TKK Dissertations 186
Espoo 2009

AGILE SOFTWARE DEVELOPMENT IN LARGE-SCALE NEW PRODUCT DEVELOPMENT ORGANIZATION: TEAM-LEVEL PERSPECTIVE

Doctoral Dissertation

Petri Kettunen

Helsinki University of Technology
Faculty of Information and Natural Sciences
Department of Computer Science and Engineering

TKK Dissertations 186
Espoo 2009

AGILE SOFTWARE DEVELOPMENT IN LARGE-SCALE NEW PRODUCT DEVELOPMENT ORGANIZATION: TEAM-LEVEL PERSPECTIVE

Doctoral Dissertation

Petri Kettunen

Dissertation for the degree of Doctor of Science in Technology to be presented with due permission of the Faculty of Information and Natural Sciences for public examination and debate in Auditorium T2 at Helsinki University of Technology (Espoo, Finland) on the 4th of December, 2009, at 12 noon.

Helsinki University of Technology
Faculty of Information and Natural Sciences
Department of Computer Science and Engineering

Teknillinen korkeakoulu
Informaatio- ja luonnontieteiden tiedekunta
Tietotekniikan laitos

Distribution:
Helsinki University of Technology
Faculty of Information and Natural Sciences
Department of Computer Science and Engineering
P.O. Box 9210 (Tekniikantie 14)
FI - 02015 TKK
FINLAND
URL: http://www.cse.tkk.fi/
Tel. +358-9-451 4851
Fax +358-9-451 4958
E-mail: petri.kettunen@tkk.fi

ISBN 978-952-248-113-9
ISBN 978-952-248-114-6 (PDF)
ISSN 1795-2239
ISSN 1795-4584 (PDF)
URL: http://lib.tkk.fi/Diss/2009/isbn9789522481146/

TKK-DISS-2655

Multiprint Oy
Espoo 2009

ABSTRACT OF DOCTORAL DISSERTATION	HELSINKI UNIVERSITY OF TECHNOLOGY P.O. BOX 1000, FI-02015 TKK http://www.tkk.fi

Author Petri Kettunen

Name of the dissertation
Agile Software Development in Large-Scale New Product Development Organization: Team-Level Perspective

Manuscript submitted June 1, 2009	Manuscript revised November 5, 2009

Date of the defence December 4, 2009

☐ Monograph ☒ Article dissertation (summary + original articles)

Faculty	Information and Natural Sciences
Department	Department of Computer Science and Engineering
Field of research	Software Engineering
Opponent(s)	Prof. Markku Oivo (University of Oulu)
Supervisor	Prof. Tomi Männistö
Instructor	Prof. Pekka Abrahamsson (University of Helsinki)

Abstract

Many modern intelligent products and systems (e.g., automotive, consumer electronics, telecommunications) contain more and more embedded software. Often the new product development (NPD) companies developing such products operate under turbulent circumstances stemming from the business environment, technology development and other, even disruptive sources. The embedded software development functions of such NPD organizations then face the uncertainties directly or indirectly, often coupled with time-to-market, quality and productivity pressures. Agile software development has been advocated as a new way of coping with such circumstances in particular with small independent teams developing customer-driven software products. This thesis investigates in contrast how it can be utilized with embedded software development teams in large-scale market-driven industrial NPD context.

The exploratory, problem-driven research process is based on interpretive design science and action research principles. The author worked as a full-time software quality and process development specialist employee inside the case organization, thus acting as a reflective practitioner. The longitudinal study research cycles were conducted over several years in that particular NPD organization context. The cycle viewpoints evolved from first recognizing typical software project problems and uncertainties, and developing certain solutions to software team knowledge management and software process model selection. This development led to consider, what problems current agile software methods address. The realization of agile software development was then further examined with respect to the cost factors, and finally towards integrating agile software product development teams into larger-scale NPD organization.

The main result of this research is that agile software development models address many typical key issues in large-scale industrial NPD context, and the cost/benefit factors are in principle justifiable. However, if agile software methods are applied just bottom-up trying to integrate isolated agile software teams into larger organizational context, this inside-out approach leads often to problems with organizational barriers and impediments. Thus, in order to be able to leverage the potential benefits, agile software development should be approached more from the strategic business perspective (outside-in), viewing the software development functions as elements of the total value-creation system in the NPD organization. Different software development (project) teams may have different roles and needs for agility in this complex over time.

The contributions imply that rational software team-level improvements require in many cases wider, even enterprise-level perspectives in creating and improving the agile capabilities of the NPD organization. It is thus fundamental to conceptualize agility in the NPD context by combining software development with the overall NPD processes. In particular in large organizations, the improvements may require more actions at the organizational level than in software teams.

Keywords process improvement, agile software methods, embedded software, new product development

ISBN (printed) 978-952-248-113-9	ISSN (printed) 1795-2239
ISBN (pdf) 978-952-248-114-6	ISSN (pdf) 1795-4584
Language English	Number of pages 153 p. + app. 95 p.

Publisher Helsinki University of Technology, Department of Computer Science and Engineering

Print distribution Helsinki University of Technology, Department of Computer Science and Engineering

☒ The dissertation can be read at http://lib.tkk.fi/Diss/2009/isbn9789522481146/

VÄITÖSKIRJAN TIIVISTELMÄ	TEKNILLINEN KORKEAKOULU PL 1000, 02015 TKK http://www.tkk.fi

Tekijä Petri Kettunen

Väitöskirjan nimi
Ketterä ohjelmistokehitys isossa tuotekehitysorganisaatiossa: tiimitason näkökulma

Käsikirjoituksen päivämäärä 1.6.2009	Korjatun käsikirjoituksen päivämäärä 5.11.2009

Väitöstilaisuuden ajankohta 4.12.2009

☐ Monografia	☒ Yhdistelmäväitöskirja (yhteenveto + erillisartikkelit)

Tiedekunta Informaatio- ja luonnontiede
Laitos Tietotekniikka
Tutkimusala Ohjelmistotuotanto
Vastaväittäjä(t) Prof. Markku Oivo (Oulun yliopisto)
Työn valvoja Prof. Tomi Männistö
Työn ohjaaja Prof. Pekka Abrahamsson (Helsingin yliopisto)

Tiivistelmä

Monet nykyaikaiset älykkäät tuotteet ja järjestelmät (esim. autotekniikka, kulutuselektroniikka, telekommunikaatio) sisältävät yhä enemmän upotettuja ohjelmistoja. Usein tällaisia tuotteita kehittävät (NPD) yhtiöt toimivat turbulenteissa oloissa aiheutuen liiketoimintaympäristöstä, teknologian kehityksestä ja muista, usein yllättävistäkin lähteistä. Tällaisten NPD-organisaatioiden upotettujen ohjelmistojen kehitysfunktiot kohtaavat epävarmuudet suoraan tai epäsuorasti, usein julkaisuaikataulu-, laatu- ja tuottavuuspaineiden kera. Ketterää ohjelmistokehitystä on pidetty uutena tapana toimia sellaisissa tilanteissa erityisesti pienissä itsenäisissä, asiakaskohtaisia ohjelmistotuotteita kehittävissä tiimeissä. Tämä väitöstyö tutkii sitä vastoin, miten sitä voidaan hyödyntää upotettujen ohjelmistojen kehitystiimeissä isossa markkinaorientoituneessa teollisessa NPD-kontekstissa.

Tämä kartoittava, ongelmalähtöinen tutkimusprosessi perustuu tulkinnallisiin suunnittelutieteen ja toimintatutkimuksen periaatteisiin. Tutkija työskenteli kokopäiväisenä ohjelmistojen laatu- ja prosessispesialistina tapausorganisaatiossa, ollen näin reflektoiva praktiikassa. Pitkittäistutkimuksen syklit tehtiin useiden vuosien kuluessa ko. NPD-organisaatiokontekstissa. Syklien näkökulmat kehittyivät alkaen tyypillisten ohjelmistokehitysongelmien ja epävarmuuksien tunnistamisesta eräiden ratkaisutapojen kehittämiseen ohjelmistotiimien tiedonhallintaan ja ohjelmistoprosessimallien valintaan. Tämä kehitys johti selvittämään, mitä ongelmia nykyiset ketterät ohjelmistomenetelmät koskettavat. Ketterän ohjelmistokehityksen toteuttamista tutkittiin sitten edelleen kustannustekijöiden ja lopuksi ketterien ohjelmisto-tuotekehitystiimien integroinnin suhteen laajamittaiseen NPD-organisaatioon.

Tutkimuksen päätulos on, että ketterät ohjelmistokehitysmallit käsittelevät monia tyypillisiä avainongelmia laajoissa teollisissa NPD-organisaatioissa, ja kustannus/hyötytekijät ovat periaatteessa perusteltavissa. Jos ketteriä ohjelmistomenetelmiä kuitenkin sovelletaan vain alhaalta-ylös pyrkien integroimaan erillisiä ketteriä ohjelmistotiimejä laajempaan organisaatioyhteyteen, tällainen sisältä-ulos -lähestymistapa johtaa usein ongelmiin organisatoristen rajojen ja esteiden takia. Siksi ketterän ohjelmistokehityksen täysimittaiseksi hyödyntämiseksi sitä olisi tarkasteltava strategisesta liiketoimintanäkökulmasta (ulkoa-sisään) tarkastellen ohjelmistokehitysfunktioita elementteinä NPD-organisaation koko arvontuottosysteemissä. Eri ohjelmistokehitys(projekti)tiimeillä voi olla eri rooleja ja vaatimuksia tässä kompleksissa ajan suhteen.

Tutkimuksen kontribuutiot implikoivat, että rationaaliset parannukset ohjelmistotiimitasolla vaativat useissa tapauksissa laajempaa, jopa yritystason perspektiiviä NPD-organisaation ketteryyskyvykkyyksien luomisessa ja parantamisessa. Siksi on perustavan tärkeää käsitteellistää ketteryys NPD-kontekstissa yhdistämällä ohjelmistokehitys koko NPD-prosessiin. Erityisesti isoissa organisaatioissa parannukset saattavat vaatia enemmän toimia organisaatiotasolla kuin ohjelmistotiimeissä.

Asiasanat	prosessinkehitys, ketterät menetelmät, upotetut ohjelmistot, tuotekehitys		
ISBN (painettu)	978-952-248-113-9	ISSN (painettu)	1795-2239
ISBN (pdf)	978-952-248-114-6	ISSN (pdf)	1795-4584
Kieli	englanti	Sivumäärä	153 s. + liit. 95 s.
Julkaisija	Teknillinen korkeakoulu, Tietotekniikan laitos		
Painetun väitöskirjan jakelu	Teknillinen korkeakoulu, Tietotekniikan laitos		
☒ Luettavissa verkossa osoitteessa http://lib.tkk.fi/Diss/2009/isbn9789522481146/			

Preface

Academic dissertations have much in common with expeditions. There is a bold overall goal coupled with many uncertainties about the execution and final outcome. Careful preparations and appropriate equipment as well as persistent but flexible and nimble conduct are necessary for success. This is the essential nature of research work.

Considering the subject of this thesis, the competitive environments of many product development companies are nowadays ñbitter coldò. Like with expeditions, definite preplanned goals are not enough, but also sensible flexibility in action is required. Agility is the general business concept for prosperity under such conditions. Such characteristics have been necessary during this thesis work, too. Indeed, major changes and even turbulence occurred both globally and personally while writing this dissertation between the years 2007-2009.

This thesis represents in a way the current state of my learning and experience gained during my 20-year long industrial career in software product development, although the actual research work is based on the past five years in the telecommunications sector. It is thus impossible to give credit to all the persons who have influenced me either directly or subconsciously over the years. However, I would like to thank a few them by name in here since they have had major impacts in the course of this thesis work.

Maarit Laanti (Nokia Corporation) is the co-author of a part of the publications of this thesis. We used to work together in the same software project at the time of the initiation of the research activities, and had many inspiring and thought-provoking discussions of many of the ideas presented in this thesis (and much more). Her impact has thus been profound. Maarit is the one who initially urged me to write a doctoral thesis. I appreciate the encouragement and insightful suggestions over the years.

Maarit also introduced me to Prof. Pekka Abrahamsson (now University of Helsinki), who acted subsequently as my Instructor. He has had ï being a distinguished researcher ï a major impact for bridging the gap between scientific research work and my industrial practice. It is largely thanks to him that I realized how to present my contributions in the form of a concise summary.

Pekka in turn connected me to Prof. Tomi Männistö at Helsinki University of Technology (SoberIT) who then became my Supervisor. Tomi guided me through the milestones of dissertation work in a supportive and enthusiastic way. He also gave certain insightful scientific comments, complementing Pekkaôs instructions.

Finally, the pre-examiners Prof. Frank Maurer (University of Calgary) and Prof. Markku Tukiainen (University of Joensuu) had a major responsibility for reviewing the dissertation. I appreciate their efforts and improvement comments in particular emphasizing my role as a reflective practitioner.

In addition, I had an opportunity to finalize the thesis writing with the financial support of the Graduate School on Software Systems and Engineering (SoSE). I acknowledge that appraisal.

Overall, although I have made most of my work career in the Helsinki metropolitan area, there are many close connections to my home town Kuusankoski (now Kouvola). I thank my parents and sister for their background support, and I am especially grateful to Pinja, Pauli, and Pihla for the many inspiring discussions round the kitchen table.

Espoo, November 2009

Petri Kettunen

Table of Contents

List of Original Publications

I Kettunen, P., 2006. Troubleshooting large-scale New Product Development embedded software projects. In: Proc. 7th Intõl Conf. Product Focused Software Process Improvement (PROFES), pp. 61-78.

II Kettunen, P., 2003. Managing embedded software project team knowledge. IEE Proceedings: Software 150(6), 359-366.

III Kettunen[1], P., Laanti, M., 2005. How to steer an embedded software project: tactics for selecting the software process model. Information and Software Technology 47(9), 587-608.

IV Kettunen[2], P., Laanti, M., 2006. How to steer an embedded software project: tactics for selecting agile software process models. International Journal of Agile Manufacturing 9(1), 59-77.

V Laanti, M., Kettunen[3], P., 2006. Cost Modeling Agile Software Development. International Transactions on Systems Science and Applications 1(2), 175-179.

VI Kettunen, P., 2007. Extending Software Project Agility with New Product Development Enterprise Agility. Software Process: Improvement and Practice 12(6), 541-548.

[1] idea formalization and analysis, main content writing
[2] idea development, main content writing, shared conference presentation
[3] idea development, main content writing

List of Abbreviations

ASD	Adaptive Software Development
B2B	Business-to-Business
B2C	Business-to-Customers
CASE	Computer-Aided Software Engineering
CMM	Capability Maturity Model
CMMI	Capability Maturity Model Integrated
CSF	Critical Success Factor
DSDM	Dynamic Systems Development Method
FDD	Feature-Driven Development
FMS	Flexible Manufacturing System
HR	Human Resources
HRD	Human Resource Development
HW	Hardware
IS	Information Systems
ISD	Information Systems Development
IT	Information Technology
LOC	Lines-of-Code
NPD	New Product Development
NPV	Net Present Value
OD	Organization Development
OPF	OPEN Process Framework
OT	Organization Theory
R&D	Research and Development
ROI	Return-on-Investment
RUP	Rational Unified Process
SPI	Software Process Improvement
SW	Software
TDD	Test-Driven Development
TQM	Total Quality Management
UCD	User Centered Design
WiMAX	Worldwide Inter-operability for Microwave Access
XP	eXtreme Programming
3G	3rd Generation (telecommunications technology)

List of Figures

List of Tables

1 Introduction

This opening section of the thesis compendium outlines briefly the domain, overall drivers, and the motives of the research (Sect. 1.1). The quest for agility in product development in general and agile software development in particular is reasoned, leading to the specific research questions of the thesis (Sect. 1.2). The focus and scope of the work are then refined (Sect. 1.3). The research steps (papers) are introduced, highlighting the key results (Sect. 1.4). Finally, the structure of this thesis summary is explained (Sect. 1.5).

1.1 Motivation

Many modern intelligent electronic devices and equipment include more and more software-based functionality. For example digital cameras, mobile phones, and home-entertainment equipment are typically such products ï just to name a few. Furthermore, embedded software can increasingly be found in such products as automotive and even white goods (e.g., microwave ovens). The role of embedded software in future European automotive industry value creation is for instance expected to almost double (up to 35%) by the year 2015 (TNO/IDATE 2005; Tuormaa 2009).

In addition to such mass market consumer products (business-to-customer), many systems products (business-to-business) are software-intensive. For example 3G telecommunications, digital TV broadcasting, and broadband Internet networks rely heavily on complex software-based components.

New product development (NPD) companies make their business by creating and manufacturing those products. Notably, because of the embedded nature, many such companies do not necessarily appear as software-intensive organizations.

The competitive environment of many NPD companies is nowadays dynamic and often increasingly turbulent (Doz and Kosonen 2008). The sources of uncertainties and turbulence can be manifold and intertwined, stemming from such factors as for example

- customers / markets
- competition
- technology
- regulations
- globalization
- organization
- social and environmental factors

Such turbulence factors affect the embedded software development functions of the company either directly or indirectly. Consequently, the software development faces not only the inherent product complexities and embedded software engineering technical difficulties, but also the many external business-related disturbances. The

unpredictable and dynamic nature of the current competitive environments of many NPD organizations characterized above calls for new models of software product development.

Agile software development has been advocated as a prominent solution for many such software production problems; see for example (Highsmith and Cockburn 2001; ITEA 2004; Boehm and Turner 2005). Agility has also been applied in other disciplines facing similar business problems like manufacturing.

In general, agility is a capability of an organization (entity) relative to its environment. There is no one generally agreed definition of agility, but the essential characteristics are proficiency at change and responsiveness under unpredictable environmental conditions. As such a broad concept, agility is not unique to NPD or software development. We elaborate and scrutinize the concepts of agility in detail in Sect. 2.2.2, but for the purposes of this investigation we follow the generic definition by Conboy and Fitzgerald (2004):

- ***ñ****the continual readiness of an entity to rapidly or inherently, proactively or reactively, embrace change, through high quality, simplistic, economical components and relationships with its environment****ò***

Software process improvement (SPI) is the overall activity of developing the software production capabilities of the organization. This entails in general setting the strategic goals and target levels, evaluating the current capabilities with respect to the targets, and executing improvement activities accordingly. In particular in large companies the improvements could range from software engineering technical items up to organizational issues, relating to process improvement and organizational development in general. In turbulent NPD environments the improvement activities should continuously be adjusted with the current strategic goal alignments and environmental factors, which are often subject to even disruptive changes.

This thesis investigates embedded software product development (in devices) and its software process improvement with respect to agility in large-scale industrial NPD organizational context. Agile software development on the one hand and flexible NPD on the other have been active research topics for some years now, and there appears to be even more growing interests to adopt them in the industry because of the fundamental changes taking place in many competitive environments. This work is primarily initiated at the software (project) team level, but it is subsequently extended towards the enterprise level organizational capabilities, and even to the business milieu.

1.2 Research Problem

There can be many possible, not always obvious software process improvement issues in any large NPD organization. The nature and weighting of the different problems and improvement areas vary across different companies depending for example their current contextual factors, strategic choices, and historical background (Abrahamsson 2002; Börjesson 2006; Fugetta 2000; Humphrey et al. 2007; Liu et al. 2006; Ojala 2006; Taramaa et al. 1998). Consequently, conceptualization and more systematic framing of the problem space are needed.

Agile software development models offer several prospective solutions to many software production problems. However, so far agile software development has been investigated mostly at a small-scale team level, but it is not well understood how it really works when integrated into large-scale NPD organizational context.

Following that line of thinking leads to the overall goal of this research to gain deeper, holistic understanding of primarily embedded software development agility and its consequent SPI potential in large-scale NPD environments. Traditionally, those areas have been investigated separately in different disciplines, and they are not well understood in combination. Hence, we formulate the principal research problem as follows:

How can agile software development be utilized in large-scale NPD context?

This is a complex multi-issue problem. We approach the compound problem with the following research questions and viewpoints:

1. What are the typical problems of large-scale NPD embedded software projects?
2. What problems and goals does agile software development address?
3. How can typical large-scale NPD problems be tackled with agile software development methods?

From an industrial NPD organization point of view, this research problem setting addresses the following organizational development goals:

(i) recognizing and understanding embedded software project team problem areas and consequently process improvement items specifically with respect to agility
(ii) understanding the relationships with the surrounding NPD organizational context
(iii) distilling the impacts up to the NPD enterprise level

The fundamental question of why to contemplate agility improvement stems from the industrial background of this particular research environment as described in Sect. 3.2. The environment has many such characteristics, which are often advocated as the ñhome groundò for agile software development (major uncertainties in new products, high productivity/time-to-market pressures, and yet high product quality expectations).

However, the product development case environment also has many such elements, which are often mentioned as less favorable for agile software development models:

- large, complex product systems
- embedded software development (mission-critical and partially real time)
- large, even globally distributed development teams and program structures

It is not clear how beneficial software development agility really is in such industrial NPD environments, and whether other improvement areas would bring more advantages. Indeed, like stated above, a general aim of this research work is to understand the NPD environmental and situational factors favoring or possibly contradicting agile software product development. Such holistic understanding would make it possible to exploit agile software development appropriately in order to gain

competitive advantages, and also to realize areas for additional improvement needs in context.

Recapping, the research problem set above can be decomposed and framed like shown in Table 1. The Goal-Question-Metric template aggregates the essence of this entire research effort (Kujala 2007).

Table 1. GQM-oriented framing of the research problem

Element	This Research
Analyze:	(how) agile software development
In order to:	understand, rationalize, exploit (SPI)
With respect to:	benefits/costs, applicability
From perspective of:	embedded software teams
In the context of:	case NPD organization
Because:	competitive environment (needs, drivers)

1.3 Research Scoping

This research work is conducted in a large industrial product development organization context as defined in Sect. 3.2. The scope of the empirical research is thus limited to one particular industrial organization setting. This constrains the field of industry to one sector of telecommunications, and within that sector into a single ï though a major one ï company only. Furthermore, inside this particular large company environment only certain product development units/projects are covered.

Large (more than 100 employees) industrial product development organizations are by nature structured into smaller work units in one way or another (Trott 2005; Ulrich and Eppinger 2000). The primary scope of this research stems from the software team level, but it is subsequently expanded towards the larger-scale NPD organizational levels (see Fig. 1). In particular with large complex systems products, there are usually multiple different software teams and also concurrent development projects (multiproject programs). Furthermore, with embedded software development, it is generally necessary to consider also the related hardware and product systems engineering functions to some extent.

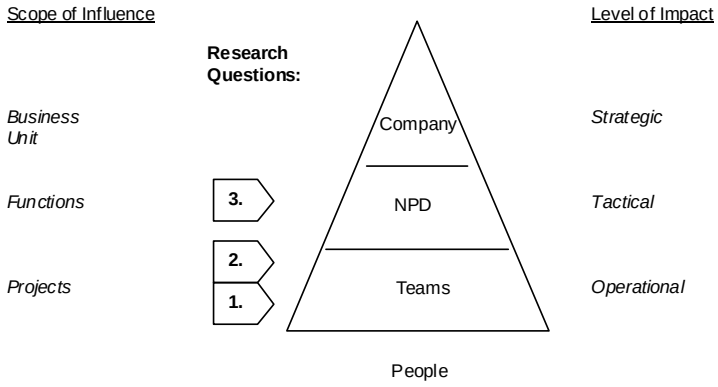

Fig. 1. Organizational leveling of the research questions

In practice, such a large multidisciplinary research problem space cannot be covered completely in detail in one investigation. Many sectors of this problem space are distinct fields of research on their owns. In particular, the following areas are beyond the focal scope of this thesis:

- embedded software engineering (design) technology (e.g., CASE) and tools
- other NPD disciplines than electronics (e.g., industrial design, mechatronics)
- product innovation (e.g., UCD, open sourcing)
- general large-scale organizational management (e.g., financing, communications, subcontracting, multisite structures)
- organizational culture (ideology) and dynamics (behavior, ecosystems), people and social factors (e.g., wellbeing, motivation, rewarding)

In general, the intention of this thesis work is not to provide particular case-specific evidence of how beneficial agile software development can be nor how exactly in practice it can be implemented (deployed) in real-life organizations. The practical realization of the changes and improvements is thus mostly outside the scope. The research design (Chapter 3) takes this intentional scoping into account. We discuss the impacts of these practical constraints and limitations in Chapter 5.

