before starting the estimation. Figure 3.3 displays the outline of the UPI model, and the following chapter describes the stages and their interrelations in more detail.

FIGURE 3.3: Conceptual illustration of the UPI model.



3.4.2 Description of the UPI model for software cost estimation

3.4.2.1 Understand

Project environment

Managers comprehend the project environment along three dimensions: the degree of uncertainty, time dependency, and involvement of a client or supplier. Categorizing the project along these dimensions serves as a steering mechanism for managers planning the cost estimation process and communicating expectations about the estimation accuracy to different stakeholders. First, managers characterize uncertainty by various means, e.g., the availability of a target picture and the product novelty. For example, managers differentiate between estimating costs for new and known software products based on available analogies and the varying uncertainty of cost drivers. When firms develop a product from scratch, the cost estimation process requires more effort, and the estimation accuracy will be lower. Second, projects can follow a continuous improvement approach without a definite time horizon or have a strict deadline in the short,

medium, or long term. Thereby, the estimation accuracy declines with an increasing estimation time horizon. Third, managers consider that the possible negotiation of tenders between clients or suppliers requires defining an additional process, e.g., for handling internal versus external change requests.

> *"We need to differentiate the development of standard software and customized software. For example, a customer's change request has different implications for the planning than an internal change request." (Case 1)*

Project requirements

The development team analyzes the project's functional and non-functional requirements. To maximize the understanding of the project, the development team decomposes the project largely, logically, and hierarchically, e.g., into functional blocks, epics, features, stories, or tasks. Scholars refer to this approach as the decomposition principle, which describes decomposing complex problems into relatively small sub-tasks that can be accomplished separately and combined later (Connolly and Dean, 1997). After decomposing the project requirements, managers describe the requirements as accurately and quickly as possible. The analysis of project requirements occurs upfront during the tender stage or directly at the beginning of the project phase. However, the process of understanding is continuous to account for requirement changes and can take place throughout the project life cycle.

> *"You can only do a proper estimation if you understand the specification." (Case 9)*

The interviews reveal that firms prefer conducting a specification workshop with all relevant stakeholders to understand the requirements and clarify possible ambiguities for estimating the development costs. The objective of the workshop is threefold. First, the workshop targets understanding the requirements and functional project scope as a team. For example, engineers can work during the design of a user interface with paper mock-ups to align requirements before starting to code. Following the advantages of user-centric design, contractors work as cheaply as possible, and clients perceive a smaller hurdle to demand changes. Engineers can improve the alignment iteratively beyond paper mock-ups, e.g., by using Photoshop, click dummies, or prototypes. Second, the workshop gives the stakeholders a platform to identify and discuss cost

drivers and evaluate the effect of environmental changes on past efforts for deriving analogies. Third, the workshop enforces identifying all relevant stakeholders and organizational interfaces.

> *"There should be dedicated workshops to discuss the specification. The developers should tell the requirements engineers in their own words what they have understood. This is necessary to ensure that both parties are discussing the same thing, even if this is tedious and time-consuming." (Case 3)*

3.4.2.2 Plan

Estimation strategy

The second stage *Plan* is based on the results of the understanding stage and comprises several activities. Which estimation strategy managers choose depends on the results of categorizing the project environment and analyzing the requirements. Managers distinguish between top-down and bottom-up estimation.

Firms encourage top-down estimations during early project stages when requirements are still vague, a detailed breakdown of project activities is not yet possible, when facing a strict delivery deadline, or when preparing the tenders for bidding phases. During the ongoing development, firms estimate development costs with a top-down approach if analogies are available, i.e., when the targeted product is not entirely new. Firms derive a top-down estimate, for example, after the specification workshop. The top-down estimation shifts the cost estimation in an agile setting towards more sequential planning because the firms consider all project requirements until the end of development. The high-level estimate serves as orientation value for further decision-making, e.g., aligning with stakeholders on project complexity, duration, and resources, weighing the benefits and risks of the project, and improving the development process by cutting the scope or outsourcing of tasks. The development team decomposes the requirements to estimate the effort and project lengths for individual functional blocks and on a high level of granularity, e.g., in person months. When firms follow a top-down approach, they can extend the specification workshop by identifying dependencies between the project requirements. This step enables prioritizing tasks and aligning contextual interaction points between the project organizations as early as possible. It increases the autonomy of the individual development teams, synchronizes the development process, and enables the integration of software code during later development

stages. Further, the team defines the critical development path based on the complexity and dependency of the requirements analyzed during the understand task category.

> *"When we need to launch a product in a specific time frame, we do a back-of-the-envelope estimate to give the business feedback about the rough complexity of the project and the resources required." (Case 4)*

Firms execute the bottom-up estimation continuously before sprints within the agile development process. In particular, in volatile development settings with changing requirements and a long development time horizon, development teams focus on prototyping and refining the requirements in the short term rather than estimating the long-term costs upfront. The bottom-up estimation entails a detailed project planning process and decomposes the requirements at the task or story-point level.

Firms can combine both strategies by applying the top-down approach during initial planning and the bottom-up approach during ongoing development. When firms estimate costs bottom-up, they can consider the identified dependencies between requirements. Developers start with the prioritized requirements, which often display the cost drivers, complex features, or common development hurdles. Managers synchronize and align development processes concerning the interaction points between the development teams and then implement parallel processes across teams.

> *"When we know the critical paths, touch points, and interfaces between different features, epics, and teams, we go into the agile mode and estimate and implement one epic after another, elaborate requirements, evaluate their complexity, and distill them to story points for sprint planning." (Case 4)*

As a notable exception, two case firms try to avoid estimating effort upfront for new products, characterized by uncertain requirements, lack of available analogies, and a long-term development horizon beyond six months. They are convinced that the estimation process is counterproductive and estimating such projects upfront leads to highly imprecise results. Instead, both case firms focus on developing a prototype, refining the requirements, continuously delivering finished increments and discussing the required capabilities of the software product while leaving product features, including their effort estimates, aside until they are better specified.

> *"We moved away from estimating features for projects with unknown requirements because it was a waste of time. [...] The requirements are changing so quickly that it did not end up being worth the time relative to the accuracy we got and, more importantly, whether we released the feature. We estimate for what we are going to release in the next three months." (Case 5)*

Data strategy

Almost all firms highlight the necessity of defining the criteria and the process for collecting data consistently throughout the entire development process to build a standardized data repository. The interviews show that firms are starting to collect and analyze data to enable data-driven decision making of individuals in expert and model-based effort estimation. However, most firms still rely on individuals' recollections of past projects. The development teams leverage their implicit experience to derive analogies across and within projects to conduct the group-based expert estimation.

> *"The past effort data is in people's heads and gut feeling. We need to systematize data collection and enforce comprehensive and consistent collection on the most granular level because people's memories are not perfect." (Case 5)*

Individuals firms, in particular in a supplier position, already document their effort automatically with tools like Jira or Confluence and manually with tools like Excel. Managers analyze the individual changes in productivity or velocity within a defined time frame to consider learning curves in the data-driven estimation. Further, development teams document and analyze change requests separately to better trace requirements and understand what has changed. Consistent documentation is decisive in identifying and evaluating firm- and project-specific cost drivers and their impact on the estimation. Managers can subsequently track the cost drivers. Lastly, firms enforce the consistent use of data by all team members.

Currently, firms are focusing on building databases by collecting data comprehensively and systematically to enable a data-driven estimation and comparing data across projects. Managers define criteria for collecting data, including the project environment and the project itself. The project environment-related data is based on the results of the *Understand* stage and comprises, for example, the customer, development environment, time frame, and uncertainty. Considering

the project environment is important to account for the individuality of the different projects. The project-related data consists of the functional and non-functional requirements, which are broken down to the smallest granularity level possible, e.g., story points. Also, the data consists of the estimated and actual effort, the respective skill class of the individuals and team, and hourly or daily wages. Collecting the skill classes enables a differentiated perspective on the past effort when deriving analogies for future projects. Firms can derive an individual's or a team's productivity or velocity, e.g., by measuring the completed story points per sprint. Further, firms define a methodology to harmonize data logging and text standards to make the requirements comparable.

Estimation method

All firms rely on expert estimations, while group-based approaches are the dominant alternative. Firms apply the same methods in the early stages as during the actual development and regardless of the project environment. A difference between estimating effort in the early stages and during the actual development applies only to the aggregation level of the estimates, e.g., in epics, stories, or story points. Managers face the challenge that the accuracy of the estimates decreases with an increasing degree of requirements uncertainty and an extended project time horizon.

> *"Our firm's estimation techniques do not depend on the project environment. I think that the techniques are largely the same. It is the accuracy of the outcome that varies greatly." (Case 5)*

Managers primarily perceive expert estimation as a group task, combining structured methodological procedures with the experience of the estimators. Further, group-based estimation has often grown historically in firms. It displays a valuable part of the culture of the companies to encourage and not penalize employees for their decisions. The objective of the group task is to discuss individual estimations and reach a holistic consensus as a team. Further, the estimations display a reference basis for subsequent discussions. That managers involved in the estimation process need to find a consensus about the cost estimate is a control mechanism by itself. Literature refers to this kind of control as clan control (Maruping et al., 2009).

> *"There is not a method that solves everything. Most of the time, it is not the method itself but the culture, the people, and collaboration within the project governance framework. You can have the best tools available, but without the other factors, you will not be able to make a good estimation." (Case 9)*

In practice, there are various options to implement group-based expert estimation, e.g., the Delphi method, Dice game, Fibonacci estimation, Infinity estimation, or Planning Poker. In all these methods, cost estimation builds on a rating of complexity. For instance, poker cards or dice figures refer to complexity ratings.

> *"Generally, teams apply their expert knowledge, regardless of the concrete agile development process and estimation method." (Case 8)*

Managers define the team size and roles for the group-based expert estimation. Firms often have team sizes of five to ten individuals. The number of teams necessary for an individual project depends on the project size. Managers define roles within a development team, e.g., coding, estimation, and project responsibility. Additional roles like a trained communicator between the different parties are necessary for client-supplier settings.

> *"The trained communicator between supplier and client must be a qualified person who understands the different perspectives and speaks the language of all stakeholders to translate the requirements between the parties." (Case 7)*

Four out of nine sample firms apply data-driven approaches selectively as an add-on to expert estimations, not on a stand-alone basis. The objective is to standardize estimates. Also, following a top-down estimation strategy, development teams often apply more than one estimation method to increase the reliability of their results.

> *"We do not recommend teams to rely on one particular estimation method. Instead, teams should estimate costs using at least two different ways, e.g., expert-based estimation and data-driven estimation, and then compare the results." (Case 8)*

If applicable, practitioners rely on data-driven approaches like linear regression analysis to estimate future costs based on firm-specific data of past projects. Cost drivers comprise numerical

features like the effort for developing requirements of past projects and categorical features like the complexity rating of requirements or the experience of the programmers. Firms apply the data-driven approach for similar and not entirely new features, for which data of past projects is available.

> *"The objective of the data-driven approach is to delete politics from the numbers. However, it is only an add-on to the expert estimation." (Case 1)*

The development team conducts the effort estimation in person-hours or person-days. Afterward, the estimation effort in person-hours or person-days is multiplied by the respective hourly or daily rate of the individuals or development team. This step is necessary to translate the effort to a monetary unit, simplify communication with the management, and ask for product approval. The development team or the respective controlling unit can conduct the monetary calculation. Distinguishing effort and costs is necessary to avoid estimation bias by accounting for differences in development productivity and cost rates. Besides estimating the non-recurring development effort for an individual project, managers also consider auxiliary costs and risk costs as well as recurring costs for service and maintenance, e.g., for the product or infrastructure.

Estimation process control

Firms implement different process control mechanisms for their effort estimates depending on the project's volume or strategic importance and the implementation timing, i.e., before or after the project start. Thereby, the development team and respective controlling team have different control responsibilities. On the one hand, the development team is responsible for reviewing their technical estimations, including the developers' effort and cost rates. Based on the review, the development team tries to identify cost savings or adjust the implementation to avoid development hurdles. The team can, for example, develop a minimum viable product that the customer requests by minimizing the number of realized features and, thus, limit their risk by being able to stop the development earlier. The development team focuses on informal controls like clan control, enforced by the agile estimation techniques within a project team, as the teams need to find a consensus for estimating the requirements. Additionally, managers can implement formal controls, e.g., the development teams are requested to discuss estimates for which the value deviates by 50% from the mean value. Managers can display the target-actual comparison and velocity per sprint in burndown charts or velocity curves.

> *"We shifted from waterfall to agile development, creating a new culture and departing from formal control. Following the agile estimations, things run far more independently. So you do not have to have a meeting every Monday morning and pull in everybody to review their estimations. It just happens naturally on the teams. It was as much about creating a culture as it was about actually formal control." (Case 5)*

On the other hand, financial controlling can additionally control estimates. In general, the financial controlling team integrates various perspectives, e.g., development, sales, and finance, to gain a cross-project perspective, validate the alignment of development and firm strategy, and analyze the impact a project can have on the firm. The Chief Operating or Financial Officer is responsible for setting the right incentives, defining the standards for Key Performance Indicator (KPI) reports, and implementing a governance board to steer the resources. Further, the financial controlling team defines an interactive review of progress, e.g., monthly. During the review, they conduct a target-actual comparison for user stories and discuss the burndown chart.

Financial controlling teams apply an integrative financial planning tool, including different financial planning levels for the development department, e.g., the long-term planning for the entire firm, the strategic financial planning for the development department, and business case planning for the product area. The tool improves the speed of planning. Taking the example of strategic financial planning, financial controlling teams apply value driver trees and conduct backward planning to define the budget for a development department. First, they plan the revenue per department based on estimated market shares. Second, they define department-related KPIs. Third, they derive the budget for the operative units based on the KPIs. Further, financial controlling teams validate the internal and external cross-project capacity planning for a development department. They evaluate the competence profiles and location strategy of the existing workforce, e.g., low versus high-cost regions, and consider project-related reallocations. Alternatively, they consider hiring a workforce externally. Thereby, firms consider a realistic duration for the hiring process, including the notice period and the ramp up of new resources, which often takes up to three to six months.

Controls are important because managers need to recognize the political motives of the different individuals, teams, and departments when evaluating the estimates. The development, financial controlling, and sales teams cooperate from the start of the product planning phase. However, the parties have different departmental objectives and, thus, incentives to over- or underestimate.

> *"I've seldomly seen a party in an organization that didn't have an incentive to either over- or underestimate." (Case 9)*

The development teams often have incentives to overestimate the effort to block resources and implement a safety buffer for development. Further, the development teams prefer to avoid measuring their actual development progress.

> *"Improving R&D efficiency has been the number one way development teams have been able to boost their power in the organization. We could improve the efficiency by introducing the right measurement metrics." (Case 9)*

Other organizational functions like financial control and purchasing follow different objectives. Purchasers challenge the offer commercially, not technically, and have incentives to buy the product or service as cheaply as possible. Financial controllers prefer to estimate as accurately as possible, yet, often lack the information to be exact.

> *"Financial control is an organizational unit that is trying to be precise, but they often lack the information to be precise." (Case 9)*

Contract design choices in client-supplier settings

Four case firms highlight the effect of contract types in client-supplier settings on designing the estimation process. Clients and suppliers generally arrange their relationship via three contract types: time and material, fixed price, and agile fixed price. A *time-and-material* contract includes agreed cost rates for material and time spent by employees. These rates usually differ for groups of employees with different skills. The total price then depends on the total time the employees work on the project (Jørgensen et al., 2017). Especially in early development phases, clients and suppliers frequently agree on a *time-and-material* contract because requirements are hard to specify, and this contract type provides the highest flexibility. On the one hand, the clients benefit from the flexibility in the early development phases. On the other hand, the clients monitor the project's progress and bear the risk of cost overruns as they have no guarantee for deliverables and cannot claim rework. Literature and interviews have revealed that *time-and-material* contracts tend to favor the suppliers (Gopal et al., 2003).

On the contrary, for *fixed-price* contracts, the clients and suppliers negotiate a fixed fee for delivering a specified software (Jørgensen et al., 2017). Thus, the contract explicitly states product requirements but does not refer to the resources, time, and material spent to achieve the task. Clients benefit from *fixed-price* contracts because they can claim rework if the requirements are not achieved in the specified quality. Hence, the suppliers bear a higher share of the project risk under this contract. The contract requires a specific upfront definition of deliverables and testing options (Gopal et al., 2003).

The *agile fixed-price* contract is a hybrid contract type. The full project price is fixed, comparable to a *fixed-price* contract. However, the project scope is variable and settled iteratively. This contract fosters communication between clients and suppliers and partially eliminates the disadvantages of the other contract types. Specifically, the clients and suppliers agree on a standardized collaboration process, which they frequently base on the SCRUM principles. The clients' and suppliers' product owners (PO) take decisive roles as contact and negotiating partners between the firms. They meet roughly two months before the project start for a *PO board 1*, in which both parties plan the assignment. The client communicates the requirements, including an effort estimate for the epics, stories, or story points, depending on the respective level of product uncertainty. Thereby, one sprint displays one increment and takes two weeks. Further, the clients and suppliers discuss a monthly fixed price for pre-defined roles, for which continuous support independent of the actual development task is necessary. The *PO board 1* takes place twice a year, in which the client and supplier discuss improvement ideas and open requirements of the old assignment. Firms choose this contract type when the requirements are largely defined.

The team starts with a *sprint planning session*, in which the team plans the stories of the upcoming sprint. After each sprint, the developers demonstrate the results of the stories in a *demo session*. The demo session participants then decide on approval and the subsequent change of sprint content or if the developers must further improve their results. During the sprint phase, when no demo session occurs, the client and supplier team conduct a biweekly *refinement session 1* to enable a common understanding between the parties, clarify questions, estimate, and adjust story points for the upcoming stories. The team estimates the story points with the required skill class to derive the necessary development time. All participants discuss the results, align on the estimate, and, subsequently, the costs. Further, the team participates in a biweekly *PO board 2* to ensure transparency of requirements. The clients communicate the requirements in their

language, and the product owner of the supplier translates the requirements into stories. The meeting takes place when no sprint change occurs. Additionally, the suppliers' team conducts internally a biweekly *refinement session 2* to reflect on the progress, clarify the requirements, and improve the development.

The clients and suppliers can define a neutral third party from an independent firm to support their communication, the so-called PO partner. PO partners should understand the needs of both clients and suppliers and translate the developers' technical obstacles to the POs and Scrum Masters while incorporating the client's mindset. They participate in the meetings of suppliers and clients and act as mentors. They are usually compensated hourly.

3.4.2.3 Implement

Project factors

The third stage *Implement* is based on the results of the *Understand* and *Plan* stages. According to the chosen estimation and data strategy, managers implement the decomposed requirements of *Understand*. Taking the SCRUM methodology as an example, the decomposed requirements are prioritized in the product and sprint backlog. Engineers consider the identified dependencies and start with the most critical requirements.

Managers consider project factors when estimating the development effort. The factors comprise categorizing the decomposed requirements by complexity and novelty. First, managers can divide the complexity categorization into requirements and system complexity. To define the complexity of the requirements, managers define complexity classes and allocate the requirements to the respective classes. They can do this by evaluating past cost drivers, reference tasks in the product backlog, or based on their professional experience. To define the system complexity, managers decompose the software system into sub-systems and agree on system interfaces to define a complexity map. The development team defines complexity ratings for the different sub-systems based on their experience. Managers analyze the system complexity to account for differences in the system regarding code implementation, testing, integration, and the handling of change requests. Engineers locate the decomposed requirements of the new project on the map and evaluate which requirements can be allocated to the different sub-systems to consider additional complexity if necessary.

> *"We use a map of complexity for our software system. If a new project steps in the defined complexity areas, we consider this for the estimation." (Case 4)*

Second, development teams evaluate the project's novelty by leveraging their experience and deriving analogies to understand the differences between the new and previous projects. The development teams rely on defined categories or criteria for analogies, e.g., previously identified cost drivers. The team then projects the criteria-related data to the new requirements and development environment. They try to understand differences in, for example, the uncertainty of requirements, new technology, data content and quality, programming language, technical environment, infrastructure, development methods and processes, testing methods and level of testing, security levels, or tools. The analysis supports understanding whether the past data is relevant and can be used for estimation. However, firms require reference data specific to the development team to effectively and efficiently derive cross-project analogies as the firm perspective is insufficient. Firms often lack relevant internal data for estimating new products, and the requirements of new products are often not specified in detail to enable a comparison.

> *"The biggest challenge you are facing is understanding to what degree this requirement is more different and novel than you have seen before." (Case 9)*

Further, managers recognize within-project, cross-project, and cross-firm lessons learned to recalibrate their estimates. Development teams apply the within-project factor after the project starts during the ongoing development when project teams evaluate the retrospective after each sprint and adjust their estimates if necessary. Development teams evaluate the cross-project and cross-firm factors before the project start. The cross-project factor describes the learning curves the firm has taken since conducting the reference projects beyond the team level. Managers make the development team aware of all previously identified project cost drivers. Developers consider the cost drivers comprehensively and then analyze which ones are relevant for the specific use case. The cross-firm factor incorporates the technical progress since the organization had conducted the reference projects. Engineers can identify the factor by conducting external benchmarks to identify new technologies, programming paradigms, as well as development methods and derive market-wide lessons learned. In addition, developers search for relevant parameters for effort estimation in the research literature. The cross-firm factor is crucial for new products to identify if analogies are in the market, which can be used for effort estimation. Managers require development teams to incorporate these lessons learned in future estimations.

> *"We cannot do the same thing repeatedly and hope for a better outcome. Learning curves only can be realized by management action. It does not happen by itself." (Case 9)*

Team factors

Managers also consider defining a team factor, which comprises the individuals' capabilities and availability. Managers derive the team factor on an individual team member level. Managers compare the assigned roles and responsibilities with the ones in previous projects and investigate differences in skills, seniority level, domain-specific knowledge, and estimation characteristics of individuals, e.g., if they tend to over- or underestimate or include a personal safety buffer. The results are consistent with Shmueli et al. (2016), who find that project managers underestimate the time needed to complete a task.