1.4 Overview of the Thesis

The conclusions of this research work derive from the longitudinal stream of process improvement studies reported in the research papers I-VI (see pp. vii). The stream follows the decomposed research questions (1-3) and viewpoints stated in Sect. 1.2.

We began by exploring the overall problem space of embedded software product development (Research Paper I). This investigation resulted in problem and uncertainty profiles of individual software project teams.

Based on the project problem profiling, certain problem areas calling for new solutions were selected for further studies. One of the reflections is that knowledge

management is an intrinsic part of embedded software product development, although the formal term ôknowledge managementô is not always underlined. We therefore focused first on the problems of managing knowledge and information at the software team level (Research Paper II). This work produced certain practical tools to address the information and knowledge management needs of embedded software product development projects.

The problem-based project management perspective and the specific but partial solutions with the knowledge management area (Research Paper II) led to the insight that also more comprehensive and overarching project steering solutions would be needed in uncertain environments. This line of thinking brought up the idea that a fundamental part of software product development and project management is the life-cycle model. There are many different software development process models proposed over the past few years. In general, each software process model addresses certain decision (problem) areas of software projects, and the project organization must select an appropriate model in the particular project context. Our next research question was therefore to compare and contrast systematically certain software process models with respect to how they address certain key areas of embedded software product development (Research Paper III). This work is partially based on the knowledge gained with the project problem profiles (Research Paper I).

Those problem-based investigations brought up the question of whether agile software development models could be an effective solution path. We therefore developed the software process comparison (Research Paper III) further to focus more on agile software process models (Research Paper IV). We discovered that agile software process models address certain problem areas more effectively than the compared traditional models. This is in particular under uncertain development conditions when the project steering tactics are based on frequent adjustments and feedback-driven development cycles. On the other hand many larger-scale organizational areas are beyond the scope of the current agile software development models. Consequently, we directed the research work towards those concerns, and delved into NPD agility in more general.

An important part of process improvement (SPI) efforts is not only to develop point solutions but also to understand the cost/benefit factors of improving the organizational capabilities and requisite enabling factors. For instance efficient knowledge management (Research Paper II) is typically one of the key enablers of building such capabilities. Consequently, there is a need to analyze the realizable benefits and associated costs of agile software development solutions. We therefore examined next certain general cost models of agility in respect of software product development (Research Paper V). The findings and observations indicate that agile software development should be matched with the overall NPD performance goals of the organization in terms of the cost-effectiveness and agility. System-level value stream mapping is one way of doing this in practice.

Finally, by understanding the needs and drivers of agility (Research Paper I-III) and the insights gained with the solutions (Research Paper IV) coupled with the cost analysis (Research Paper V), it becomes obvious that in large organization environments it is not enough to work at the software project team level alone. A more comprehensive agile perspective is thus needed up to the NPD enterprise level to realize the full benefits at the team level (Research Paper VI).

In summary, by synthesizing the different findings of all the research steps and inferring the conclusions, the main contribution of this thesis is the realization that in large-scale industrial NPD context beneficial agile software development requires a holistic view considering not only the software engineering function, but also the related product engineering functions together with the overall business model of the organization. Notably there is an intentional cross-discipline perspective (even ñboundary spanningò) here.

While agile software development has been investigated actively over the past few years independently mostly at team-level, the cross-functional combination with the NPD discipline is to our knowledge an under-researched area. This thesis attempts to fill that gap.

1.5 Structure of the Summary

The rest of this thesis compendium is structured as follows. Following this introductory Chapter 1, Chapter 2 surveys the previous published work highlighting the knowledge gaps. This grounding begins with an overview of contemporary new product development concepts in general, and modern flexible NPD models in particular. The role of software production within the NPD context is then scoped. Agility in NPD and agile software development are defined. This exploration justifies the research problem, and rationalizes the research aims of the thesis.

Chapter 3 explains the research design. The constructive research approach is explained with the connections to the industrial case environment. The research rigor and criteria are set.

Next, Chapter 4 presents the detailed results and findings of the research. The presentation is organized following the research questions, and the individual research papers I-VI are linked accordingly. The main research problem can be answered by combining and synthesizing all the stepwise results.

The results section is followed by analysis discussion in Chapter 5 further elaborating the answer to the research problem. The related research literature and the targeted knowledge gaps are contrasted. The overall research work of the thesis is then scrutinized, and the quality is assessed. Moreover, certain inferred implications and recommendations are presented. Due to the interdisciplinary nature of the research problem, the implications are also manifold considering both theory and practice.

Finally, Chapter 6 summarizes the contributions of the thesis. The research work done here opens up many new ideas and potential avenues for future studies.

2 Theoretical Framing and Foundations

This literature-based background section reviews the key concepts of the thesis research field and surveys the prior published research work. Based on that exploration, the knowledge gaps can be highlighted and the research questions of the thesis set in Sect. 1.2 can be put into appropriate context and grounded to the knowledge base.

The review begins with a general overview of software-intensive NPD in (large-scale) organizational context (Sect. 2.1). There are many related business competence knowledge areas (e.g., marketing) and disciplines. It is important to realize the overall positioning and interdependencies, although most of them are beyond the scope of this thesis. The basic NPD concepts are then explored in more detail. Typical NPD project success factors are surveyed. The conventional models have evolved towards modern flexible NPD processes for uncertain and turbulent conditions. Finally, software development in this NPD context is characterized.

Following the development of general flexible NPD models, agility is featured (Sect. 2.2). The concepts and principles are first reviewed generally in different disciplines. NPD agility can then be reasoned. This is continued with the overall approaches to develop agile capabilities in (large-scale) organizational environments, taking into account typical obstacles observed in practice.

Agile software development as currently known and typically practiced is then examined (Sect. 2.3). The linkage to NPD is established. Organizational adoption of agile software development methods is discussed.

Finally, the research rationale of the thesis is shown based on the above groundwork (Sect. 2.4). The new research is connected to the prior scholarship and research streams.

2.1 Flexible Software-Intensive New Product Development

In order to be able to realize the nature of the software development problem space in large-scale NPD context, it is first necessary to understand the basic technology-independent principles and concepts of new product creation (Sect. 2.1.1). On that basis it can then be seen, which factors are typical keys to successful NPD projects in traditional business environments. However, many modern competitive environments are much more uncertain, requiring more advanced NPD models (Sect. 2.1.2). Following that groundwork, the nature and problems of embedded software development in the modern NPD environments can be comprehended (Sect. 2.1.3).

2.1.1 Overview of NPD Competence Area

It is first instructive to view new product development in the total business competence knowledge space (2.1.1.1). The traditional basic NPD process models are then overviewed (2.1.1.2), coupled with typically recognized success/failure factors (2.1.1.3).

2.1.1.1 *Knowledge Space*

This primary field of study of this research is software engineering. However, in particular in large NPD organizational context, there are several other related disciplines.

Industrial product development can be defined as the commercial exploitation of market needs and opportunities with new products utilizing the available technological possibilities (Krisnan and Ulrich 2001). It is important to understand in general how embedded software development relates to them in order to be able to achieve comprehensive competitive advantage improvements. Notably in some cases the main focus the improvement actions should actually be set outside the software development function.

It is therefore essential to understand the nature of the product development (including software) and the related discipline functions in the organizational context, such as business and strategy management, collaboration, and innovation (Nambisan 2003; TEKES 2006). Those competences[4] together make up the organizational capabilities providing competitive advantages for the NPD organizational entity. However, it is not generally agreed what all areas comprise business competence, and the definitions vary (Näsi and Neilimo 2006). The competitive advantages and the required capabilities are relative to the particular competitive environment of the organization or unit (Chakravarthy 1997).

There are several general schemes to organize this body of knowledge. For instance Day (1994) develops a framework of market-oriented organizational processes. He views the NPD function as a spanning process like illustrated in Fig. 2. The key point is to see market-orientation as the total focus of the entire organization, and the role of the NPD function in the network of the interrelated functions. However, this overall model needs to be refined for the purposes of this thesis in order to understand the intrinsic factors and interdependencies of software-intensive NPD.

[4] The terminology of competencies and capabilities is not clear-cut in the extant literature. The terms are sometimes used interchangeably. In this thesis, the term ócompetenceô is defined to mean the know-how of conducting different functional operations. Consequently, the term ócapabilityô means here the ability to utilize the competencies. This aligns with the recent literature of NPD dynamic capabilities (Mathiassen and Vainio 2007; Pavlou and El Sawy 2006).

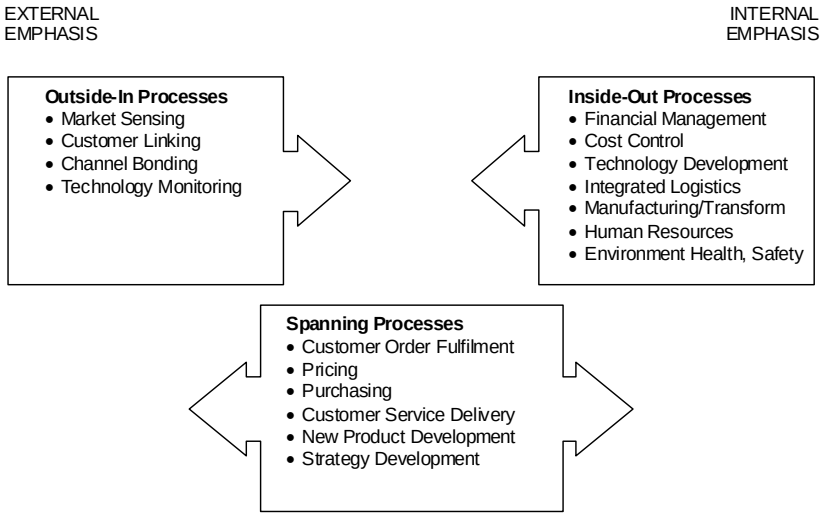

Fig. 2. NPD competence positioning (Day 1994)

In general, products and their development can be examined from different perspectives, such as marketing, organizations, engineering design, and operations management (Krisnan and Ulrich 2001). The NPD process area can further be categorized in different ways varying in scope and primary viewpoints. Software engineering can then be regarded as one technical discipline in the NPD context.

Notably the different bodies of knowledge are currently not clear-cut, but they are partially overlapping and intertwined. For example, the software engineering discipline utilizes general project management knowledge. In the general-purpose project management knowledge and practices software development is a particular application area technical element, and new product development is a management specialization (PMBOK 2004 / Ch. 1.4).

2.1.1.2 *Basic Models*

The basic purpose of industrial product development is to create either completely new products, or enhance and improve the existing ones (Smith and Reinertsen 1998; Trott 2005; Ulrich and Eppinger 2000; Wheelwright and Clark 1992). Nowadays product development offerings are increasingly often combinations of tangible components coupled with various intangibles providing ótotal solutionsô.

Product creation process models are intended to guide the development of such new products. They are in particular for coordination and steering purposes. Fig. 3 illustrates a traditional, sequential product development process model (Ulrich and Eppinger 2000). In this system development management model view the product software construction is a part of the detailed product design, together with the hardware engineering and other technology-dependent activities. Typically they follow their internal project management and technical engineering process models.

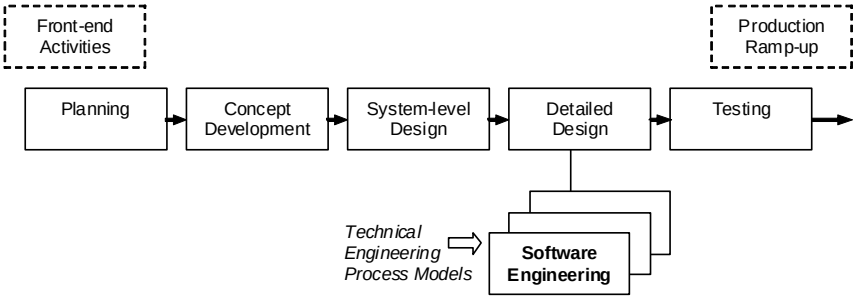

Fig. 3. Basic general-purpose new product development process model

There are different types of product development projects. Typically the following four generic types are distinguished (Wheelwright and Clark 1992):

- research and advanced development
- radical breakthroughs
- platform (next generation) development
- derivatives (enhancements and improvements; e.g., cost reduction versions)

Considering new products, it is important to realize that the ónewnessô is relative to the particular NPD company and the competitive environment. From that point of view, product developments can be (Trott 2005; Ulrich and Eppinger 2000):

- new-to-the-world (fundamentally new products)
- new-to-the-firm (new product lines, platforms)
- additions to the existing product lines (derivatives of existing product platforms)
- incremental improvements and revisions to the existing products (e.g., cost reductions)

Basically the product and project types follow from the company strategy. Typically in large NPD organizations there are multiple development projects of different types active concurrently, depending on the company product portfolio and future strategic intents (Benko and McFarlan 2003; Davis 2002). Some companies make strategic choices of avoiding new-to-the-world developments (disruptive innovations) requiring risky, possibly heavy investments for years until profits can be gained (Deloitte 2007). Depending on the product portfolio, different types of uncertainties and risks involved must be taken into account at different stages (Davis 2002). All this calls for proactive NPD strategies balancing the short-term business pressures and long-term growth paths (Kotler et al. 1996; Nakano 2007; Scinta 2008).

Overall, it is illuminating to understand how the basic role of organizational NPD (R&D) has evolved over the years in general starting from separate corporate research laboratories towards cross-functional integration of marketing, manufacturing, and eventually networking across different organizations (Augier and Teece 2007; Nambisan 2003; Nobelius 2004). However, in many cases it is still sensible to distinguish between pure research-oriented activities and efficient execution of commercial product development efforts.

2.1.1.3 *Success Factors*

In order to analyze comprehensively how successful software development can be realized in NPD contexts, it is first necessary to understand the overall success factors of NPD operations. This contextual perspective makes it then possible to see the possibilities of agile software development models in NPD organizations.

In general, determining successful NPD efforts is non-trivial. Basically, successful new product development comprises the following (Trott 2005):

- developing right products at the right time (product innovation management)
- developing the products right (product development process)

Product development performance can be measured from multiple viewpoints typically in terms of product quality, development cost and development time (Mäkelä 2008). Such performance indicators are for instance product technical performance, innovativeness, cost (design and production), service level, lead time, and market ñfitò (attractiveness) (Krisnan and Ulrich 2001).

In commercial industrial NPD contexts the ultimate success is usually defined in terms of financial performance (Brown and Eisenhardt 1995; McGrath 2004). Fig. 4 illustrates the type typical overall cost structure of NPD projects (Ulrich and Eppinger 2000; Chillarege 2002; Porter 1993). The future sales determine the overall financial success in the long run, which in some NPD cases can only be seen after several years (NPV). A typical traditional measure of NPD financial performance is the share of profits and sales accounted for products introduced within the last (5) years (Kotler et al. 1996). More advanced measurements have also been developed (Salem 2001).

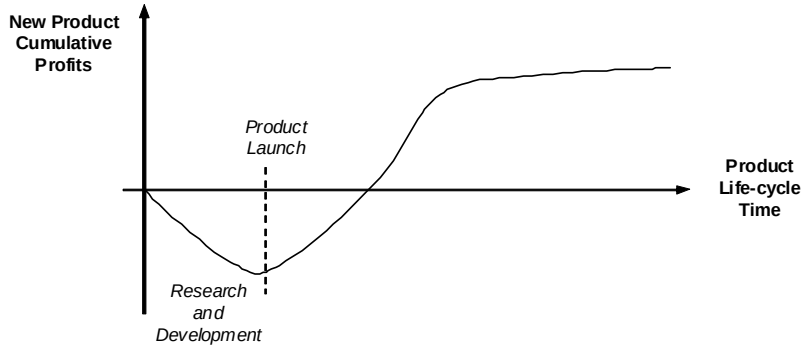

Fig. 4. Basic NPD return-of-investments flow

However, with more complex NPD product and project portfolios, the combined financial performance figures are not always so straightforward to analyze (Benko and McFarlan 2003; Davis 2002). For instance there may be market and technology probing projects and next-generation platform developments, where the overall success effects for the entire NPD company in the long rung is more difficult to amortize. Many NPD companies operating under fierce business conditions attempt thus to shorten the new product development cycles (ñtime-to-cashò).

It is also important to realize that success can be measured at different levels of a large NPD organization. Success at the product development project level does not necessarily result in long-term business success at the company level. All and all this is about value engineering (Lindstedt and Burenius 2003; Raivio et al. 2006). Ultimately this leads to the question of how ósuccessô is defined, and what makes a high-performance company (Atkinson 1999; Kirby 2005).

Over the years there have been numerous research investigations about the success and failures of NPD efforts. A seminal, yet still much applicable summary of the published NPD success factor research is the one by Brown and Eisenhardt (1995). Regarding the research problem of this thesis, the most insightful part of the survey is its proposition for an integrated model of different factors affecting the business performance of the NPD organization as a whole ï in particular the NPD process performance (lead-time, productivity), NPD team process, and the product fit with the market needs.

More recently for instance Ernst (2002) compiles an extensive survey of empirical research results of NPD success factors. They use five main categories ï notably including organizational and cultural factors:

- NPD process
- organizational aspects of NPD
- cultural aspects
- role and commitment of senior management
- NPD strategy

Table 25 (Appendix) presents a literature summary of typical NPD success/failure influencing factors in different industries (including the surveys discussed above). Notably the studies conducted by Cooper and Kleinschmidt over the years are instrumental here. While there are no straightforward answers to what is a órightô product or an ideal product development process model, certain key success factors have been advocated frequently time and again by many different studies. Those include in particular (early) customer involvement and market sensing, collaborative cross-functional teamwork, adaptive situational processes, trust-based project leadership (culture), and supportive senior management commitment.

As a conclusion, it is important to realize that, in general, successful new product developments require typically combined efforts of many different parties in large organizations (ñtotal-company effortò) (Kotler et al. 1996). The different factors stem from many sources of the NPD process, technology, and organization. In particular, the co-operative interplay between the marketing/sales and R&D functions is one of the common keys to success (Day 1994; Souder and Moenaert 1992). However, often people factors are underlined as the single most fundamental source of advantage (Cohan and Unger 2006; Liker 2004; Takeuchi and Nonaka 1986).

McGrath (2004) sees that the main focus of NPD management (ñmega-trendò) has evolved over the past decades from basic successful product development project management to time-to-market reduction, and more recently to overall R&D productivity emphasis. However, it is important to realize that this is context-dependent, and in large NPD organizations different product development projects may have different performance drivers over the product life-cycles (Chillarege 2002; Tyrväinen et al. 2004). Often there is a subtle balance between short-term

productivity gains and the long-term success of the product development (Mintzberg 2007). Organizational R&D capability investments create potentially future earnings (Nakano 2007).

It is beyond the scope of this thesis to analyze all NPD success influencing factors. Furthermore, currently there is no common agreement on what all factors are really influential, and the theoretical frame working is diverse. However, for the purposes of this research work, it is necessary to realize how the software product development function interrelates to the overall NPD success, in particular with respect to flexibility (Sect. 2.1.2) and agility (Sect. 2.2).

2.1.2 Flexible NPD Concepts

New product development is inherently uncertain in many modern competitive environments (2.1.2.1). In such cases more flexibility is needed for successful project conduct, and the basic NPD process models need usually to be replaced by more flexible ones (2.1.2.2).

2.1.2.1 *Uncertainty*

The very nature of NPD entails uncertainties and risks. The two main categories of uncertainty are usually the product and/or process technology and market risks. While technical uncertainties can often be managed to a large extent with conventional engineering management techniques, business risks are often much more subtle to be coped with successfully, requiring multidisciplinary techniques and cross-functional knowledge (e.g., weak signal interpretation). This is particularly relevant in modern turbulent business environments, where market uncertainties are often more dominating failure factors than technical risks (Smith and Reinertsen 1998).

In general, a turbulent competitive environment is characterized by high complexity and high change rate (Chakravarthy 1997). For instance MacCormack et al. (2001) define uncertain and dynamic product development environments as such in which market and technology evolutions are unpredictable and happen rapidly. In general, such uncertainties stem from the following main areas:

- What product to develop (concept, features) in order to maximize the current market/customer value and development investments?
- How to design and implement it accordingly (technology, processes, people)?
- environmental circumstances (organization internal and external competitive environment)

It is furthermore possible to distinguish between the degree of uncertainty ranging from small variations to foreseen and unforeseen uncertainty, and even to chaos (De Meyer et al. 2002). Different types of uncertainties call for different types of product development project management tactics ï possibly even questioning the very purpose of the project (Atkinson et al. 2006). Traditional project risk management techniques must then be extended towards more intrinsic project uncertainty management (Charette 1996; Smith 2007). Discovery-driven planning acknowledges the premise that in uncertain business environments the planning assumptions are often not stable, and they must be systematically stated and revisited throughout the execution while new information and feedback is received (McGrath and MacMillan

1995). This is one of the key premises in flexible and agile software development models.

Notably the degree of uncertainty varies typically between the different areas of the product creation (for example, high market uncertainty but low technical uncertainty). It is therefore important to understand the uncertainty profile of the specific NPD project context, and how it evolves over the life-cycle (Chillarege 2002; Davis 2002; Little 2005; Royce 2005). Moreover, different elements of large, complex products face often different types and levels of uncertainties over time.

Most of those sources of uncertainties, trends (e.g., globalization) and changes in industrial NPD environments have actually been anticipated already for some decade ago (Wind and Mahajan 1997). In particular it has been recognized that there is a growing need for new NPD approaches to cope with discontinuous changes and uncertainties in many competitive environments (Eisenhardt and Brown 1998; Iansiti 1995; Smith 2007).

2.1.2.2 *Flexibility*

The traditional basic NPD process model described in Sect. 2.1.1.2 is relatively well understood in stable, predictable environments. However, under turbulent conditions with considerable uncertainties and frequent, even disruptive changes, the basic sequential planning and execution premises are often not so successful, and more flexible ways of conducting the NPD work are needed.

Currently there is no one standardized definition of ôflexibilityô in NPD. It can be addressed from different perspectives and at different levels. Consequently there is no one agreed measure of NPD flexibility. For instance Upton (1994) characterizes flexibility (in manufacturing) in abstract terms as the ôability to change or react with little penalty in time, effect, cost or performanceô. Likewise, Smith (2008) underlines the ability to make (even late) changes without excessive ôdisruptionsô. One possible general formulation of flexibility in NPD is to define it in terms of the cost and time of making (late) changes and/or corrective actions (Verganti 1999). One proposed metric is the (incremental) economic cost of modifying the product (Krisnan and Ulrich 2001).

In general, flexibility in NPD can be realized in multiple dimensions and at different levels including flexible products (both design and use), flexible NPD project and process models, organizational flexibility, and even the company strategic flexibility. For example product platforms and mass customization are typical ways of enabling flexibility with a variety of product combinations (Anderson 1997; Männistö 2000). This thesis focuses on the product creation processes in general.

It is important to realize what creates the flexibility enabling internal process capabilities, and how those capabilities are utilized externally for competitive advantages (Upton 1994). The basic constraint of flexibility in the traditional sequential NPD process (Fig. 3) stems from the early product concept definition, which is not supposed to be changed during the downstream development (Iansiti 1995). The ñfuzzyò front-end has often been pointed out as one of the main root-causes of NPD project problems and failures due to uncompetitive/inappropriate product concept or feature selection and/or slow start of the actual design work (Nobelius and Trygg 2002; Rautiainen et al. 1999; Smith and Reinertsen 1998).

The principal solution to such inflexibility of the traditional sequential product concept freezing followed by the fixed implementation and market introduction is to allow the product definition (concept) to evolve concurrently with the actual product implementation like illustrated in Fig. 5 (Iansiti 1995). Further and more current information can thus be incorporated. It is then possible to define the responsiveness of the NPD process as the time delay (Development Lead Time) between the market introduction and the last moment of accepting changes (Concept Freeze).

The key difficulty with such overlapping is in general the increased complexity of managing the work, since the incoming new information of the initial product definition may affect the work already done and the subsequent planning. The product development functions should then be prepared for accommodating changes in product concepts and specifications during the concurrent development.

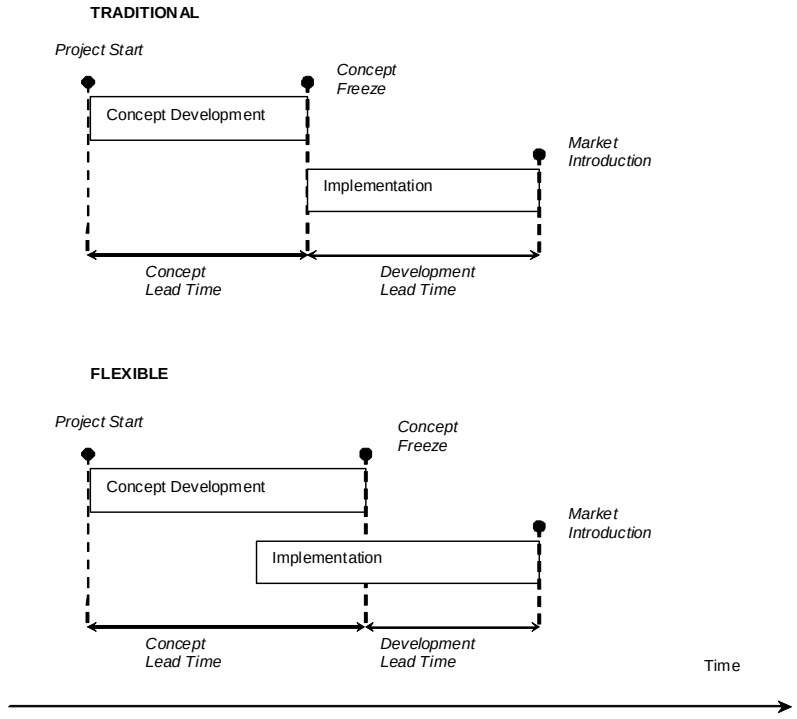

Fig. 5. Flexible product development process principles (Iansiti 1995)

The choice between a basically linear and more flexible concurrent NPD process models should typically be strategy-based following the environment-specific drivers for flexibility (Treacy and Wiersema 1995). Usually it is not reasonable to attempt to strive for maximum flexibility in all dimensions (Thomke and Reinertsen 1998). Identifying early the critical project uncertainties and preparing how to handle those

criticalities with reaction capabilities guide necessary planned flexibilities (Verganti 1999). The competitive environment moderates the needs for flexibility in different areas. It is thus important to understand, where and when the NPD organization needs in particular

- predictable and straightforward project execution (operational efficiency) and/or
- abilities to accommodate new midcourse information and make changes.

NPD process acceleration has got considerable attention over the past years (Eisenhardt and Brown 1998; MacCormack et al. 2001; McGrath 2004; Rauscher and Smith 1995; Smith and Reinertsen 1998). Product development speed is coupled with flexibility in many interrelated ways (Smith and Reinertsen 1998). Fast product development cycle times make it possible to adapt more quickly to customer/market changes and technological developments. Moreover, if the product development lead time is short enough, there may be less needs for making changes altogether during the development period.

There are two general principles of accelerating product development: compression models (c.f., Fig. 5) and experiential models (Nambisan and Wilemon 2000). Often in stable, predictable environments compression models are suitable while in uncertain environments flexible experiential models are more effective (Eisenhardt and Tabrizi 1995). This is also one of the key premises in agile software development. However, in addition several organizational and environmental factors affect the effectiveness of the acceleration methods and their suitability.

Another dimension of flexibility is at the NPD management level. The NPD flexibility can be increased by maintaining a balanced project portfolio supporting both the current business strategy setting as well as anticipated future strategic scenarios (Raynor and Leroux 2004). The latter perspective is in particular important in turbulent competitive environments (Brown and Eisenhardt 1998; Chakravarthy 1997; Johnson et al. 2006).

NPD flexibility can also be viewed from the overall organizational perspective. The flexibility of the NPD is interrelated with other strategic flexibilities in large organizations. It is therefore important to understand the effects of the NPD flexibility across the entire value chain (network) of the company ï even extending beyond the company boundaries (Goldman et al. 1995; Raivio et al. 2006; Worley and Lawler 2006). For instance supply chain flexibility is often closely connected to it (Kaipia 2007; Vehtari 2006).

The research stream of firm strategic dynamic capabilities has got growing interests during the past few years (Mäkelä 2008; Wang and Ahmed 2007). The works by Sanchez are foundational here (Sanchez 1995; Sanchez and Mahoney 1996). According to their view, high resource flexibility allows to utilize the product development resources for alternative purposes with rapid and low-cost reorientation, while high coordination flexibility makes it possible to take an advantage of such flexible resources. However, they do not prescribe how to realize such organizational advantages in practice.

Haeckel (1999) advocates likewise a model of an adaptive enterprise based on dynamic sense-and-respond capabilities. Again, there is no prescription for implementing such a conceptual model in reality, though. Worley and Lawler (2006) emphasize the people factors in such organizational capabilities.

Notably in large NPD organizations the flexible organization view can be considered as a hierarchical pattern with internal customers (Haeckel 1999). There may then be some more stable internal subfunctions where less adaptation is needed although the organization as a system is flexible (Sanchez and Mahoney 1996).

Table 26 (Appendix) summarizes various literature suggestions and findings about flexible NPD (including the ones discussed above). Interestingly, one of the early, highly influential and often-cited investigations of rapid and flexible new product development was published already some 20 years ago by Takeuchi and Nonaka (1986). This is because the investigation addressed technology-independent elements of NPD organization and management. Consequently, they are still valid to a large extent. However, the recent development of many IT-based solutions (e.g., virtual prototyping) has made it possible to achieve fundamentally more flexibility in many phases of the NPD process, though (Nambisan 2003).

It is beyond the scope of this thesis to elaborate all those factors more extensively. The concluding point is to realize that, in general, flexibility and speed in NPD depend on many interrelated factors. Usually no single element or ñbest practiceò is a key solution, but mutually enforcing different organizational, process, technology, and people factors are typically combined and balanced.

2.1.3 Software-Intensive NPD

Software development is an increasingly important part of the creation of many modern new products (2.1.3.1). In such product environments embedded software engineering is typically a core competence for successful product creation (2.1.3.2).

2.1.3.1 *Product Creation and Software Development*

Software development relates to new product development at different levels depending on the role of the software in the products. In general, the outputs of industrial software production can be categorized as pure software products (stand-alone), customized software, and embedded software (Hietala et al. 2004). However, it is useful to realize how much software development is done in different industrial organizations in addition to specialized software product companies (Tyrväinen et al. 2004). In particular, embedded software production (in devices but also increasingly in services[5]) is a major activity for instance in the telecommunications industry. Consequently, software engineering can be one of the component disciplines of the larger product creation process (see Fig. 3), or the entire NPD process is about software production.

Basically all that is said about NPD processes in Sect. 2.1.1 and Sect. 2.1.2 applies to software product development as well, but the intrinsic factors of software put weight on specific process key areas. However, such technology-independent factors as early customer involvement and systematic product architecture design are keys to success (MacCormack 2001).

In practice many industrial software product development set-ups may be even much more complicated. A complex software-intensive new product is typically

[5] In this thesis, the term óembedded softwareô is by default used with the former meaning (in telecommunications equipment).

created as a series of embedded software and hardware releases. For example, large telecommunications system products may be under development for several years. Such a continuous development streaming stretches the traditional concept of software projects (Koskela and Howell 2002; Levine 2005). For instance the following additional technical and organizational factors are typically involved in large-scale software-intensive NPD organizations:

- different (even asynchronous) development time-schedules for the product software and hardware parts (e.g., hardware development started first without related software development)
- externally furnished components (software and hardware)
- product software and/or hardware configurations compatibility constraints
- legacy systems and components
- software and hardware platform developments
- multiple parallel product lines (product portfolio)
- existing customer bases and product installations

Software product development productivity is a general strategic concern in many current industrial sectors in general (Baskerville et al. 2006). At the same time the complexity of software-intensive product development is often increasing in many dimensions (ARTEMIS 2006; ITEA 2004; ITID 2008; Nidiffer and Dolan 2005; Rauscher and Smith 1995). Those factors introduce new challenges for success. The key issue is to realize the role of the software development in the total product development value chain over time (Raivio et al. 2006). In particular, there are often extensive needs to transfer knowledge both spatially and temporally (Büchel 2007).

Over the years there have numerous research investigations about general software product development project success/failure factors and risks. However, the NPD context is much more sparsely studied.

One of the seminal investigations of large-scale industrial software product development problems (including embedded software) was conducted by Curtis et al. (1988) already in late 1980s. A key conclusion was that large software development problems are much about learning, communication, and negotiation between different stakeholders with different knowledge domains. Interestingly, they stem from organizational dynamics and management structures, which have not fundamentally changed since the time of that field study.

Table 2 summarizes typical software-intensive NPD success factors discussed above. In addition, what is known about software project successes and failures in general is also relevant here ï see for example Smith (2001), Sommerville (2001). It is beyond the scope of this thesis to cover all such factors in more detail, however. Sufficient to conclude that the NPD context brings the business dimension into specific consideration.

Table 2. Representative software NPD success/failure factor findings (chronological order)

Publication	Influencing Factors	Success Criteria
(Curtis et al. 1988)	NEGATIVE: • thin spread of application domain knowledge • fluctuating and conflicting requirements • communication bottlenecks and breakdowns POSITIVE: • exceptional designers (system-level thinkers with good communication and coordination skills) • managing learning (in particular when major changes in the application area, technology) • not formalizing (requirements) specifications until the major uncertainties have been reduced; negotiation and coordination processes for resolving requirements conflicts • organizational communication channels between the customers and developers, and between successive project teams; informal communication networks	software productivity and quality
(MacCormack 2001)	POSITIVE: • early release of the evolving product to customers • frequent (daily) incorporation of new software code (new information) and rapid feedback on design changes • project teams with broad-based experience of developing multiple projects • investments in the product architecture design	product quality (reliability, technical performance, breadth of functionality) compared to the competitors, project resource productivity
(Baskerville et al. 2006)	NEGATIVE: • time-to-market compression demands • ambiguous and fluid requirements • changing environment • insufficient programmer productivity • lack of design time and experience POSITIVE: • concurrent development with frequent releases • customer involvement and prototyping • structured architecture • efficient tools and reusable components • tailored methodology with right team expertise	quality, cost, and development speed balanced

Section 2.1.2 explores flexible NPD in general. The same principles can to a considerable extent be applied also for the software parts of products, with the additional inherent flexibility of software. Notably for that reason many modern product designs allocate more and more functionality into the embedded software components (Tuormaa 2009). This increases the overall flexibility of the product system design, but puts even more emphasis on the flexibility requirements in the software development (Turner 2007). Two key points are then the last change to the architecture (conceptual changes), and the last change at the module level (feature changes) (MacCormack and Verganti 2003).

Most of the currently advocated approaches and practices for flexible software product development have actually been known for years (Larman and Basili 2003).

However, the current trends in many competitive environments and organizational business models (e.g., networked product development) on one hand, and the modern software development and IT tool advancements on the other have recently made them more and more attractive (McGrath 2004). Every NPD organization should nowadays consider the influence of Internet to their business and products on the one hand (e.g., Open Innovation), and the fundamental nature of software development enabled by even global networking (e.g., Global Software Development) on the other (Baskerville et al. 2006; Yourdon 2002).

Table 3 highlights the key literature viewpoints and findings of flexibility in software new product development. It is typically achieved essentially with the same basic principles as in NPD processes in general (c.f., Table 26, Appendix).

Table 3. Representative approaches to flexible software NPD (chronological order)

Publication	**Approaches**	**Potential Benefits** **Costs and Problems**
(Gilb 1988; 2006)	PROCESS: • evolutionary systems development and delivery in short (even 1 week) and small (some 2% of the total budget) value-based increments • continuous learning and adjustment of the goals according to the current needs and feedback on delivered increments	Focuses on current most valuable delivery goals; rapid adjustments with constant feedback; no excessive budget overruns (visibility) Assumes quantifiable systems goals, expecting good systems engineering capabilities and skills.
(Yoffie and Cusumano 1999)	ORGANIZATION: • swift strategic decisions • small product teams, authorized to make decisions for their products • leveraging external resources to compensate in-house capabilities (e.g., beta-testers, open source)	Closely following and even influencing the customer expectations and market trends; Maintaining ñsmall-companyò flexibility and creativity in large scale; Balancing internal resource bottlenecks with external resources. lack of coordination between different groups in large organizations
(MacCormack 2001; MacCormack, Verganti and Iansiti 2001)	PROCESS: • investments in architectural design • earlier feedback on product performance from the market ORGANIZATION: • development teams with greater amounts of ñgenerationalò (system-level) experience	Closer match with the evolving requirements; Dedicated architecture work can maximize the product performance and support flexibility. Overlapping: Need to start detailed design before the product architecture is completed; Need to start integrating the system with partially ready components;
(MacCormack and Verganti 2003)	PROCESS: • analyzing the sources and levels of project context-specific uncertainty (new design work and markets) • matching the development process model (practices supporting flexibility) accordingly to address the uncertainties (contingent view)	Investments in architectural design, and early technical and market feedback are associated with better performance of projects facing high uncertainties. The cost of increasing the flexibility (e.g., developing a highly modular product architecture) should be weighed against the potential gains of such options. Building the flexible capabilities may require significant long-term efforts (proaction).

Publication	Approaches	Potential Benefits / Costs and Problems
(Mikkonen and Pruuden 2001)	PROCESS: • defining explicit flexibility requirements for certain parts of the software system to accommodate future information and/or late changes	Allows the software development to proceed with incomplete and unstable information.
		Requires careful architectural design decisions identifying the critical flexible parts. May lead to trade-offs and compromises with other product goals (e.g., performance).
(Moløkken-Østvold and Jørgensen 2005)	PROCESS: • flexible development models (incremental, evolutionary, agile)	Promote continuous dialogue between the customers and the software developers.
		Requires competent clients.
(Subramaniam and Hunt 2006)	TECHNOLOGY, PROCESS: • avoiding making premature irreversible commitments to new technologies (in case of uncertainties)	Keeps change options open.
		May delay development decisions unnecessarily. Requires understanding of the technology development maturity.

Overall, interdisciplinary organizational studies of NPD software development are still dispersed. There is no clear research stream, and the publications are diverse (Glass et al. 2004). However, recently the concept of firm dynamic capabilities has gained considerable attention also in the software development context (Aramand 2006; Kivelä 2007; Mathiassen and Vainio 2007; Mäkelä 2008).

2.1.3.2 *Embedded Software*

A fundamental characteristic of embedded system product development is its multidisciplinary nature. Software engineering is then one of the component disciplines, while systems engineering brings the different elements together as a complete product design (Leppälä et al. 2005). In principle, it is about integrating software computing models, the target hardware execution limits, and the system environment constraints (e.g., response time) into a coherent realization (Henzinger and Sifakis 2006).

Such industrial embedded software new product development entails certain inherent difficulties compared to other software production categories (Sect. 2.1.3.1). They stem typically from the complicated dependencies with the environment and the target hardware:

- The customers (users) do not usually perceive the software part as a such, but they action with the combined hardware/software product. Consequently, the customer requirements do not necessarily address the software directly.
- With business products (B2B) and deeply embedded systems there may be different levels of users/customers (B2B2C). The software solutions may not be equally visible at every level ï if at all (e.g., telecommunications network systems).
- There may be critical non-functional system requirements pertaining for example real-time performance and reliability.
- The technical requirements for the software include often a complex set of interfaces to external systems and to (proprietary) hardware devices. The external

system requirements may be defined by international technical standards (which could be subject to change or still under development).

- The software implementation is often constrained by the target hardware resource limitations (e.g., processing power).
- The target hardware platform may consist of various computing, peripheral, and interface units often realized as a distributed system.
- The software testing in the target hardware environment may be complicated, requiring special-purpose laboratory set-ups, auxiliary measurement devices, etc.
- The software part may have to support in addition to the actual customer functionality various internal hardware testing functions for instance for field testing and hardware manufacturing purposes.
- If the target hardware is under concurrent development with the embedded software parts, the software development may have to be started with incomplete hardware specifications, and the early testing phases may have to be done with prototype hardware.
- It is not unusual that some target hardware design defects are discovered late, requiring additional software workarounds.

A recent European investigation indicates that there appears to be considerable gaps between the theoretical research advancements of embedded software engineering and the industrial practice in many sectors (Graaf et al. 2003). The key to industrial embedded software NPD success is to be able to develop such technically complex and large products with high business productivity and quality (Sifakis 2007; Solingen 2002). One of the key challenges is then to develop flexible software process models integrated with the systems and hardware engineering design flows (ITEA 2004; ITID 2008; Rauscher and Smith 1995).

A long-term vision is to have systematic end-to-end product development processes and tools for complex software-intensive and embedded systems taking into account industrial business drivers and constraints (ARTEMIS 2006). Sufficient to conclude here that embedded software development expands the software engineering technical dimension while the NPD context (Sect. 2.1.3.1) emphasizes the business dimension.

2.2 Agility in Product Development

In order to fully understand the role and effects of agile software development (Sect. 2.3) in large-scale NPD context, it is first necessary to comprehend the general principles and foundations of agility (Sect. 2.2.1). This grounding then makes it possible to reason agile capabilities in NPD (Sect. 2.2.2). Furthermore, the key enablers and tactics for developing such necessary agile capabilities can be devised (Sect. 2.2.3).

2.2.1 Concepts and Definitions

Currently there is no unified definition of óagilityô in product development. Different authors use it in varying ï sometimes even confusingly ï in different scopes and

depths. In general, it is an attribute which can be linked to organizational entities ranging from individuals to entire enterprises. For example Goldman et al. (1995) view it from an enterprise-level perspective comprising marketing, production, (product) design, organization, management, and people.

Table 27 (Appendix) presents a representative set of different general definitions of agility used in the literature of various disciplines over the years. The essence of agility is that in turbulent competitive environments there are many unpredictable changes taking place often and dynamically, and thus traditional forecast-based planning and strategy assumptions do no longer hold (Doz and Kosonen 2008). Agility is seen as a viable way of competing successfully in such new environments. The ultimate business goal can be attributed to profitability and adaptability requirements (Dove 2004).

It is not in the interests of this thesis to propose a new definition of agility. Like stated in Sect. 1.1, we adopt as a working definition the one by Conboy and Fitzgerald (2004) because of its wide-ranging interdisciplinary yet software-oriented nature; see Table 27 (Appendix). It also aligns with the scoping of this research (Sect. 1.3).

That said, this thesis maintains the view that agility is a system capability of an organizational entity relative to its competitive environment. Fig. 6 illustrates this view. It is an inferred synthesis based on the different formulations and definitions of agility summarized in Table 27 (Appendix). Here an entity may be the entire (virtual) company as well as the internal NPD function or an individual software project team.

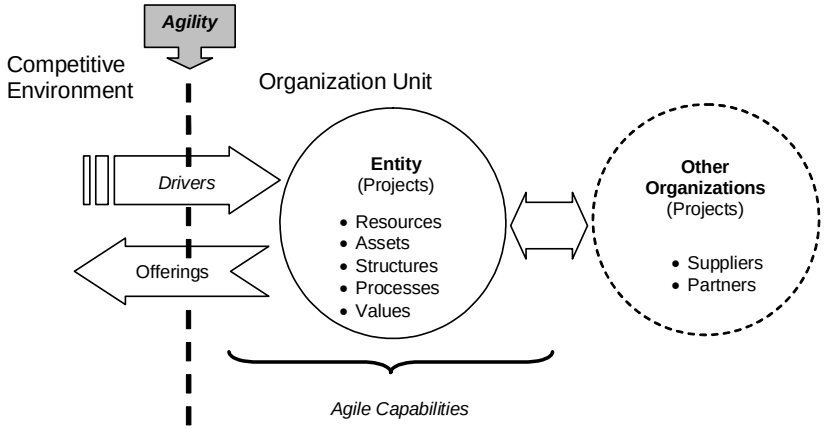

Fig. 6. Agility viewed as an organizational capability

Notably this is an open dynamic system. In general, agility can thus be decomposed into the following dimensions with respect to the interface between the unit and its environment (Ismail et al. 2006):

- responsiveness (reacting appropriately to the changes)
- proaction (preparing for future changes, possibly also influencing them)
- robustness (ability to cope with unpredictable external changes)

Agility is fundamentally about dealing with uncertainties and learning over time. A Learning Organization is proficient at adapting to its changing environment. It is therefore important for the organization (unit) to understand the potential sources of uncertainties and changes (drivers) in its competitive environment. The needs for external agility and the appropriate internal management style tactics thus vary (De Meyer et al. 2002).

Ideally, the offerings of an agile NPD organization are constantly competitive and timely responses to market/customer changes, utilizing ï whenever feasible ï the latest technological opportunities. This is typically achieved with adaptive, customizable products realized by modular architectures and platforms. In addition, they are created profitably with dynamically configured production set-ups, leading to sustainable competitive advantages for the company even under disruptive circumstances (Suikki 2007). However, not all changes require always immediate responses, and sometimes no response may be a sensible choice (Lovén 2006). An agile company may furthermore create and adapt its offerings proactively, thus initiating itself desirable changes to the environment (Goldman et al. 1995; Highsmith 2002). Both product and process innovations are thus important (Lyytinen and Rose 2006).

In this respect, product development flexibility ï as discussed in Sect. 2.1.2 ï can be interpreted as one enabler for agility. Oosterhout et al. (2006) distinguish between flexibility and agility based on how predictable the changes are and how well the responses can be predetermined. That is, flexibility alone may not be enough for comprehensive agility.

Agility is often related to leanness because of their mutual goal-settings. The essence of Lean Thinking is to continuously optimize the production value flow by eliminating ñwastesò and maintaining high quality (Liker 2004). The lean ideas have subsequently been adopted further to product development and even to software production (Mascitelli 2006; Poppendieck and Poppendieck 2004).

However, again, the conceptualization and terminology are not uniform and harmonized here, and some authors consider ólean ô as a prerequisite for agility, whilst other interpret it in a broader sense with a wider scope (Katayama and Bennett 1999; Narasimhan et al. 2006; Ward 2007). There are even some propositions to define ólean agilityô as the composite concept (Naylor et al. 1999; Hoque et al. 2008). óLean ô (with a capital initial) is also used in a wider sense as a general organizational improvement technique (Phillips 2008).

On the other hand for instance Haeckel (1999) sees that agility is not enough in highly unpredictable environments, but organizational óadaptationô is needed. Upton (1994) views flexibility as the main concept with robustness and agility as its features. Tsourveloudis et al. (1999) define (manufacturing) flexibility as the production function-level attribute, while agility is a company-level capability of the whole enterprise. Such corporate agility is the net-effect of all business processes, and usually in practice both adaptive flexibility and routine efficiency is needed in an appropriate balance in different process areas (Miers 2007). In that perspective efficient product development could enhance company-level agility.