In addition, managers apply a focus factor for the individual team members to plan capacities realistically during the project. The focus factor defines the capacity of individuals for the project and thereby considers that engineers also work on other projects and have other obligations within the firm. This factor is reviewed regularly, e.g., every three months.

> *"You need to recognize the domain knowledge of the individual team members. You cannot simply exchange individuals who worked in different domains. That is like you trying to make a person who formerly worked at the bakery now to be in the tailor's shop." (Case 9)*

Rules

Managers define the implementation rules for the project and team factors before the estimation starts. They communicate the rules to the development team to create a common understanding. The project and team factors relate to a defined number of story points so that the project and team factors can be translated into an effort estimation and respective time estimation. Alternatively, the development team can implement the factors as additional iterations.

3.5 Challenges and propositions

I summarize the most frequent challenges the sample firms faced during the estimation process and assign the challenges to the respective activities of the UPI model. Based on the identified challenges, I derive propositions by triangulating the results of my qualitative data analysis with insights gained during the literature review (Eisenhardt, 1989). The propositions aim to support managers in designing the estimation process, reduce situational and human biases, and, thus, increase estimation accuracy. The overview of the challenges and propositions is presented in Table 3.2.

Proposition 1: *Conduct workshops to increase the clarity of requirements.* The interviews reveal that conducting an initial specification workshop to discuss, define, and understand the requirements with the decision-maker, client, or supplier reduces the risk of neglecting development tasks, improves information sharing and alignment with other functions, and, ultimately, increases the estimation accuracy. Further, all stakeholders should develop or discuss the product vision and roadmap. The alignment of product objectives facilitates the requirements specification and reduces uncertainty. The initial planning stage is crucial for long-term projects, projects with a definite time horizon, and projects with many stakeholders. Further, the workshop displays the baseline for applying the three complex adaptive systems (CAS) principles by Vidgen and Wang (2009). Conducting a specification workshop to collaboratively define the scope and requirements sets the basis for prioritizing development tasks and monitoring and tracking requirement changes. Managers should distribute control and decision making in a team to optimize self-organization and coordination by applying group-based expert estimation techniques like Planning Poker to reach a consensus for each estimation. When coordinating the estimation, the team members discuss the detailed specification and who will take over the development task. The team leverages its capabilities within this process to fit best with each development task. The process minimizes the risks of a wrong understanding of requirements and serves as a guideline for inexperienced developers.

> *"The deviation of estimation results is lower for long-term planned than short-term planned estimations because you can just put more time into formulating the requirements and planning their implementation." (Case 3)*

Table 3.2: Challenges and propositions.

Challenge	**#cases**	**UPI model activity**	**Proposition**	**Related literature**
1. Firms frequently underestimate the project complexity by not understanding the project novelty (e.g., product, environment) and the requirements themselves (e.g., not comprehensive, formulated in a too complicated manner, not precise enough). Further, firms lack initial information sharing between departments and recognize a gap between what the client wants and what the supplier understands. Also, firms find it difficult to find a sweet spot for spending time for initial effort estimation.	8	Understand - Project requirements	Conduct workshops to increase the clarity of requirements	Vidgen and Wang (2009)
2. Firms neglect useful information about past projects and do not use available analogies for estimation (e.g., compare and analyze past cost drivers). They fail to implement usable knowledge management mechanisms to counteract their lack of experience. This is, in particular, challenging when the development team changes and new developers are hired.	7	Implement - Project factors	Train estimators	Harris et al. (2009)
3. Firms cannot rely on data-driven estimations due to a lack of discipline for project documentation and a defined process for systematic data collection (e.g., no relevant data available, no specification text standards defined, individual and inconsistent use of data). Firms highlight the future relevance of applying machine learning-based data analysis and forecasting methods, for which systematic data collection is decisive.	6	Plan - Data strategy	Build a database for data-driven decision making	Lederer et al. (1990) Jørgensen (2004b)
4. Firms derive analogies of project effort between different teams, which does not account for the individual differences in capabilities, capacities, locations, and cultures. Further, managers often do not recognize the political motives of the different estimators when reviewing the estimates, not eliminating situational and human biases in the estimates.	4	Implement - Team factors	Recognize team characteristics and the interest of different stakeholders	Menzies et al. (2017)
5. Firms face difficulties to control the estimation process when the requirements are unclear at the project start or the requirements and scope are changing in the ongoing development process.	3	Plan - Estimation process control	Adapt controls to development situations	Jørgensen and Molokken (2003) Usman et al. (2018)

However, the planning of the initial estimation is a two-edged sword. The benefits of increasing the understanding of the requirements confront the downsides of time loss and changing requirements during development. Especially in volatile environments without client-supplier interaction and specific deadlines, firms can shorten the initial planning stage and start with agile development quickly.

Proposition 2: *Train estimators.* Managers should systematically train engineers to identify and analyze past cost drivers to counteract the lack of experience with the task, project, client, and supplier. Estimators need to get an overview of all cost drivers depending on the project environment to avoid neglecting development efforts. Managers can give them documented data. In addition, managers can give them checklists to support the estimation process (Jørgensen and Molokken, 2003, Usman et al., 2018). Upfront training increases productivity by decreasing the amount of communication necessary during the project. For example, firms often underestimate the effort to define the software test environment as close to reality as possible and define test scenarios that cover the requirements and their interaction. Thus, the development teams should conduct test-driven development to avoid testing becoming a bottleneck, i.e., testing every feature immediately after development and continuously testing the software against all test cases. The objective is to reach a high code coverage with automatic tests.

Proposition 3: *Build a database for data driven decision-making.* The interviews show that firms focus on collecting and analyzing data systematically to enable data-driven decision making of individuals in expert estimations and data-driven effort estimation. Managers highlight that systematic data collection is the decisive next step before applying machine learning (ML)-based methods. These findings are in line with Menzies et al. (2017), who summarize that "how data is collected is more important than what learner is applied to that data." The potential application of ML-based methods has not yet arrived at the case firms. Managers highlight that using ML-based estimation methods has many data collection, analysis, and forecasting opportunities. For example, development teams can use natural language processing to search for text similarities in product specifications. Further, they can apply AI-based estimation methods to flag peculiarities and give the estimator feedback about the estimation based on previously collected data or find similar tasks based on defined project parameters in databases for effort estimation. However, firms investigate the opportunities only as a subsequent step after building a data repository. ML-based estimations require detailed and comprehensive inputs, and, thus, managers think that the effort necessary to specify respective models frequently outweighs the benefits from improved

cost estimations. Firms must specify an extensive model containing detailed information about individual tasks. Lastly, the future potential depends on the individual use cases.

> *"We need reliable data for applying methods based on artificial intelligence. There is certainly potential, but the first step is to build a database, and the step afterward is to apply these methods." (Case 1)*

Proposition 4: *Recognize team characteristics and the interests of different stakeholders.* Managers must recognize team characteristics in effort estimation. Team characteristics include the varying capabilities, capacity, locations, and culture among individuals and development teams. Besides individual competence, managers must also recognize the estimator's political motives, which can lead to inefficient decision making, e.g., the unwillingness to look bad in front of a superior, reluctance to analyze the retrospective honestly, or communicating views contrary to the group's perspective. Lederer et al. (1990) pick up the different political motives of estimators and differentiate between a "rational" and "political" model of the estimation process. The rational model aims to achieve the highest estimation accuracy, whereas the political model includes individual motives, goals, and power conflicts. Jørgensen (2004b) suggests six principles to reduce situational and human biases: Evaluate the estimation accuracy, but avoid high evaluation pressure, avoid conflicting goals, ask the estimators to justify and criticize their estimates, avoid irrelevant and unreliable information, use documented data from previous development tasks, and find estimation experts with highly relevant domain background and good estimation records.

Proposition 5: *Adapt controls to development situations.* Agile methodologies like SCRUM structure the development with dedicated meetings, e.g., the planning, review, retrospective, and daily meetings. For example, the retrospective meetings intend to discuss the team's behavior and define lessons learned for the upcoming sprints. Two other forms of control extend this form of behavioral control. First, managers' formal control is based on their authority when reviewing final estimates. Second, a group's clan control is based on self-control and subtle peer-to-peer signals when conducting group-based expert estimations. I suggest extending the control mechanisms by emergent outcome control, enabling managers to take continuous corrective actions and give the project team dynamic feedback when the scope and requirements change during estimation and implementation (Harris et al., 2009). Thereby, managers apply two control mechanisms: Scope boundaries and ongoing feedback. Scope boundaries limit the

set of possible solutions without specifying outcomes. Boundaries can display a product vision, feature specifications, or technical constraints like the architecture. Ongoing feedback is essential when the scope boundaries are not set tight enough and targets the development with a minimum of iterations. The mechanisms are interdependent and inherit trade-offs between the control choices (Harris et al., 2009).

3.6 Conclusion

This study develops a process model that outlines how firms estimate their software costs. I observe the software cost estimation process over three stages: Understand, Plan, and Implement. I describe the activities within each stage and differentiate between organizational and project-specific settings. I triangulate the case study results with the literature on software cost estimation to derive propositions that support managers in designing the estimation process.

I conduct a multiple-case study to address the lack of developing theories in software engineering (e.g., Hannay et al., 2007, Jørgensen and Shepperd, 2007, Dybå and Dingsøyr, 2008, Usman et al., 2015, Idri et al., 2015, Eduardo Carbonera et al., 2020). I add to the literature on software cost estimation by reflecting on the current state of practice (Jørgensen and Shepperd, 2007, Eduardo Carbonera et al., 2020). The results allow researchers to identify similarities and divergences between research and practice and base their future research efforts on real-life findings (Eisenhardt and Graebner, 2007).

Improving estimation accuracy is a critical goal for organizations to more effectively plan and control the project budget, efficiently allocate resources, reduce costs and delays, and improve customer satisfaction (Heemstra, 1992, Jørgensen and Carelius, 2004, Huang et al., 2008). This study provides a rich understanding of the software cost estimation process for individual projects. It enlarges the perspective on software cost estimation beyond describing individual estimation methods and comprises the entire process from understanding the project environment to estimating the costs. The process model embodies peculiarities of various industries and incorporates different firms' perspectives, e.g., clients and suppliers, and organizational roles, e.g., software developers, effort estimators, project leaders, and controllers, to understand different facets of the estimation process. Taking this comprehensive perspective, I add to Boehm and Papaccio (1988) who describe a software planning and control framework, Edwards and Moores

(1994) who introduce the EEPS model, and Jørgensen and Molokken (2003) and Usman et al. (2018) who suggest using checklists structured along the software cost estimation process.

My results show that firms have a homogeneous understanding of the high-level estimation process, which can be structured in three stages: Understand, Plan, and Implement. However, the activities within each stage are heterogeneous depending on the organizational and project-specific setting. The estimation methods is a notable exception. I show that expert judgment is still the dominant method across industries, and firms rely on group-based expert estimation methods (Jørgensen, 2004b, Idri et al., 2015). The interviews reveal that this is due to the high degree of implicit expert knowledge and the high effort in setting up a data-driven model. The results are consistent with extant literature that previous experience from a similar task is necessary for achieving accurate estimation results (Haugen, 2006). Further, I show that development teams apply the same methods in the early stages as during the actual development, regardless of the project environment. Differences only occur at the aggregation level of the estimates, e.g., in epics, stories, or story points.

Firms can apply the UPI model as a whole or partially, depending on their organizational and project-specific needs. The study suggests when firms should differentiate their estimation strategy dependent on the project environment and suggests propositions to improve their estimation process. Firms can compare their current estimation process with the UPI model to adapt or extend their process.

My results are subject to limitations. I focus on how firms estimate their software costs and how the process is adapted according to their project environment. Further, I analyze the procedural estimation challenges they face to derive propositions. However, the multiple-case study design does not reveal which stages and activities of the UPI model are mainly responsible for poor estimation results, e.g., to which degree inferior requirements understanding, planning, and control, or estimation methods contribute to the inaccurate estimates. Further, despite suggesting propositions based on identified estimation challenges, my analysis does not suggest a normative cost estimation process that practitioners can apply to achieve superior estimation results. In addition, I do not investigate whether and how defining an initial estimate influences managers in designing the remaining development process. Finally, I study software cost estimation processes for individual projects. This focus constitutes a control in my study. However, my findings may be limited to individual projects. It is possible that my results do not generalize to other software development practices such as continuous developments or research projects.

Scholars can strengthen the validity of my results by examining cost estimation processes for other development methods.

My study reveals interesting avenues for future research. I suggest that scholars and practitioners validate the descriptive UPI model and compare the estimation accuracy with the accuracy from extant processes. Based on the results, scholars validate the proposed process model, generate further advancements, and derive a normative model which fulfills both the social and technical aspects of software cost estimation. Further, I suggest that scholars and practitioners focus on overcoming challenges during early estimation phases with particularly high uncertainty due to unknown product requirements. The interviews reveal that firms struggle to estimate the effort for new products due to the lack of available analogies and partially move away from estimating effort beyond the next six months due to changing requirements and inaccurate estimates.

4 | Divide and Conquer: Designing Cost Systems for Software Firms

Abstract

This study proposes a cost system design for software firms that can serve multiple purposes in the context of computing software's product costs. We contribute to the sparse literature on cost systems for information goods, which needs to catch up to the ongoing transformation of the world economy into a digital economy. We develop the cost system conceptually, drawing on extant management accounting research and incorporating software's product and process characteristics. Our cost system design integrates three cost-management and modeling systems. First, we adopt a software life-cycle perspective and define the software development project as the cost object to collect, measure, and allocate costs. Second, we propose a regression-based cost model that allows firms to understand the influence of project characteristics on project costs. Third, we introduce a dynamic target costing approach to plan and control costs during software design and development. We structure the cost system based on the divide-and-conquer principle, enabling each system to account for its distinct capabilities while establishing cross-system synergies. The proposed cost system design defines guidelines for firm-specific implementation and allows managers to make informed decisions on software product costing by taking multiple perspectives on software costs.

Authors: Marcus Witter and Michael Blumberg[13]

Status: Working Paper[14]

[13] Author contributions: The authors jointly conducted the literature review and the concept analysis. Marcus Witter developed the research idea and led the writing of the essay. Michael Blumberg led the initial concept adaptions.

[14] This essay was presented at the 2022 Manufacturing and Service Accounting Conference (MSAR) in Pisa, Italy. I thank the participants for their valuable comments and helpful discussions with them.

4.1 Introduction

We propose a framework that suggests guidelines for firms designing cost systems for information goods in general and software products in particular. Designing cost systems for information goods is important as the digital transformation involves organizational changes that generate new paths for value creation by shifting processes, products, and services within and across firms (e.g., Yoo et al., 2010, Vial, 2019, Warner and Wäger, 2019). The rise of digital technologies also entails implications for various business activities, supply chains, and support functions (Vial, 2019, Möller et al., 2020), such as the finance function (Bhimani and Willcocks, 2014). Firms must react to the shift in cost structures and adapt traditional cost-management practices as they move towards digitally enabled businesses (Bhimani and Willcocks, 2014). The effect of technology investments on cost structures is particularly strong for investments in digital technologies, characterized by rising fixed costs that strongly exceed variable cost elements (Afuah and Tucci, 2001, Bhimani and Willcocks, 2014).

Extant cost system designs are production-centric and focus on the commercialization phase of products (Schweitzer et al., 2015). When computing product costs, many companies have traditionally considered software as a byproduct of tangible products, allocating its development costs as a supplement to the product costs without an elaborated allocation scheme (Broy, 2006). From a decision-making perspective, the fixed costs of developing an information good are sunk (Bhargava and Choudhary, 2008). Thus, software firms can hardly apply existing cost systems focusing on the dominant manufacturing and distribution costs typical for industrial products (Bhimani and Bromwich, 2010, Jones and Mendelson, 2011). They need to consider that information goods' costs occur by creating value during the development phase, and the production and distribution of an additional unit incur virtually zero costs (Shapiro and Varian, 1999, Jones and Mendelson, 2011).

That research on cost systems for software development is scarce might reflect that only a few companies apply cost accounting to their software development projects (Astromskis et al., 2014). Astromskis et al. (2014) presume that a low expectation of a payoff from cost accounting activities might be one potential reason for the phenomenon, which results in limited knowledge about project costs, resource-saving potentials, and cost reduction opportunities associated with working habits. Another reason might be that no common framework for designing specialized cost systems in this field exists that is based on proven management accounting theory and which

incorporates theoretical and practical insights about cost planning and project characteristics in software engineering.

Firms need to adapt cost systems for information goods because the requirements for designing cost systems for information goods differ from the requirements for traditional industrial goods. Previous research on cost systems has focused on the manufacturing of industrial goods, for which unit production and distribution costs are often dominant. In comparison, information goods' unit production and distribution costs are marginal when compared to their development costs (Jones and Mendelson, 2011). As soon as an information good has been developed, additional units can be reproduced at virtually zero costs (Shapiro and Varian, 1999).

The optimal design of management accounting systems depends on industry and firm-specific factors (Messner, 2016). Specifically, cost systems must encompass the organization's requirements and business activities. For software development, these characteristics pertain, for instance, to the development process and the cost structure. To address these contingent factors, we analyze software product and project cost characteristics in the development and commercialization phases and elaborate on differences with regard to tangible products.

We design the cost system framework by following the principles of Kaplan (1988) that "no single system can adequately answer the demands made by the diverse functions of cost systems". Firms should rely on different cost systems which meet their products' and processes' needs for cost information to perform different managerial functions sufficiently (Kaplan, 1988). Building upon Kaplan (1988), we transfer the idea of the divide-and-conquer principle for designing efficient algorithms in computer science to the management accounting practice. The divide-and-conquer principle suggests dividing a larger problem into smaller partial problems ("Divide") if a problem cannot be directly solved. These partial problems can then be solved ("Conquer") and combined into an overall solution (Smith, 1985, Cormen et al., 2009). In this study, the overall solution serves three purposes in the context of computing product costs.

We evaluate the characteristics of software products and review extant cost systems with regard to their fit for computing software's product costs. We suggest two cost-management approaches and one cost-modeling system, each of which focuses on a specific task in cost management ("Divide"). First, we suggest a life-cycle costing approach to allow for cost planning and measuring throughout a software project's full economic life cycle. We recommend a project-oriented cost model that measures software development and commercialization costs in each phase of the

product life cycle. Furthermore, the cost model includes recommendations on allocating costs but maintains flexibility in respect of individual choices for firm-specific settings. Second, we propose a regression-based cost model for investigating the relationship between software cost drivers and software costs to enhance the understanding of software costs. Third, we apply a modified target cost-management approach to improve software cost planning and management. Managers can utilize these systems individually or in combination with each other to form a joint cost system ("Conquer"). We suggest integrating the three individual cost-management and modeling systems into a joint cost system, which can be used to manage software project costs from multiple perspectives. Table 4.1 gives an overview of the suggested cost systems.

Our study follows calls for research to tailor management accounting systems to industry-specific needs (Otley, 1980, Messner, 2016). Our paper contributes to the contingency-based management accounting literature, which gives a more general perspective on the design of management accounting systems contingent on a firm's circumstances, e.g., production technology, organizational structure, and environmental aspects (Otley, 1980). In this context, Messner (2016) investigates how an organization's industry context shapes its management accounting practices. He argues that organization-specific practices, e.g., research and development, and industry-specific characteristics, e.g., regulations or best practices, determine the design of management accounting practices. Further, our study relates to the literature stream of strategic management accounting, which investigates the design of management accounting practices subject to strategic aspects (e.g., Miles et al., 1978, Porter, 1980, Lord, 1996, Astromskis et al., 2014, Otley, 2016).

Further, our study contributes to the research on measuring and allocating product and service costs (e.g., Cooper and Kaplan, 1988, Datar and Gupta, 1994, Labro and Vanhoucke, 2007, Balakrishnan et al., 2011, 2012). We contribute to this stream by developing guidelines for designing a cost system for software development projects. Software development projects have rarely been studied in the cost accounting literature. In a notable exception, Fichman and Kemerer (2002) apply activity-based costing (ABC) principles for managing reuse costs in software development.

The remainder of this study is organized as follows. Chapter 4.2 lays the theoretical foundation by reviewing extant cost systems and analyzing the cost structure of software products. Chapter 4.3 outlines the cost-management and modeling systems. Chapter 4.4 synthesizes the framework proposition, and finally, Chapter 4.5 concludes.

Table 4.1: Overview of the suggested cost-management and modeling systems.

Cost-management and modeling system	**Accounting objective**	**Point in time**	**Motivation**
Life-cycle framework - Project cost model (cf. Chapter 4.3.1)	Collect data comprehensively and allocate costs by cause	Ex-post	Cost incurrence and composition of intangibles differ from physical products (Shapiro and Varian, 1999, Jones and Mendelson, 2011). Extant cost systems mainly account for the dominant manufacturing and distribution costs of industrial products (Bhimani and Bromwich, 2010).
Life-cycle framework - Life-cycle costing (cf. Chapter 4.3.1)	Plan, monitor, and document costs by life-cycle phase	Ex-ante & ex-post	Extant cost systems focus on production and commercialization and do not map continuous development practices (Schweitzer et al., 2015).
Inherent cost allocation model (cf. Chapter 4.3.2)	Understand past cost drivers	Ex-post	Managers have difficulties to identify and analyze software cost behavior and cost drivers (Vicinanza et al., 1991).
Dynamic target costing (cf. Chapter 4.3.3)	Plan future costs	Ex-ante	Software firms face uncertainty and constantly changing requirements when developing products (Meso and Jain, 2006, Harris et al., 2009).
Integrated cost system design (cf. Chapter 4.4)	Manage software product costs	Ex-ante & ex-post	Apply the suggested cost-management and modeling systems in a complementary manner (i.e., "divide-and-conquer") (Kaplan, 1988).

Notes: This table includes an overview of the suggested cost-management and modeling systems, including their accounting objective, applied point in time, and the motivation for selecting the respective systems.