In principle, this is a matter of how óagilityô (and ólean ô) is defined and scoped. This thesis maintains the view that agility is the highest-level umbrella concept, and flexibility, adaptation and leanness are constituent elements of comprehensive,

external agility. Depending on the level and nature of the uncertainties, more adaptive (flexible) or streamlined (lean) internal capabilities are apt in order for the organization (entity) to stay competitive. This is furthermore subject to change over time, depending on the dynamics (turbulence) of the competitive environment. Agile software development can be mapped to this spectrum (Sect. 2.3.1).

2.2.2 Agile Capabilities

Agility is not unique to either NPD or software development. The origins of most of the current agility concepts trace back to early 1990s manufacturing field (Preiss 2005). At that time the traditional mass production business models were realized to become uncompetitive in many sectors particularly in the USA. There was a need to rethink the whole production system, and agility was devised to be the next competitive edge in the new competitive environments. The key driver with flexible manufacturing systems (FMS) and other related production means is that in many industries facing unpredictable changes in product demands and customer needs the traditional mass production mechanisms are no longer competitive enough, and more adaptive production is needed.

Table 4 presents a concise literature overview of the general meaning and objectives of agility in different business competence areas (Sect. 2.1.1.1). Although the general goal is the same (sustainable profitable business in changing competitive environments), the different areas address it from different points of view and in different scopes. They are also partially overlapping.

Table 4. Agility in different business competence areas

Area	Meaning / Objectives
Strategic Agility	Strategic sensitivity (awareness and attention), leadership unity (collective commitment), and resource fluidity (reconfiguration) working as an integrated real-time system; Agility = Sensitivity × Unity × Fluidity (Doz and Kosonen 2008);
Business Agility	Constant reconfiguration of strategies and processes and examination of their market positioning while external conditions continually change (Gould 1997); Quickly implement new business models and value delivery systems (HP 2003); Being able to swiftly change businesses and business processes beyond the normal level of flexibility to effectively manage unpredictable external and internal changes (Oosterhout et al. 2006); Capability to adjust the coordination of resources and mechanism with smooth dexterity in response to change and to maintain performance (Caswell and Nigam 2005)
Enterprise Agility	Capabilities to thrive and prosper in a changing, nonlinear, uncertain and unpredictable business environment (Kidd 1997); Accurate timely awareness that changes should be made, effective prioritization among competing changes and response-alternatives, abilities to change business processes and to customize operational responses in real time (Dove 2004); Adaptive enterprise (Haeckel 1999; HP 2003); Ability of firms to sense environmental change and respond readily (Overby, Bharadwaj and Sambamurthy 2006)
Agile Organization	Adaptive (re)configuration of resources, structures, and routines to address unpredictable changes and opportunities in the environment; Nonlinear interaction with self-organization (decentralized control) and coevolution (Atkinson and Moffat 2005);

Area	Meaning / Objectives
Agile Workforce	Responsiveness to changing customer needs and market conditions (intelligence), speed of developing and acquiring new skills and competencies, effectiveness of cross-functional cooperation and moving between projects (collaboration), culture (empowerment), IS (IT support) (Breu et al. 2001); All employees meet (can interact with) customers (Goldman et al. 1995); Agile leaders;
IT Agility	Efficient and flexible IT services for dealing with changes and supporting organizational reconfigurations; New systems can quickly be implemented, critical systems changed, or the IT infrastructure restructured to provide new strategic and tactical capabilities or to respond to changing market and competitive conditions (Skaistis 2006); Enabler for Business Agility (Crawford et al. 2003);
Agile Manufacturing	Rapid and low-cost production of customized and high quality products in varying lot sizes by combining the efficiency of lean production with operational flexibility whilst delivering customized solutions at the cost of mass-production (Adeleye and Yusuf 2006); Production model that enables firms to react deliberately, effectively and in a coordinated manner to changes in the environment (Vázquez-Bustelo and Avella 2006); Efficiently changes operating states in response to uncertain and changing demands (Narasimhan et al. 2006);
Agile Supply Chains	Responding rapidly to changes in demand, both in terms of product volume and variety (Christopher 2000); Coping with irregular (unpredictable) demand patterns in volatile markets; Ability to sense and respond quickly, predictably, with high quality, easily adapting to changes in demand (Hofman and Cecere 2005);

Notably the concept of agility is currently not exactly or uniformly defined in all those fields. For instance the literature of Agile Manufacturing proposes many different definitions varying in scope from the actual production functions up to enterprise-level virtual manufacturing networks (Yusuf et al. 1999). Overby et al. (2006) conceptualize enterprise agility in terms of the environmental sensing and responding capabilities of the entire organization.

Furthermore, agility-oriented principles have generally speaking been addressed in different disciplines and business competence areas without using the term ‘agility’ explicitly. For instance Kotler (1994) characterizes Marketing as ‘the process by which an organization relates creatively, productively, and profitably to the marketplace’. This is essentially in line with the definitions in Table 27 (Appendix) and Table 4, and concerns the NPD functions, too.

Following that line of holistic systems thinking, ultimately the entire value-creation network of the company – including the NPD function with software development – can be viewed with respect to agility. Considering the conceptual system view in Fig. 6, the NPD functions contribute to both the sensing and responding capabilities of the organization. For example Krisnan and Ulrich (2001) have recognized the product development supply chains thinking. Such systemic viewing should cover not only the organizational structures and processes but also – and often even more importantly – the people (workforce) involved (Atwater and Pittman 2008).

2.2.3 Agility Improvement

In search for applicable improvement approaches, the range of currently proposed agility measurements in different disciplines are reviewed (2.2.3.1), along with some

more complex analysis tools (2.2.3.2). The current state of practice in introducing and improving agile product development capabilities is then overviewed (2.2.3.3).

2.2.3.1 *Metrics*

Because of the diversity of the ways agility is defined and scoped, there is no unified direct measurement of agility. However, it is typically attributed with the response times and the economic cost of making changes. Referring to Fig. 6, the former is an external metric observable at the customer interface while the latter is an internal metric related to the agile capabilities. The ultimate (indirect) measure is then the competitive business result achieved in the long run.

Zsifkovits and Engelhardt-Nowitzki (2007) survey and compare different conceptual views of agility and measurement frameworks. For instance with supply chains measurements a general problem is that the traditional metrics defined in the past for stable environments are not necessarily representative in new turbulent environments. In software development there is currently a similar problem of determining the expressive power of conventional software engineering metrics with agile software development models (Sect. 2.3). Some more business-oriented, macro-level metrics have thus been proposed (Hartmann and Dymond 2006).

There are various propositions for agility metrics like presented in Table 5. They are currently under debating, and no single measurement is commonly used. Typically the proposed indices are aggregates. Again, much depends on the way agility is defined in the first place. Notably there is currently no specific agreed measure on agility in NPD.

Table 5. Agility measurement approaches (alphabetical order)

Publication	Field, Scope	Approach / Principles
(Caswell and Nigam 2005)	IT	Operational system change model calculus: • (minimum) cost in time, money, and other resources of making changes
(Conboy and Fitzgerald 2004)	IS	Implementing changes: • # of changes implemented vs. costs (ú)
(Dove et al. 1996)	Enterprise	Change-proficiency: • cost, time, robustness, scope
(Hofman and Cecere 2005)	Supply chains	• speed and predictability • ease (of responding) • quality (supplier, manufacturing, product)
(HP 2003; HP 2005)	IT; Financial services, Network service providers, Manufacturing	Implementing changes: • time, range, ease (Agility Index)
(Ismail et al. 2006)	Manufacturing	Agility Strategic Framework: • agility capability indicators • environmental turbulence indicators
(James 2005)	Manufacturing	• cost of change • time to change • stability of change • scope of change • frequency of change
(Lin et al. 2006)	Manufacturing	• Agility Index

Publication	Field, Scope	Approach / Principles
(Oosterhout et al. 2006)	Enterprise, IT	Agility Gap Ratio: • probability of business change • difficulty to achieve business change (beyond the normal level of flexibility)
(Overby, Bharadwaj and Sambamurthy 2006)	Enterprise	Enterprise Agility Score (indirect aggregate function): • Sensing score (ability to sense environmental changes) • Responding score (ability to respond to the changes) • alignment level between different sensing and responding areas
(Tsourveloudis et al. 1999)	Manufacturing	Fuzzy-set based aggregate: • production (time and cost of unanticipated changes) • market (external customer service and marketing feedback) • people (training, motivation; Agile Workforce) • information (capturing, managing, sharing)

For example Turkulainen (2008) has recently investigated similar measurement questions in manufacturing plants – in particular the relationship between organizational integration and the performance. Her submission is that performance is a multidimensional measure (e.g., on-time new product launch, product innovativeness), and in the manufacturing context a reasonable scale is comparative (to the competition in the industry) rather than a one-dimensional absolute value.

2.2.3.2 *Analysis Models*

It is not obvious how exactly the different dimensions and levels of agility illustrated in Sect. 2.2.2 are related to each other. This is partially an open research question. In particular, it is not clear how much and under what specific circumstances they each contribute to the business success of a large NPD company (Mäkelä 2008).

In theory, different organizations (units) can provide similar responses with very different internal costs. The essence of total-company agility is the ability to sustain a reasonable balance between the responsiveness and the associated costs under changing (even turbulent) circumstances. For example Conboy and Fitzgerald (2004) propose a general cost-based model of agile capability like illustrated in Fig. 7. A proactive organization which is well-prepared is able to implement more changes and/or at a lower cost during the same time than a less-prepared, reactive one. The organization can utilize new learning to become continuously more capable (prepared for future changes). If the organization initiates the changes by itself (creation), the competitors may have to respond.

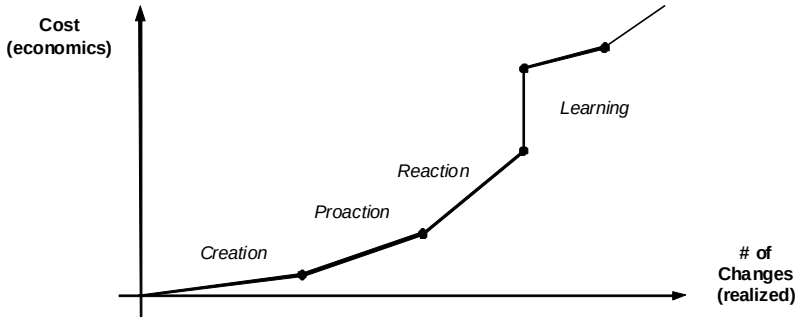

Fig. 7. Cost modeling agility dimensions (Conboy and Fitzgerald 2004)

In traditional NPD models (Sect. 2.1.1.2) the premise is to first create the product concept, design it, and then realize the chosen product design basically avoiding (major) changes during the realization phase (c.f., Fig. 3). The reasoning here is that in many product development areas the cost of changing the product design after certain commitment points (e.g., hardware technology selection) increases drastically, making it infeasible to realize major (late) changes.

Flexible NPD models (Sect. 2.1.2) on the contrary have a different basic assumption to accept uncertainties, avoiding early (premature) design commitments, and consequently making even late design changes throughout the product creation (c.f., Fig. 5). Depending on the product technologies, there are various ways to accomplish this, but in particular software-intensive systems have the advantage of potentially low cost of making the product changes with software thereby keeping the cost-level nearly uniform until the product release time (Thomke and Reinertsen 1998). Ideally, the cost of making even major changes remains flat throughout the product development time period (robustness). However, the traditional sequential NPD models (Fig. 3) strive to lock the product design early, consequently making changes later more costly. The flexible NPD models attempt to avoid this (Fig. 5). Fig. 8 illustrates this overall characterization (Boehm and Turner 2004; Glazer et al. 2008; Highsmith and Cockburn 2001; Smith 2008; Wils et al. 2006).

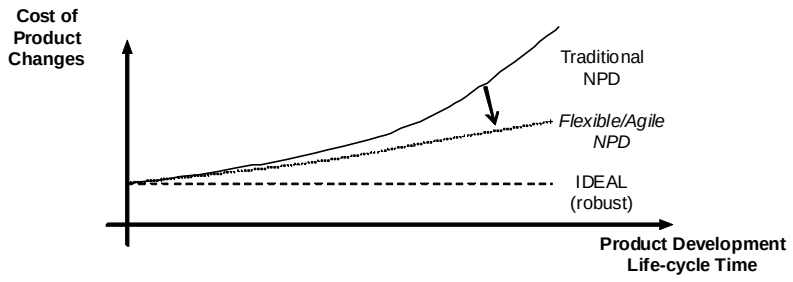

Fig. 8. Characteristic cost-of-change curves of different NPD approaches

In general, there are many possible ways to achieve the flatter slopes of the cost-change curve. Like discussed in Sect. 2.1.2.2, flexibility in NPD requires typically dedicated investments (e.g., in product architectures, extra capacity, options). Agility is from that point of view a strategic NPD capability investment. Consequently, the return-of-investment should be balanced and amortized over a longer period of time for sustainable competitive advantages. For example reducing the ñtechnical deptò of an aging product design may require additional midstream work but decrease the cost of future changes.

Also more formal cost calculus approaches have been proposed (Caswell and Nigam 2005). A general problem is the lack of (published) quantitative evidence of actual cost of change values and formulas in different disciplines (Smith 2008).

On the whole, care should be taken not to oversimplify complex product development systems with idealized models such as in Fig. 7 and Fig. 8. In particular, the underlying cost dynamics are in practice hardly ever fully continuous throughout the entire product life-cycle. For example, there may be certain major architectural decision points which limit the future change space radically. Such discontinuations could make the ideal models only piecewise valid. Moreover, disruptive changes in the competitive and technological environments can make the proactive preparations and current learning partially obsolete, thus again increasing the costs (Fig. 7).

Overall, this is about investment appraisals and evaluation. In principle, there is a risk of over/under-investing to agility (Overby, Bharadwaj and Sambamurthy 2006; Mirakaj 2008). Currently there is no one universal technique used for such purposes in IS/IT development. The evaluations can be approached from multiple different points of view, such as (Irani and Love 2002):

- economic ratios and discounting
- strategic (alignment of the investments with the business goals)
- analytic (e.g., value analysis)
- integrated (multi-criteria)

A parallel discipline here is the investment evaluation of agile/flexible manufacturing systems and supply chains (Heikkilä and Ketokivi 2005; Naik and Chakravarty 1992; Raafat 2002; Vehtari 2006). For example Turkulainen (2008) takes the viewpoint that integration benefits (in manufacturing) should not be taken for granted since there are in general cost/benefit trade-offs. The positive effects of integration are according to her findings not uniform in all performance dimensions.

The strategic viewpoint is the key concern considering the benefits and costs for developing the agile capabilities in the NPD functions. We expect the same ideas to be applicable with respect to agility in the NPD software production context.

In sum, different organizations may choose and are able to respond to changes in different ways depending on their agile capabilities. There is a need for guiding such strategic and tactical decisions in industrial NPD organizations. Current published work lacks such guidance, and more multidisciplinary research is needed in integrating the business, engineering, and large-scale organizational concerns of agility.

2.2.3.3 *Capability Development*

Overall, the agile improvement is subject to organizational change management in general. Depending on the market circumstances in one hand, and the competitive position and current capabilities of the organization on the other hand, different companies (organization units) may choose to conduct a radical transition to agility or to implement incremental changes.

Following Scheinôs (1999) generic change model, the fundamental driving questions should be answered first:

- Why do we need agility? How agile should we be?
- What is our current state, respectively?

The gap between the current agile capabilities state and the desired strategic goal state should then be addressed. An important part is to revisit and iterate this process frequently. A fully agile organization is proficient at continuously reorganizing and reconfiguring itself (Dove 2004).

One way of making this process more specific is to conduct systematic agility assessments. Such assessments could help realizing the actual needs for agility, and to evaluate the existing agile capabilities. Improvement actions can then be focused accordingly.

Several different assessment approaches have been proposed in different disciplines like surveyed in Table 9. Again, they vary considerably in scope and depth. Assessing agile software development is discussed in Sect. 2.3.3.

Table 6. Agility assessment approaches (chronological order)

Publication	Field, Scope	Approach / Principles
(Dove et al. 1996)	Enterprise, organizational development	• Agile Enterprise Reference Model
(Sharifi and Zhang 1999)	Manufacturing	• tabularization of company agility needs, capabilities, practices • self-assessment
(Conboy and Fitzgerald 2004)	ISD	• Agility Assessment Framework
(Lin et al. 2006)	Manufacturing, supply chains	• Agility Index

The agility metrics (Table 5) are related to these assessments. In particular, it is important to understand whether to assess/measure the effects of agility (business outcomes) or the capabilities that enable agility, c.f., Fig. 6.

Currently there is no one uniform way (framework) of agility improvement. Table 30 (Appendix) presents a literature overview of different approaches to improving agility in different disciplines. Like the definitions of agility (Table 27), they vary considerably in scope and focus. It is important to realize how agile capabilities and their development range from discipline-specific aspects to generic organizational ones (Table 4).

In all, there is a fundamental question of why, where and when an organization (entity) should strive for agility improvements (agility drivers). One should also determine how ñmuchò agility is really necessary (Lyytinen and Rose 2004; Sharifi

and Zhang 1999; Tsourveloudis et al. 1999). Those strategic questions can be reasoned in structural and temporal dimensions.

The organizational structure levels can range from individual persons (workforce agility) up to the entire company, and possibly even further to inter-enterprise leveling (enterprise agility; c.f., Fig. 1 and Fig. 6). The temporal dimension factors stem for instance from the life-cycle phase of the product portfolio, technology development, marketing strategy and market positioning, and also the company (unit) business stage (start-up vs. established) (Kotler et al. 1996 / Ch. 13; Levine 2005; Chillarege 2002; Tyrväinen et al. 2004).

Once the organization (unit) has recognized and understood its current agility drivers and needs, it can plan appropriate improvement actions ï if necessary. In principle, this means creating and developing the enabling capabilities and removing the possible obstacles and impediments (if any for the time being). This results in building the agile capabilities (resources, assets, structures, processes, values) like depicted in Fig. 6.

Johnson et al. (2006) suggest that certain levels of threshold capabilities are necessary in any competitive environment just to be able to maintain reasonable business. Like with strategic development in general, there are different possible strategic change approaches to agile transition and improvement: adaptation, reconstruction, evolution, and revolution. The two general transformation strategies applied in practice are top-down and bottom-up approaches (Day 1994). None has been shown to be superior so far. Much depends on the historical background of the company (organization unit) and its current context-specific needs and capabilities.

Middle managers may play key roles in large organizations (Ferrarini 2008). When agility contradicts with the prevailing fundamental assumptions and values of the organizational culture, a deep paradigm shift may be required to transform the organization profoundly (Levine 2005; Northover et al. 2007; Schein 1999).

Finally, it is fundamental to realize that agility is in principle not a static property. Any organization could also to some extent lose its current agility for example due to unnoticed, gradual shifts in the competitive environment, or reduce it unintentionally as negative side-effects of internal actions (like restructurings). Thus, it needs constant attention and contextual adjusting ï sometimes even rebuilding (Doz and Kosonen 2008). This resonates with the organizational dynamic capability views (Eisenhardt and Jeffrey 2000).

2.3 Agile Software Development

Agile software development is basically a software engineering discipline-specific implementation of the general-purpose agile NPD capabilities discussed in Sect. 2.2. It follows certain key principles and values (Sect. 2.3.1). They can be realized with various different agile software methods (Sect. 2.3.2). Like with organizational agility improvement in general, adopting them requires typically dedicated improvement activities, including creating enablers and removing possible impediments (Sect. 2.3.3).

2.3.1 Principles

There is no uniform definition of óagile software developmentô, but the Agile Manifesto (2001) is the *de facto* outlining of the key emphasis. A less-cited, more business competence oriented characterization is the Declaration of Interdependence (Agile DOI 2005).

The Agile Manifesto outlines certain overall values and principles of agile software development, but it is not an exact definition (Conboy and Fitzgerald 2004). For instance Anderson (2004) characterizes agile software development methods simply as ñmore profitableò than traditional software process models. Highsmith (2004) defines agility in terms of responding to and creating changes while balancing flexibility and stability.

Table 7 summarizes how óagile software developmentô is typically outlined in the relevant literature. Essentially, they align with the more general definitions of agility (Sect. 2.2.1) summarized in Table 27 (Appendix).

Table 7. Definitions of software development agility (chronological order)

Publication	Definition
(Aoyama 1998b)	*Quick delivery, quick adaptations to changes in requirements and surrounding environments*
(Cockburn 2002)	*Being effective and maneuverable; Use of light-but-sufficient rules of project behavior and the use of human and communication-oriented rules*
(Highsmith 2002)	*Ability to both create and respond to change in order to profit in a turbulent business environment*
(Anderson 2004)	*Ability to expedite*
(Larman 2004)	*Rapid and flexible response to change*
(Schuh 2005)	*Building software by empowering and trusting people, acknowledging change as a norm, and promoting constant feedback; producing more valuable functionality faster*
(Lyytinen and Rose 2006)	*Discovery and adoption of multiple types of ISD innovations through garnering and utilizing agile sense and respond capabilities*
(Subramaniam and Hunt 2006)	*Uses feedback to make constant adjustments in a highly collaborative environment*
(Ambler 2007)	*Iterative and incremental (evolutionary) approach to software development which is performed in a highly collaborative manner by self-organizing teams with "just enough" ceremony that produces high quality software in a cost effective and timely manner which meets the changing needs of its stakeholders;*
(IEEE 2007)	*Capability to accommodate uncertain or changing needs up to a late stage of the development (until the start of the last iterative development cycle of the release)*
Wikipedia (2007)	*Conceptual framework for software engineering that promotes development iterations throughout the life-cycle of the project;*

Notably some definitions emphasize the ends without specifying the means. It is important to distinguish between the organizational goals (ends) of the software production and the means (e.g., agile software development methods) to achieve them (Fitzgerald, Russo and OôKane 2003). This resonates with the general characterization of agile capabilities in product development outlined in Sect. 2.2.1 as a spectrum of flexibility and efficiency.

2.3.2 Agile Software Development Methods

The background and intentions of commonplace agile software development methods is highlighted (2.3.2.1). They share similar basic working principles and targets (2.3.2.2). Current publicly available empirical information about utilizing them is surveyed (2.3.2.3).

2.3.2.1 *Origins and Scoping*

Currently there is a range of what can be called agile software development methods, for example eXtreme Programming (XP), Scrum, Feature-Driven Development (FDD), Adaptive Software Development (ASD), and Dynamic Systems Development Method (DSDM) (Abrahamsson et al. 2002; Boehm and Turner 2004; Cohen, Lindvall and Costa 2004). In addition to those general-purpose methods, there are also various in-house developments and more product-specific models, such as Mobile-D (Dagnino 2002; VTT 2008).

Notably most of the methods have actually originated prior to the publication of the Agile Manifesto, and the term óagilityô was attached to them subsequently (2001). For instance the DSDM method dates back to the early 1990s with some later agile practices support. In general, the origins of the current agile software development models can be traced to late 1980s and early ó90s. The different methods philosophically joined by the Agile Manifesto have evolved from multiple different paths like illustrated by Abrahamsson et al. (2003). For example Scrum was influenced by the New Product Development Game ideas of Takeuchi and Nonaka (1986) (Sutherland 2001).