4.2 Theory and research question

4.2.1 Review of extant cost systems

Cost systems help controllers to compute product costs (Balakrishnan et al., 2012), make pricing decisions (Banker et al., 1994), and plan product and resource capacity in the long term (Balakrishnan et al., 2011). Historically, cost systems originate from production theory, which argues that the costs of resources deployed in production determine the product price. These are, in particular, materials, human labor, and machinery utilized for production (Cobb and Douglas, 1928). Over the past decades, scholars and practitioners have developed various cost system approaches to respond to changing production environments, individual information needs, and perceived weaknesses of extant systems (Balakrishnan et al., 2012).

Most organizations determine a product's full costs by allocating fixed costs to the variable production costs (Balakrishnan and Sivaramakrishnan, 2002). The variable production costs vary with changes in the quantity of a cost driver, i.e., the output volume. In contrast, fixed costs remain constant even when the quantity of a cost driver changes. However, the main challenges arise when assigning overhead costs to a product because they originate from multiple cost objects and cannot be directly assigned to an individual cost object, such as a product (Friedl et al., 2017). Organizations allocate the fixed overhead costs to support managers in optimizing product and resource planning, enabling targeted organizational behavior and "taxing" undesired behavior (Balakrishnan et al., 2012).

Traditional cost system theory determines costs in two stages, cost data collection and cost assignment to a cost object. Scholars define different cost objects depending on the context, e.g., product, service, project, customer, or department. Scholars differentiate between the direct assignment of traceable costs and the application of an allocation key for indirect costs. Therefore, they collect indirect costs in a cost pool, calculate a rate per unit in relation to a selected allocation base, and allocate the indirect costs to the cost object according to the unit consumption. Thereby, the cause-and-effect relationship indicates the prevailing criterion for conceptualizing an allocation system (Datar et al., 2021). Scholars calculate the costs of a cost object as the sum of direct and indirect costs. The traditional cost system theory benefits from

simple conceptual mechanisms and widespread adoption in practice (Quinn et al., 2017). However, the cost allocation principle is criticized as arbitrary and inferior to recent advancements in certain contexts (Mishra and Vaysman, 2001).

Kilger et al. (2012) proposed the practical marginal costing concept, which is rooted in the traditional cost system theory and was implemented in the ERP software of SAP. The concept focuses on calculating marginal costs, defining cost centers, and planning costs. In particular, they suggested separating variable and fixed costs to calculate products' contribution margin, i.e., the difference between revenues and variable costs (Kilger et al., 2012). Further, Riebel (1979) introduces the relative direct cost and contribution margin calculation approach. He suggests displaying variable and fixed costs separately and treating indirect costs as direct costs, i.e., not to allocate indirect costs to cost objects (Riebel, 1979). Simple contribution margin accounting deducts all fixed costs from the contribution margin. If firms select a more differentiated approach, they can consider fixed costs at different levels, e.g., product, product-group, divisional and firm. The approach indicates the amount each product contributes to cover a firm's fixed costs (Friedl et al., 2017).

Cooper and Kaplan (1988) introduced the ABC concept to better account for the increasing share of indirect costs, and suggested determining costs for a cost object based on its consumption of activities (Noreen, 1991). ABC differs from fundamental cost system theory by defining multiple indirect cost pools associated with activities instead of selecting a single cost pool. The cost pools aggregate homogeneous activities, share a common underlying cost-driving relationship and use an activity-specific allocation base (Datar et al., 2021). The application of ABC potentially improves cost-management decisions by enhancing the measurement accuracy (Noreen, 1991), e.g., leading to new price calculations (Kaplan and Anderson, 2004). ABC also has disadvantages, e.g., the application is subject to several conditions (Noreen, 1991), agency theoretical issues (Mishra and Vaysman, 2001), and practical implementation hurdles due to behavioral and organizational factors (Shields, 1995, Kaplan and Anderson, 2004). Despite being initially designed for application to manufacturing firms, extant literature confirms the concept mechanisms as also being suitable for the field of software development (e.g., Ooi et al., 1998, Raz and Elnathan, 1999, Roztocki, 2001). For example, Fichman and Kemerer (2002) integrate ABC for managing reuse costs in software development. In light of the practical implementation issues of the original ABC approach, Kaplan and Anderson (2004) introduced time-driven ABC. They revisit the ABC approach by introducing a time-driven determination approach for measuring

activity consumption through focusing on feasible productive amounts of available time units. The drawback is that controllers often cannot base the feasible resource capacity on solid empirical foundations and rely on the assumption of homogeneous cost occurrence (Gervais et al., 2010). An alternative for allocating indirect costs displays the German process cost calculation approach, which focuses exclusively on indirect, non-production-related processes (Horváth and Mayer, 1995).

The concept of life-cycle costing extends the scope of cost observation beyond the commercialization phase of a product and incorporates the product design, development, and post-commercialization phases (Riezler, 1996, Schweitzer et al., 2015). Thereby, life-cycle costing observes costs from a plan or as-is perspective (Ewert and Wagenhofer, 2014). The former allows managers to influence design choices, budget planning, and to identify cost drivers. As management decisions in the early design stages of a product predetermine the majority of costs in later stages, managers can utilize cost savings from a life-cycle point of view (Asiedu and Gu, 1998). The latter supports adjusting incurred costs (Ewert and Wagenhofer, 2014). Scholars determine the planned costs based on various methods, e.g., parametric models, analogy-based models, or expert estimations (Asiedu and Gu, 1998) and allocate costs to cost objects from a cross-period perspective rather than a single period perspective to mitigate distorting effects (Riezler, 1996, Ewert and Wagenhofer, 2014).

The concept of target costing intends to manage life-cycle costs during the early product design and development phases (Kato, 1993, Ewert and Ernst, 1999). In this respect, target costing incorporates behavioral control aspects and market information for setting cost objectives (Hiromoto, 1988). Target costing allows engineers and managers to generate new ideas for product development and levers for cost reduction (Tani, 1995). The latter is particularly effective because the major share of life-cycle costs are determined in the development phase but occur during production (Friedl et al., 2017). Scholars define target costs as the maximum costs of a future product while considering profitability targets and customer requirements at the same time (Everaert et al., 2006). In contrast, target costing involves a potential threat to creativity if there is a time pressure factor (Kato, 1993). There is limited evidence for the adoption of target costing in software development, yet Becker and Gaivoronski (2018) demonstrate potential applications, including critical aspects for future research.

4.2.2 Accounting for information goods' cost structure

Software as an information good differs in cost occurrence and composition from an industrial good (Jones and Mendelson, 2011). An information good is a good whose costs arise in the course of development, while the production and distribution of an additional unit of an information good incurs virtually zero additional costs (Shapiro and Varian, 1999, Jones and Mendelson, 2011). In contrast, unit production and distribution costs dominate the costs of industrial products (Jones and Mendelson, 2011). For example, Jones and Mendelson (2011) model firms producing information goods with zero unit costs and positive development costs that increase with product quality, and firms producing industrial goods with zero development costs and positive unit costs that increase with product quality.

Alongside the difference in cost occurrence and composition, the focal point of value creation differs between information and industrial goods. For example, software development involves a creative attempt that requires flexibility under uncertain conditions (Harris et al., 2009). In particular, agile methodologies emphasize the continuous reactions to changing environmental conditions (Meso and Jain, 2006). Information goods lack a physically observable manufacturing process, so managers cannot observe how development effort relates to the costs associated with an individual product unit. Thus, they face the challenge of effectively allocating costs to a software product.

The different characteristics of information goods and traditional goods also affect managers' decision making. Management accounting scholars and practitioners target a rational view of costs, which is not confounded by past decisions and allows managers to focus on costs that can be influenced in the future. Cost-management approaches for industrial goods differentiate between development and manufacturing effort, with the focus on manufacturing costs. Taking the general perspective of a product as a cost object, the production costs of industrial goods are decision-relevant because they can be influenced in the future. In contrast, software development costs are considered as a past one-time investment that cannot be avoided ex-post. Thus, initial software development costs are considered sunk (Bhargava and Choudhary, 2008, Friedl et al., 2017).

Further, information goods do not incur direct input materials and manufacturing costs. Thus, the perspective of a product as a cost object is an insufficient basis for decision making. As the development effort is central to value creation, the crucial phase of cost occurrence is not covered

from a product cost object and post-development perspective. Figure A.10 in the appendix illustrates typical cost structure differences between industrial goods and information goods. However, when taking the perspective of a project as a cost object for tangible goods, we expect to observe similarities concerning the cost structures. Further, target costing for physical products also aims to bring development into focus by influencing costs during the product design and development phases through behavioral control and market information for defining cost objectives (Ewert and Wagenhofer, 2014).

To account for the different characteristics under the perspective of a product cost object, Schweitzer et al. (2015) suggest enlarging the cost-management perspective towards the earliest and most direct cost originator where value creation takes place, i.e., the development phase. The total software product costs can be calculated by adding the costs incurred in the software development project and the software product costs incurred after the development phase. Conceptually, the software project costs in the development phase consist of fixed costs, which incur only once over the project duration, and variable costs, which depend on the project size and duration (Maltzman and Epstein, 2013, Chellappa and Mehra, 2018). Based on this approach, we propose the following conceptual cost Equation 4.1:

$$\begin{aligned} &\text{Total software product costs} \\ &= \text{Software project costs}_{\text{Development phase}} \\ &+ \text{Software product costs}_{\text{Commercialization phase}} \end{aligned} \tag{4.1}$$

Practitioners can manage industrial goods appropriately from a product perspective while accounting for the commercialization phase using the traditional cost system theory. However, managers face challenges in applying these systems developed for industrial goods to information goods such as software. Due to their intangible nature, information goods require a distinct consideration from a development project perspective, which disentangles value creation, specifically production, from the creation of an additional product unit and thus shifts the cost occurrence towards the development phase.[15] Drawing on the review of extant cost-management approaches and the differences between industrial goods and information goods, we derive the research question of how controllers can design cost systems for software firms to encompass software's product and process peculiarities.

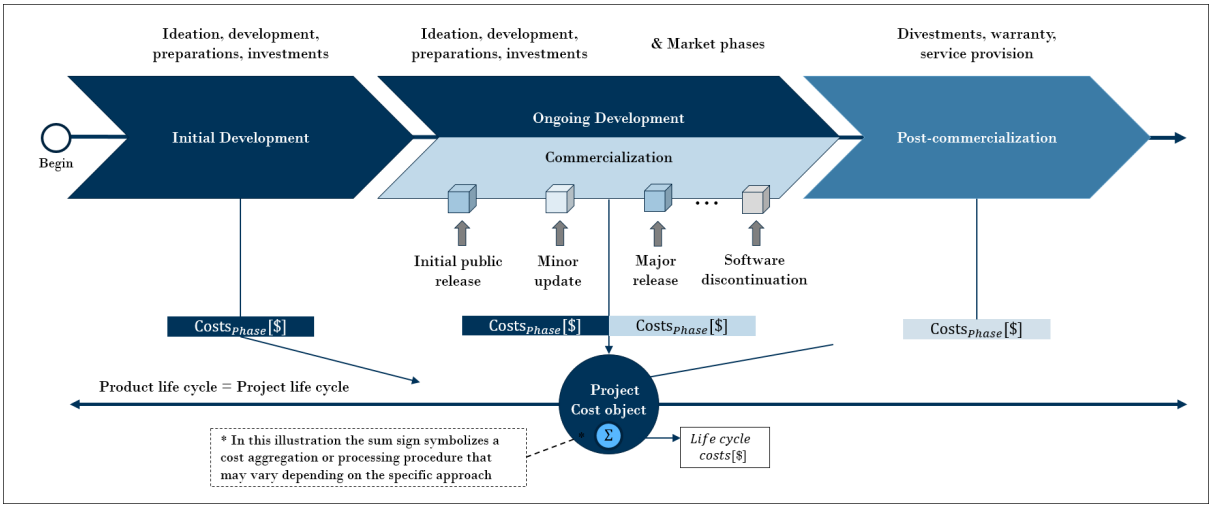
[15]Refer to Boehm and Papaccio (1988) for an overview of the software development value chain.

4.3 Proposed software cost-management and modeling systems

4.3.1 Life-cycle framework for software projects

We propose that software firms adopt a life-cycle costing perspective, which dissolves the separation between the development and commercialization phases and allows for the monitoring and planning of software costs for a project's full economic life cycle. Thus, managers should consider total software costs by extending their cost perspective beyond the core development life cycle (Bradley and Dawson, 1999, Zarnekow and Brenner, 2005). We differentiate between the development of the initial software release, the continuously ongoing development, the commercialization, and the post-commercialization phase. Further, we suggest a parallelized life-cycle understanding to support the practice of continuously ongoing development of software in parallel to the commercialization phase. This distinction allows managers to understand the occurrence of development costs better while obtaining a more comprehensive overview for decision making. However, different life-cycle cost considerations must be taken into account depending on the software's purpose. Taking license or software-as-a-service solutions as an example, we depict the idea of continuously ongoing development and commercialization and observe them partly in parallel. According to this approach, multiple minor updates and major releases might follow after the initial public release. Figure 4.1 illustrates the general concept of a full software project life-cycle costing perspective for commercial software.

FIGURE 4.1: Conceptual illustration of life-cycle costing in the context of software projects.



Notes: The illustration is based on the general life-cycle costing illustration by Riezler (1996) and adjusted to the context of software projects. In addition, it integrates insights from the life-cycle illustration for application systems by Zarnekow and Brenner (2005). Note that the post-commercialization is not depicted for simplification purposes in the proposed conceptual cost Equation 4.1

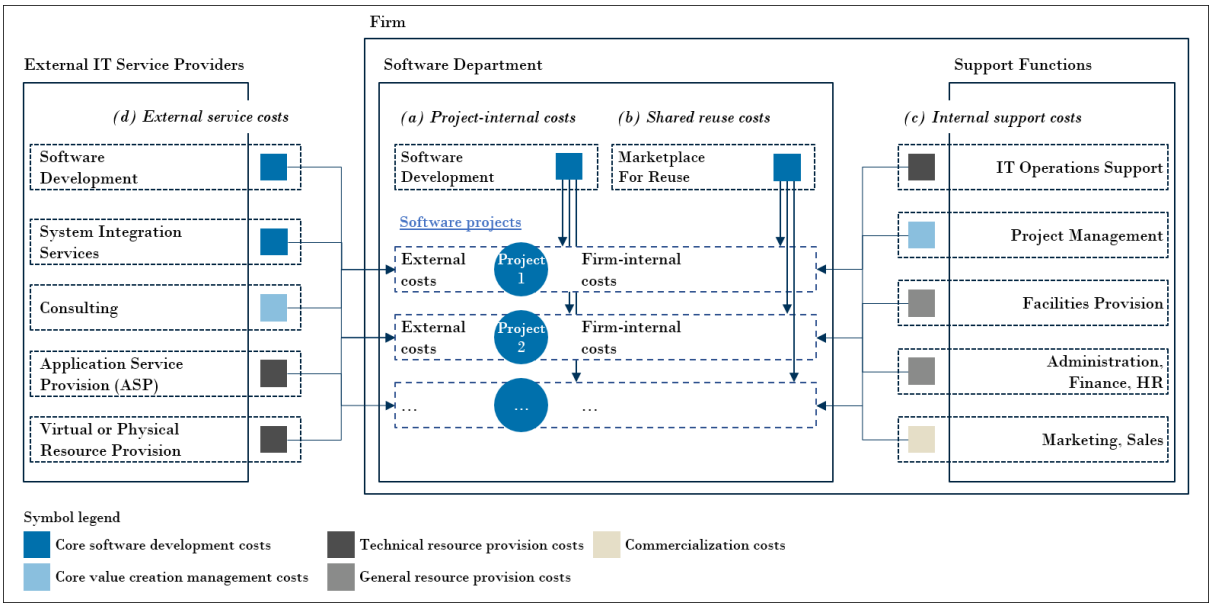
As a supplement to the model depicted in Figure 4.1, we suggest a project-based cost model for collecting and allocating software project costs independent of the life-cycle phases. The framework incorporates the peculiarities of cost occurrence and integrates costs from various internal and external sources from a macro perspective. Therefore, we reformulate Equation 4.1, which is based on a heterogeneous cost object definition, to Equation 4.2 by focusing on the definition of the project as a unique cost object and comprehensively taking all cost sources into account.

$$\begin{aligned} &\text{Total software project costs} \\ &= \text{(a) Project—internal costs} \\ &+ \text{(b) Shared reuse costs} \\ &+ \text{(c) Firm—internal support costs} \\ &+ \text{(d) External service provider costs} \end{aligned} \tag{4.2}$$

The total project costs comprise (a) the project—internal costs which incur for activities within software development, (b) costs for reuse of software components created for previous or other ongoing projects, (c) internal support costs, and (d) external service provider costs, which can

be assigned to the project based on the fees paid to the contractor.[16] Typical cost collection and allocation mechanisms segmented by cost source for a software firm collaborating with an external service provider are presented in Figure 4.2.

FIGURE 4.2: Typical cost collection and allocation mechanisms segmented by cost source for a software firm.



Notes: Typical illustration that can be adjusted dependent on the specific setting. Refer to Seltsikas and Currie (2002) for an evaluation of the business model Application Service Provision.

We structure the cost allocation principles for a phase-specific time period based on the outlined cost sources within the organization. We propose assigning (a) project—internal costs by measuring the time used for projects and multiplying the respective time units by the associated hourly or daily wages. Firms can measure time by requesting employees to enter their times worked on a project in a simple spreadsheet or database or by applying firm-internal automatic time capturing tools (Astromskis et al., 2014). Fichman and Kemerer (2002) suggest allocating (b) shared reuse costs according to the ABC principles, i.e., in relation to the actual degree of reuse in the respective projects and the period in question (see Equation 4.3). Subsequently, managers can multiply this rate by the quantity of reused components per project, adjustable by a size or complexity weight, to derive the total activity-based costs of reuse allocated to a project (Fichman and Kemerer, 2002).

[16] For example, refer to Nie and Hammouda (2017) on the practice of strategic software development outsourcing or see Mishra and Mahanty (2016) on the effect of outsourcing on project costs and other performance measures.

$$\text{Reuse component unit costs} = \frac{\text{Total reuse component development costs}}{\text{Total volume of reused components used by all products}} \tag{4.3}$$

Concerning (c), we recommend considering firm-specific allocation preferences for allocating internal support costs. If managers prioritize simplicity over accuracy, they can, for example, select a simple cost allocation scheme by calculating costs per unit of a selected reference basis. If they want to increase accuracy, they can apply more advanced allocation approaches to reflect the cause-and-effect relationship between costs and project cost drivers, e.g., ABC. Concerning (d), we assign external service costs to the project based on the service provider's service fee. In the case of shared service use, managers can allocate costs according to different procedures, e.g., by equally splitting costs among consuming projects, leveraging report documents of the service provider as basis for assigning costs, or creating a "marketplace for externally sourced solutions" to distribute external costs among projects.[17]

We recommend considering the project-oriented cost model as a blueprint for a context-specific cost system that can be built from scratch or implemented based on existing cost systems. In particular, existing concepts for cost allocation can be realigned to a project cost object context and incorporated into a full life-cycle perspective. However, organizations must adapt the cost model for their purpose, e.g., observe additional cost sources to enhance cost-management insights and adapt the cost allocation principles dependent on their specific setting.

We define conceptual principles for documenting actual life-cycle costs transparently (Zarnekow and Brenner, 2005). Here, life-cycle costing does not focus on a certain period of observation but serves as a cross-period technique to establish a cross-phase analytical and planning perspective on costs (Coenenberg et al., 1997, Ewert and Wagenhofer, 2014). This provides an opportunity for bridging the gap between different micro-structures that inherently characterize development, commercialization, and post-commercialization phases. Wynn et al. (2014) suggest one approach for automatically capturing process costs. They suggest extending the information from Business Process Management Systems with cost information to enhance the scope of data for management accounting. They use log data that record the execution of events and then combine these logs with associated cost data, termed cost annotation. Based on cost-annotated event logs, managers can quickly obtain reports for analyzing costs (Wynn et al., 2014).

[17] The calculation procedure for the "marketplace for externally sourced solutions" can been established by generalizing the ABC calculation proposal (Fichman and Kemerer, 2002).

Firms track the value creating activities in the initial development phases to a lesser extent and hence have difficulties generating informative cost insights and require more data on the specific activities performed by developers (Astromskis et al., 2014). However, a close-meshed self-logging of process executions would potentially intrude in the creative environment of developers. Astromskis et al. (2014) propose an automated non-invasive method for measuring the time spent on specific activities in software development. To do so, firms collect cost items that represent specific elements, such as source code methods, websites, or collaborators, that interact within software applications utilized in the development project. These cost items are recorded with the corresponding time consumption (Astromskis et al., 2014).

We suggest implementing a phase-sensitive benchmarking of actual costs relative to planned costs after establishing a transparent measurement system for costs incurred over the full economic software life cycle. Depending on the phase, managers can apply different approaches for cost planning. In the development phase, they could draw upon classical model-based or expert-based estimation methods from the software engineering domain (e.g., Boehm, 1981, Jørgensen and Shepperd, 2007, Idri et al., 2015, Eduardo Carbonera et al., 2020). Alternatively, they could apply more general estimation methods rooted in business management knowledge in the commercialization and post-commercialization phases. Here, insights from research can serve as a potential means for enhancing and automating cost estimations.

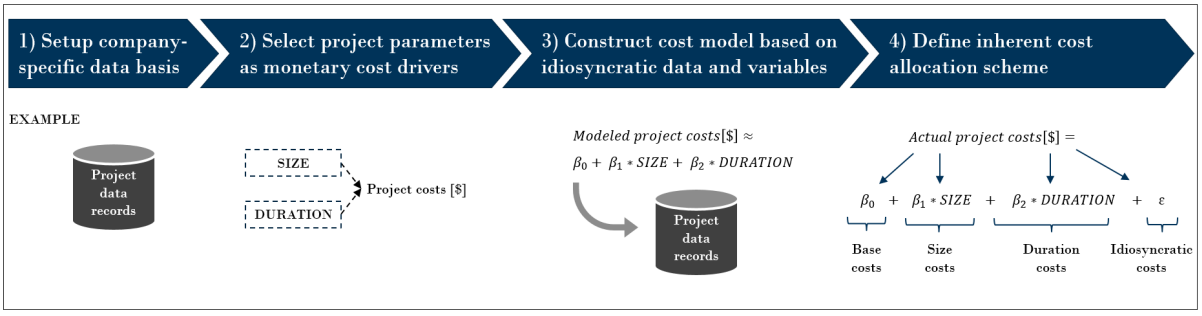
In light of the demand for an interlinked cost-management approach across stages (Zarnekow and Brenner, 2005), we recommend to introduce cross-phase sensitivity analyses. Managers should evaluate the impact of decisions with regard to software characteristics or organizational project parameters on the costs incurred in later stages. However, they should also conduct scenario analyses in the opposite direction that make it possible to examine how decisions in the area of commercialization or operations might affect development costs.

4.3.2 Inherent cost allocation model

We suggest a cost-modeling system that examines how underlying project characteristics influence project costs from an ex-post perspective. We determine the cost-modeling system as an inherent cost allocation by allocating the total project costs to inherent project parameters, e.g., functional size, and duration, as cost drivers virtually.

We suggest a four-step approach as a guideline for initializing the inherent cost allocation system. We define the software project costs as the dependent variable and the underlying software project characteristics as the independent variables.[18] We define inherent virtually allocated costs by combining the intercept, independent variables, parameters, and residual. Thus, actual project costs represent the sum of allocated inherent costs. Figure 4.3 illustrates the four-step approach schematically for a typical software project.

FIGURE 4.3: Inherent cost allocation based on the inherent cost drivers of a software project.



First, we recommend that companies set up a firm-specific database for project cost data and parameters because of opaque cost structures in the software industry (Huijgens et al., 2017). Companies should refrain from drawing on cross-firm project databases in which the relevant dependent variable portrays costs because this can confound the results due to different organization-specific cost-influencing factors. Furthermore, not many large-scale company project databases, including cost data, are available and accessible (Huijgens et al., 2017).

Second, we suggest selecting firm-specific project parameters as cost drivers because the relevance of variables for effort estimation depends on the specific context (Jørgensen, 2014). On the one hand, analogy-based effort estimation methods suggest selecting individual variables dependent on the specific setting. The variable selection can be performed manually or supported by automation (Shepperd and Schofield, 1997). Based on the insights of analogy-based effort estimation, managers can select features based on a fuzzy clustering technique as proposed by Idri et al. (2015). On the other hand, managers can consider the empirical results of scholars in the field of effort or cost estimation as a starting point for supporting the firm-specific selection of relevant variables. For instance, Huijgens et al. (2017) perform an analysis, with costs

[18] We observe project costs from an overall monetary perspective and do not limit our focus to effort as a proxy for development costs (Huijgens et al., 2017).

as the dependent variable, on software project data from the Evidence-Based Software Portfolio Management (EBSPM) and International Software Benchmark Standards Group (ISBSG) repositories.[19] The authors emphasize that different variables are more relevant than others depending on the data set. However, they report size and duration (size and effort) to be of primary relevance in the observed EBSPM (ISBSG) data excerpts (Huijgens et al., 2017). In general, the effort is the central cost driver within software development projects (Jørgensen and Shepperd, 2007). Effort describes the working hours spent on the project and drives the wages paid to the development team. The wages are recorded as direct variable costs, which are positively related to team size and project duration.

Third, we propose that companies develop a regression model based on their idiosyncratic data and variables. Therefore, we establish the general structure of the inherent cost allocation model as depicted in Equation 4.4, introducing a formal perspective on project costs.

$$\begin{aligned} &\text{Total software project costs} \\ &= \beta_0 \\ &+ \sum \beta_i \times \mathit{Project\ characteristic}_i \\ &+ \sum \beta_{ij} \times \mathit{Project\ characteristic}_i \times \mathit{Project\ characteristic}_j \\ &+ \epsilon \end{aligned} \tag{4.4}$$

Fourth, we suggest defining a firm-specific inherent cost allocation scheme. In our suggested approach, we consider a linear model. The total project costs are subdivided into their virtual cost components according to the regression analysis, a process we refer to as inherent cost allocation.[20] To do so, the allocation scheme can be directly derived from the cost formula (Gordon, 1974, Wright, 1983). The intercept β_0 can be considered as fixed costs that occur independently of the project's specific characteristics. The values of the products of independent variables $\mathit{Project\ characteristic}_i$ and the corresponding estimation parameters β_i are defined as inherent cost components associated with each variable as an inherent cost driver. The value of residual ε represents idiosyncratic project costs that cannot be explained by the preceding core associations between cost components and project costs.

[19] Refer to Rastogi et al. (2014) for an overview of data sets used.

[20] Estimated parameter values obtained from regression analysis can have a positive or negative algebraic sign. Therefore, virtual cost component values carry positive or negative algebraic signs. Cost components with a positive value can be considered cost contributors, and components with a negative value are cost alleviators.

4.3.3 Dynamic target costing

We introduce the concept of dynamic target costing that allows managers to plan and steer development costs with an integrated view of software development's technical and economic perspectives. Our concept aims to realign the traditional target costing approach with the virtual nature of software products and the need of development teams to react to dynamic market and technology changes that entail constantly changing requirements.

We recommend shifting the target costing focus towards the development costs and defining target costs aggregated for all product units and not for a single unit. In traditional target costing, managers derive the target costs from the market analysis on a per-unit basis and then break these costs down so that they apply to the physical components of the product. However, the traditional approach does not hold for intangible products as the development phase displays the focal point of value creation and the costs for producing a virtual software unit are close to zero. In this context, we aim to dissolve the relatively static definition of functions and associated components. Referring to Fichman and Kemerer (2002) demonstrating ABC for component-based software engineering, we define components as the reference point to cost management. In software engineering, components are not part of a physically defined product but rather flexible and coherent parts of one or more software versions, which comprise multiple virtual components.[21] One virtual component can be part of different versions, and one version usually consists of multiple components. In general, firms create a high-end version of a software product as a flagship product and then subsequently reduce features to create degraded versions of the product (Wei and Nault, 2014). Thus, the dynamic target costing approach focuses on aggregated target costs for multiple product versions that need to be fulfilled by the associated components.

We adhere to the market-driven approach of target costing by identifying customer needs and selecting functions that are required to meet these needs. Firms can apply different means to extract market information, e.g., conduct market assessments, customer survey, focus groups, or interviews with key customers (Becker and Gaivoronski, 2018). However, we propose to dissolve the static link between customer preferences and component cost weights. In traditional target costing, engineers derive the component cost weights based on market preferences of functions

[21] Versioning is representative of a quality-induced product-price strategy of firms in the software industry. Firms offer multiple versions of software at different prices in order to enable different types of users to self-select their best feature-price pair (Shapiro and Varian, 1999, Shivendu and Zhang, 2015). For example, Microsoft offers its operating system Windows 10 and productivity software Office in more than five versions.

and the contribution of components to these functions (Friedl et al., 2017). The sum of the costs from the underlying components indicates the costs of the function. In software engineering, customer preferences cannot be directly linked to the importance of functions and the associated development costs as the application and solution domain are both volatile.[22] For example, non-functional requirements such as security functions can be costly to develop but are not directly perceived by customers. The gap between the application and solution domain, as well as the necessity for continuous flexible adjustments (Brügge and Dutoit, 2014) pose a challenge concerning the viability of traditional target costing. The dynamic target costing concept must allow managers to flexibly add, delete, or modify versions, components, or functions and their relationships without performing time-consuming recalculations.

We suggest initiating the dynamic target costing process after thoroughly analyzing the functional and non-functional requirements and translating these to system components.[23] Managers should only implement dynamic target costing activities after defining core characteristics of the software product and, thus, focus on activities in the subsequent phases of the development life cycle, e.g., the object design, implementation, and testing phase. This staggered approach based on in-depth requirements and system design analysis is necessary because software engineers require specific implementation knowledge and an increased technical understanding to define virtual software components. Subsequently, managers can use the defined functions and components to measure costs. In contrast, hardware engineers can initiate the target costing procedure early on because functions can be relatively easily linked to physical hardware components.

We recommend defining target costs not only at the product level but also for selected components and functions. Managers should compare current component or function costs with their target costs to gain an insights into whether a cost leeway exists or costs are overrun. This granular analysis is necessary because component costs of software products are progressive, while the costs of the development effort already made are sunk, so no ex-post component cost reduction is possible. Managers should only calculate drifting costs in the sense of traditional target costing at an aggregated product level, comparing overall target costs with the estimated costs for the total development effort. However, managers should also ensure that the creative environment of developers is not harmed by evaluating cost coverage at the component and function level. They might foster creativity by creating transparency about the economic effect

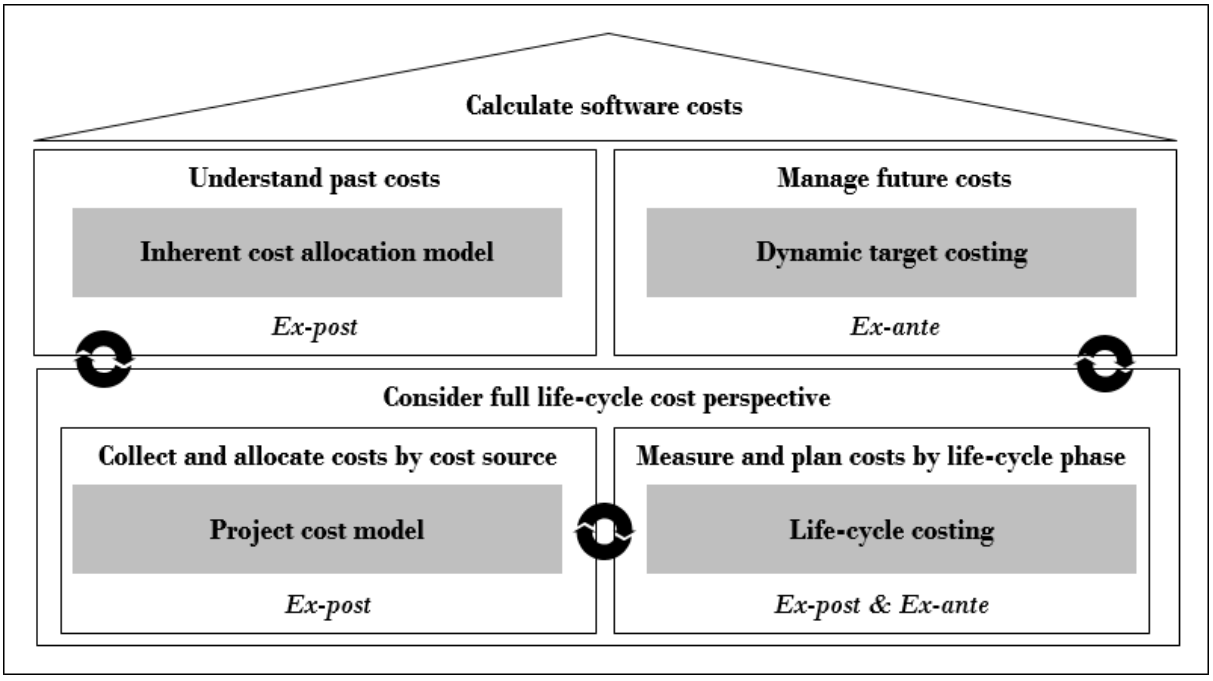
[22] The application domain relates to the user perspective of the targeted functions, whereas the solution domain describes aspects of system design and implementation (Brügge and Dutoit, 2014).
[23] We refer to Brügge and Dutoit (2014) for a description of the software development life-cycle phases.

of their improvement activities and outline incentives that enable developers to participate in the economic success of the function development. We derive and modify the dynamic target costing process from the traditional target costing approach (Friedl et al., 2017). Figure A.11 in the appendix illustrates the process and Figure A.12 in the appendix describes the procedural steps in detail.

4.4 Integration of the cost-management and modeling systems

We propose integrating the cost-management and modeling systems into a common framework by demonstrating how they can be applied in a complementary manner. Therefore, we establish a balance between using the systems individually and interlinking them to achieve synergies between them (Cooper and Kaplan, 1998). Each system is assigned specific purposes and capabilities to support controllers with a distinct cost perspective. Both individual software departments and software firms can apply the framework. Figure 4.4 gives an overview of the suggested cost-management and modeling systems and their interlinking points.

Figure 4.4: Systematic framework outline.



Managers can apply the inherent cost allocation model as a complementary technique to the life-cycle cost-management and project-oriented cost-modeling system. We conceptualize the project-oriented cost model as a system that allocates costs to the project cost object. These costs can be an ingredient in collecting project data and recording total costs associated with characteristic project parameters. Managers can use this data to set up a regression analysis establishing a cost formula for inherent cost allocation. Thus, managers obtain an analytical cost-management perspective by supplementing the project-oriented cost model with an econometric analysis and allocating total project costs to project parameters. The life-cycle framework incorporates planning life-cycle costs from an ex-ante perspective based on scenarios and sensitivity analysis. Managers can derive a general cost formula based on historical projects, with project characteristics as input variables, and utilize the formula to initially estimate future project costs. Hence, managers do not use the cost formula for inherent cost allocation but for future cost estimation. Since the accuracy of estimates obtained from the formula might be limited, combining the results with the results of other estimation methods, e.g., expert-based, is crucial (Jørgensen, 2014).

The dynamic target costing approach can be combined with the life-cycle framework. Knauer and Möslang (2018) report that the use of target costing and life-cycle costing is positively related. Life-cycle costing allows the measuring and planning of costs over the full economic life cycle. However, it does not comprise an approach for directly managing the development costs from a technical and economic perspective in a target-oriented way. Managers can fulfill this task by applying the dynamic target costing concept, which enables cost steering of aggregated component costs side by side with cost targets associated with commercially available software versions. Managers can define these cost targets with a life-cycle costing perspective in an interlinked way, assign the targets to specific versions and monitor their fulfillment in a dynamic target costing setting. In doing so, managers must jointly utilize and synchronize the cost measurement systems applied in dynamic target costing and life-cycle costing to ensure consistency among the approaches.

We suggest establishing a holistic cost-management perspective for software development projects by balancing the distinct and interlinked usage of the proposed systems. Managers must consider that inherent capabilities and limitations of scope characterize each system. Thus, individual systems cannot serve as a "one-fits-all" solution. Therefore it is crucial to consider, evaluate,

and connect insights from multiple perspectives and, if necessary, perform adjustments for interlinking systems.

4.5 Discussion and conclusion

The low dissemination of cost systems in software firms arises for various reasons. Established techniques for designing cost systems might not be directly applicable to software applications, raising the need for adjusting extant systems by considering context-specific peculiarities. In this context, Astromskis et al. (2014) observe a lack of a common framework that defines principles for a context-specific system design, thus hindering the implementation. Further, development teams are reluctant to apply cost accounting techniques in development teams, potentially because the benefit obtained from these techniques is expected to be low (Astromskis et al., 2014). Compared to tangible products, the atypical composition of costs and shift in value creation towards the development phase of software can be an additional impediment to introducing existing cost systems. As outlined by Astromskis et al. (2014), this can result in a lack of cost awareness and associated adjustment potentials.

This study proposes a cost system design for software firms. We define guidelines for designing a customized cost system that fulfills the context and firm-specific peculiarities in software engineering. We build the cost system on three cost-management and modeling systems ("Divide"), each solution containing specific purposes and capabilities (Kaplan, 1988). We utilize synergies between the solutions and suggest a way to interlink them into a joint cost system design framework ("Conquer"). Organizations can implement and apply the cost system as suggested or adjust it to their specific organizational requirements (Cooper and Kaplan, 1998).

The proposed cost system design inherits multiple perspectives on costs to enable informed cost management. First, we utilize the life-cycle costing concept for our cost system design. To assist in this, firms can apply the fundamental ideas of collecting and assigning costs to a cost object from traditional cost system theory and the ABC approach without changing the underlying mechanisms but by adjusting them to the characteristics of software development. Second, we follow Datar et al. (2021) and define a cause-and-effect relationship between costs and project cost drivers by implementing a modified empirical indirect cost allocation model, enabling an analytical view of software project costs. Third, we re-define the underlying mechanics of target costing for tangible products to account for software components' intangibility and

the development's creative nature. However, we recommend that organizations maintain the intuition of cost monitoring and steering.

We extend the cost system design literature by specifically accommodating the peculiarities in software engineering. We do this by providing insights into how firms can systematically manage and model software development costs. We add to Otley (1980), who establishes a contingency-based view on cost management sensitive to firm-specific peculiarities, and Messner (2016), who suggests incorporating industry-specific facets in management accounting research.

We consider our framework as a starting point for defining firm-specific cost system design solutions, but neither as a finite nor definitive solution to software cost management and models in organizations. Firms can apply all cost-management and modeling systems or select and modify individual systems based on their specific costing challenges. To do so, firms must consider two findings. First, organizations must recognize the intangible nature of software products and the associated shifted focus of cost incurrence to the development phase. Second, organizations should shift the focus from the software product to the software project as a cost object and keep the project-oriented differentiation between fixed and variable costs in mind. For a practical realization, managers can draw on already implemented solutions and modify these to accommodate the context-specific software development requirements. However, we recommend evaluating the existing and proposed cost-management and modeling solutions with the aim of achieving a sound cost representation in the organizational setting.

Our results are subject to limitations. The cost-management and modeling systems inherit weaknesses concerning scope, practicability, and accuracy of cost measurement. While we focus on developing a context-specific cost system design to handle the peculiarities of software products and developments, the pre-existing limitations of the cost system concepts are inherent due to the assumptions of cost systems as simplified models of reality.

Further, we lack evidence that our suggested framework improves managers' decision making in calculating software's product costs. Therefore, we suggest that scholars and practitioners validate our proposed framework in a case study and derive further advancements in cost system design in software development by considering current development practices. We propose that scholars follow an integrated approach by combining the academic and practical perspectives in five steps. First, we suggest extending our conceptual proposals by expert-based assessments of current costing practices in software development. To do so, scholars can conduct a focus group

workshop. Second, scholars and practitioners should add industry-specific requirements for cost systems based on the assessment results. Third, we recommend designing a newly adjusted cost system by focusing on one cost-management or modeling approach to generate lead user stories and incorporate feedback and validation loops. Fourth, scholars and practitioners can implement prototype modules for software cost systems and conduct test runs with lead users or collaborating clients while continuously monitoring the performance, recognizing feedback, and improving the cost system. Fifth, practitioners can realize commercial software costing modules as a stand-alone product or integrate these into existing software applications for effective cost accounting in software departments or firms.

5 Conclusion

5.1 Summary of main results

The digital transformation impacts various industries and shifts firms' products and services towards new value-producing opportunities for digitally enabled businesses (e.g., Yoo et al., 2010, Vial, 2019, Warner and Wäger, 2019). In this dissertation, I focus on the implications of digital transformation on firms' decision-making and cost-management practices. I investigate how firms can react to the increasing volume of available information originated through new technological developments, such as AI and big data, and organizational cost mix changes. I address these issues with three essays, which aim at finding solutions to improve decision making and adapt traditional cost-management practices to the changing requirements of software product costing and cost estimation. In doing so, I apply three research methodologies: a laboratory experiment (cf. **Essay I** in Chapter 2), a qualitative multiple-case study (cf. **Essay II** in Chapter 3), and a conceptual analysis (cf. **Essay III** in Chapter 4). Based on my findings, I conclude that information overload impairs decision quality. Firms must adapt their decision-making frameworks to ensure that managers make optimal decisions, e.g., by utilizing decision aids or introducing accountability as a control mechanism. Further, I develop a comprehensive process model which outlines how firms estimate their software project costs and derive propositions for process improvements based on identified estimation challenges. Finally, I propose a solution for how software-producing firms can design their cost systems to calculate current, evaluate past, and manage future software product costs from multiple perspectives. Hence, this dissertation contributes to the theoretical and practical debates on decision making and costing in the digital age.

In **Essay I** (cf. Chapter 2), I conduct a laboratory experiment with nearly 200 participants and investigate how information load and accountability affect decision-making quality. I contribute

to the information load literature that adding information cues impairs decision-making quality in an investment decision task based on BSC data. This result is consistent with previous findings from information load literature, which examine the effects of information overload on decision quality in other settings (e.g., Iselin, 1988, Chewning and Harrell, 1990, Roetzel, 2019). The results suggest that firms must prevent information overload when providing information to managers and adapt their decision-making frameworks and reports to foster a more effective and efficient decision making. Chewning and Harrell (1990) suggest preprocessing the available information to a considerable amount or equipping managers with decision models or causal chains to reduce cognitive complexity and shift the managers' focus to the relevant information cues (e.g., Humphreys et al., 2016, Dalla Via et al., 2019). Further, I pick up recent calls for research (Gupta et al., 2018, Brynjolfsson et al., 2021) and introduce accountability as a countermeasure to mitigate the negative effects of information overload on decision making (e.g., Iselin, 1988, Chewning and Harrell, 1990, Roetzel, 2019). I find that accountability improves the frequency of making an optimal decision under information overload. However, I do not find a main effect of accountability, and there is no significant interaction between information load and accountability on decision quality. The missing main effect of accountability could indicate a dominant dilution effect, increasing participants' attention to irrelevant information (e.g., Siegel-Jacobs and Yates, 1996, Tetlock and Boettger, 1989).