The different agile software development models can be compared from many points of view, such as:

- What is their level of concern (individual vs. enterprise) (Boehm and Turner 2004)?
- What are their prescriptive project characteristics, and to what extent do they support key project activities (development, management, communication, decision-making) (Cohen, Lindvall and Costa 2004)?
- What is their life-cycle scope (Abrahamsson et al. 2002)?
- How much formalism (ñceremonyò) do they define (Larman 2004)?
- To what extent do they support the key characteristics of self-organizing, explorative work (Schwaber 2001)?
- What project problems does each method tackle (subject of Research Paper IV)?
- How discerning are they about the project environment (Schuh 2005)?
- What requisite skills and competencies do they expect (Turner and Boehm 2003)?
- What is their overall ñdegree of agilityò (in terms of flexibility, speed, leanness, learning, responsiveness) (Qumer and Henderson-Sellers 2008)?
- What are their value stream cost structures (Anderson 2004)?
- How do they compare and contrast with NPD process models (Vainio et al. 2005)?

Those various comparison viewpoints are also keys to agility improvement strategies (Sect. 2.3.3). However, there is a general problem of how to compare and select between different methods in practice since there are currently no uniform

frameworks (Cohen, Lindvall and Costa 2004). Moreover, even the basic meaning of ‘agile’ in software development methods has evolved over the years in diverse ways.

Notably, like with the general concepts of agility (Sect. 2.2.1), the terminology is not uniformly agreed here. In particular, some authors distinguish between ‘methods’ and ‘methodologies’ while other ones use only the term method (Cockburn 2002; Henderson-Sellers and Serour 2005; Larman 2004). In general, methodologies can be interpreted to encompass methods. In addition, software life-cycle models are then parts of them. However, this thesis uses the terms ‘method’ and ‘process model’ for simplicity to avoid taxonomic misconceptions and to descope philosophical and social elements (ideology) of the research like set in Sect. 1.3.

2.3.2.2 *Premises and Focuses*

Overall, the basic premise of current agile software development models is that a small, co-located self-organizing team working closely together with the customer(s) can create a high-value product cost-effectively with frequent increments and short iterations. Skilled and apt people are keys to this.

Typically, the following benefits are then advocated (Highsmith 2007; Schwaber 2007):

- increased customer satisfaction
- reduced time-to-market (better ‘time-to-benefit’)
- increased quality
- improved project portfolio and product management (project types, features)
- improved product development investment management (control and flexibility)
- reduced ‘waste’ (increased efficiency, productivity, development cost)
- better predictability (visibility)
- better risk management (risk reduction)
- better workforce morale (developer satisfaction, well-being)

There is a growing body of empirical support for those advantages (Table 28), but the statistical rigor is still uneven to be fully conclusive (Sect. 2.3.2.3). Nevertheless, in general, these elements match well with the typical problems of turbulent software product development environments (Baskerville et al. 2006):

- time-to-market pressures
- productivity demands
- fluid and ambiguous product requirements
- changing environment

A key principle of the agile software development models is that they expect uncertainties and consequently changes throughout the product development life-cycle. That is, the traditional forecast-based batch project planning is replaced with continuous feedback-based value-driven planning cycles (Nerur and Balijepally 2007; Poppendieck and Poppendieck 2004; Sidky 2007). With appropriate self-management of versatile workforce, this should result in making the product development more adaptive.

Working under such embraced uncertainties, agile software development models base their risk management essentially on confronting the uncertainty areas early and

as frequently as necessary in order to understand and thus reduce the uncertainties as soon as new information is learned (ñfail fastò). Such continuous uncertainty (risk) management is an intrinsic part of agile software development (Leishman 2001). Table 8 summarizes the general approaches to address different uncertainty areas with agile software development models.

Table 8. Uncertainty management approaches in agile software development

Uncertainty Area	Agile Software Development Approaches
Product	• close customer involvement (customer-driven development) • incremental delivery • periodic reprioritization of the product features • assessing the current business value after each development cycle • encouraging and supporting creativity and emergent innovation
Process, technology	• short iterations providing rapid and frequent feedback • continuous and early product integration and test activities • learning and adapting designs and work practices accordingly • teamwork (empowerment, self-organization, collaboration and communication) • skill-based staffing
Environment	• dynamic replanning and reorganization • networking

Aligning with the general characterization of agile product development capability spectrum discussed in Sect. 2.2.1, in addition to being responsive to changes, agile software development models emphasize efficient and effective work. This is facilitated for example by team self-management (quick decision-making) and efficient information / knowledge sharing (preferably face-to-face). High (or negotiable, ñgood enoughò) product quality is also an inherent part. That is incorporated for instance with development practices (e.g., continuous integration and testing) as well as with people-centric tactics (such as group accountability and responsibility of the team deliveries). They rely heavily on skilled and flexible workforce.

Fig. 9 illustrates how the key elements of ideal agile software development outlined above show with respect to the traditional project management ñIron Triangleò. In general, the schedule and cost dimensions are fixed (by iterations), whilst the scope (functionality) is variable. A key control dimension is thus the time of changing the different attributes (e.g., functionality). Time-planning (ñrhythmsò, ñtaktò) is fundamental to this. In dynamic market environments the weighting and balancing of the different parameters are often subject to change (Levine 2005). However, there is lack of comprehensive management models for steering that in practice. The theory base is incomplete (e.g., effort estimation).

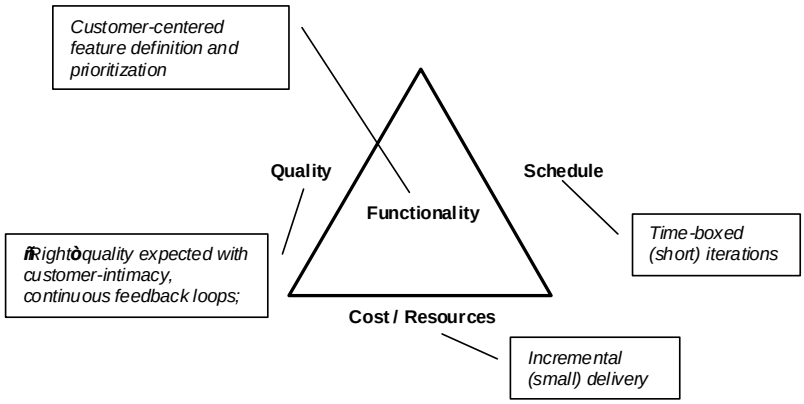

Fig. 9. Key management attributes of agile software development

With respect to the basic general new product development investment model shown in Fig. 4, agile software development models strive to compress the scale (ñtime-to-cashò) with short iterations and incremental delivery. The aim is thus to drive the customer value up soon while bringing the risks down promptly (Guckenheimer and Perez 2006). Overall, such mode of operation is supposed to maximize the right value delivery and minimize the risks of false investments. In many current fast-paced competitive environments the market windows are often emergent, and the product life-cycles get shorter. This compression and opportunistic concentration means, notably, that the underlying traditional assumption of a linear time-scale in Fig. 4 may no longer be fully applicable. Then again, the theory base is not fully developed (e.g., value analysis), and the empirical support is limited.

Notably, basically none of those principal ideas of agile software development is fundamentally new in itself. The key point here is their weighting and combined interplay. A fundamental trait is the emphasis on ñsoftò project management tactics for example with respect to the project goals and stakeholder expectations (Crawford and Pollack 2004). Agile software development models emphasize people factors and sustainability. óAbove-averageô self-organization, collaboration and learning capabilities are thus necessary. Generalists are preferred to specialists.

In all, consequently, it is important to see the underlying ï even implicit and unstated ï working assumptions behind the principles of different agile software methods (Glazer et al. 2008; Leishman 2001; Levine 2005; Nerur and Balijepally 2007; Turk, France and Rumpe 2005; Turner 2007). In particular, there are certain situations and software project conditions which may make some assumptions invalid and thus limit the applicability, such as reliance on tacit knowledge sharing and limited documentation with large project teams and complex systems products. The project may need to prioritize for example predictability, flexibility, or visibility (Ambler 1998).

It is also important to realize that the key elements of agile software development are in principle technology-independent ï i.e., they could be applied to ónon-softwareô

products (subsystems) as well (Smith 2008). Consequently, the same basic ideas apply for agility in NPD in general (Sect. 2.2.2).

The latter viewpoint is especially relevant for this thesis work. It is important to see how the current agile software development models scope and focus in general with respect to the generic NPD process (Fig. 3). In particular, much of the larger-scope organizational activities (like marketing and production) are reduced to the role of the Customer (product feature planning, acceptance testing). That is, agile software development models mostly concentrate on the actual product construction at hand, making simplifying assumptions about the product life-cycle front-end and product commercialization activities ï like (Kotler 1994):

- Is the product technically and commercially sound enough?
- Do the product sales satisfy the company objectives?
- Should we modify or enhance the mature product?

Overall, it is enlightening to position agile software development into a larger context. There are many considerable parallels between agile software development and agility in other fields as discussed in Sect. 2.2.2 (c.f., Table 4). Much of this can be explained by understanding the historical progress path of lightweight, agile software development approaches as a countermeasure to rigid, heavyweight organizational process development problems in large-scale contexts (Glazer et al. 2008).

In general, current agile software development models concentrate mostly at the single-team level and do not comprehensively address wider organizational dimensions (other projects/products, management hierarchies) or longer-term temporal dimension of product life-cycles (Overby et al. 2006; Chillarege 2002). The coverage varies significantly following the historical development paths and the chosen (or emerged) focus of the different methods (Abrahamsson et al. 2003).

In contrast, for example lean product development emphasizes more enterprise level elements (Liker 2004; Morgan and Liker 2006; Ward 2007). Agile software development methods are mostly limited to the micro-context of software product design and delivery (exploitation) whereas higher-level innovation capabilities are needed during explorative phases (such as new base technology adoption) (Lyytinen and Rose 2006). Not surprisingly, the latest trends in agile software development seek for such combinations and extensions in larger scale for instance by synthesizing general Agile and Lean principles discussed in Sect. 2.2.1 (Benefield 2008; Highsmith 2005; Leffingwell 2007).

2.3.2.3 *Empirical Evidence*

Current published empirical evidence of agile software development is uneven. The scope and level in these works range from small teams to large-scale enterprise agility issues, but this is not always obvious due to the conceptual disparity (c.f., Table 7). Furthermore, the dimensions of analysis span from pure software engineering technical disciplines to more general management and organizational studies ï again often without clearly stating so. All this causes the prior research to be scattered across a large knowledge space with diverging goals and sometimes even confusingly varying degrees of rigor and empirical support.

There are certain inherent difficulties in collecting and analyzing empirical evidence of agile software development:

- Many of the interrelationships between the key software project attributes (Fig. 9) are not fully understood even in traditional software engineering economics. Agile software development models stretch them further.
- Some aspects (e.g., cultural, teamwork, and people factors) require new measurement approaches beyond traditional software engineering metrics (e.g., behavioral sciences, organizational dynamics).
- The long-term effects of agile software development in complex large-scale product development set-ups (e.g., product lines, legacy platform) are not yet clearly visible in many organizations.

Nevertheless, under right conditions, there have been favorable results of applying agile software development models like summarized in Table 28 (Appendix). So far the reported findings are most often about the XP and Scrum methods and practices. Typical benefits mentioned are better time-to-market targeting, increased productivity, and higher quality. In addition, certain organizational improvements ï such as better developer motivation ï are highlighted. A frequently mentioned trait is that the basic agile software models and practices have been adapted and combined in industrial settings (typically XP with Scrum). Notably the published results in large-scale organizations are still scarce.

Embedded software product development has certain additional intrinsic complications compared to pure software product development like discussed in Sect. 2.1.3.2. Agile software development methods have nevertheless been successfully applied also in embedded software development environments (ITEA-AGILE 2007b; Manhart and Schneider 2004; Ronkainen and Abrahamsson 2003; Salo and Abrahamsson 2008). One constraint with deeply embedded systems development is that there may not be any direct user interface for getting immediate visible customer feedback (Turk, France and Rumpe 2005).

However, overall, it is not clear how much and under what circumstances each element discussed above really contributes to the advocated benefits of using agile software development models. A general problem is that there is still lack of systematic body of statistically sound (quantitative) research evidence, and many of the results are anecdotal in nature (Abrahamsson 2006; Dybå and Dingsøyr 2008; Turk, France and Rumpe 2005). In addition, it is not obvious what exactly to measure, since for instance social factors (e.g., higher motivation and job satisfaction) may be difficult to specify objectively. Qualitative measures may be more appropriate.

2.3.3 Agile Adoption

Currently many industrial organizations are contemplating agile software development and agile capability improvements in general (Schwaber 2005; Version One 2008). This overall strategic move is often referred to as Agile Adoption / Agile Transformation.

In particular in large product development organizations, there are many obstacles to overcome in order to be able to implement and take full advantage of the agile software process models. Table 29 (Appendix) illustrates typical issues faced in practice. Notably in large organizations the problems are often related to the

integration of the agile software development teams with the rest of the organization, product/project portfolio management, and organizational governance ï i.e., the areas that the current agile software development models do not comprehensively address like concluded in Sect. 2.3.2. Limitations with large-scale product development infrastructure and tools may be serious practical obstacles.

These are often key considerations for successful agile adoption and improvements. Sidky (2007) suggests that if there are major impediments, the organization should first correct them before proceeding with the adoption. However, in typical industrial environments such an ideal is often not fully achievable soon enough, but the adoption process should nevertheless be advanced. The key is to openly recognize the situational, often path-dependent problem factors and attempt to overcome them gradually possibly with multiple simultaneous approaches and partial solutions.

There are some ógreadinessô assessments to examine the starting point of the organization (project) in order to determine the current abilities to adopt effectively agile software development. For example the DSDM process model has such accompanying Suitability Risk List questionnaires (DSDM 2004).

Following the initialization, there comes the need to assess the current status and progress of the adoption. However, currently there are no standardized agile assessment models publicly available (Pikkarainen and Mäntyniemi 2006). In fact, the very idea of assessing (measuring) agility is somewhat controversial and under debate, and some authors even doubt the need. In particular, there has been much debate about how the CMM models and agile software development are related ï if they should be at all (e.g., Kane and Ornburn 2002; Pikkarainen 2008; Turner 2007). Recent developments approach them as complementary rather than conflicting, in particular for large organizations (Dutton and McCabe 2006; Glazer et al. 2008).

Nevertheless, the agility metrics (Table 5) are related to these assessments. In particular, it is important to understand whether to assess/measure the effects of agility (business outcomes) or the capabilities that enable agility, c.f., Fig. 6 (Lappo and Andrew 2004).

Table 9. Agile software development assessment approaches (chronological order)

Publication	Field, Scope	Approach / Principles
(Boehm and Turner 2004)	Software process, organization	Agile Home Ground: • project size (# of personnel) • product criticality (impact of software failures) • development dynamism (requirements change rate) • personnel (skills, competence, experience) • culture (favoring order vs. emergency)
(DSDM 2004)	Software process, organization	• Organization/Project Suitability Risk List
(Lappo and Andrew 2004)	Software process, organization	• organization-specific agile goals attainment (process, organization and people, tools, software design and quality) • relative performance measurement (benchmarking)
(Hansson et al. 2006)	Software process, organization	• degree of implementing the Agile Manifesto values (individuals over processes, working software over documentation, customer over contract, change over plan)

Publication	Field, Scope	Approach / Principles
(Pettit 2006)	Software process	• Agile Maturity Model
(Highsmith and Wysocki 2006)	Software process, organization	Level of implementation of certain key agile practices: • customer involvement and collaboration • software development process (e.g., iterative plans) • quality and testing (e.g., automated testing) • engineering management (e.g., effectiveness, value vs. efficiency) • feedback and learning (e.g., learning support)
(Pikkarainen 2008; Pikkarainen and Mäntyniemi 2006)	Software process	CMMI applied for assessing agile software development: • evaluating the strengths and weaknesses of the current (agile) processes in order to plan improvements • mapping CMMI goals to agile practices • finding suitable agile practices
(Sidky 2007)	Software process, organization	Agile Measurement Index: • embracing change to deliver customer value • planning and delivery of software frequently • human-centric • technical excellence • customer collaboration

Currently there is no one uniform way (framework) of agile software development improvement. Table 30 (Appendix) presents a literature overview of different approaches applied. Like the definitions of agility (Table 7), they vary considerably in scope and focus. It is important to realize how agile capabilities and their development range from software-specific aspects to generic organizational ones.

The agile capabilities of the company are a combination of the different elementary organizational components. Consequently, different software product development teams may have to contribute to this in different ways. In particular, not all the teams may need to be equally agile depending for example on their local product type characteristics.

In general, the fundamental high-level agile principles should guide the lower-level practices. A proactive NPD organization may for example choose to build a framework of software process fragments to be quickly configured and tailored for specific project instances. Software method tailoring can be positioned at project (micro-level), organization (macro-level) and even industry levels (Fitzgerald, Russo and OôKane 2003; Henderson-Sellers and Serour 2005). Notably the agile software development models themselves can be utilized for the improvement (Salo 2007).

Like with strategic development in general, there are different possible strategic change approaches to agile transition and improvement (Sect. 2.2.3.3). However, remarkably, in industrial practice many agile software development initiatives have emerged from the team level in individual software projects (Mar 2006a). Often the starting point has been in their practical problems with the existing development processes and practices, which are or have become unsatisfactory under new competitive and technological circumstances.

In particular in large organizations such a bottom-up team level development must nevertheless be eventually supported by the surrounding organization, and therefore a top-down strategy is also necessary for example to ensure management support and organizational alignment. A hybrid model combining both top-down and emergent bottom-up approaches is thus often a working strategy in practice (van

Schoonenderwoert 2007). Human-centric approaches (emergent strategies) are increasingly advocated. However, profound organizational changes necessitate usually more top-down actioning (Highsmith 2002 / Ch. 16).

2.4 Summary of Knowledge Gaps and Research Needs

Sect. 2.1-2.3 establish the conceptual background and survey the key literature of this research field. The following concludes this groundwork by first recapping the software-intensive NPD context problem space (Sect. 2.4.1). It then summarizes the prior interdisciplinary agility-oriented research streams while pinpointing the main knowledge gaps and research needs. Having done this, it is possible to put the agile software development research problem of this thesis into the relevant context and rationalize the research questions (Sect. 2.4.2).

2.4.1 Problem Space

The general problems of software-intensive new product development in current competitive environments are summarized (2.4.1.1). They introduce a range of research issues for agile software development (2.4.1.2).

2.4.1.1 *NPD Problems*

Over the years there have been various investigations of the software-intensive NPD problem space. There are many field studies of project success/failure factors ï even ñchallengedò and ñtroubledò projects ï and NPD performance. Some factors, such as uncertain and unstable product requirements, market entry strategies and technological dependencies, are addressed more often than others, but there is no general agreement and mature understanding of all the related factors and their interdependencies. In particular their interplay within organizational dynamics is not well understood (Brown and Eisenhardt 1995; ITID 2008; Kotler et al. 1996; Levine 2005; Mäkelä 2008; Trott 2005).

One viewpoint of the NPD problem space is to consider it as a net of decisions to be made. Krisnan and Ulrich (2001) survey the empirical NPD literature from that point of view. As a conclusion, they call for cross-functional research to address in particular product development supply chains. This mirrors to Fig. 2.

This viewpoint is also one of the key premises in our research, considering large-scale NPD organizations and their software production. In an industrial company environment this is further complicated for instance by the business model considerations (Suikki 2007). It is necessary to master such multidomain knowledge to a certain extent in order to be able to improve the integrated software development function in the organizational problem context.

Moreover, many real-life problematic situations do not always link neatly into any one prescribed conceptual knowledge maps, but there are interdependencies and cause-effect-symptom chains (Brown at al. 2000). In turbulent environments the time-dependencies and rapid, even disruptive changes of the problem space create additional complexity. The new business model and strategy development suggest

even radically different approaches for the future competitive NPD enterprises (Chakavarthy 1997; Doz and Kosonen 2008; Mäkelä 2008).

With respect to academic research NPD is still a relatively young and immature field. An inherent source of research problems is its multidisciplinary nature. Current active NPD research topics include uncertainty management, value chain analysis, and innovation processes (ITEA 2004; Kahn et al. 2003). For the purposes of this thesis work, the following research threads are of particular relevance:

- NPD project success/failure factors
- NPD process flexibility and acceleration (ñnext-generationò NPD process models)
- organizational models of successful NPD

Due to the multidisciplinary nature and relative immaturity, the NPD research is subject to methodological concerns about the research rigor of the field studies (Ernst 2002). For instance the lack of common measures of NPD success and single-informant bias are fundamental scientific threats for generalizing the results, although most observations appear to be intuitively right in practice.

2.4.1.2 *Research Needs*

Overall, a critical analysis of the related prior work reviewed in Sect. 2.1-2.3 reveals the current nascent state of the agile software development field. The lack of rigorous conceptual base and uniform common definitions has led to a proliferation of diverse practical approaches and theoretical studies. This is understandable given the mainly practitioner-based origins of agile software development.

There is a need for comprehensive, systematic context-specific understanding of the NPD problem space in general, and the role of agile software development in the improvement in particular. Current generic knowledge frameworks on the one hand and miscellaneous case studies on the other do not provide such directly applicable pragmatic aids for the practitioners. Moreover, the conceptual theory-building requires further interdisciplinary understanding.

In summary of the survey and discussion in Sect. 2.3, what is currently known and have been experienced about agile software development is in general as follows:

- Agile software development models aim to tackle many of the typical software project problems faced in practice in turbulent business environments.
- Agile software development methods have successfully been applied to a variety of different software development types ranging from pure software products to embedded systems in many application domains. However, it is not well understood, which software types are the most suitable ones (ñsweet spotsò), and furthermore, if there are any particular product development types which may disfavor agile software models.
- There is a diverse body of empirical evidence of both successes and obstacles observed in practice in various environments.
- However, it is not clear how much and under what circumstances agile software methods really contribute to success (lack of statistically significant rigorous evidence).

- The long-term effects of agile software development are yet to be seen due to the mostly nascent state of the development. Many large-scale systems have very long lifecycle times (up to ten years).
- There are many areas in particular in large NPD organizations, which are currently not fully (if at all) addressed by the current models. Specifically, large-scale product management, engineering management, and organizational development require additional measures.
- Traditional organizational software process improvement methods may not be effective enough in rapidly changing, fast-cycled product development environments.
- The human-centric value-based philosophy of agile software development (Agile Manifesto) may require fundamental paradigm shifts in traditionally structured and managed organizations. Other disciplines ï such as organizational development (OD) ï address such issues.

Following the development characteristics and success/problem factors presented in Sect. 2.3, the current research trends in agile software development focus on the following key topics and areas in particular summarized in Table 10.

Table 10. Agile software development research areas

Area	Publications	Relevance for This Thesis
large-scale agile software development ï also distributed, even globally	(Kähkönen 2004; Lindvall et al. 2004; Leffingwell 2007; McMahon 2002; McMahon 2005)	• integrating (accommodating) agile software development teams in large-scale organizations
specific application areas (e.g., mobile and automotive embedded software development)	(Dagnino 2002; Fitzgerald, et al. 2006; Greene 2004; Khan and Balbo 2005; Manhart and Schneider 2004; Pyhäjärvi 2006; Salo and Abrahamsson 2008; Vanhanen et al. 2003; Wils et al. 2006; Välimäki and Kääriäinen 2008)	• embedded systems domain (in particular, telecommunications)
cost/benefit analysis of agile software development (e.g., in terms of productivity, quality, product maintainability), metrics	(Anderson 2004; Hartmann and Dymond 2006; Heikkilä and Holmström 2005; Itkonen et al. 2005; Lyytinen and Rose 2004)	• agile software engineering economics
organizational models for agile software product development (agile frameworks)	(Ambler and Kroll 2007; Dagnino 2001; Highsmith 2005; IEEE 2007; Karlström and Runeson 2006; Kivelä 2007; Mäkelä 2008; Wallin et al. 2002)	• NPD organization management (governance)

Area	Publications	Relevance for This Thesis
transforming traditional software development organizations into agility (agile adoption)	(Ågerfalk and Fitzgerald 2006; Benefield 2008; Boehm and Turner 2005; Börjesson 2006; Ceschi et al. 2005; Dutton and McCabe 2006; Hansson et al. 2006; Highsmith 2007; Mar 2006a; Pikkarainen and Mäntyniemi 2006; Pyhäjärvi 2006; Salo 2007; Sidky 2007)	• NPD organizational development
cultural aspects, values, and other organizational development considerations (innovation, business agility)	(Aramand 2006; Cockburn 2007; Coplien 2004; Levine 2005)	• large-scale organizational enablers of agile capabilities • strategic agility

In all, from the critical research point of view, there have so far been only provisional conceptualization and preliminary theory-building, mostly basing incrementally on existing mature constructs and disciplines. However, more systematic research is needed to first capture the current developments, and then to advance to potentially relevant multidisciplinary areas (e.g., economics, organizational science). More empirical support is necessary accordingly.