In **Essay II** (cf. Chapter 3), I address the research question of how firms estimate their software costs. Based on a multiple-case study design, I develop a process model that structures the process on three levels: project stages, activities, and activity features. The first level summarizes the generic three project stages: Understand, Plan, and Implement. The second level illustrates ten activities: understanding the project environment and project requirements during the first stage, planning the estimation strategy, data strategy, estimation method, estimation control, and client and supplier contract during the second stage, and, finally, implementing the project factors, team factors, and rules during the third stage. The third level outlines the features of each activity depending on the firm and project-specific environment. In doing so, I describe why firms select different activity features and how they proceed in their given setting. In addition, I triangulate the identified estimation challenges with the literature on software cost estimation and theories in information systems to derive suggestions that aim at improving the estimation process by enhancing the consistency of estimation results. The findings address the lack of developing theories in software engineering (e.g., Hannay et al., 2007, Jørgensen and Shepperd, 2007, Dybå and Dingsøyr, 2008, Usman et al., 2015, Idri et al., 2015, Eduardo Carbonera et al.,

2020). I add to the literature on software cost estimation by reflecting on the current state of practice and enlarging the perspective on software cost estimation beyond developing and testing individual estimation methods (Jørgensen and Shepperd, 2007, Eduardo Carbonera et al., 2020). Further, I complement the literature on software cost estimation processes by including peculiarities of various industries and firms' perspectives to understand the different facets of software cost estimation (Boehm and Papaccio, 1988, Edwards and Moores, 1994, Jørgensen and Molokken, 2003, Usman et al., 2018). Firms can compare their current estimation process with the provided UPI model to adapt or extend their process.

In **Essay III** (cf. Chapter 4), we conceptually propose a cost system design for software firms. In doing so, we take multiple perspectives on software product costing and integrate three cost-management and modeling approaches that allow firms to calculate current, examine past, and influence future software product costs. Our cost system design is based on the "divide-and-conquer" principle. Following Kaplan (1988), each of the three cost-management and modeling approaches fulfills a specific purpose ("Divide"). Then, we utilize synergies between the approaches by interlinking them to a joint cost system framework ("Conquer"). The first cost-management approach describes a life-cycle costing concept for software firms and exemplifies principles for collecting and allocating software costs over the entire economic life cycle. The second cost-modeling approach illustrates an inherent cost allocation model to define a cause-and-effect relationship between costs and project cost drivers. The third cost-management approach outlines an adapted target costing approach to account for software's intangibility and the development process' creative nature. We follow the calls for research to define principles for developing cost systems for software firms (Astromskis et al., 2014). Further, we add to the cost system design literature by defining guidelines on how firms can compute their software product costs (Otley, 1980, Messner, 2016).

5.2 Limitations

The conducted experiment in **Essay I** has limitations. First, my results might not generalize to other information load settings. I manipulate information overload by using a BSC with 16 measures (Ding and Beaulieu, 2011), representing an abstraction of the increasing volume of information induced by AI and big data and reducing real-world complexity. Second, the experimental setting includes only a single-decision rule by solely focusing on maximizing the

net profit, which facilitates the analysis and simplifies reality (Dalla Via et al., 2019). In practice, managers can face multiple decision rules and must weigh the given measures against each other.

The developed process model of **Essay II** also has limitations. I focus on depicting the process of estimating software costs, identifying estimation challenges, and deriving propositions for which a qualitative approach is suitable and enables scholars to gain insights. However, I do not analyze whether and how defining an initial estimate in early project phases influences managers in designing the ongoing development process, i.e., whether an anchoring effect occurs. Further, despite deriving propositions for managers to improve their estimation process, I do not suggest a normative cost estimation process that scholars or practitioners can apply to achieve better results. Both shortcomings can also be related to the small sample size, which does not allow a quantitative comparison of different project stages, activities, and activity features and their effect on estimation accuracy. Moreover, the findings are limited to individual projects. I cannot assume that the results can be generalized to other software development practices, such as continuous developments or research projects.

The results of **Essay III** are subject to limitations. We consider our cost system design as a starting point for defining firm-specific solutions, but neither as a finite nor definitive solution to software product costing in firms. We focus on conceptually deriving the cost system design by taking multiple perspectives and focusing on three objectives: calculate current, evaluate past, and manage future software product costs. However, our main limitation is that we did not validate our suggested cost system design and, thus, lack evidence that our proposed cost system design increases the accuracy of software product costing. In this context, we cannot assume how transferable and practical the guidelines are for developing the suggested cost system design for a broad range of software-producing firms in different industries. Further, the limitations of the suggested cost-management and modeling systems have weaknesses as they rely on simplified assumptions of reality.

The dissertation also has limitations as a comprehensive work itself. The dissertation tackles three issues in costing and decision making of firms affected by the digital transformation. To answer the different research questions, I apply different methodologies and present evidence from multiple perspectives. However, I do not use the results of one essay as an input variable for the other essays. First, I lack to evaluate which process activities of estimating software costs are prone to overloading managers with information. The same applies to the suggested cost system design as I do not analyze which situations of the different cost-management or modeling

systems accommodate the risk of information overload. Future researchers could analyze which decision-making frameworks, reports, or financial ratios should be adapted in software product costing. Second, I do not link the cost estimation process to the proposed cost system design for software firms, which aims to calculate current, examine past, and influence future software product costs. **Essay II** aims to analyze the software cost estimation process on a granular project level, from understanding the project requirements and environment to implementing the actual estimation. In doing so, I take the perspective of a project controller or manager who focuses on estimating the costs of his or her responsible project. In contrast, **Essay III** aims to outline why firms must adapt their extant cost systems to the requirements of intangible products on a firm or departmental level. Here, I take the perspective of a chief controlling officer or head of controlling who is responsible for defining congruent cost-management and modeling systems that allow firms to gain multiple perspectives on product costing and, thus, improve decision making. A continuation of my work would be to extend the suggested cost system design (cf. **Essay III** in Chapter 4) by integrating the software cost estimation process on an individual project level (cf. **Essay II** in Chapter 3) according to the divide-and-conquer principle.

5.3 Avenues for future research

Essay I reveals interesting opportunities for future research. First, researchers could validate my experimental results in a field study with multiple decision rules to address the external validity concerns. Second, they could vary the information overload manipulation and examine whether and how digital working environments induce information overload and influence decision-making quality. Third, future scholars could investigate additional control mechanisms to mitigate the adverse effects of information overload on decision quality in the absence of decision aids.

Based on the results of **Essay II**, scholars and practitioners should validate the UPI model and compare the estimation accuracy when implementing the UPI model with the accuracy from extant processes. The findings allow scholars to improve the model while aiming at deriving a normative model that fulfills software cost estimation's social and technical aspects. Further, the interviews highlight that firms that need to make a project bid or outline medium or long-term budget requirements particularly need help estimating the software costs in early project phases

with unstable product requirements. Scholars should try to find solutions for such settings where analogies are rarely available.

The limitations of **Essay III** reveal several avenues for future research. Scholars and practitioners can follow our guidelines for building a cost system design and validate the results in a case study. In particular, we propose that scholars validate the cost system design in four steps and combine academic and practical insights. First, scholars can extend our cost system design by expert-based evaluations of current software costing practices by leveraging focus group workshops. Second, scholars can add further industry-specific requirements. Third, scholars can focus on one cost-management or modeling approach of the cost system design to generate lead user stories, receive feedback, and continuously improve the approach. Fourth, scholars and practitioners can implement a prototype of the selected cost-management or modeling approach by conducting test runs with the respective lead users. Based on the test runs, they can monitor the performance, incorporate feedback, and improve the approach.

This dissertation provides further opportunities for research and practice. Scholars can pick up the results of introducing accountability as a control mechanism for decision-making under information overload (cf. **Essay I** in Chapter 2) for introducing new cost-management practices (cf. **Essay II** in Chapter 3 and **Essay III** in Chapter 4). They can investigate how firms implement new cost-management practices using different forms of accountability. Enforcing new cost-management practices can shape *what* managers should be held accountable for (i.e., outcome accountability) (Siegel-Jacobs and Yates, 1996, Chang et al., 2013, Patil et al., 2014), *how* the practice should be performed (i.e., process accountability) (Lerner and Tetlock, 1999, Dalla Via et al., 2019), and *who* should be responsible for the outcomes or processes (i.e., personal accountability). Scholars can examine which accountability type supports the adaption of the suggested cost-management and modeling practices in the context of management accounting change and whether this changes throughout the implementation.

5.4 Concluding remarks

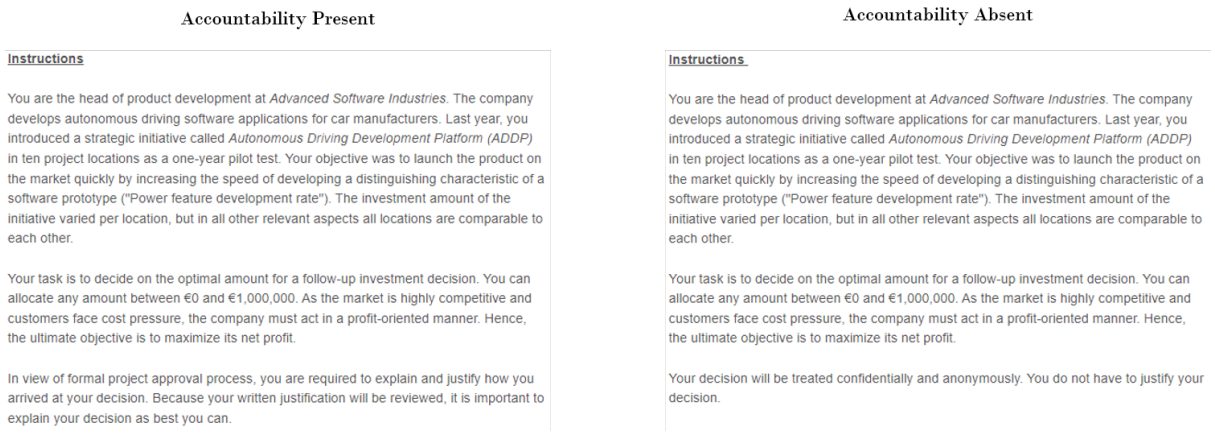
How can scholars and practitioners now proceed? Referring to Herbert Diess' quote on managing the digital transformation as the central task of the automotive industry, firms like Volkswagen already rebuild their organization to account for the development and offering of digital products. However, adjusting the organizational structure is only part of the puzzle to survive in a

digital world. I examine how controllers should provide information to decision-makers in the context of an increasing volume of available information. I comprehensively describe how project managers currently estimate software project costs and suggest guidelines to chief controlling officers for re-designing their cost systems on departmental and firm levels to meet digital product requirements. In conclusion, this dissertation improves the understanding of controllers for the implications of the digital transformation on firms' decision-making and cost-management practices and provides suggestions for how firms can approach the changes.

Appendix

Appendix to Essay I

FIGURE A.1: Case-related instructions.



Notes: This figure displays the case-related instructions participants read in the Accountability—Present and Accountability—Absent conditions.

FIGURE A.2: BSC data.

Low Information Load

Investment Decision

Below, you find the performance data of the one-year pilot test. The data includes financials, customers, internal business processes, as well as learning and growth aspects. Consultants indicated that all performance effects are immediately visible and occur within the year of implementation.

The figures state the pilot study results as percentage of the previous year, indicating performance changes based on the strategic initiative. Brief explanations of the measures are provided when you hover with your mouse pointer slowly over the words marked with [i].

Your task is to decide on the optimal follow-up investment amount (€0 - €1,000,000).

Location	A	B	C	D	E	F	G	H	I	J
Financial										
Net profit [i]	101%	104%	107%	109%	112%	109%	107%	105%	104%	99%
Customer										
Market share [i]	95%	102%	114%	108%	105%	97%	108%	103%	105%	107%
Internal Business Process										
Customer service lead time [i]	108%	98%	118%	109%	117%	109%	114%	113%	104%	122%
Learning and Growth										
Power feature development rate [i]	101%	105%	109%	111%	113%	115%	116%	117%	118%	118%
ADDP investment amount	€100,000	€220,000	€320,000	€410,000	€520,000	€610,000	€730,000	€810,000	€920,000	€1,000,000

Please enter the amount you decide to invest in the further project. The entry must be in full Euro amounts and without currency sign and without thousands separator (0 - 1000000).

High Information Load

Investment Decision

Below, you find the performance data of the one-year pilot test. The data includes financials, customers, internal business processes, as well as learning and growth aspects. Consultants indicated that all performance effects are immediately visible and occur within the year of implementation.

The figures state the pilot study results as percentage of the previous year, indicating performance changes based on the strategic initiative. Brief explanations of the measures are provided when you hover with your mouse pointer slowly over the words marked with [i].

Your task is to decide on the optimal follow-up investment amount (€0 - €1,000,000). Before entering the respective figure in the entry field, you must write down your justification of how you arrived at your decision.

Location	A	B	C	D	E	F	G	H	I	J
Financial										
Revenue [i]	95%	102%	114%	108%	105%	97%	108%	103%	105%	107%
New customer revenue growth [i]	90%	99%	112%	106%	111%	92%	103%	107%	103%	102%
Profit margin [i]	106%	102%	94%	101%	107%	112%	99%	102%	99%	93%
Net profit [i]	101%	104%	107%	109%	112%	109%	107%	105%	104%	99%
Customer										
Market share [i]	95%	102%	114%	108%	105%	97%	108%	103%	105%	107%
Product appeal relative to competitors [i]	93%	101%	113%	107%	108%	95%	105%	105%	104%	104%
Perceived service quality [i]	99%	105%	117%	111%	104%	101%	113%	103%	108%	112%
Service appeal relative to competitors [i]	97%	103%	115%	109%	102%	99%	111%	101%	106%	110%
Internal Business Process										
Power feature release rate [i]	96%	100%	104%	106%	107%	109%	110%	111%	112%	112%
Total number of power features released [i]	94%	98%	101%	103%	105%	107%	108%	109%	110%	110%
Average customer service staff productivity [i]	111%	100%	121%	112%	120%	112%	117%	116%	107%	125%
Customer service lead time [i]	108%	98%	118%	109%	117%	109%	114%	113%	104%	122%
Learning and Growth										
Customer service training expense [i]	92%	96%	105%	97%	91%	86%	99%	96%	99%	106%
Average customer service skills [i]	103%	101%	117%	109%	110%	104%	113%	107%	105%	116%
Power feature development expense [i]	97%	101%	110%	102%	96%	91%	104%	101%	104%	111%
Power feature development rate [i]	101%	105%	109%	111%	113%	115%	116%	117%	118%	118%
ADDP investment amount	€100,000	€220,000	€320,000	€410,000	€520,000	€610,000	€730,000	€810,000	€920,000	€1,000,000

Below you can find a text entry field to explain and justify how you arrived at your decision. As your written justification will be reviewed, it is important to explain your decision as best you can. The minimum length requirements are 20 words.

Your word count is: 0

Notes: This figure displays the BSC data in the Accountability—Absent condition. In addition to the BSC data, explanations of the measures were provided when participants hovered with their mouse pointer over the respective measures.

FIGURE A.3: A briefing note.

Welcome to the experiment. Before I will explain the rules of the experiment, I would like to finalize the attendance check. If you do not use your real name as Zoom alias, please state your name in the chat. Only I can see your chat message. If you use your real name as Zoom alias, you don't need to chat anything. I will give you further information while I am updating the attendance list.

You must participate via desktop computer or laptop. The experiment is not optimized for your smartphone or tablet. You will only receive the compensation if you completely finish the experiment. Further, you risk to be excluded from future experiments. You will not need your camera or microphone at any time. Yet, you will need to join with computer audio to listen to my instructions.

I will share the link and password to the experiment with you at the end of the instructions. You will find any information that you need on the experiment website. If anything goes wrong, you can always return to the experiment website. Just go to the link address again. To optimize the experimental presentation, I kindly ask you reduce the size of your screen view (,i.e., zoom out) by pressing the following two keys together: Strg and Minus sign (-) for Windows users or pressing the following three keys together: Option, Command, and Minus sign (-) for Mac users.

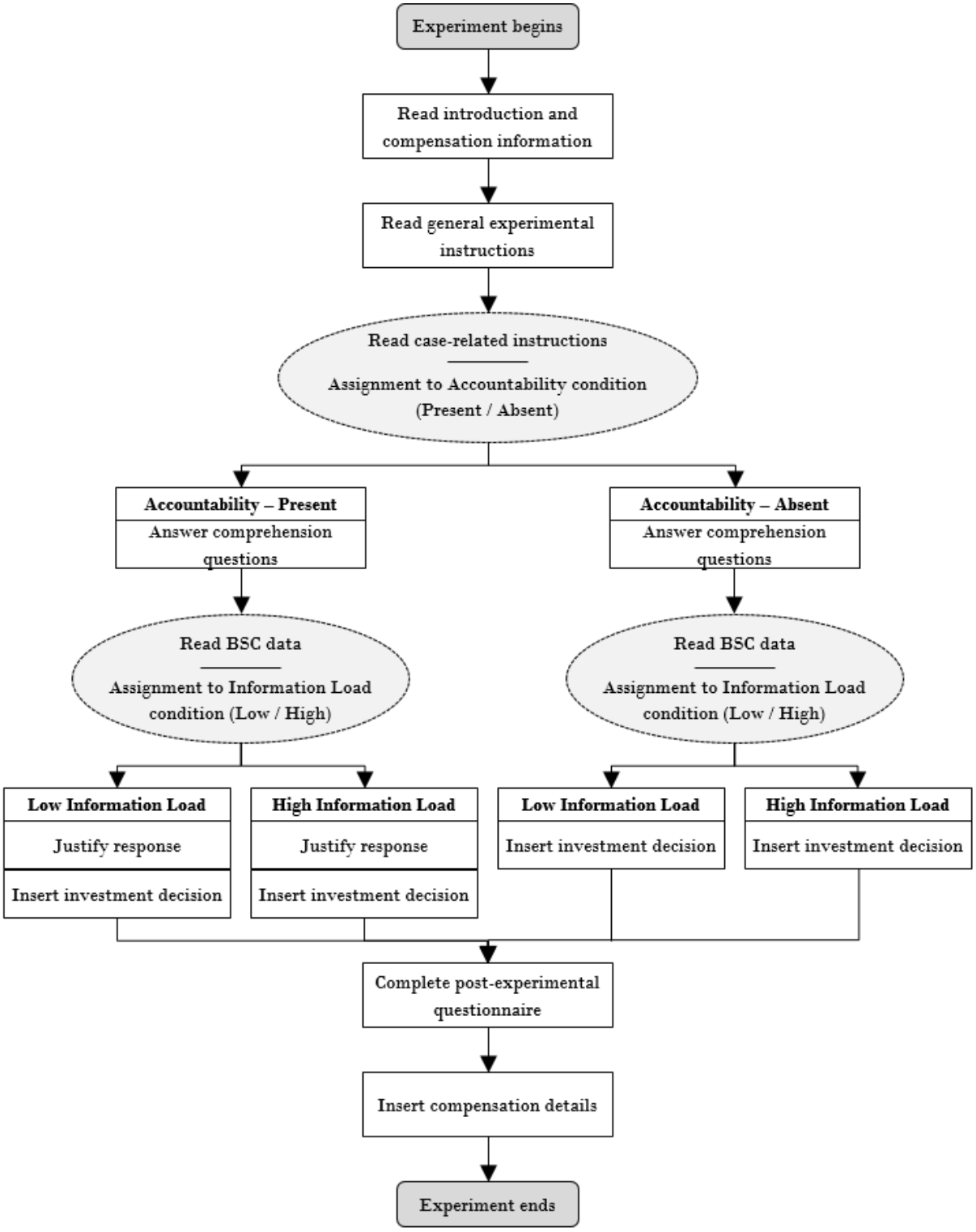
I'll keep this Zoom meeting running and you're welcome to stay here during the experiment. If you have any technical questions or issues, chat me. Feel free to leave the Zoom meeting when you have finished with the experiment.

At the end of the experiment, you will need to share your Paypal e-mail address. This is necessary to receive your compensation. Make sure you have the information ready. If you get to the page with the Paypal address field, everything has gone fine. If you do not get there, please let me know. You can also reply to yesterday's e-mail with the link to today's Zoom meeting.

According to experiment's rule, I will never cheat on you. Whatever you are told in the experiment is true. I will not trace any data to you. Except for the attendance check, your identity is irrelevant for the experiment.

I am going to chat the link to the experiment in a moment and mute myself. Once you have the link, please go to the website and start. I wish you good luck.

FIGURE A.4: Sequence of events in the experimental design.



Appendix to Essay II

Table A.1: Case study protocol.

Structure	**Protocol items**
Research objective	I aim to develop a deeper understanding of the software cost estimation process and explore how firms structure and conduct this task.
Research question	How do firms estimate software project costs?
Interview guide	(1) Personal introduction of the researcher (background and prior experience, research interests and objective) (2) Information on interview operations (explanation of recording practice and anonymity, clarification of the next steps after the interview) (3) Interview questions part I (see table A.2) (4) Interview questions part II (see table A.2) (5) Interview questions part III (see table A.2) (6) Interview questions part IV (see table A.2) (7) Interview questions part V (see table A.2) (8) Potential questions of the interviewee

Notes: This table provides an overview of the case study protocol based on the recommended structure of Yin (2018).

Table A.2: Interview questions.

Structure	**Questions**
I: Introduction and personal experience	1. Please state your current position and describe the roles and responsibilities you have held in the field of software development and cost estimation. 2. What is your field of expertise in software development and cost estimation?
II: Estimation process and methods	3. When you think of a recent individual project, please describe the entire software cost or effort estimation process with regard to a specific development activity. • How did you leverage popular software cost estimation methods? • Why are you using that specific type of process or method? • On which criteria did the choice of process or method depend? • How did you control the estimation process? Which mechanisms did you apply?
III: Data management	4. Talking about recent project examples, how did you leverage data to estimate software costs, e.g., how did you collect, train and/or calibrate cost or effort data?
IV: Evaluation of estimation results	5. Looking at past projects, what were the main cost drivers? 6. If applicable, what were the main reasons for cost under- or overestimation? 7. What are your improvement suggestions to increase the estimation accuracy?
V: Outlook and trends	8. Where do you see future potential in the field of software cost estimation?

Notes: This table includes an overview of the interview questions. Each interview consisted of five parts. However, the table does not display which specific question I asked, considering the previous answers of the interviewee.

FIGURE A.5: Overview of the data structure.

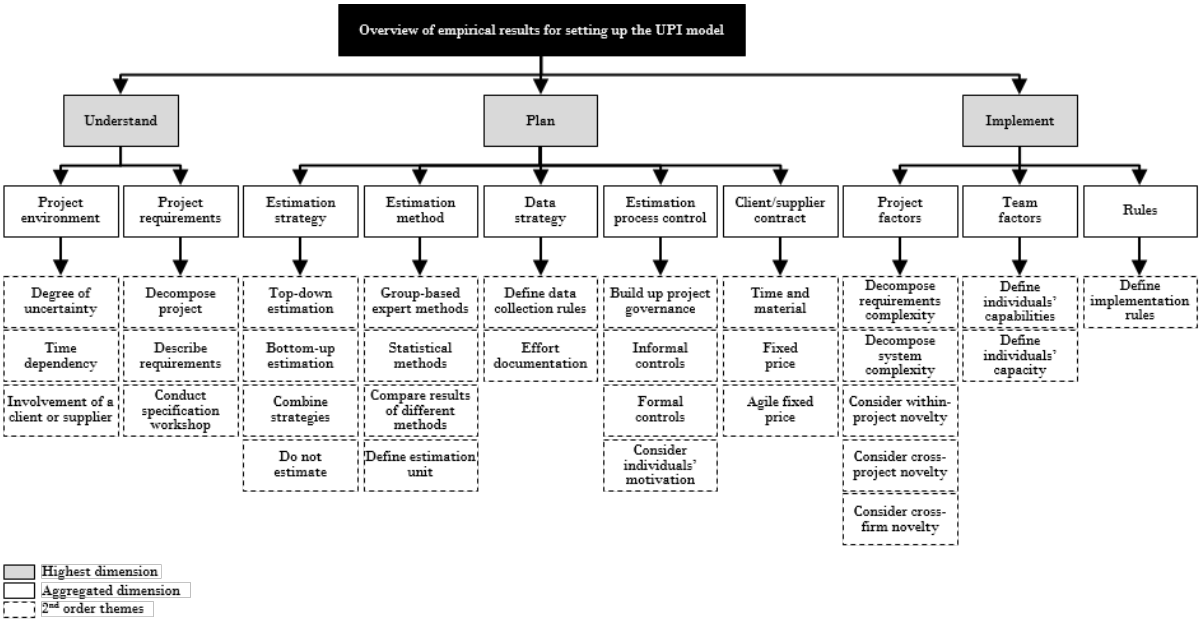
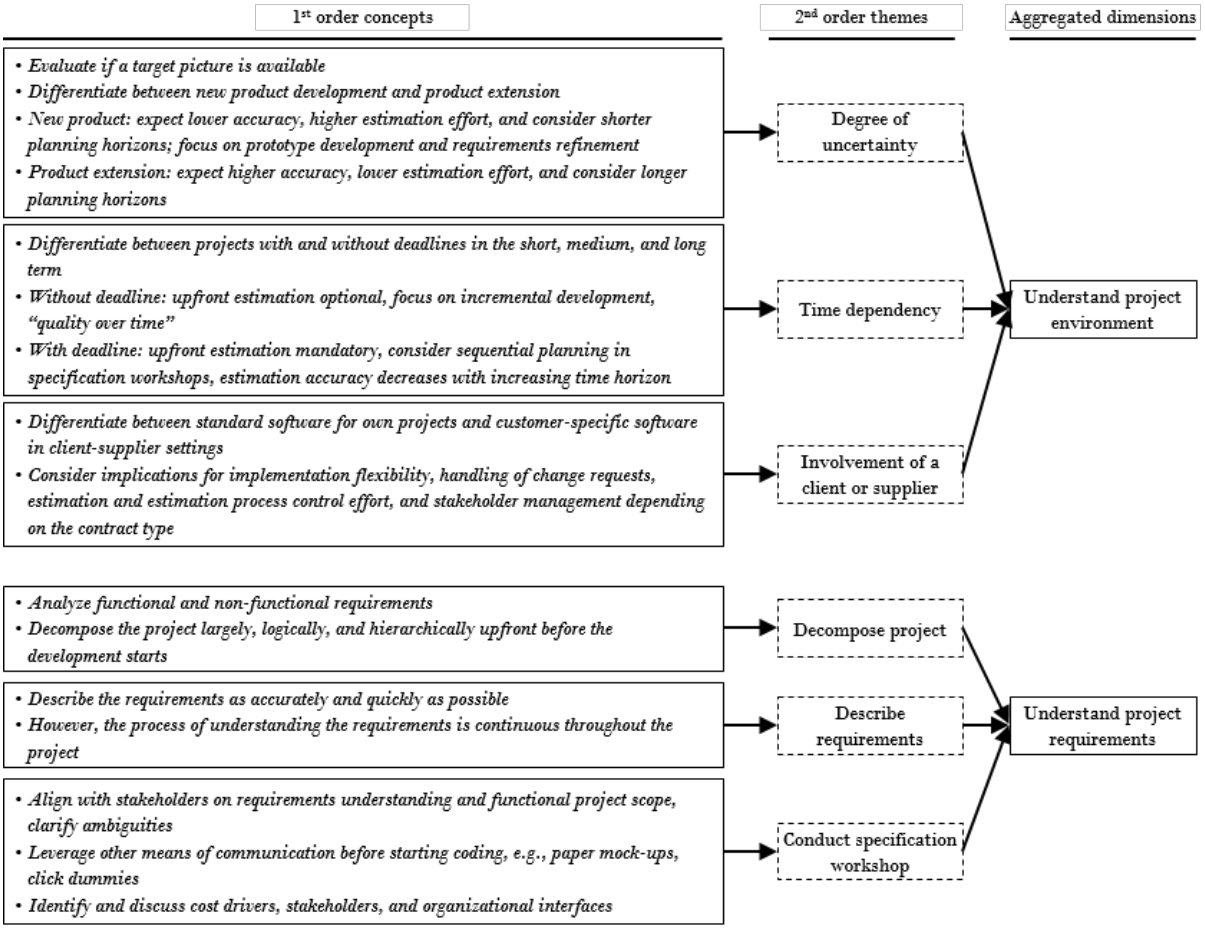
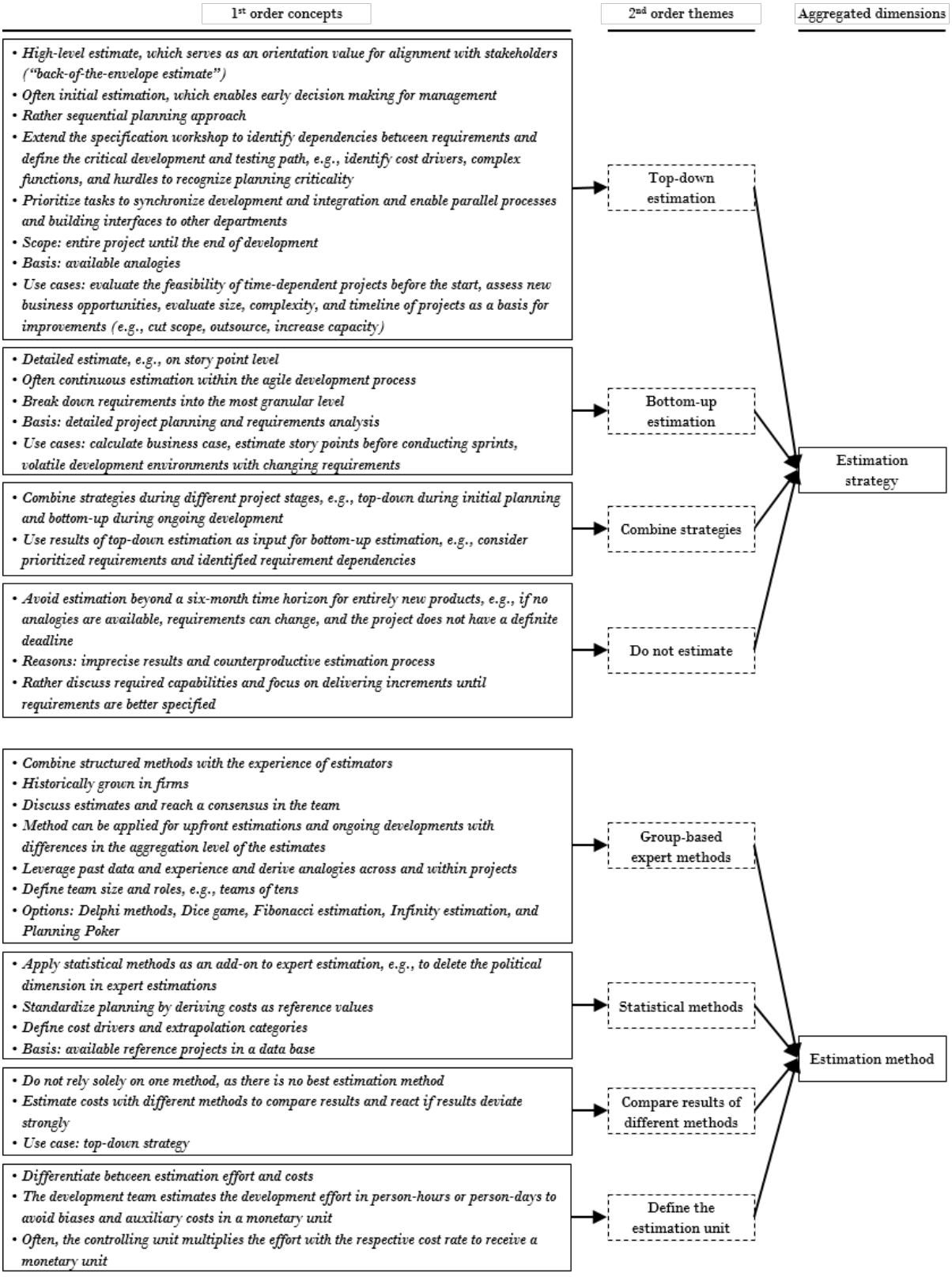


FIGURE A.6: Data structure of the *Understand* phase.



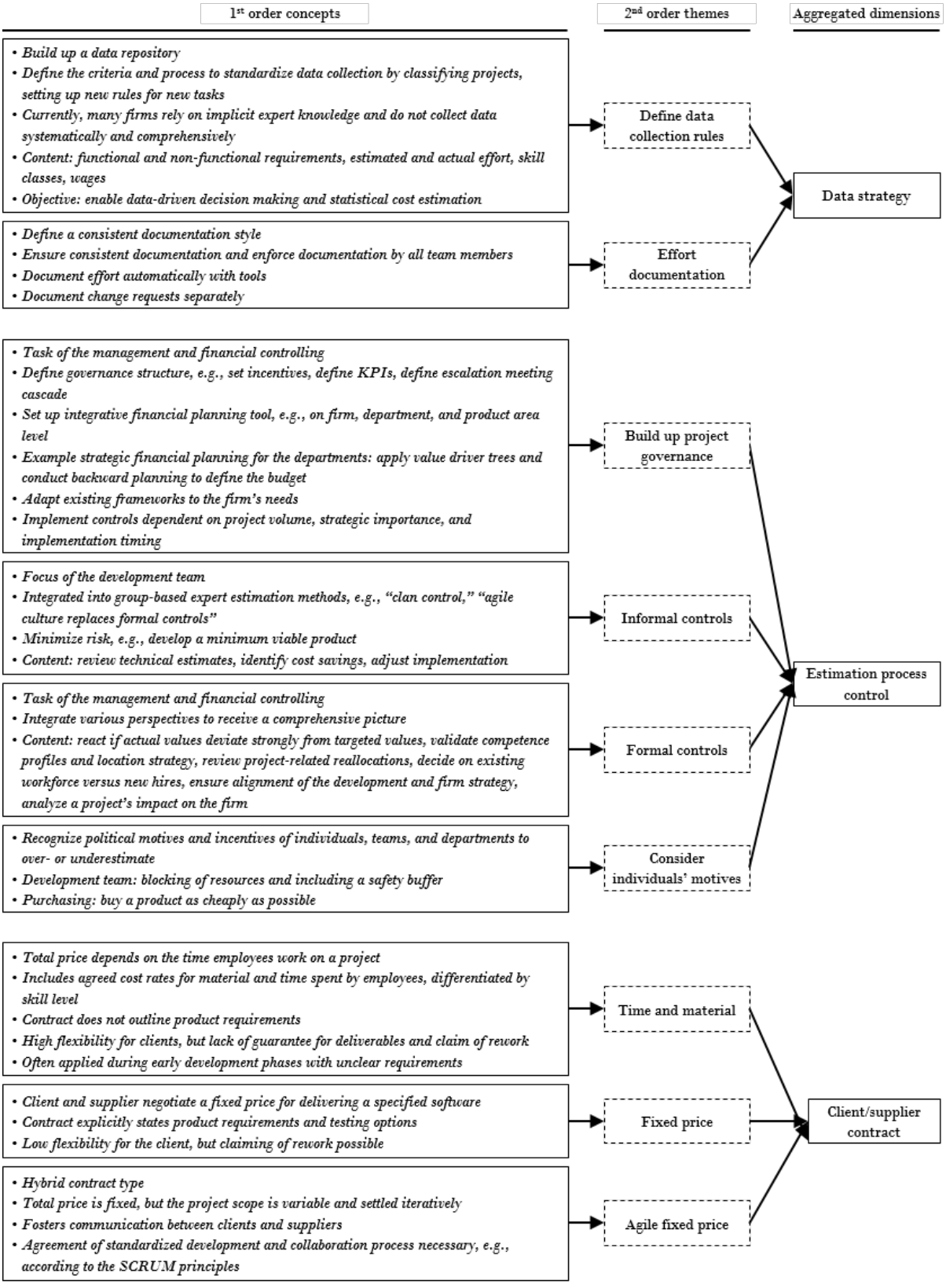
Notes: This figure displays the data structure of the *Understand* phase as suggested by Gioia et al. (2012). Each 1st order concept can consist of multiple codes with identical meaning.

FIGURE A.7: Data structure of the *Plan* phase (1/2).



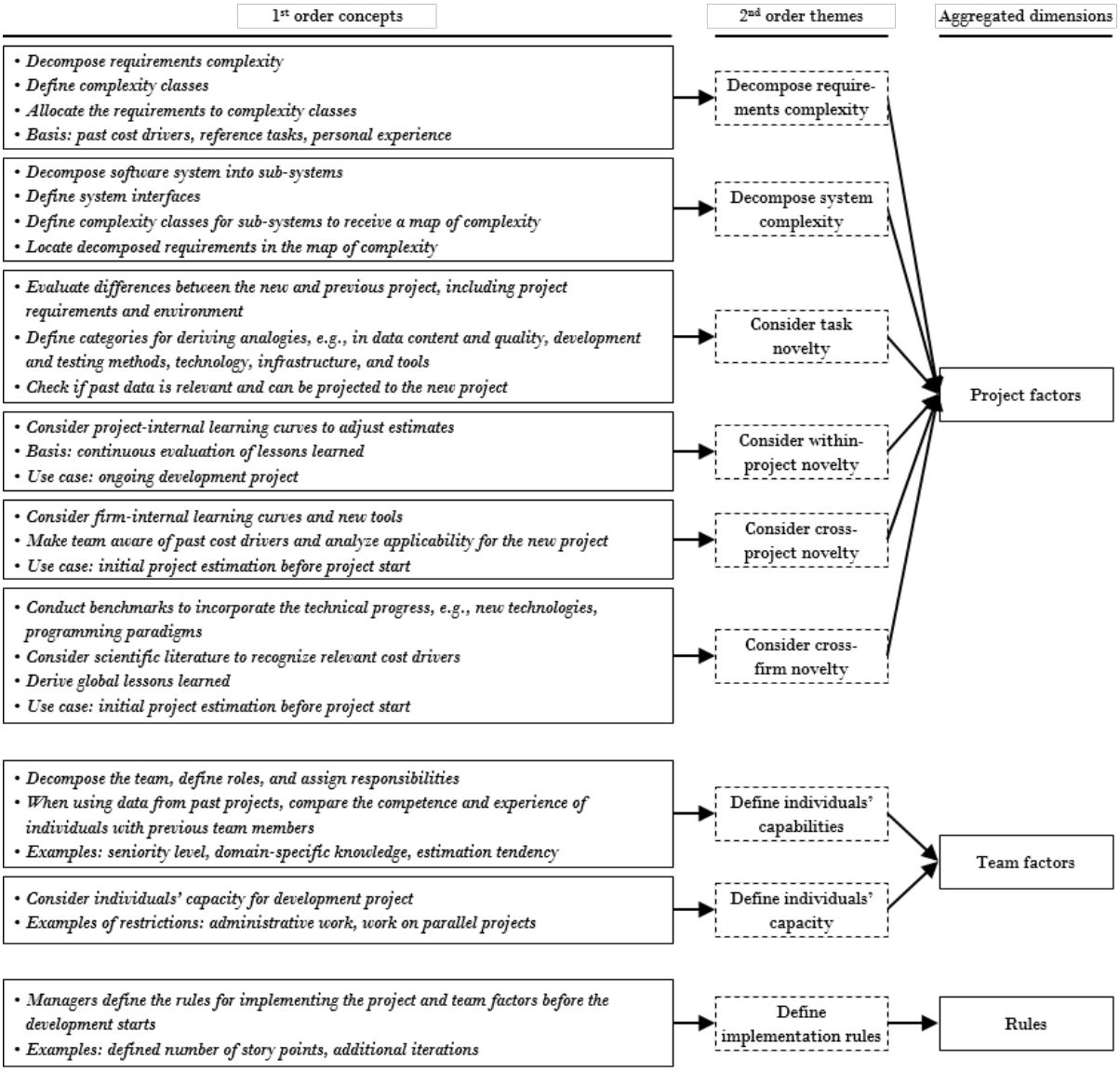
Notes: This figure displays the data structure of the *Plan* phase as suggested by Gioia et al. (2012). Each 1st order concept can consist of multiple codes with identical meaning.

FIGURE A.8: Data structure of the *Plan* phase (2/2).



Notes: This figure displays the data structure of the *Plan* phase as suggested by Gioia et al. (2012). Each 1st order concept can consist of multiple codes with identical meaning.

FIGURE A.9: Data structure of the *Implement* phase.



Notes: This figure displays the data structure of the *Implement* phase as suggested by Gioia et al. (2012). Each 1st order concept can consist of multiple codes with identical meaning.

Appendix to Essay III

FIGURE A.10: Cost structure table for industrial goods and information goods.

Assumptions (**Product cost object perspective** $_{\text{Commercialization Phase}}$):

- We display the costs of software products with no link to tangible products.
- We consider only the costs of manufacturing and SG&A as relevant from a product perspective.
- We consider development costs as sunk and do not depict them.
- We exclude maintenance and enhancement costs from the analysis.

	Direct costs		Indirect costs	
	Industrial goods	**Information goods**	**Industrial goods**	**Information goods**
Fixed costs	• Fixed salaries for product-specific employees in manufacturing • Fixed salaries for product-specific employees in sales and marketing	No manufacturing costs • Fixed salaries for product-specific employees in sales and marketing	• Rental costs for facilities and equipment • Fixed salaries for non-product-specific employees in manufacturing • SG&A costs independent of the level of activity	• Rental costs for facilities and equipment No manufacturing costs • SG&A costs independent of the level of activity
Variable costs	• Direct input material (e.g., metal, plastics) • Costs for flexible product-specific temporary workforce in manufacturing • Variable salary of product-specific workforce in manufacturing dependent on the level of activity • Sales commission per product	No costs for physical direct input materials No manufacturing costs • Sales commission per product	• Indirect input material (e.g., gas, lubricants, electricity) • Costs for machine maintenance and setup • Costs for flexible non-product-specific temporary workforce in manufacturing • Variable salary of non-product specific temporary workforce in manufacturing dependent on the level of activity • SG&A costs dependent on the level of activity	• Costs for operating the server infrastructure (e.g., hardware replacement, service fees) No manufacturing costs • SG&A costs dependent on the level of activity

Notes: This figure compares the cost structures under equal conditions with regard to the timeline and type of cost occurrence.

FIGURE A.11: Illustration of the dynamic target costing process.

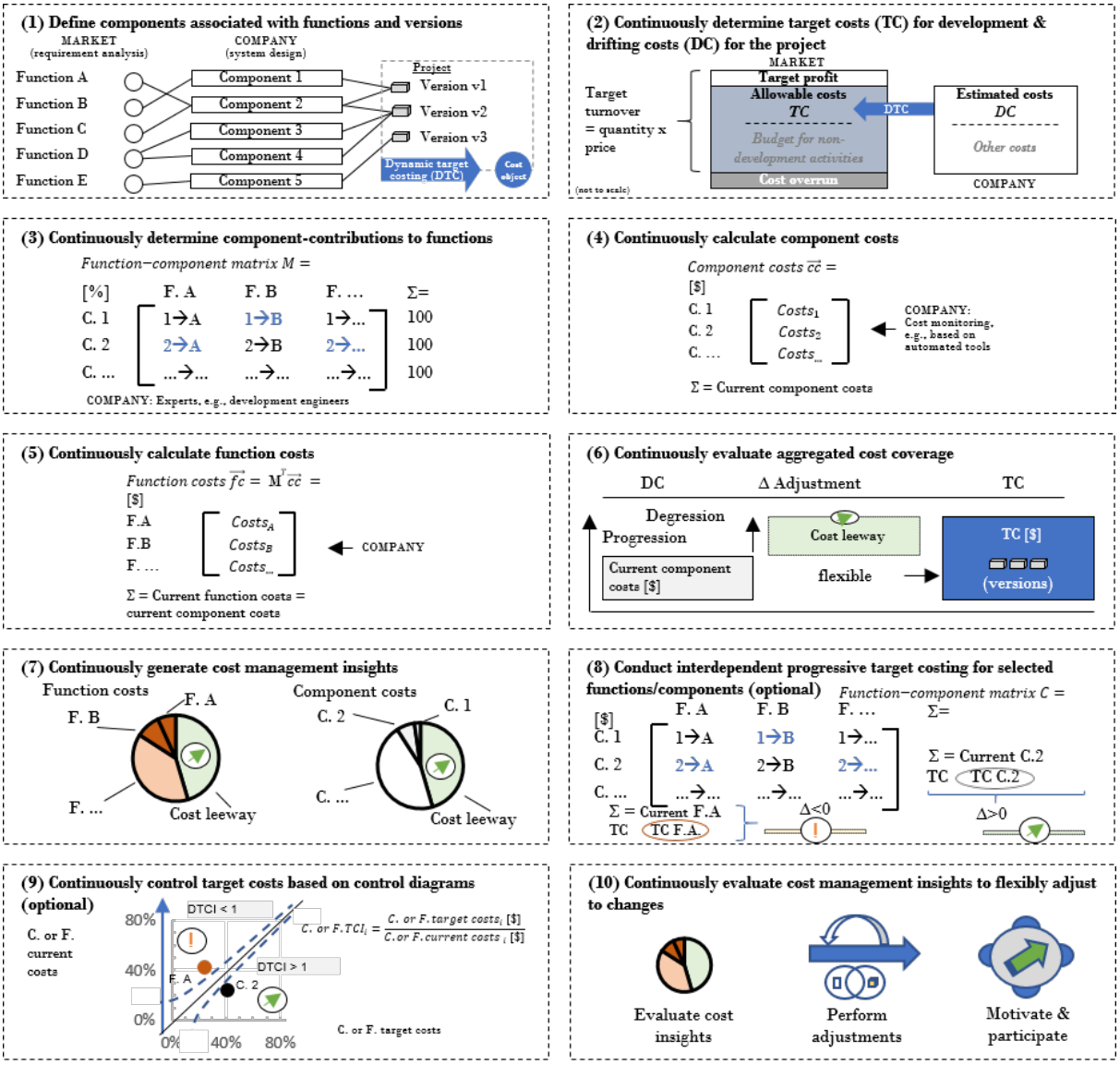


FIGURE A.12: Description of the dynamic target costing process.