Dybå and Dingsøyr (2008) have recently conducted an extensive literature survey on empirical studies of agile software development. They conclude that there is in the one hand a clear need for more empirical studies about the benefits and limitations of agile software development, but on the other hand also the research methods should be strengthened. Overall, Dybå and Dingsøyr call for more research on management-oriented aspects of agile software development as well as more extensive investigations about how and when it is beneficial to adopt agile software development methods in practice.

Moreover, Dingsøyr et al. (2008) outline a roadmap for future research (up to 2015) of agile software development based on a current state analysis. They find the following major gaps and needs in the current research and knowledge in order to comprehensively understand agile software development:

- more empirical studies with rigorous research methodologies following more established field paradigms (in particular information systems research)
- analyzing experienced teams and (large) organizations in complex industrial real-life settings
- taking more into account related business competence areas (like management science and organizational development)

This thesis attempts to address to some extent all those three key areas by working in a large-scale, established industrial organization developing complex systems products. The research work links software development with the NPD field.

Empirical support of the underlying theories is desirable accordingly. This thesis builds mostly on combining the prior empirical evidence about flexible software-intensive NPD on the one hand (Table 25, 26), and what is has so far been experienced in agile software development and its adoption on the other hand (Table 28, 29). However, while the NPD field evidence is more mature recorded over a long

period of time, the empirical software development research is much more limited. Considering the empirical support of the underlying work of the thesis, there is not much prior integrative evidence linking NPD and software development together. Multidisciplinary studies are rare.

2.4.2 Positioning the Thesis Research

The research problem is put into the relevant scientific context (2.4.2.1). This makes it possible to rationalize the specific research questions of the thesis (2.4.2.2).

2.4.2.1 *Connections*

The main research problem of this thesis (Sect. 1.2) is by nature multidisciplinary. Consequently, the key idea is to investigate the question from different perspectives, looking for and combining applicable knowledge and solutions from related disciplines and research fields. However, the main focus areas are project management and software engineering. Table 11 outlines this bridging.

Table 11. Software-intensive NPD research cross-connections

Field	Key Questions, Knowledge Areas
Computer Science (CS)	fundamentals of software technology (discipline)
Software Engineering (SE)	How to construct software (products) economically?
Information Systems (IS)	How to develop software systems (IT)?
Project Management (PM)	How to organize and govern the software development and engineering work efficiently (time-cost-features)?
Operations Management (OM)	How to produce and deliver the (software) products effectively?
Product Development (PD)	How to conduct successful NPD in general?
Organization Design (OD)	How to structure high-performance PD organizations?
Organization Theory (OT)	How to improve PD organizations?
Marketing	product (features) offerings
Business Strategy Development	How to achieve firm performance goals with NPD?

In order to be able to comprehensively address the research problem of this thesis, a wider range of NPD disciplines and business competence areas are of certain relevance in conjunction with the actual software product development functions. There are two main research threads here. The primary discipline of this thesis research is software engineering (management). Agile software development is essentially an extension to the traditional software engineering by emphasizing, augmenting and reshaping certain key areas. The second main thread is NPD, which is the host discipline in this thesis research context. The traditional NPD models have subsequently been revised toward more flexible product creation.

In addition, there is a range of other potentially related business competence areas and disciplines like discussed in Sect. 2.1. However, it is not possible to cover all such areas within the scope of this thesis work, and most of them are excluded here like defined in Sect. 1.3.

A fundamental insight here is to understand how the concepts of agility and flexibility are realized in different disciplines. Furthermore, what is more profound is to see, that the level and scope of different research streams vary considerably ï

sometimes even in unclearly defined and confusing ways. In particular, the dimensioning can vary from individual (software) teams up to entire enterprises.

Again, most of this goes beyond the scope of this thesis research focusing on the software team level. However, it is important to realize that such connections and dimensions do exist, and they offer potentially further research avenues.

2.4.2.2 *Rationale*

The profound tenet of this research work is that software development agility in conjunction to NPD should be approached with a multidisciplinary perspective. Such cross-functional thinking is not totally unusual, but to the best of our knowledge not much research results have this far been published accordingly (Table 10).

Some authors have pointed out the need and potential for such interdisciplinary research (Mäkelä 2008; Nambisan 2003; Rauscher and Smith 1995; Smith 2007). In general, the links and possible interdependencies between the different areas shown in Table 11 and possibly also with other business competence areas (Sect. 2.1.1.1) are currently not comprehensively understood. The cross-functional linkages and connections offer thus potentially innovative solution possibilities.

A few research publications make an explicit connection between software development and NPD. For instance Nambisan and Wilemon (2000) compare and contrast prevailing software development and NPD. For example Boehm and Turner (2005) underline the integrated systems engineering perspective. Aramand (2006) calls in general for more cross-disciplinary research between strategic management and software product development.

Some authors have recently made agility-oriented connections between software development, NPD and other related disciplines. For instance Larman (2004) underscores that software production is by nature closer to new product development than predictable mass production. Smith (2008) proposes enhancing NPD models with agile software development principles, and Turner (2007) sees equal possibilities in the traditional systems engineering processes. Vainio et al. (2005) compare and contrast general-purpose NPD process models with commonly used (agile) IS software development methods. Cockburn (2007) parallels agile software development with agile manufacturing principles.

In sum, there is lack of knowledge and deeper understanding of embedded software product development in NPD context in particular under the new, changing competitive circumstances facing many large industrial organizations today (Abrahamsson 2007). Large-scale agility is still an immature area in practice, and specifically the benefits and costs of agile software development in such large organizational settings is not yet well understood. Furthermore, the way agile software development can be realized requires further investigations (Table 29).

In all, the general NPD research directions coupled with the above approaching establish the rational basis for this research. This thesis maintains that in order to be able to comprehensively develop the strategic capabilities of the embedded software production functions, it is necessary to consider the wider organizational context and the competitive environment. The essence is to realize how agile software development capabilities can support the NPD strategy, which in turn should align with the overall business strategy of the company.

That line of reasoning leads to the research questions of this thesis about how agile software development can be utilized in large-scale NPD context as defined in Sect. 1.2. Table 12 recaps the research needs and knowledge gaps identified in Sect. 2.1-2.3, and links the research questions of the thesis to address them.

Table 12. Research needs and questions

Research Needs, Knowledge Gaps	**Research Questions** (Sect. 1.2)
understanding the essential NPD problems affecting embedded software projects in turbulent competitive environments (Sect. 2.1)	1. *What are the typical problems of large-scale NPD embedded software projects?*
rationalizing agile software development capabilities accordingly (Sect. 2.2, 2.3)	2. *What problems and goals does agile software development address?*
integrating agile software project teams into larger NPD organizational context (Sect. 2.3, 2.4)	3. *How can typical large-scale NPD problems be tackled with agile software development methods?*

3 Research Design

There is no one specific research method, which is completely perfect in this pragmatic research setting. All in all this thesis work follows the design scientific approach with the researcher as a participatory observer. Those principles are characterized (Sect. 3.1).

The empirical research case environment is described in Sect. 3.2. The particular industrial organization context is rich in potential research issues, but there are also many practical constraints. The nascent and interdisciplinary nature of the research field calls for mixed research designs.

The actual realization of the overall research work is then described (Sect. 3.3). The organizational connections are shown.

Last, the general quality criteria for the research design are defined (Sect. 3.4). Both the relevance and rigor aspects are important in this kind of an industrial research work.

3.1 Research Methods

Software engineering is an applied discipline. Consequently software engineering (information systems) research often uses multidisciplinary research methods (Mingers 2001). Fig. 10 presents a general-purpose taxonomy of all research methods (Järvinen 2004).

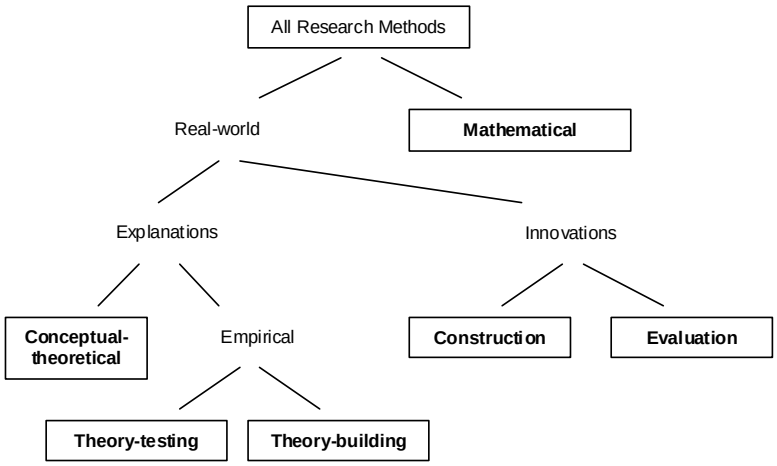

Fig. 10. Taxonomy of research methods after Järvinen (2004)

Like described in Sect. 3.2, this research work has been conducted in a large-scale industrial product development organization software process improvement context. With respect to the classification scheme in Fig. 10, the work is thus primarily about constructing and evaluating innovations (artifacts). In addition, there are also some empirical explanatory elements involved in order to be able to understand the underlying fundamentals stemming from the real-life observations.

The constructive part of this research is about Design Science (van Aken 2004; Hevner et al. 2004). The business environment sets the needs for the research questions, thus ensuring their relevance. The purpose of the research is to create innovative solutions (design artifacts, foundations, methodologies) for those problems based on the existing body of knowledge ï both organizational local as well as more general scientific knowledge of the discipline, ensuring scientific rigor.

Action Research is an increasingly typical method of conducting design-scientific research in software engineering (information systems) (Avison et al. 1999; Susman and Evered 1978). The iterative change/improvement cycle is repeated until a satisfactory and feasible solution is reached for meeting the organizational needs. Depending on the nature of the design artifacts, different evaluation methods may be appropriate (Hevner et al. 2004). Action research is often coupled with case studies. Qualitative data analysis (descriptions, illustrations and interpretation) is typical in such cases (Eisenhardt 1989).

Action research and case studies are becoming more and more commonplace in software engineering research, since the current research problems address increasingly managerial and organizational questions (Myers 2007). In such practice-oriented organizational research settings it is often necessary to understand not only software engineering (information systems) technical disciplines, but also (large-scale) organizational dynamics, people factors, and even social sciences to some extent (Jankovic 2005; Raelin 1997; Schön 1983).

Overall, the core principles of the action research approach can be summarized as follows (Routio 2007):

- The starting point should be in the current actual state of the activity, not just in theoretical assumptions.
- The problematic situation should be reflected from a wider point of view in order to be able to recognize more general patterns with respect to the uniqueness of the situation and consequently potential existing solutions.
- The original situation could be abstracted into a suitable theoretical model as a reference base of the changes.

This kind of a participatory, collaborative design (clinical research) is increasingly commonplace with research-in-industry settings. Ideally, there is a continuous interplay between the researcher and the target community with shared goals (Kiviniemi 1999). Recently for instance Börjesson (2006) and Suikki (2007) have applied similar research approaches in comparable industrial organizations.

Dybå and Dingsøyr (2008) argue that agile software development is still such an immature and unchartered (nascent) research area that exploratory qualitative studies are needed in the field research. Action research is in their opinion a particularly suitable approach in this context. Flexible case studies are in general appropriate

research approaches in such emerging research areas with potentially novel insights (Eisenhardt 1989).

These elements serve well the objectives of this thesis research work in the case organization environment (Sect. 3.2.2). The first steps of this research address general, more mature problem areas while the later ones focus on the specific, currently nascent issues of agile software development. Consequently, this research design is adjusted accordingly in terms of leaning to existing theoretical constructs and utilizing empirical data. The analysis is mostly qualitative. The fundamental nature of action research supports such adjustments and focusing during the research work (Heikkinen and Jyrkämä 1999).

The actual usage of the selected research methods discussed above varies during this research work to some extent in the different phases and steps. They are defined in described in detail in the individual Research Papers and summarized in Ch. 4. In all, it is important to realize the state of the prior research in order to be able to fit the research design appropriately in each phase (Edmondson and McManus 2007).

3.2 Research Environment

To begin with, it is necessary to understand the key characteristics and trends of the industry sector and the competitive environment (Sect. 3.2.1). The specific case organization key attributes can then be contextualized (Sect. 3.2.2), and the research constraints underscored (Sect. 3.2.3).

3.2.1 Case Industry Characteristics

The empirical background of this research work resides at certain business units of a telecommunications equipment vendor company in Finland over a period of several years between 2000-2007. It is beyond the scope of this presentation to describe the telecommunications business sector and product development industry in detail, but there are two noticeable global developments worth highlighting during the period of this research:

- the rise and rapid fall of the so-called ñdot-comsò around year 2000 (Levine 2005)
- the huge expenditure of telecommunications operators for the 3G networks licenses in the early 2000s

At the time of this research the global developments in the telecommunications field indicate in particular the following trends in the competitive environment:

- The business sector is in general gradually recovering from those downturns (Ante 2007)[6].
- There are some new entering equipment vendors with aggressive business models challenging the incumbents.
- The vendors as well as the operator customers are driven by strict business objectives of profitability and cost-effectiveness. The business models must be

[6] Interestingly enough, as of this writing (in fall 2008) the global economic turmoil is again causing stagnation in the telecommunications markets.

directed accordingly, affecting for example the product development portfolio/feature strategies.
- There is no one predominant network technology, but a range of different technologies combined with certain new ones emerging (e.g., WiMAX). This implies that large-scale equipment vendors may decide to include very different product types and platforms in their NPD portfolios at the same time.
- The telecommunications industry continues to be a significant developer of embedded software-intensive products (TNO/IDATE 2005).

Those overall business milieu trends have affected also this particular research case environment, causing considerable external turbulence, and consequently driving the needs during the research period. At the time of this thesis research work the organizational interests towards agile software development grew significantly due to those overall telecommunications business environment drivers (Vodde 2006; Vilkki 2007; Tanskanen 2008). This general movement also motivated the research setting of this thesis (Sect. 1.2).

3.2.2 Case Environment

This thesis research work has been conducted in an industrial telecommunications product development context like typified in Sect. 3.2.1. The work concentrates on one particular business unit of the larger organization. This particular case environment can be characterized as follows in Table 13 (at the time of this research). For confidentiality reasons only the overall scale is shown.

Table 13. Key characteristics of the industrial case environment (orders of magnitude)

ATTRIBUTE	Company	Case Business Unit
Field	• telecommunications (equipment and services)	• network element product embedded software (and hardware) development • network element management systems (workstation software)
Size of organization: • # of people • # of business units • # of sites • # of projects	 • 10000 (worldwide) • 10 • 10 (global) • 100	 • 1000 • n/a • multiple • 10
Typical project size: • # of people • # of sites • # of teams • team (persons) • duration (months)	• n/a	 • 100 • multiple • 10 • 10 • 10
Typical product size, complexity: • LOC • # of subsystems • software domains • expected life-time (years)	• n/a	 • 1M • 10 • multiple • 10

The case software product development environment has in addition to the overall industry-specific characteristics described in Sect. 3.2.1 the following particular complexities and drivers:

- hyper competition in the business environment (e.g., cost reduction needs)
- new market/customer needs (volatile and emerging product requirements)
- high product reliability requirements (even mission-critical)
- interdependencies to proprietary target hardware engineering and manufacturing

Within this case environment, the author has worked as a full-time Quality and Process Development Specialist in association with various embedded software product development projects. This insider position has made it possible to work in close contact with the practitioners of the software product development. In addition, there has been full access to all the relevant knowledge sources in the case environment (within the overall company confidentiality limits).

3.2.3 Strategic Concerns

In the particular type of the industrial environment described in Sect. 3.2.1-3.2.2, the research strategy must consider not only the academic relevance and rigor, but also the case organization current needs and constraints. In particular, it is usually necessary to focus on a narrow set of essential improvement items (ñburning issuesò) even though there often are many more potential topic areas and possible future needs (Hinkin et al. 2007). The key is to be able to recognize those problem areas, which are really caused by some local impediments (e.g., lack of resources, inappropriate tools), and the more fundamental ones with possibly enterprise-wide strategic significance.

In general, research and improvement in this kind of industrial environments is typically characterized and constrained by many practical factors (Kettunen 2000; Börjesson 2006; Lassenius 2006; Suikki 2007):

- There has to be an appropriate alignment and balance with the business objectives and the research objectives.
- The research work should be integrated to the business operations in order not to hinder the daily software production work.
- The relevant expertise is not necessarily evenly spread and available in the organization.
- New improvement changes must typically be aligned with the existing organizational process assets and legacy systems and tools.
- The life-cycle phase of the products under development drives the needs for improvements over time.
- In large organizations there are usually many stakeholders, each with possibly varying needs and priorities. Furthermore, the needs and priorities could be subject to change over time.
- It is not unusual that restructurings and other organizational factors affect the research and improvement plans even radically during the course of the execution ï in particular if the time period spans over several years.

In all, this kind of an industrial environment brings both opportunities and obstacles to conduct successful research. The relevance of the selected research

problems is usually straightforward to verify, and the research can be kept aligned with the current needs of the business environment. On the other hand, there might be practical problems to achieve high quality (rigor) of the research work. The research methods must be selected and adjusted accordingly.

3.3 Research Process

The research work is embedded into the organizational process development and improvement activities (Sect. 3.3.1). The cyclic research process comprises a longitudinal sequence of phases and steps conducted over several years (Sect. 3.3.2).

3.3.1 Interconnections

Like stated in Sect. 1.2, this research work focuses on product development process improvement problems in a case organization (Sect. 3.2.2). The main driver is to help the organization in its daily work following the philosophy of action research (Heikkinen and Jyrkämä 1999).

The research process joins the organizational software development and improvement activities. The research cycle augments the organizational improvement cycle, which is typically focused on short-term project-specific issues. The aim of the longer-term research cycle having a more generic focus is to support the organizational activities by extending the organizational knowledge base (learning). The external research publications and this compendium are ultimately targeted to increase the academic body of knowledge, which is in general one external source of organizational SPI references.

This is what can be called professional researcher-centered approach (Routio 2007). In this approach the researcher is assumed to be an (insider) domain expert capable of identifying and understanding the key problems of the affected practitioners, but the researcher is expected to conduct the actual studies mostly independently. The resulting improvement plans should then take into account also the organizational constraints about what can be changed in practice.

There are also some ethnographic research aspects in here. With respect to organizational learning, this approach could be characterized in a way as problem-based learning (Delaney et al. 2003).

3.3.2 Realization

This thesis work comprises six distinct research steps reported in the research papers I-VI, see pp. vii. We have repeated the action research cycle, but for each iteration have opened up new successive research viewpoints and propositions in order to gain deeper understanding and shaping of the research problem like depicted in Fig. 11. The research questions 1-3 (Sect. 1.2) phase the research. In particular, the starting point is no specific *a priori* theory, but the concepts emerge during the research (Eisenhardt 1989). However, a key is to reconstruct and reflect the organizational history (Kiviniemi 1999).

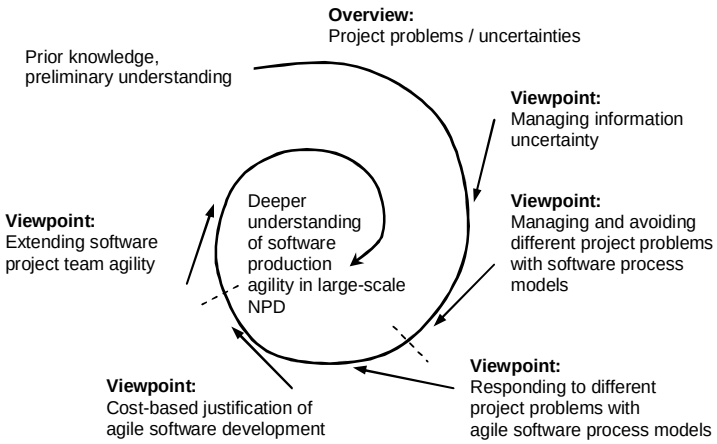

Fig. 11. Spiral model of the thesis research following Routio (2007)

In addition, while advancing, we have gradually shifted the hierarchical level (see Fig. 1) from the initial team level towards more organizational issues following the research goals (i-iii) stated in Sect. 1.2. This bottom-up progressing emerges from the increasing knowledge in the research spiral (Fig. 11).

Fig. 12 depicts the overall research flow. The research phases link the research steps focusing on the consequent research questions (c.f., Fig. 11). Altogether they seek to answer the research problem by addressing the identified research needs and knowledge gaps summarized in Table 12.

With respect to the general Action Research methodology (Sect. 3.1), the research steps can be regarded as iterations. The resulting artefacts are described in Ch. 4 (Table 20).

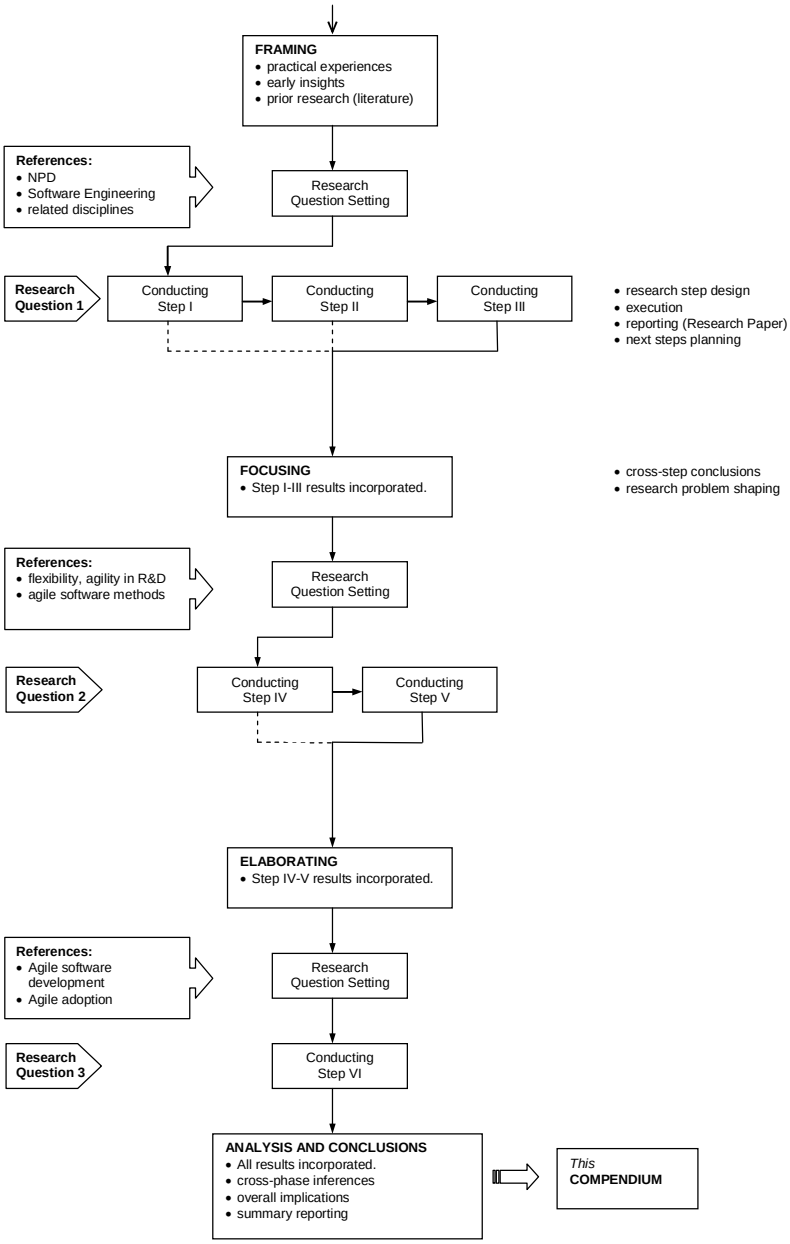

Fig. 12. Main research flow

Overall, this research work does not formally join to any particular larger research program. The case organization has a permanent SPI group, and the author has been a full-time member of it. The research activity has been an integral part of that ongoing daily work. However, some of the publications have been reported in the ITEA-AGILE program (ITEA-AGILE 2007a).