We subdivide the dynamic target costing process into ten steps.

(1) Engineers define components associated with specific functions and versions during the requirements engineering and system design phases. In doing so, they combine a market- (requirements engineering) and firm-based (system design) approach and continuously adjust the definitions in an iterative way.

(2) Managers determine the target turnover by multiplying an estimated target sales volume by the target price. From this, managers deduct the target profit, which, in reverse, yields the costs that an organization in a competitive setting is allowed to incur, i.e., the allowable costs. In the next step, managers compare the allowable costs to the estimated costs that would arise in the current situation. If the latter exceeds the allowable costs, the difference indicates a need for cost reduction. Scholars divide the allowable costs into cost budgets planned for non-development activities, e.g., pre-development, distribution, and administration, and target costs for development. Similarly, scholars separate the estimated costs into realizable development costs, also called drifting costs, and estimated other costs. We consider only target costs and drifting costs for dynamic target costing.

(3) Experts, e.g., development engineers, assess the contribution of components to functions and define a function-component matrix. The sum of component contributions is 100 % for each component.

(4) Experts continuously monitor the costs. Based on the cost monitoring, they calculate the costs of components and define the costs as the vector of component costs. Automated tools can support this step.

(5) The vector of function costs is calculated by multiplying the transposed function-component matrix by the component cost vector.

(6) Managers aggregate component costs and target costs based on the versions that comprise the components and distribute these costs in a way that establishes synergetic coverage of costs. They evaluate component and target costs to determine cost leeway or overrun at this aggregated level. When doing so, managers must keep in mind that component costs are progressive since sunk costs cannot be altered ex-post.

(7) While jointly observing the costs of individual components and functions and the cost leeway or overrun, managers generate and continuously evaluate cost management insights.

(8) Managers can perform target costing for selected functions or components in a matrix structure that depicts the interdependency between function and component costs. This function-component matrix represents the counterpart to the matrix introduced in step 3 and consists of absolute cost values instead of proportional cost contributions. Managers can obtain the matrix by multiplying all proportional values in a component-related row of the function-component matrix with the overall costs of the respective component. Thus, managers can identify cost leeways or overruns for individual components and functions. We suggest considering this granular target costing level as optional because it could have a negative impact on the creative environment of software development.

(9) Managers can supplement the optional target costing at a function or component level by calculating a dynamic target costing index and visualizing it in a control diagram. The visualization enhances the identification of cost leeways and overruns at this granular cost observation level.

(10) We recommend continuously performing all dynamic target costing procedure constituents and adjusting these to allow for changing requirements from a technological and market perspective.

Bibliography

T. K. Abdel-Hamid. A study of staff turnover, acquisition, and assimilation and their impact on software development cost and schedule. *Journal of Management Information Systems*, 6 (1):21–40, 1989.

R. Adner, P. Puranam, and F. Zhu. What is different about digital strategy? From quantitative to qualitative change. *Strategy Science*, 4(4):253–261, 2019.

Aeresospace Vehicle Systems Institute. Motivation for advancing the savi program, 2016. URL `https://savi.avsi.aero/about-savi/savi-motivation/`. Date accessed: 17 Oct 2022.

S. Afhüppe, P. Brors, and S. Menzel. Noch ist es nicht zu spät - VW-Chef Diess warnt vor Abhängigkeit in der Batteriezellentechnik. *Handelsblatt*, 2018. URL `https://www.handelsblatt.com/unternehmen/industrie/interview-noch-ist-es-nicht-zu-spaet-vw-chef-diess-warnt-vor-abhaengigkeit-in\-der-batteriezellentechnik/22934412.html`. Date accessed: 29 Sep 2022.

A. Afuah and C. Tucci. *Internet Business Models and Strategies: Text and Cases*. Boston McGraw-Hill, 2001.

T. Ahrens. Styles of accountability. *Accounting, Organizations and Society*, 21(2-3):139–173, 1996.

A. J. Albrecht. Measuring application development productivity. In *Proceedings of the Joint SHARE, GUIDE, and IBM Application Development Symposium*, pages 83–92, Monterey, CA, 1979.

L. Angelis and I. Stamelos. A simulation tool for efficient analogy based cost estimation. *Empirical Software Engineering*, 5(1):35–68, 2000.

Y. Asiedu and P. Gu. Product life cycle cost analysis: State of the art review. *International Journal of Production Research*, 36(4):883–908, 1998.

S. Astromskis, A. Janes, A. Sillitti, and G. Succi. An approach to non-invasive cost accounting. In *EUROMICRO-Conference on Software Engineering and Advanced Applications*, pages 30–37, Washington, DC, USA, 2014. IEEE Computer Society.

R. D. Austin and L. Devin. Research commentary —weighing the benefits and costs of flexibility in making software: Toward a contingency theory of the determinants of development process design. *Information Systems Research*, 20(3):462–477, 2009.

R. Balakrishnan and K. Sivaramakrishnan. A critical overview of the use of full–cost data for planning and pricing. *Journal of Management Accounting Research*, 14(1):3–31, 2002.

R. Balakrishnan, S. Hansen, and E. Labro. Evaluating heuristics used when designing product costing systems. *Management Science*, 57(3):520–541, 2011.

R. Balakrishnan, E. Labro, and K. Sivaramakrishnan. Product costs as decision aids: An analysis of alternative approaches (part 1). *Accounting Horizons*, 26(1):1–20, 2012.

R. D. Banker, H. Chang, and C. F. Kemerer. Evidence on economies of scale in software development. *Information and Software Technology*, 36(5):275–282, 1994.

R. D. Banker, G. Potter, and D. Srinivasan. An empirical investigation of an incentive plan that includes nonfinancial performance measures. *The Accounting Review*, 75(1):65–92, 2000.

R. D. Banker, H. Chang, and M. J. Pizzini. The balanced scorecard: Judgmental effects of performance measures linked to strategy. *The Accounting Review*, 79(1):1–23, 2004.

G. Bartlett, E. Johnson, and P. Reckers. Accountability and role effects in balanced scorecard performance evaluations when strategy timeline is specified. *European Accounting Review*, 23 (1):143–165, 2014.

D. Basten and W. Mellis. A current assessment of software development effort estimation. In *2011 International Symposium on Empirical Software Engineering and Measurement*, pages 235–244, Washington, DC, USA, 2011. IEEE.

D. M. Becker and A. A. Gaivoronski. Optimisation approach to target costing under uncertainty with application to ICT-service. *International Journal of Production Research*, 56(5):1904–1917, 2018.

I. Benbasat and I. Vessey. Programmer and analyst time/cost estimation. *MIS Quarterly*, 4(2): 31, 1980.

A. Bharadwaj, O. A. El Sawy, P. A. Pavlou, and N. Venkatraman. Digital business strategy: Toward a next generation of insights. *MIS Quarterly*, 37(2):471–482, 2013.

H. K. Bhargava and V. Choudhary. Research note: When is versioning optimal for information goods? *Management Science*, 54(5):1029–1035, 2008.

A. Bhimani and M. Bromwich. Management accounting in a digital and global economy: The interface of strategy, technology, and cost information. In *Accounting, Organizations, and Institutions: Essays in Honour of Anthony Hopwood*. Oxford University Press, 2009.

A. Bhimani and M. Bromwich. *Management accounting: Retrospect and prospect*. CIMA Publishing, Amsterdam and Oxford, 2010.

A. Bhimani and L. Willcocks. Digitisation, 'big data' and the transformation of accounting information. *Accounting and Business Research*, 44(4):469–490, 2014.

J. G. Birnberg, V. B. Hoffman, and S. Yuen. The accountability demand for information in China and the US–a research note. *Accounting, Organizations and Society*, 33(1):20–32, 2008.

B. W. Boehm. *Software engineering economics*. Prentice-Hall, 1981.

B. W. Boehm and P. N. Papaccio. Understanding and controlling software costs. *IEEE Transactions on Software Engineering*, 14(10):1462–1477, 1988.

M. Bradley and R. Dawson. Whole life cost: The future trend in software development. *Software Quality Journal*, 8(2):121–131, 1999.

K. Brands and M. Holtzblatt. Business analytics: Transforming the role of management accountants. *Management Accounting Quarterly*, 16(3):1–12, 2015.

H. Brown-Liburd, H. Issa, and D. Lombardi. Behavioral implications of big data's impact on audit judgment and decision making and future research directions. *Accounting Horizons*, 29 (2):451–468, 2015.

M. Broy. Challenges in automotive software engineering. In *Proceeding of the 28th International Conference on Software Engineering*, pages 33–42, New York, USA, 2006. ACM Press.

B. Brügge and A. H. Dutoit. *Object-oriented software engineering: Using UML, patterns, and Java.* Prentice Hall, 3rd edition, 2014.

E. Brynjolfsson and K. McElheran. Data in action: Data-driven decision making in U.S. manufacturing: Working papers. *Available at SSRN*, 2016.

E. Brynjolfsson, L. M. Hitt, and H. H. Kim. Strength in numbers: How does data-driven decisionmaking affect firm performance? *Available at SSRN*, 2011.

E. Brynjolfsson, C. Wang, and X. Zhang. The economics of it and digitization: Eight questions for research. *MIS Quarterly*, 45(1):473–477, 2021.

CARIAD SE. Car.software organisation is now CARIAD, 2021. URL `https://cariad.technology/de/en/news/stories/car-software-organisation-is-now-cariad.html`. Date accessed: 13 Dec 2022.

J. L. Carlo, K. Lyytinen, and R. J. Boland. Dialectics of collective minding: Contradictory appropriations of information technology in a high-risk project. *MIS Quarterly*, 36(4):1081, 2012.

M. Castells. *The rise of the network society.* The information age: Economy, society, and culture. Wiley-Blackwell, Oxford, 2nd edition, 2010.

L. J. Chang, M. M. Cheng, and K. T. Trotman. The effect of outcome and process accountability on customer–supplier negotiations. *Accounting, Organizations and Society*, 38(2):93–107, 2013.

R. K. Chellappa and A. Mehra. Cost drivers of versioning: Pricing and product line strategies for information goods. *Management Science*, 64(5):2164–2180, 2018.

E. G. J. Chewning and A. M. Harrell. The effect of information load on decision makers' cue utilization levels and decision quality in a financial distress decision task. *Accounting, Organizations and Society*, 15(6):527–542, 1990.

T. Chow and D.-B. Cao. A survey study of critical success factors in agile software projects. *Journal of Systems and Software*, 81(6):961–971, 2008.

C. W. Cobb and P. H. Douglas. A theory of production. *The American Economic Review*, 18 (1):139–165, 1928.

A. G. Coenenberg, T. Fischer, and J. Schmitz. Target costing und Product life cycle costing als Instrumente des Kostenmanagements. In *Kostenmanagement*, pages 195–232. Springer, Berlin, Heidelberg, 1997.

T. Connolly and D. Dean. Decomposed versus holistic estimates of effort required for software writing tasks. *Management Science*, 43(7):1029–1045, 1997.

R. Cooper and R. S. Kaplan. Measure costs right: Make the right decisions. *Harvard Business Review*, 66(5):96–103, 1988.

R. Cooper and R. S. Kaplan. The promise–and peril–of integrated cost systems. *Harvard Business Review*, 76(4):109–119, 1998.

J. M. Corbin and A. Strauss. Grounded theory research: Procedures, canons, and evaluative criteria. *Qualitative Sociology*, 13:3–21, 1990.

T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge 2009, 3rd edition, 2009.

N. Dalla Via, P. Perego, and M. Van Rinsum. How accountability type influences information search processes and decision quality. *Accounting, Organizations and Society*, 75:79–91, 2019.

S. Datar and M. Gupta. Aggregation, specification and measurement errors in product costing. *The Accounting Review*, 69(4):567–591, 1994.

S. Datar, M. V. Rajan, and C. T. Horngren. *Horngren's cost accounting: A managerial emphasis*. Pearson, Essex, United Kingdom, 17th edition, 2021.

S. Ding and P. Beaulieu. The role of financial incentives in balanced scorecard-based performance evaluations: Correcting mood congruency biases. *Journal of Accounting Research*, 49(5):1223–1247, 2011.

M. J. Driver and S. Streufert. Integrative complexity: An approach to individuals and groups as information-processing systems. *Administrative Science Quarterly*, pages 272–285, 1969.

T. Dybå and T. Dingsøyr. Empirical studies of agile software development: A systematic review. *Information and Software Technology*, 50(9):833–859, 2008.

C. Ebert. The impacts of software product management. *Journal of Systems and Software*, 80 (6):850–861, 2007.

C. Eduardo Carbonera, K. Farias, and V. Bischoff. Software development effort estimation: A systematic mapping study. *IET Software*, 14(4):328–344, 2020.

J. S. Edwards and T. T. Moores. A conflict between the use of estimating and planning tools in the management of information systems. *European Journal of Information Systems*, 3(2): 139–147, 1994.

K. M. Eisenhardt. Building theories from case study research. *The Academy of Management Review*, 14(4):532–550, 1989.

K. M. Eisenhardt and M. E. Graebner. Theory building from cases: Opportunities and challenges. *The Academy of Management Journal*, 50(1):25–32, 2007.

M. J. Eppler and J. Mengis. The concept of information overload: A review of literature from organization science, accounting, marketing, mis, and related disciplines. *The Information Society*, 20(5):325–344, 2004.

P. Everaert, S. Loosveld, T. van Acker, M. Schollier, and G. Sarens. Characteristics of target costing: Theoretical and field study perspectives. *Qualitative Research in Accounting & Management*, 3(3):236–263, 2006.

R. Ewert and C. Ernst. Target costing, co-ordination and strategic cost management. *European Accounting Review*, 8(1):23–49, 1999.

R. Ewert and A. Wagenhofer. *Interne Unternehmensrechnung*. Springer, Berlin, Heidelberg, 8th edition, 2014.

D. D. Fehrenbacher, S. E. Kaplan, and C. Moulang. The role of accountability in reducing the impact of affective reactions on capital budgeting decisions. *Management Accounting Research*, 47:100650, 2020.

R. G. Fichman and C. F. Kemerer. Activity based costing for component-based software development. *Information Technology and Management*, 3(1/2):137–160, 2002.

R. G. Fichman, B. L. Dos Santos, and Z. Zheng. Digital innovation as a fundamental and powerful concept in the information systems curriculum. *MIS Quarterly*, 38(2):329–A15, 2014.

D. Fleisig. Adding information may increase overconfidence in accuracy of knowledge retrieval. *Psychological Reports*, 108(2):379–392, 2011.

G. Friedl, C. Hofmann, and B. Pedell. *Kostenrechnung: eine entscheidungsorientierte Einführung*. Vahlen, 2017.

Gartner glossary, 2020. URL `https://www.gartner.com/en/information-technology/glossary/digitalization`. Date accessed: 19 Oct 2022.

M. Gervais, Y. Levant, and C. Ducrocq. Time-driven activity-based costing (TDABC): An initial appraisal through a longitudinal case study. *Journal of Applied Management Accounting Research*, 8(2):1–20, 2010.

D. A. Gioia, K. G. Corley, and A. L. Hamilton. Seeking qualitative rigor in inductive research: Notes on the Gioia methodology. *Organizational Research Methods*, 16(1):15–31, 2012.

S. M. Glover. The influence of time pressure and accountability on auditors' processing of nondiagnostic information. *Journal of Accounting Research*, 35(2):213–226, 1997.

A. Gopal, K. Sivaramakrishnan, M. S. Krishnan, and T. Mukhopadhyay. Contracts in offshore software development: An empirical analysis. *Management Science*, 49(12):1671–1683, 2003.

L. A. Gordon. Allocating service departments' costs: Methodology and case study. *Accounting and Business Research*, 5(17):3–8, 1974.

A. Gupta, K. Kannan, and P. Sanyal. Economic experiments in information systems. *MIS Quarterly*, 42(2):595–606, 2018.

T. Halkjelsvik and M. Jørgensen. From origami to software development: A review of studies on judgment-based predictions of performance time. *Psychological Bulletin*, 138(2):238–271, 2012.

A. T. Hall, M. T. Royle, R. A. Brymer, P. L. Perrewé, G. R. Ferris, and W. A. Hochwarter. Relationships between felt accountability as a stressor and strain reactions: The neutralizing role of autonomy across two studies. *Journal of Occupational Health Psychology*, 11(1):87, 2006.

A. T. Hall, D. D. Frink, and M. R. Buckley. An accountability account: A review and synthesis of the theoretical and empirical research on felt accountability. *Journal of Organizational Behavior*, 38(2):204–224, 2017.

J. E. Hannay, D. I. Sjoberg, and T. Dyba. A systematic review of theory use in software engineering experiments. *IEEE Transactions on Software Engineering*, 33(2):87–107, 2007.

M. L. Harris, R. W. Collins, and A. R. Hevner. Control of flexible software development under uncertainty. *Information Systems Research*, 20(3):400–419, 2009.

M. Hartmann and B. E. Weißenberger. Decision-making in the capital budgeting context–effects of type of decision aid and increases in information load. *Available at SSRN*, 2020.

N. C. Haugen. An empirical study of using planning poker for user story estimation. In *Proceedings of the Conference on AGILE 2006*, pages 23–34, USA, 2006. IEEE Computer Society.

F. J. Heemstra. Software cost estimation. *Information and Software Technology*, 34(10):627–639, 1992.

O. Henfridsson and R. Lindgren. User involvement in developing mobile and temporarily interconnected systems. *Information Systems Journal*, 20(2):119–135, 2010.

J. Hihn and H. Habib-Agahi. Cost estimation of software intensive projects: a survey of current practices. In *13th International Conference on Software Engineering*, pages 276–287, Washington, DC, USA, 1991. IEEE Computer Society.

B. Hinings, T. Gegenhuber, and R. Greenwood. Digital innovation and transformation: An institutional perspective. *Information and Organization*, 28(1):52–61, 2018.

T. Hiromoto. Another hidden edge—Japanese management accounting. *Harvard Business Review*, 66(4):22–26, 1988.

V. B. Hoffman and J. M. Patton. Accountability, the dilution effect, and conservatism in auditors' fraud judgments. *Journal of Accounting Research*, 35(2):227–237, 1997.

P. Horváth and R. Mayer. Konzeption und Entwicklungen der Prozeßkostenrechnung. In *Prozeßkostenrechnung*, pages 59–86. Gabler Verlag, Wiesbaden, 1995.

J. L. Huang, P. G. Curran, J. Keeney, E. M. Poposki, and R. P. DeShon. Detecting and deterring insufficient effort responding to surveys. *Journal of Business and Psychology*, 27(1):99–114, 2012.

S.-J. Huang, N.-H. Chiu, and L.-W. Chen. Integration of the grey relational analysis with genetic algorithm for software effort estimation. *European Journal of Operational Research*, 188(3): 898–909, 2008.

H. Huijgens, A. van Deursen, L. L. Minku, and C. Lokan. Effort and cost in software engineering: A comparison of two industrial data sets. In *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*, pages 51–60, New York, USA, 2017. ACM.

K. A. Humphreys, M. S. Gary, and K. T. Trotman. Dynamic decision making using the balanced scorecard framework. *The Accounting Review*, 91(5):1441–1465, 2016.

A. Idri, F. a. Amazal, and A. Abran. Analogy-based software development effort estimation: A systematic mapping and review. *Information and Software Technology*, 58:206–230, 2015.

E. R. Iselin. The effects of information load and information diversity on decision quality in a structured decision task. *Accounting, Organizations and Society*, 13(2):147–164, 1988.

E. R. Iselin. The effects of the information and data properties of financial ratios and statements on managerial decision quality. *Journal of Business Finance and Accounting*, 20(2):249–266, 1993.

C. D. Ittner and D. F. Larcker. Innovations in performance measurement: Trends and research implications. *Journal of Management Accounting Research*, 10:205, 1998a.

C. D. Ittner and D. F. Larcker. Are nonfinancial measures leading indicators of financial performance? An analysis of customer satisfaction. *Journal of Accounting Research*, 36:1–35, 1998b.

S. Iyengar. *The art of choosing*. Twelve, New York, 1st edition, 2010.

J. Jermias. The influence of accountability on overconfidence and resistance to change: A research framework and experimental evidence. *Management Accounting Research*, 17(4): 370–388, 2006.