Taking into account the overall industrial case environment (Sect. 3.2), this kind of an investigation begins hardly ever from scratch in a large organization with a long history. For example in this case the author has gained prior knowledge and experiences already in another business unit in the case organization (Kettunen 2000). The initial ideas of this research stream really originate from the early 2000s (Kettunen 2001).

Furthermore, this investigation is partially open-ended. More work (iterations) could possibly have been done for each one of the phases and individual research steps. The path chosen here is context-dependent relative to the organizational environment, following a prominent research thread at the time of the investigation. That is what pragmatic action research is in essence (Heikkinen and Jyrkämä 1999). Consequently, this thesis concludes the work up to the point scoped in Sect. 1.3. The impacts of these choices are considered in Sect. 5.3.4.

The author has subsequently continued the agility-oriented research elsewhere (Kettunen 2009; Kettunen and Laanti 2008). We discuss this further work in Sect. 6.2.

3.4 Research Scrutiny and Evaluation Criteria

This kind of research-in-industry work is subject to the commonly underlined rigor-relevance dilemma (Glass 1994). That is, we should focus the research effort on relevant questions from the organizational business point of view, but also conduct scientifically valid and reliable research work from the academic point of view. There should be a win-win setup (Benbasat and Zmud 1999; Hinkin et al. 2007).

Hevner et al. (2004) suggest certain general guidelines for good design-scientific information systems research. Table 14 shows how we attempt to satisfy them in this research work.

Table 14. Guidelines for design-scientific information systems research

Guidelines (Hevner et al. 2004)	**Addressing in This Research**
Design-science research must produce a viable artifact.	The aim is to include empirical sections in every research paper for the practitioners. The resulting research papers I-IV present actionable artifacts (worksheets), which can readily be deployed in daily software production projects. The modeling propositions in the research papers V and VI remain more tentative, requiring some further refinements, but their feasibility is nevertheless demonstrated with practical industrial examples.
The purpose is to develop solutions to important and relevant business problems (research relevance).	The research questions are extracted from the daily operational work of the software production in the host organization. Hence, they stem from the ñburningò issues for the practitioners, and should consequently be relevant from the business point of view.

Guidelines (Hevner et al. 2004)	**Addressing in This Research**
The utility, quality, and efficacy of the artifacts must be rigorously demonstrated.	Descriptive evaluation methods (scenarios) are used tentatively.
The research must produce clear and verifiable contributions (artifacts, foundations and/or methodologies).	All the research papers I-VI make practice-oriented propositions.
The construction and evaluation methods of the artifacts applied must be rigorous (research rigor).	The practical propositions are grounded to the relevant research literature surveyed in the research papers.
The artifact solution is created by utilizing the available means while satisfying the laws in the problem environment.	The research work is closely connected to the organizational environment.
The results must be presented effectively for both technology-oriented as well as management-oriented audiences.	The research work is published in peer-reviewed journals and conference proceedings. Some company-internal presentations of the publications are organized for the more practitioner-oriented audience.

The Reflective Practitioner mode of research (reflection-in-action) entails certain inherent threats (Heiskanen and Newman 1997):

- The researcher should pay attention to the unique situation at hand.
- The researcher must avoid getting too ñnativeò.
- All informants should be treated equally.
- The practitioners may have stakes in the process, thus biasing the neutrality.
- The research reporting should take into account the breadth of observations, and data recording scrutiny.

This thesis research work is prone to the risk of the researcher getting too ñnativeò due to the author being a full-time employee of the company rather than an independent external academic researcher. However, this risk is lessened by the organizational position in the separate SPI organization not directly responsible for the daily project operations of the software production.

The quality of the empirical observations is a valid concern here because of the limited data size and mostly qualitative type. This is why a broad range of prior literature is surveyed extensively including some most recent (2008) publications in Ch. 2. That is in general recommended to compensate the limited number of cases in typical theory-building studies (Eisenhardt 1989).

Descriptive evaluation methods are in general not recommended as the sole method of evaluation of design-scientific artifacts, if other methods are feasible (Hevner et al. 2004). We must therefore pay special attention to the following (van Aken 2004):

- What can be learnt from this experience for other context (generalization)?
- On which observations and which logical reasoning are the recommendations based (justification)?

Benbasat and Zmud (1999) stress that relevant (IS) research should be interesting, applicable, current, and reported in an accessible way:

- Does the research address problems that are of concern to the professionals?

- Does the research produce the knowledge and offer prescriptions that can be utilized by practitioners?
- Does the research focus on current technologies and business issues?
- Are the research articles understandable (even enjoyable) to the professionals?

In general, there is no universal agreement on the criteria of successful action research (Huttunen, Kakkori and Heikkinen 1999). In particular, normal validity and reliability tests may not be most appropriate. The pragmatic evaluation viewpoint is to consider how much the research really helped the organization to improve its working.

We evaluate our research results and inferences against these criteria and recommendations in Ch. 5. The successfulness of the entire research work can then be judged.

4 Results

This section presents the main results and findings of the individual research papers I-VI (pp. vii). The items are grouped and combined here according to the research questions 1-3 set in Sect. 1.2. The presentation follows the logical flow of the research stream illustrated in Fig. 12. Each subsection (4.1, 4.2, 4.3) begins with a rationale. This bridging integrates the sections together into a chain of cross-section result steps and consequent needs for the subsequent research steps like overviewed in Sect. 1.4.

To begin with, the problem space of large-scale NPD embedded software projects is examined (Sect. 4.1). Typical problem and uncertainty factors are recognized by developing systematic means to characterize the contextual project problem space. Next, the problem space is examined by developing certain recognized key problem areas in more detail. Some practical constructs for this are proposed.

Following that line of thinking suggests considering agile solutions for typical software project team problems (Sect. 4.2). This experimentation allows realizing how and when agile software development can be realized efficiently at the software team level.

The team-level agility is then developed towards larger-scale organizational capabilities by integrating agile software project teams into NPD enterprise context (Sect. 4.3). This connection makes it possible to draw wider conclusions about the main research problem of the thesis.

Finally, Sect. 4.4 summarizes the research steps, tabularizing the developed artefacts and the resulting findings. It then integrates the results together, making it possible to answer the main research problem of the thesis.

4.1 Typical Problems of Large-Scale NPD Software Projects

Industrial non-trivial software projects tend to face many types of problems throughout the project life-cycle. Some problems stem from the project external environment and may be beyond direct control, while additional problems may be endogenous.

An important first step is therefore to be able to recognize such context-specific uncertainties and problem sources (Sect. 4.1.1). Once recognized and analyzed, it is possible to look for alternative solutions to the current and anticipated future project problem areas (Sect. 4.1.2, 4.1.3). By focusing on the particular context-specific uncertainty and problem areas, the project manager can conduct and steer the project rationally with appropriate means, even under considerable trouble conditions beyond her immediate control.

It is not in the scope of this thesis work to investigate each software development problem area in detail. Instead, the idea is to take a macro-level look, and thereby cover a wider range of essential NPD project uncertainty and problem classes. This scoping and focusing allows linking the possibilities offered by agile software development models to the wider NPD context.

4.1.1 Project Problems / Uncertainties

The purpose of Research Paper I (ñTroubleshooting Large-Scale New Product Development Embedded Software Projectsò) is to address the following specific questions:

- *How to recognize the typical problems of large-scale NPD embedded software projects?*
- *How to assess the feasibility and achievability (***ñ***health***ò***) of such projects?*

This paper proposes focused aids for identifying and evaluating the typical NPD problem factors. The investigation is first grounded to the existing literature of well-known software project failure factors observed in various industrial and academic environments over the years. Over the years there have been published many such typical software project problem factor classifications (Boehm 1991; Brooks 1995; Brown et al. 2000; Curtis et al. 1988; Fairley and Willshire 2003; May et al. 1998; McConnell 1996; McConnell 1998; Ropponen and Lyytinen 2000; Royce 1998; Schmidt et al. 2001; Smith 2001).

This study focuses on the specific problem areas of embedded software development projects. For that purpose it is important to understand the contextual factors and interdependencies between the actual software engineering and other related functions.

Many problems and uncertainties stem from the software project external reasons and dependencies. It is furthermore important to realize the dynamics involved ï i.e., how often and radically each source may change (turbulence), and how predictable the changes could be. That depends typically on many contextual factors in non-trivial and often interdependent ways (cause-effect relationships). Each embedded software team is then one element in a complex network thus facing both internal and external turbulence factors stemming in particular from the following:

- product/project portfolio management (concurrent product programs)
- product systems engineering (product evolution)

Based on the above general characterization of the overall problem space, this paper proposes a tool called ôProject Problem Profilerô for recognizing and evaluating software project problem factors in a systematic and comprehensive way. The Profiler is basically a database of typical problem factors and their likely impacts. It is based on the literature survey coupled with certain practical industrial experiences.

Table 15 illustrates the overall structure of the resulting Profiler matrix[7]. The matrix has two main parts (indicated by the thick vertical separator line).

[7] A tool implementation is available in http://old-www.cwi.nl/events/2006/profes/program.html

Table 15. Project Problem Profiler structure excerpt (Research Paper I)

Characteristic Project Problems, Risk Factors	**Categori-zation (Nominal)**	**Typical NPD Embedded SW**	**Typical IMPACT**	**Project STATUS**	**Project INDEX**
Program/Project Management					
Ineffective project management	Company	-	Critical	x_1	y_1
Inadequate planning and task identification	Project	-	Moderate	x_2	y_2
Inter-component or inter-group dependencies	Project	NPD special concern!	Major	x_3	y_3
Personnel Management					
cont.					

The left-hand side part of the matrix is basically a constant directory of typical software project problem factors, with a special emphasis on NPD embedded software projects. The rightmost part of the matrix is variable. It consists of the following two fields:

- *Project STATUS:*
  - This value is the current evaluation of the project status with respect to the problem items (No problem / Minor issue / Concern / Serious!). This field is intended to be filled in by the user.
- *Project INDEX:*
  - The projectôs profile is indicated as a numeric value for each problem item. It is calculated based on the fields *Typical IMPACT* and *Project STATUS*. This index can be used to plot graphical profiles of the current project situation (illustrated in Research Paper I).

For the *Typical NPD Embedded SW* field the matrix highlights the following six NPD special concerns (c.f., Table 15):

- new market with uncertain needs
- developing wrong software functions (functions that are not needed or are wrongly specified)
- unrealistic schedules, budgets (time and budget estimated incorrectly)
- inter-component or inter-group dependencies
- real-time performance shortfalls
- straining computer science capabilities (lacking technical solutions and computing power)

We have conducted some empirical case study experiments with the Profiler. The experiments were done by asking the case project (quality) managers to evaluate their project problem situation by using the Profiler matrix. The figures in Table 16 shows

a summary of the findings indicating the number of actual problem items detected (see Research Paper I for the case details).

Table 16. NPD project problem profiling case studies (Research Paper I)

Project Case	Field, Scope	Approach / Principles# of Problem Items flagged (out of 23)	# of Problem Items assessed as óSerious!ô	# of óNPD special concernô items (out of 6)
1	Terminal software platform subsystem, new features; Project ending.	8	2	2
2	Network element software, completely new product; Project completed.	17	5	6

In these cases concerning typical NPD-related problems 5 common problem items (out of 23) were identified. For confidentiality reasons the actual problem profile values cannot be shown here, though.

The general conclusions of these case studies were that the Profiler matrix captured critical problem areas of the case study NPD projects. None of the project cases identified any such significant problems that were not covered by the matrix. However, it is not possible to say, if the matrix approach highlighted such problem areas which had not yet been seen by the project manager.

4.1.2 Managing Information Uncertainty

The explorative study of the overall NPD software development problem space described in Sect. 4.1.1 suggests that many of such problem sources are essentially about information and knowledge management (óNPD special concernô in Table 15). The embedded software teams must acquire and combine information from various internal and external sources. Each source of information may introduce uncertainties in terms of completeness, maturity and volatility.

Following that line of thinking, Research Paper II (ñManaging Embedded Software Project Team Knowledgeò) addresses the following questions to investigate systematic methods for capturing and managing embedded software project team knowledge / information:

- *What are the key sources of information for the embedded software project team in such environments?*
- *What kind of information is needed from those different sources?*
- *When is each piece of information needed relative to the product life-cycle?*

To begin with, it is important to take a holistic view of the overall knowledge space of the product development project environment. We can start characterizing the knowledge base and information streams of an embedded software project team by

modeling the software development process workflow. One of the key points is to take the system and hardware engineering interdependencies into account.

The software development team must have a general understanding of the related systems and hardware engineering processes for instance to be able to deal with the hardware/software interface specifications. Ideally, there is a mutual codesign network with intense knowledge sharing between all the related disciplines. This is particularly important with concurrent engineering in turbulent environments with uncertain, incomplete and volatile design information.

This information flow and knowledge sharing modeling can be further developed to address the software project team outputs in particular for the other related project teams and subsequent projects (e.g., future product releases). Typically in large, established organizations much accumulated knowledge is encoded explicitly, but ï notably ï also implicitly in the various structures, processes, routines, and different tools. Often in practice some design knowledge remains outside the formal documentation (e.g., private notes) and even undocumented. Interpersonal networks are then important for sharing tacit knowledge. New personal experiences and skills (organizational knowledge assets) have also been accumulated in the project team.

Research Paper II proposes two constructs to address the information flow and knowledge sharing problems discussed above. These methods have been developed based on empirical experiences in industrial embedded software production for telecommunications network equipment.

The first Proposition helps identifying and consequently providing the necessary knowledge of the project team. It is essentially a systematic matrix of key software development knowledge areas following the modeling ideas.

Fig. 13 is an outline excerpt of this table. Each process area is accompanied by a set of questions concerning the related knowledge elements. The basic idea is to answer the questions from the viewpoint of each member of the project team.

Overall this method is most useful during the early phases of the project. Project staffing and teaming can be more systematically knowledge-based. It can also be utilized more generally for organizational process improvement purposes by identifying possible gaps in competencies, incomplete information assets, and other imperfect process areas.

NOTE:
The following is a list of software product development process areas based on ISO/IEC 15504 Reference Model.
The accompanying questions (in italics) are supposed to help identifying the practical information needs on those areas.
Each member of the project team should know those things from their point of view.

	P. Kettunen (PM)	*N.N. (Designer)*				
Customer-Supplier Process Cat (CUS):						
Acquisition						
Supply						
Who are our customers (external and internal)?						
Requirements Elicitation						
What do the customers really want from us?						
Who is responsible for the elicitation of the customer requirements?						
Operation						
Engineering Process Cat (ENG):						
System Requirements Analysis and Design						
Where do I get my system requirements?						
How do I know the software architecture (and system design)?						
Software Requirements Analysis						
Which items (documents) comprise my software requirements package?						
How are the requirements managed (changes)?						
Software Design						
What design methods and tools do I use?						
How do I change the component / subsystem external interfaces?						
Where can I find the hardware data sheets (if any)?						
Software Construction						
What compilers etc. tools do I use?						
What implementation rules do I have to obey (e.g., coding standards)?						
Software Integration						
What kind of integration and testing should I do?						
Software Testing						
Where do I get the target test hardware?						
System Integration and Testing						
How do I interface with the SW integration and system I&V?						
System and Software Maintenance						
What software platform dependencies do I have to follow?						
Support Process Cat (SUP):						
Documentation						
How do I manage (e.g., version) my documents?						

For each question, answer considering the actual needs of that person:
- n/a: No need to know.
- YES: Need to know. WHEN is the knowledge needed then?

Fig. 13. Embedded software project knowledge planning template (Research Paper II)

The second Proposition helps using the knowledge during the software project life-cycle by eliminating or at least reducing communication gaps and missing information flows. It is based on well-known ideas of responsibility charts for project planning in general by defining the key producers and consumers of the key information and mapping them together as a chart.

Fig. 14 illustrates such a project knowledge sharing chart. With this tabular method it is possible to visualize the knowledge item dependencies of each member of the project team.

	Software Project Internal		Software Project External		
Knowledge Items \ Actors	*P. Kettunen (PM)*	*N.N. (Designer)*	*(System Specifier)*	*(Hardware Manager)*	*(Quality Manager)*
Previous projects history	User	n/a	n/a	n/a	Provider
Software Specification A	Author	Reader	n/a	n/a	n/a
System Specification B	Reader	n/a	Responsible	Contributor	n/a
Hardware Data Sheet	n/a	Reader	Reviewer	Responsible	n/a
ASIC hardware behaviour	n/a	User	n/a	Provider	n/a
Standard Operating Procedure	Reader	Reader	n/a	n/a	Responsible
User's Guide	n/a	Author	n/a	Reviewer	Reviewer
Test process experience	Provider	User	n/a	n/a	n/a

Fig. 14. Embedded software project knowledge sharing chart (Research Paper II)

We have used certain real-life historical records to assess the proposed constructs and methods retrospectively in order to evaluate how the ideas could tackle similar situations in the future. Table 17 summarizes that inferencing (see Research Paper II for details). It suggests that the propositions could probably have helped to avoid many of those problems at least within this particular NPD context.

Table 17. History data based evaluation of the propositions (Research Paper II)

Case #	Problem Observations	Construct Reflections
1	• underspecified system requirements • lack of hardware behavior knowledge	• Proposition 2 • Proposition 1
2	• unclear, changing requirements • architectural complexities • testing • multisite organizations, communication	• Proposition 2 • (Proposition 2) • (see Research Paper II) • Proposition 2
3	• project scoping • product release planning	• Proposition 1 • NONE!

Overall, this investigation points to the following fundamental knowledge-related factors affecting NPD embedded software production and possibly causing major problems unless taken comprehensively into account even at a software team level:

- overall knowledge-intensiveness, including tacit knowledge and implicit assets
- team-level perspective (e.g. how well a project team masters the relevant knowledge, project team experience)
- people factors (e.g., skills, experience, interpersonal networking)
- organizational interdependencies (e.g., multiproject, product management)
- competitive environment changes (market and technology)

4.1.3 Tactics for Selecting the Software Process Model

The constructive exploration of the NPD software development problem space presented in Sect. 4.1.1 and 4.1.2 suggests that there is a need for systematic and holistic means to manage project uncertainties and different sources of potential problems. A major part of such a framework and a macro-level solution area of software project management is the choice of the overall project life-cycle model. This calls for systematic comparisons of different life-cycle model alternatives.

Research Paper III (ñHow to Steer an Embedded Software Project: Tactics for Selecting the Software Process Modelò) addresses this question with the following specific setting:

- *How can the project manager avoid typical project problems by selecting an appropriate software process model, based on the project situational factors?*

Many research investigations and software engineering guidebooks compare and contrast different software process models. Those different investigations use various different comparison viewpoints, such as suitability for different development types (size, criticality, and projectôs priorities), efficacy of managing different software

risks (uncertainty), and the level of prescription vs. situational adaptation. In addition, there are various multidimensional classification approaches (like ñhome groundò profiles).

In Research Paper III we propose a quasi-formal comparison based on distilling features of different project problem factors. We have composed a comparison matrix of well-known software process models with typical project problem issues like explored in Sect. 4.1.1 and 4.1.2 as the comparison points. Table 18 shows the structure.

Table 18. Software process model comparison / selection matrix structure (Research Paper III)

	Software Process Model
Project problem, risk, failure factor	*How does this process model prevent that particular problem from happening, or helps mitigating it (in the context of large embedded software projects)?*

The matrix includes the following software development process models: Waterfall, Incremental development, Spiral model, Rational Unified Process (RUP), FDD, ASD, and XP. The idea is to cover a wide range of models, both traditional and modern ones. In contrast, there is an extreme case of ad hoc ñhackingò.

As the comparison points the matrix covers some 50 problem items (rows) grouped according to the project life cycle: project initiation, execution, completion. Fig. 15 illustrates the layout of the matrix (top left-hand corner; see Research Paper III for the complete matrix).

	Software Process Models		
	Plan/Specification-driven Models		Evolutionary Models
Project Problems, Failure Factors	Waterfall (serial development)	Incremental development models	Spiral model (risk-driven iteration)
Project Initiation: Unclear project objectives (lack of a project mission)	Waterfall model does not tackle especially this problem. You should stay on the specification phase, until your project objectives are clarified.	Can start working on the known increments, and clarify the rest later. Note! May arise other problems later, if project is not well defined or if the definition changes much later. Rule of thumb is: 80% of the requirements should be known in the beginning. Make a project priority chart, and plan the increments accordingly. Sometimes the priorities must be changed during the project.	é
Overplanning / underplanning (e.g., "glass case" plan)	If you can do the planning reasonably well up-front, there is less overhead than with the iterative / incremental models. However, in the case of major uncertainties it is difficult to plan the project fully in advance. You must really proceed with the development to understand it better for realistic planning.	é	é
Lack of resources (people)	é	é	é

Fig. 15. Software process model comparison / selection matrix excerpt (Research Paper III)

The matrix includes in addition a key point section of each process modelôs home ground, drawbacks, and typical pitfalls. Table 19 is the outline of that part.

Table 19. Software process model selection matrix structure (cont) (Research Paper III)

	Software process model
Home ground	*Most applicable project environment(s) ï ñsweet spotò*
Consequences, Side-effects, Drawbacks:	
Scope	*Coverage of the model (project life-cycle activities)*
Nature	*Methodological characteristics*
Advantages	*Key benefits*
Constraints	*Limitations and disadvantages, prerequisites*
Cautions!	*Significant risks and pitfalls*
Notes	*Miscellaneous remarks*
EMBEDDED SYSTEMS	*Particular considerations for embedded software projects*

Research Paper III presents some industrial problem-based descriptive case studies to demonstrate the feasibility of the comparison matrix in practical settings. They illustrate how the matrix can be used to address typical project problem scenarios ï such as incomplete product requirements.

The outcome of this comparison is not any particular process model recommendation, but the idea is that the project manager can use the matrix to support his/her own selection of the particular process model. The matrix contains distilled advice about the selected process models in a concise form.

This comparison study highlights how the focal points and underlying assumptions of different software development process models vary. Notably no one process model is optimal in all problem situations, whilst for certain problem areas several feasible choices could be possible. On the other hand, a mismatch often leads to ineffective project steering and could even create additional problems.