R. Jones and H. Mendelson. Information goods vs. industrial goods: Cost structure and competition. *Management Science*, 57(1):164–176, 2011.

M. Jørgensen. Top-down and bottom-up expert estimation of software development effort. *Information and Software Technology*, 46(1):3–16, 2004a.

M. Jørgensen. A review of studies on expert estimation of software development effort. *Journal of Systems and Software*, 70(1-2):37–60, 2004b.

M. Jørgensen. The influence of selection bias on effort overruns in software development projects. *Information and Software Technology*, 55(9):1640–1650, 2013.

M. Jørgensen. What we do and don't know about software development effort estimation. *IEEE Software*, 31(2):37–40, 2014.

M. Jørgensen and G. J. Carelius. An empirical study of software project bidding. *IEEE Transactions on Software Engineering*, 30(12):953–969, 2004.

M. Jørgensen and K. Molokken. A preliminary checklist for software cost management. In *Third International Conference on Quality Software*, pages 134–140. IEEE, 2003.

M. Jørgensen and M. Shepperd. A systematic review of software development cost estimation studies. *IEEE Transactions on Software Engineering*, 33:33–53, 2007.

M. Jørgensen, P. Mohagheghi, and S. Grimstad. Direct and indirect connections between type of contract and software project outcome. *International Journal of Project Management*, 35 (8):1573–1586, 2017.

R. S. Kaplan. One cost system isn't enough. *Harvard Business Review*, 66(1):61–66, 1988.

R. S. Kaplan and S. R. Anderson. Time-driven activity-based costing. *Harvard Business Review*, 82(11):131–138, 2004.

R. S. Kaplan and D. P. Norton. *The balanced scorecard: Translating strategy into action.* Harvard Business School Press, Boston, 1996a.

R. S. Kaplan and D. P. Norton. Using the balanced scorecard as a strategic management system. *Harvard Business Review*, (11):75–85, 1996b.

R. S. Kaplan and D. P. Norton. Transforming the balanced scorecard from performance measurement to strategic management: Part I. *Accounting Horizons*, 15(1):87–104, 2001a.

R. S. Kaplan and D. P. Norton. Transforming the balanced scorecard from performance measurement to strategic management: Part II. *Accounting Horizons*, 15(2):147–160, 2001b.

J. Karimi and Z. Walter. The role of dynamic capabilities in responding to digital disruption: A factor-based study of the newspaper industry. *Journal of Management Information Systems*, 32:39–81, 2015.

Y. Kato. Target costing support systems: Lessons from leading Japanese companies. *Management Accounting Research*, 4(1):33–47, 1993.

A. S. Kelton and U. S. Murthy. The effects of information disaggregation and financial statement interactivity on judgments and decisions of nonprofessional investors. *Journal of Information Systems*, 30(3):99–118, 2016.

J. W. Keung, B. A. Kitchenham, and D. R. Jeffery. Analogy-x: Providing statistical inference to analogy-based software cost estimation. *IEEE Transactions on Software Engineering*, 34 (4):471–484, 2008.

W. Kilger, J. R. Pampel, and K. Vikas. *Flexible Plankostenrechnung und Deckungsbeitragsrechnung*. Gabler Verlag, Wiesbaden, 2012.

T. Knauer and K. Möslang. The adoption and benefits of life cycle costing. *Journal of Accounting & Organizational Change*, 14(2):188–215, 2018.

E. Kocaguneli, T. Menzies, and J. W. Keung. On the value of ensemble effort estimation. *IEEE Transactions on Software Engineering*, 38(6):1403–1416, 2012.

E. Kula, E. Greuter, A. van Deursen, and G. Gousios. Factors affecting on-time delivery in large-scale agile software development. *IEEE Transactions on Software Engineering*, 48(9): 3573–3592, 2022.

E. Labro and M. Vanhoucke. A simulation analysis of interactions among errors in costing systems. *The Accounting Review*, 82(4):939–962, 2007.

A. L. Lederer and J. Prasad. Causes of inaccurate software development cost estimates. *Journal of Systems and Software*, 31(2):125–134, 1995.

A. L. Lederer, R. Mirani, B. S. Neo, C. Pollard, J. Prasad, and K. Ramamurthy. Information system cost estimating: A management perspective. *MIS Quarterly*, 14(2):159, 1990.

J. S. Lerner and P. E. Tetlock. Accounting for the effects of accountability. *Psychological Bulletin*, 125(2):255, 1999.

D. J. Levitin. *The organized mind: Thinking straight in the age of information overload*. Penguin, 2014.

J. Li, G. Ruhe, A. Al-Emran, and M. M. Richter. A flexible method for software effort estimation by analogy. *Empirical Software Engineering*, 12(1):65–106, 2007.

R. Libby, R. Bloomfield, and M. W. Nelson. Experimental research in financial accounting. *Accounting, Organizations and Society*, 27(8):775–810, 2002.

T. Libby, S. E. Salterio, and A. Webb. The balanced scorecard: The effects of assurance and process accountability on managerial judgment. *The Accounting Review*, 79(4):1075–1094, 2004.

M. G. Lipe and S. E. Salterio. The balanced scorecard: Judgmental effects of common and unique performance measures. *The Accounting Review*, 75(3):283–298, 2000.

M. G. Lipe and S. E. Salterio. A note on the judgmental effects of the balanced scorecard's information organization. *Accounting, Organizations and Society*, 27(6):531–540, 2002.

B. R. Lord. Strategic management accounting: The emperor's new clothes? *Management Accounting Research*, 7(3):347–366, 1996.

R. F. Lusch and S. Nambisan. Service innovation: A service-dominant logic perspective. *MIS Quarterly*, 39(1):155–175, 2015.

O. Malgonde and K. Chari. An ensemble-based model for predicting agile software development effort. *Empirical Software Engineering*, 24(2):1017–1055, 2019.

R. Maltzman and D. Epstein. *Project Workflow Management: A Business Process Approach*. J. Ross Publishing, Florida, USA, 2013.

C. A. Maritan. Capital investment as investing in organizational capabilities: An empirically grounded process model. *Academy of Management Journal*, 44(3):513–531, 2001.

L. M. Maruping, V. Venkatesh, and R. Agarwal. A control theory perspective on agile methodology use and changing user requirements. *Information Systems Research*, 20(3):377–399, 2009.

C. Matt, T. Hess, and A. Benlian. Digital transformation strategies. *Business & Information Systems Engineering*, 57(5):339–343, 2015.

McKinsey & Company. Mastering automotive software-launch excellence: Automotive players can crack the code on superior launch performance by reducing complexity and increasing robustness in embedded software development., 2020. URL `https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/mastering-automotive-software-launch-excellence`. Date accessed: 17 Oct 2022.

McKinsey & Company. Cracking the complexity code in embedded systems development: How to manage—and eventually how to master—complexity in embedded systems development., 2021. URL `https://www.mckinsey.com/industries/advanced-electronics/our-insights/cracking-the-complexity-code-in-embedded-systems-development`. Date accessed: 17 Oct 2022.

T. Menzies and M. Shepperd. Special issue on repeatable results in software engineering prediction. *Empirical Software Engineering*, 17(1-2):1–17, 2012.

T. Menzies, Z. Chen, J. Hihn, and K. Lum. Selecting best practices for effort estimation. *IEEE Transactions on Software Engineering*, 32(11):883–895, 2006.

T. Menzies, Y. Yang, G. Mathew, B. W. Boehm, and J. Hihn. Negative results for software effort estimation. *Empirical Software Engineering*, 22(5):2658–2683, 2017.

K. A. Merchant and D. T. Otley. A review of the literature on control and accountability. *Handbooks of Management Accounting Research*, 2:7F85–802, 2006.

P. Meso and R. Jain. Agile software development: Adaptive systems principles and best practices. *Information Systems Management*, 23(3):19–30, 2006.

M. Messner. Does industry matter? How industry context shapes management accounting practice. *Management Accounting Research*, 31:103–111, 2016.

M. B. Miles, A. M. Huberman, and J. Saldaña. *Qualitative data analysis: A methods sourcebook*. SAGE Publications, Thousand Oaks, 4th edition, 2019.

R. E. Miles, C. C. Snow, A. D. Meyer, and H. J. Coleman. Organizational strategy, structure, and process. *The Academy of Management Review*, 3(3):546–562, 1978.

B. Mishra and I. Vaysman. Cost-system choice and incentives-traditional vs. activity-based costing. *Journal of Accounting Research*, 39(3):619–641, 2001.

D. Mishra and B. Mahanty. A study of software development project cost, schedule and quality by outsourcing to low cost destination. *Journal of Enterprise Information Management*, 29 (3):454–478, 2016.

K. Möller, U. Schäffer, and F. Verbeeten. Digitalization in management accounting and control: An editorial. *Journal of Management Control*, 31(1-2):1–8, 2020.

K. Moløkken-Østvold and M. Jørgensen. A comparison of software project overruns - flexible versus sequential development models. *IEEE Transactions on Software Engineering*, 31(9): 754–766, 2005.

K. Moløkken-Østvold, M. Jørgensen, S. S. Tanilkan, H. Gallis, A. C. Lien, and S. E. Hove. A survey on software estimation in the Norwegian industry. In *10th International Symposium on Software Metrics*, pages 208–219. IEEE, 2004.

K. Moløkken-Østvold, N. C. Haugen, and H. C. Benestad. Using planning poker for combining expert estimates in software projects. *Journal of Systems and Software*, 81(12):2106–2117, 2008.

K. Moser, H. Wolff, and A. Kraft. The de-escalation of commitment: Predecisional accountability and cognitive processes. *Journal of Applied Social Psychology*, 43(2):363–376, 2013.

T. Mukhopadhyay, S. S. Vicinanza, and M. J. Prietula. Examining the feasibility of a case-based reasoning model for software effort estimation. *MIS Quarterly*, 16(2):155, 1992.

T. E. Muller. Buyer response to variations in product information load. *Journal of Applied Psychology*, 69(2):300, 1984.

V. Nguyen. Improved size and effort estimation models for software maintenance. In *26th International Conference on Software Maintenance*, pages 1–2. IEEE Computer Society, 2010.

A. Niazi, J. S. Dai, S. Balabani, and L. Seneviratne. Product cost estimation: Technique classification and methodology review. *Journal of Manufacturing Science and Engineering*, 128(2):563–575, 2006.

E. Nie and I. Hammouda. An exploratory study on strategic software development outsourcing. In *12th International Conference on Global Software Engineering*, pages 106–115. IEEE, 2017.

E. Noreen. Conditions under which activity-based cost systems provide relevant costs. *Journal of Management Accounting Research*, 3:159–168, 1991.

T. D. Oesterreich, F. Teuteberg, F. Bensberg, and G. Buscher. The controlling profession in the digital age: Understanding the impact of digitisation on the controller's job roles, skills and competences. *International Journal of Accounting Information Systems*, 35:100432, 2019.

G. Ooi, C. Soh, and P. M. Lee. An activity based costing approach to systems development and implementation. In *Proceedings of the International Conference on Information Systems*, pages 341–345, Atlanta, GA, USA, 1998.

D. T. Otley. The contingency theory of management accounting: Achievement and prognosis. *Accounting, Organizations and Society*, 5(4):413–428, 1980.

D. T. Otley. The contingency theory of management accounting and control: 1980–2014. *Management Accounting Research*, 31:45–62, 2016.

S. V. Patil, F. Vieider, and P. E. Tetlock. Process versus outcome accountability. *The Oxford Handbook of Public Accountability*, pages 69–89, 2014.

S. Paul and D. L. Nazareth. Input information complexity, perceived time pressure, and information processing in GSS-based work groups: An experimental investigation using a decision schema to alleviate information overload conditions. *Decision Support Systems*, 49(1):31–40, 2010.

M. E. Porter. *Competitive Strategy: Techniques for Analyzing Industries and Competitors*. Free Press, New York, 1980.

L. H. Putnam. A general empirical solution to the macro software sizing and estimating problem. *IEEE Transactions on Software Engineering*, SE-4(4):345–361, 1978.

M. Quinn, O. Elafi, and M. Mulgrew. Reasons for not changing to activity-based costing: A survey of Irish firms. *PSU Research Review*, 1(1):63–70, 2017.

P. Rahmati, A. Tafti, J. Westland, and C. Hidalgo. When all products are digital: Complexity and intangible value in the ecosystem of digitizing firms. *MIS Quarterly*, 45:1025–1058, 2021.

H. Rastogi, S. Dhankhar, and M. Kakkar. A survey on software effort estimation techniques. In *5th International Conference - Confluence The Next Generation Information Technology Summit*, pages 826–830. IEEE, 2014.

T. Raz and D. Elnathan. Activity based costing for projects. *International Journal of Project Management*, 17(1):61–67, 1999.

V. Resmi and S. Vijayalakshmi. Analogy-based approaches to improve software project effort estimation accuracy. *Journal of Intelligent Systems*, 29(1):1468–1479, 2019.

T. Riasanow, G. Galic, and M. Böhm. Digital transformation in the automotive industry: Towards a generic value network. In *Proceedings of the European Conference on Information Systems*, 2017.

P. Riebel. *Einzelkosten- und Deckungsbeitragsrechnung.* Gabler Verlag, Wiesbaden, 1979.

S. Riezler. *Lebenszyklusrechnung: Instrument des Controlling strategischer Projekte.* Gabler Verlag, Wiesbaden, 1996.

J. Roberts. The possibilities of accountability. *Accounting, Organizations and Society*, 16(4): 355–368, 1991.

P. G. Roetzel. Information overload in the information age: A review of the literature from business administration, business psychology, and related disciplines with a bibliometric approach and framework development. *Business Research*, 12(2):479–522, 2019.

N. Roztocki. Using the integrated activity-based costing and economic value added information system for project management. In *7th Americas Conference on Information Systems*, pages 1454–1460, Boston, MA, 2001.

A. Saeed, W. H. Butt, F. Kazmi, and M. Arif. Survey of software development effort estimation techniques. In *Proceedings of the 7th International Conference on Software and Computer Applications*, pages 82–86, New York, USA, 2018. ACM.

A. G. Schick, L. A. Gordon, and S. Haka. Information overload: A temporal approach. *Accounting, Organizations and Society*, 15(3):199–220, 1990.

S. C. Schneider. Information overload: Causes and consequences. *Human Systems Management*, 7(2):143–153, 1987.

H. M. Schroder, M. J. Driver, and S. Streufert. *Human information processing: Individuals and groups functioning in complex social situations.* Holt, Rinehart and Winston, 1967.

S. Schulz-Hardt, J. Rollwage, S. K. Wanzel, J. U. Frisch, and J. A. Häusser. Effects of process and outcome accountability on escalating commitment: A two-study replication. *Journal of Experimental Psychology: Applied*, 27(1):112, 2021.

M. Schweitzer, H.-U. Küpper, G. Friedl, C. Hofmann, and B. Pedell. *Systeme der Kosten- und Erlösrechnung.* Vahlen, Munich, 11th edition, 2015.

S. Scott and W. Orlikowski. The digital undertow: How the corollary effects of digital transformation affect industry standards. *Information Systems Research*, 33(1):311–336, 2022.

L. Selander and S. L. Jarvenpaa. Digital action repertoires and transforming a social movement organization. *MIS Quarterly*, 40(2):331–352, 2016.

P. Seltsikas and W. L. Currie. Evaluating the application service provider (ASP) business model: The challenge of integration. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*, pages 2801–2809, Los Alamitos, CA, USA, 2002. IEEE Computer Society.

C. Shapiro and H. R. Varian. *Information Rules: A Strategic Guide to the Network Economy*. Harvard Business School Press, Boston, 1999.

M. Shepperd and C. Schofield. Estimating software project effort using analogies. *IEEE Transactions on Software Engineering*, 23(11):736–743, 1997.

M. D. Shields. An empirical analysis of firms ' implementation experiences with activity - based costing. *Journal of Management Accounting Research*, 7:148–166, 1995.

S. Shivendu and Z. Zhang. Versioning in the software industry: Heterogeneous disutility from underprovisioning of functionality. *Information Systems Research*, 26(4):731–753, 2015.

O. Shmueli, N. Pliskin, and L. Fink. Can the outside-view approach improve planning decisions in software development projects? *Information Systems Journal*, 26(4):395–418, 2016.

K. Siegel-Jacobs and J. F. Yates. Effects of procedural and outcome accountability on judgment quality. *Organizational Behavior and Human Decision Processes*, 65(1):1–17, 1996.

H. A. Simon. A behavioral model of rational choice. *The Quarterly Journal of Economics*, 69 (1):99–118, 1955.

H. A. Simon and A. Newell. Human problem solving: The state of the theory in 1970. *American Psychologist*, 26(2):145, 1971.

D. R. Smith. The design of divide and conquer algorithms. *Science of Computer Programming*, 5:37–58, 1985.

K. Srinivasan and D. Fisher. Machine learning approaches to estimating software development effort. *IEEE Transactions on Software Engineering*, 21(2):126–137, 1995.

F. Svahn, L. Mathiassen, and R. Lindgren. Embracing digital innovation in incumbent firms: How Volvo cars managed competing concerns. *MIS Quarterly*, 41(1):239–253, 2017.

M. R. Swain and S. F. Haka. Effects of information load on capital budgeting decisions. *Behavioral Research in Accounting*, 12:171, 2000.

B. Swar, T. Hameed, and I. Reychav. Information overload, psychological ill-being, and behavioral intention to continue online healthcare information search. *Computers in Human Behavior*, 70:416–425, 2017.

T. Tani. Interactive control in target cost management. *Management Accounting Research*, 6 (4):399–414, 1995.

R. C. Tausworthe. The work breakdown structure in software project management. *Journal of Systems and Software*, 1:181–186, 1979.

P. E. Tetlock. Accountability and complexity of thought. *Journal of Personality and Social Psychology*, 45(1):74, 1983.

P. E. Tetlock and R. Boettger. Accountability: A social magnifier of the dilution effect. *Journal of Personality and Social Psychology*, 57(3):388, 1989.

P. E. Tetlock, L. Skitka, and R. Boettger. Social and cognitive strategies for coping with accountability: Conformity, complexity, and bolstering. *Journal of Personality and Social Psychology*, 57(4):632, 1989.

A. Trendowicz and R. Jeffery. *Software project effort estimation: Foundations and best practice guidelines for success*. Springer, Cham, 2014.

A. Trendowicz, M. Ochs, A. Wickenkamp, J. Münch, Y. Ishigai, and T. Kawaguchi. Integrating human judgment and data analysis to identify factors influencing software development productivity. *e-Informatica*, 2:47–69, 2008.

A. Trendowicz, J. Münch, and R. Jeffery. State of the practice in software effort estimation: A survey and literature review. In *Software engineering techniques*, volume 4980 of *LNCS sublibrary, Programming and software engineering*, pages 232–245. Springer, Heidelberg, 2011.

M. L. Tushman and D. A. Nadler. Information processing as an integrating concept in organizational design. *Academy of Management Review*, 3(3):613–624, 1978.

M. Usman, E. Mendes, and J. Börstler. Effort estimation in agile software development. In *Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering*, pages 1–10, New York, USA, 2015. ACM.

M. Usman, K. Petersen, J. Börstler, and P. Santos Neto. Developing and using checklists to improve software effort estimation: A multi-case study. *Journal of Systems and Software*, 146: 286–309, 2018.

M. van Genuchten. Why is software late? An empirical study of reasons for delay in software development. *IEEE Transactions on Software Engineering*, 17(6):582–590, 1991.

D. van Knippenberg, L. Dahlander, M. Haas, and G. George. Information, attention, and decision making. *Academy of Management Journal*, 58(3):649–657, 2015.

A. R. Venkatachalam. Software cost estimation using artificial neural networks. In *Proceedings of 1993 International Conference on Neural Networks*, volume 1, pages 987–990, 1993.

A. Vermerris, M. Mocker, and E. van Heck. No time to waste: The role of timing and complementarity of alignment practices in creating business value in it projects. *European Journal of Information Systems*, 23(6):629–654, 2014.

G. Vial. Understanding digital transformation: A review and a research agenda. *The Journal of Strategic Information Systems*, 28(2):118–144, 2019.

S. S. Vicinanza, T. Mukhopadhyay, and M. J. Prietula. Software-effort estimation: An exploratory study of expert performance. *Information Systems Research*, 2(4):243–262, 1991.

R. Vidgen and X. Wang. Coevolving systems and the organization of agile software development. *Information Systems Research*, 20(3):355–376, 2009.

Volkswagen Group. Volkswagen strengthens new software organization, 2019. URL `https://www.volkswagen-newsroom.com/en/press-releases/volkswagen-strengthens-new-software-organization-5607`. Date accessed: 17 Oct 2022.

K. S. Warner and M. Wäger. Building dynamic capabilities for digital transformation: An ongoing process of strategic renewal. *Long Range Planning*, 52(3):326–349, 2019.

X. Wei and B. R. Nault. Monopoly versioning of information goods when consumers have group tastes. *Production and Operations Management*, 23(6):1067–1081, 2014.

R. L. Wright. Measuring the precision of statistical cost allocations. *Journal of Business & Economic Statistics*, 1(2):93–100, 1983.

M. T. Wynn, W. Z. Low, A. H. M. Ter Hofstede, and W. Nauta. A framework for cost-aware process management: Cost reporting and cost prediction. *Journal of Universal Computer Science*, 20(3):406–430, 2014.

D. Yang, Q. Wang, M. Li, Y. Yang, K. Ye, and J. Du. A survey on software cost estimation in the Chinese software industry. In *Proceedings of the 2nd International Symposium on Empirical Software Engineering and Measurement*, page 253, New York, USA, 2008. ACM.

R. K. Yin. *Case study research and applications: Design and methods.* SAGE, Los Angeles, 6th edition, 2018.

Y. Yoo, O. Henfridsson, and K. Lyytinen. Research commentary–the new organizing logic of digital innovation: An agenda for information systems research. *Information Systems Research*, 21(4):724–735, 2010.

R. Zarnekow and W. Brenner. Distribution of cost over the application lifecycle–a multi-case study. In *Proceedings of the European Conference on Information Systems*, 2005.