4.2 Agile Solutions for Typical Software Project Team Problems

The cross-step conclusions in Sect. 4.1 suggest that agile software product development could potentially be a beneficial solution for many typical problems of embedded software projects, in particular:

- product requirements volatility and uncertainties (features)
- tight project schedules (time-to-market)
- extensive and rich knowledge / information sharing needs (communication)

Following the problem-driven software process model comparisons discussed in Sect. 4.1.3, it is thus logical to investigate further agile software development models and how they address different project problems (Sect. 4.2.1). It is in particular important to see how the focus areas of different agile software process models match with the practical NPD project problem areas (Sect. 4.1.1-4.1.2).

When agile software development models tackle the relevant project problem issues, it is in addition important to understand their overall effects in the NPD value-creation network. Cost factoring is therefore a key consideration for justifying their applicability and benefits (Sect. 4.2.2).

4.2.1 Responding to Problems with Agile Software Process Models

Research Paper IV continues the investigation of software process models as project problem solvers discussed in Sect. 4.1.3 (Research Paper III) by focusing more on agile software development models. The research question is thus as follows:

- *How do different agile software process models respond to different project problems faced in turbulent environments (if at all)?*

The following agile software development models are included: XP, Scrum, FDD, and ASD. Although RUP is generally not advocated as a pure agile software process model, it is possible to use it in a lightweight way, and we include that as well here. The purpose of this investigation is not to be an all-encompassing study of every agile-oriented development model but more like an overview exploration.

The composition of the comparison matrix is similar to the one in Fig. 18 (Research Paper III). The order of the process models (columns) in our matrix follows Boehmôs (2002) spectrum. Fig. 16 illustrates it.

Since all agile methodologies advocate at least some of the same common principles (Sect. 2.3), we have included one column into our matrix reflecting, how each project problem could in general be tackled (see column ñRelated AGILE PRINCIPLESò in Fig. 16). However, this reflection is basically for reference purposes only, since those agile principles have been formulated in a very general way. Notably current agile software process models do not explicitly address embedded software development.

		Software Process Models	
Project Problems, Failure Factors	**Related AGILE PRINCIPLES**	**RUP (Rational Unified Process)**	**FDD (Feature-Driven Development)**
Project Initiation:			
Unclear project objectives (lack of a project mission)	4. Business people and developers must work together daily throughout the project.	The Inception phase produces the project's Vision document defining the objectives (scope and constraints). The phase completes with a Lifecycle Objective (LCO) milestone, which criteria include a stakeholder agreement on the scope and the main requirements (features).	FDD does not cover the project initiation phase nor the customer requirements elicitation. However, a part of the Domain (Object) Model development is to understand, what the system is supposed to do. The model and the Features List are recommended to be agreed with the customers (stakeholders). With FDD, staged delivery is often recommended, thus the known/specified features can be made/shipped first.
Overplanning / underplanning (e.g., "glass case" plan)	10. Simplicity - the art of maximizing the amount of work not done - is essential.	There are two types of plans: a coarse-grained Phase Plan, and a more detailed Iteration Plan (for the current iteration). Excessive planning beyond the current horizon is not favored. The plans have evolving levels of detail. Generally, no work should be done outside the iteration plans.	é
Lack of resources (people)	5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.	é	é

Fig. 16. Agile software process model comparison matrix excerpt (Research Paper IV)

Research Paper IV presents some descriptive case studies to test the feasibility of the comparison matrix in practical problem situations. They illustrate how different agile software development models address certain typical product development scenarios requiring agile capabilities.

The comparison matrix suggests that many typical problem areas tend to be addressed by all the agile models: e.g., over/under-planning and unstable (volatile) requirements, continuous requirements changes. On the other hand, there are certain project problem areas that none of the selected agile software process models tackle especially well, such as lack of resources (people). Overall, the matrix approach points out how the problem space focus of agile methods is to some extent distinct from the problem space of traditional software process models examined in Research Paper III (Sect. 4.1.3).

One of the case examples in Research Paper IV (Case #3) indicates in addition that at least in larger projects following one particular method (XP) strictly as defined can be very challenging. None of the analyzed models are free from at least some limiting constraints.

As a conclusion, this systematic comparative study suggests that although agile software development models address many critical NPD project problems, the benefits are not self-evident and not necessarily realizable without some investments and trade-offs. This line of reasoning leads to the next research question to conduct systemic cost/benefit analysis of agile software development capabilities.

4.2.2 Cost-Based Justification of Agile Software Development

In Sect. 4.2.1, many possibilities for solving typical software project problems with agile software development models are recognized. However, it is not enough to provide effective solutions, but they must also be efficient to realize in practice. Cost factors (software engineering economics) are therefore important additional considerations of agile software development. In particular in many modern global business environments productivity and cost-effectiveness factors are often key considerations.

Agility cost factors have been investigated specifically in conjunction with manufacturing and supply chains (Sect. 2.2.3). This insight leads to the question of their suitability for software development in Research Paper V:

- *Are general agile cost models applicable to software development?*

Considering product development value streaming in terms of the following general equation (Eq. 1), agility strives to increase the quality and service factors by meeting the current customer requirements by being flexible to customer demands and market changes. Furthermore, the cost and lead time factors can potentially be decreased by improving the overall productivity of the product development workflows (lean). A key feature of the agile software methods is that they attempt to keep the cost of fast design iterations low, thus striving for increasing the numerator and decreasing the denominator factors in Eq. 1 simultaneously.

$$VALUE = \frac{Quality * Service}{Cost * LeadTime} \quad \textbf{(1)}$$

In general, agility requires investments. Agile response capabilities entail the following cost factors:

1. Building the reaction capabilities (e.g., software product platforms) in advance anticipating future changes
2. Utilizing those capabilities to rapidly implement the responses to the actually realized changes

Heikkilä and Holmström (2005) present a general cost model of an Efficiency Frontier. The point is that, up to the efficiency frontier, there is room for improving both the agility and cost-effectiveness of the current production system. Investments in agility could pay off in the future as the company is able to sustain profitability under changing conditions.

Research Paper V examines those generic modeling ideas with a descriptive case study example. The example illustrates that it is possible to quantify the costs and benefits of agile NPD software development, with exact questions to assess when preparing investments.

The case is about new product variants development, which is typically done with concurrent engineering of the new hardware and the embedded software functionality (Fig. 17, see Research Paper V for more details). A less-agile product development organization is not prepared for such emergent new variant needs, and consequently reacts to the customer needs as they come (Fig. 17(a)). This causes time-to-market delays, leading to poor responsiveness as well as lower productivity. In contrast, in a more agile organization, the need for different product variants is proactively anticipated from the beginning (Fig. 17(b)). A larger initial investment to proaction pays off in the long run due to the resulting better responsiveness and shorter time-to-market. However, this depends on the future business cases over time (ROI).

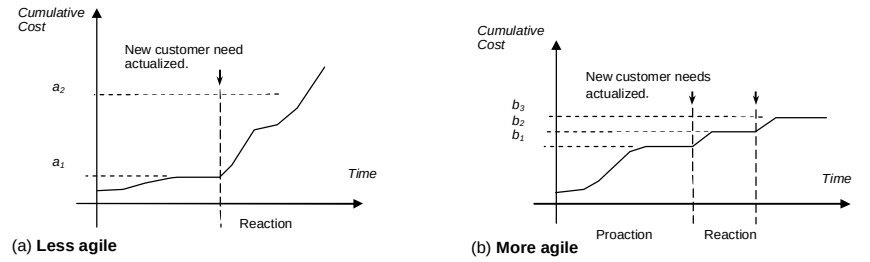

Fig. 17. Agility cost modeling example case (Research Paper V)

Research Paper V concludes that certain agile manufacturing / supply chain cost models can be successfully applied also in software development. This is because many agile manufacturing concepts have essentially been applied in some form in agile software development (e.g., lean thinking).

However, in large-scale software development the cost model of agility should incorporate the business model of the company (unit) as well (like the product variant offering in the case example above). A larger NPD organization should compose a value stream map of the whole product creation network, where the embedded software production function is one element of the NPD value stream. In large product development projects the net change cost (e.g., of swapping product features) often increases non-linearly during the development, as it is not just one team affected, but there are typically many interdependent teams. This is also influenced by product architectural choices (e.g., product line platforms). In general, each irreversible decision made during the product creation limits the future flexibility, and commits to certain costs.

4.3 Agile Software Project Teams within NPD Enterprise Context

The results and findings presented in Sect. 4.2.1 suggest that agile software development models address many key concerns of typical NPD embedded software projects. In addition the economic factors of agile software development are manageable like investigated in Sect. 4.2.2.

However, the findings also reveal the fact that complete realization of many of those potentially beneficial solutions at the software team level depends on the surrounding organizational environment and the integration between the software team(s) and the rest of the NPD value network. This insight leads to the next research need to investigate how agile software teams can work effectively in the larger organizational system context in order to contribute positively to the total performance of the NPD organization.

Research Paper VI addresses those issues of integrating agile software development teams into larger scale organization environment. Specifically, it sets the following questions:

- *How does software project agility relate to NPD enterprise agility?*
- *What are the implications for SPI?*

The research approach is to build a software project team agility extension framework by connecting agile software product development projects explicitly into the NPD enterprise context. The starting point is therefore an individual team and its agility related key interfaces (business customers / markets, organizational resourcing and governance).

The next logical step is to put the model of an individual software team into the larger organizational context considering the NPD and enterprise layers together with the competitive (business/technology) environment of the organization. Notably the project team may not have direct connections to the business customer(s), and there are typically other organizational stakeholders. Furthermore, the software project team may have additional interdependencies both internally (e.g., with the hardware development projects) and externally (e.g., subcontractors). The problem is, then, how the teams can nevertheless be agile.

Fig. 18 illustrates those considerations with the Scrum method. The basic process model has to be connected with the rest of the NPD organization (linking the inflows

and outflows indicated by the dangling arrows in Fig. 18). This may introduce additional contextual interdependencies for example to process the interim increment releases.

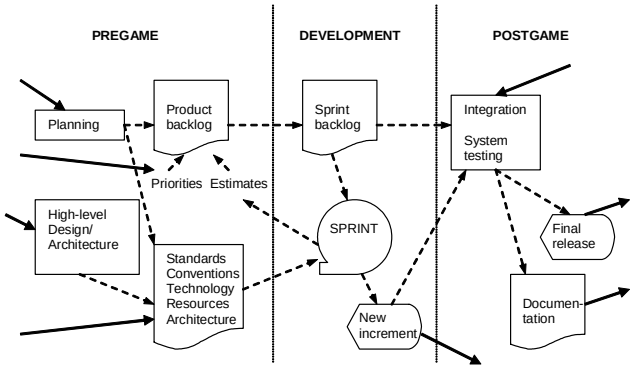

Fig. 18. Organizational extensions with Scrum (Research Paper VI)

The key is thus to understand and agree on the interfaces and constraining factors between the project team and the rest of the NPD organization. Some of the major considerations are then for example the following:

- What are the customer interface and the distance between the customer(s) and the project team?
- Who makes the business decisions about the product features, and the development schedule and resources (governance)?
- What is the required level of project progress visibility (e.g., milestones)?

Depending on the sources and level of uncertainties, different levels of flexibilities are required for achieving agility. Agile software teams must therefore have appropriate latitudes in the organizational context in multiple steering dimensions like depicted in Fig. 19. The more business (market) and technology uncertainty there is, the more flexible the product concept and development approaches should be to accommodate the volatility and changes. However, the project complexity and organizational constraints limit the choices.

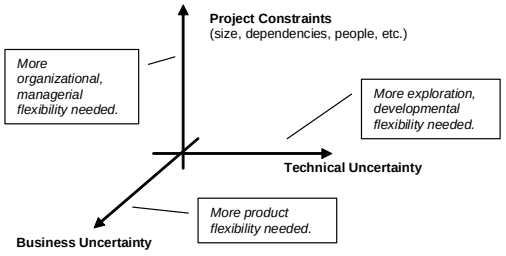

Fig. 19. Project agility dimensions (Research Paper VI)

Combining the enterprise model with the software project agility steering model (Fig. 19) produces a synthesis view of the overall NPD enterprise agility. The agility of the NPD organization is the aggregate of the product development projects. The NPD function as a whole exploits its agility by combining the different projects. For example, there may be multiple concurrent release projects each developing new features for the same product based on both known (reactive) and anticipated (proactive) market needs.

Finally, at the enterprise level the entire NPD organization is just one part of the agility capabilities considering the whole space of business competence (c.f., Fig. 2). For instance different marketing strategies may choose to leverage the NPD capabilities in different ways over time for achieving firm external agility (e.g., selecting the current and new product features to release).

Research Paper VI evaluates this modeling approach with a descriptive case in industrial NPD. The key success point is to realize the positioning and the external connections of the software development projects in the enterprise NPD value stream. An agile organization realizes the sources and nature of project uncertainty (Fig. 19), and takes proactive measures accordingly. The software projectôs external connections are then managed systematically (Fig. 18).

The main conclusion of the findings in Research Paper VI is that NPD company-specific strategic choices influence profoundly the agility of the related software projects (for example the way they interface with the customers). This may then put totally different requirements for the different product development projects within the company. The agility of the individual software projects should not be developed in isolation. The project teams can then realize their positioning in the NPD organization and in its value network, and what business effects they are expected to bring at the enterprise (business unit) level. At the project team level the specific uncertainties and organizational constraints guide then the appropriate choices for situational, flexible software project management tactics.

4.4 Synthesis

The research steps presented in Sect. 4.1-4.3 form a consecutive research thread summarized in Sect. 4.4.1. Each step addressed certain specific aspects of the main research problem, and opened up next research cycles. Altogether, the combined and integrated results answer the overall research problem of the thesis (Sect. 4.4.2).

4.4.1 Research Cycles

Following the research questions (Sect. 1.2), Table 20 is a concise summary of the research cycles included in this thesis. The development artefacts described in detail in Sect. 4.1-4.3 and originally in the corresponding research papers (pp. vii) are tabularized here.

Table 20. Summary of the research cycles

Research Cycle (Paper)	Topics	Artefacts

Research Cycle (Paper)	**Topics**	**Artefacts**
What are the typical problems of large-scale NPD embedded software projects? (Sect. 4.1)		
I ñTroubleshooting large-scale New Product Development embedded software projectsò	Project problems / uncertainties: *How to recognize the typical problems of large-scale NPD embedded software projects?*	Problem/uncertainty profiling matrix
II ñManaging embedded software project team knowledgeò	Managing information uncertainty: *What are the key sources of information for the embedded software project team in such environments? What kind of information is needed from those different sources? When is each piece of information needed relative to the product life-cycle?*	Systematic methods for capturing and managing embedded software project team knowledge / information
III ñHow to steer an embedded software project: tactics for selecting the software process modelò	Managing and avoiding different project problems with software process models: *How can the project manager avoid typical project problems by selecting an appropriate software process model, based on the project situational factors?*	Process model comparison/selection matrix
What problems and goals does agile software development address? (Sect. 4.2)		
IV ñHow to steer an embedded software project: tactics for selecting agile software process modelsò	Responding to different project problems with agile software process models: *How do different agile software process models respond to different project problems faced in turbulent environments (if at all)?*	Agile software process model comparison/selection matrix
V ñCost Modeling Agile Software Developmentò	Cost-based evaluation and justification of agile software development: *Are general agile cost models applicable to software development?*	Interdisciplinary application of agility cost theory considering NPD reaction and proaction capabilities (value creation effects vs. investments)
How can typical large-scale NPD problems be tackled with agile software development methods? (Sect. 4.3)		
VI ñExtending Software Project Agility with New Product Development Enterprise Agilityò	Extending software project team agility within larger scale organization context: *How does software project agility relate to NPD enterprise agility? What are the implications for SPI?*	Software agility extension framework taking into account contextual drivers, interdependencies, and enabling factors

4.4.2 Compound Results

Altogether this stepwise research and the results and findings presented in Sect. 4.1-4.3 and summarized in Sect. 4.4.1 provide progressively deeper understanding of software production agility in large-scale NPD environments like depicted in Fig. 11. As a conclusion, the results and findings of this research work can be aggregated as follows:

- In large-scale NPD organizations, the typical problems of embedded software projects stem in general on multiple different dimensions from both the external

competitive environment (business and technology uncertainties) as well as organization internal reasons (e.g., multiproject interdependencies) (Sect. 4.1).

- Agile software development models address many of those key problem areas, and the associated software engineering economic relationships are in principle manageable (Sect. 4.2).
- However, in large NPD organizations it is in general not enough to pursuit agility at the individual software team level alone, but the surrounding organizational context has to be taken into account in order to integrate such project teams effectively for gaining comprehensive, enterprise-level performance benefits of agile software product development (Sect. 4.3). In large organizations there are often complex interdependencies between different projects, teams, and functions.

Following the main research flow in Fig. 12, based on this new knowledge it is possible to answer the main research problem of the thesis about how agile software development can be utilized in large-scale NPD context:

- To begin with, it is essential to understand the nature of the key project problem areas (e.g., uncertainty, productivity) and the intended purpose and coverage of the suggested solutions offered by agile software development models. Often there is a range of apparently different goals and problem issues, but the respective success factors and root causes may actually lead to the same focal sources. If they map clearly to the specific areas covered and emphasized by the agile software development models, there are potentially immediate advantages. This perspective is studied in detail in Research Paper IV (Sect. 4.2.1).
- However, it is not obvious how the (agile) software product development capabilities relate to the NPD performance, and finally to the business performance. Research Paper V (Sect. 4.2.2) shows that there are certain possibilities to capture such effects with known modeling techniques. The key intrinsic characteristics of software production (Sect. 2.1.3) should nevertheless be taken into account.
- It is important to see how the agile development in software production joins the overall NPD process improvement and even more general organizational development. The agility of the individual software projects should not be developed in isolation (Research Paper VI, Sect. 4.3). The relevant goals, problems and potential solutions should be analyzed in a holistic way taken into account the environment of the organization (team, unit). The project teams can then realize their positioning in the NPD organization and in its value network, and what business effects they are expected to bring at the enterprise (business unit) level. Often the NPD front-end, product creation, and also possibly also technology development activities are iterated even concurrently in large, complex projects facing different levels of uncertainties in different product areas and life-cycle phases. At the project team level the specific uncertainties and organizational constraints guide then the appropriate choices for situational, flexible software project management tactics.

5 Discussion

Having presented the research results in Ch. 4 answering the research questions, this section elaborates the addressing of the research problem (Sect. 5.1). Moreover, the research can now be compared against the related literature (Sect. 5.2), and also evaluated considering the overall research quality (Sect. 5.3). Based on the results, the insights make it furthermore possible to suggest implications in Sect. 5.4.

5.1 Answering the Research Problem and Goals

In order to elaborate the answer to the research problem (Sect. 4.4), the question can be viewed from two different perspectives:

1. Problem-based: The problem-based perspective emerges from first sensing and understanding the software development (project) contextual problem and uncertainty space (Research Question 1, Sect. 4.1). Such a problem profile can then be analyzed with respect to the specific solutions addressed by agile software development models (Research Question 2, Sect. 4.2). Not all software product development problem areas are covered equally by different agile software process models (Sect. 4.2.1). If a distinct correspondence is found, the selected agile solutions could be worth applying ï subject to cost assessment (Sect. 4.2.2). This is often path-dependent. Finally, when the agile solutions are found to be suitable and feasible at the software team level, it should be ensured that they fit into the organizational context (Research Question 3, Sect. 4.3). This may require additional measures and extensions, which may in turn call for reassessing the cost factors, and even reconsidering the feasibility of the solutions. The consequent NPD performance effects and ultimately the firm business value gains should be considered.
2. Effects-based: The effects-based perspective begins in contrast with the potential benefits provided by agile software development teams, and how those benefits can be exploited for the overall NPD performance (Research Question 3, Sect. 4.3). The starting point is thus an outside-in view of agile software development teams. For instance, if the product development time-to-market goals are the key performance drivers in the NPD organization, the software development teams can be assessed from that particular point of view. It is then possible to realize how agile software development solutions (Research Question 2, Sect. 4.2) could support that goal. Consequently, it is also important to make it sure that the teams are in practice capable of achieving such results by providing the necessary enabling conditions and removing any possible impediments in the organization context (Research Question 1, Sect. 4.1).

Notably the research workflow of this thesis has followed the former view (i.e., top-down). However, it is possible to see it also the other way around (the latter view) by unwinding the research cycle depicted in Fig. 11. Moreover, the research papers I-VI can be positioned inside-out starting from the software team scope, and expanding

towards the larger-scale NPD organization scope. Another viewpoint is to take on outside-in lens by approaching the individual software teams from the NPD organization scope. These two views are also reflected in Fig. 1.

Concluding, the answers and contributions altogether strive for satisfying the general goals (i-iii) stated in Sect. 1.2 for this research effort by chartering and conceptualizing the project problem space, and examining situational project management with the software process model perspective ï focusing on agile software development models specifically (i). Furthermore, agile software development teams have been put into the larger organizational context and analyzed considering overall NPD objectives and value-chain relationships (ii). In addition, this impact analysis leads to certain recommendations presented in Sect. 5.4 (iii).

5.2 Comparing and Contrasting Related Work

Over the years, the prior investigations have recognized general NPD success factors (Table 25) on the other hand, and typical risks of software projects on the other hand (Table 2). Many special concerns of embedded software development have also been known for a long time. Different NPD and software process models have been compared in general terms, including flexibility (Table 26, Table 3).

In contrast, this thesis research combines and elaborates that knowledge by approaching software development projects from a wider, systems-thinking perspective in the NPD context. Based on this large-scale organizational view, the findings discover that the most critical product development project management problems stem from not so much within the single project itself, but from the technical and organizational connections and interdependencies over time (Sect. 4.1.1).

Due to those contextual and situational factors, the embedded software project teams face many external, often uncontrollable sources of uncertainties (e.g., volatile new product requirements, unstable target hardware interfaces), and various interdisciplinary heterogeneous knowledge spaces (Sect. 4.1.2). Under such circumstances a major part of successful project steering tactics is to use appropriate software process models to manage the driving criticalities, and moreover to avoid further problems caused by inappropriate process models (Sect. 4.1.3).

However, less is known about newer, flexible models of software and product development in such dynamic, uncertain environments. This is where agile software development models address distinct needs (Sect. 2.3.2.2, Table 8). What has been missing or has been considerably less-systematically covered is their integration with larger-scale NPD processes and their interplay in the organizational context.

This thesis recognizes and addresses that cross-functional, interdisciplinary research gap. There are currently incompleteness and even confusion with the concepts, definitions, and terminology of agility as related to software development (Sect. 2.3.1). Consequently, there are no standardized, comparable measurements (Table 5, Table 9). This research work clarifies the conceptual base and different definitions used not only in software development but within the larger NPD context including other interrelated business competence areas and disciplines (Table 4).

This examination makes it possible to position the current agile software development models in the larger systems context in NPD organizations. Their focusing and scope can then be understood in particular regarding their often implicit but strong working assumptions and coverages (Sect. 4.2.1). Furthermore, this more rigorous conceptualization allows more formal reasoning about their benefits and associated costs (Sect. 4.2.2).

Currently many industrial organizations are considering agile software development in their NPD functions (Table 28). In larger organizations this leads to the issues of integrating and scaling up the software team-level developments with the rest of the product development and operations (Table 29). However, there are no widely-agreed frameworks for such agile adoption, and many organizations have devised various proprietary approaches (Table 30).

This thesis acknowledges that much of such organizational development is inherently context-specific due to the different strategic goals, competitive environments, and historical paths of the organizations. Nevertheless, we also argue that despite the firm-specific characteristics there are many common higher-level patterns that can be abstracted and therefore adopted in different organizations. For instance the external interconnections of embedded software project teams are often basically similar in apparently different NPD environments (Sect. 4.3).

Related to the overall conceptualization of agility and agile software development discussed above, such abstractions are currently not widely recognized in the NPD software production. This thesis work advances that line of progressing.

5.3 Evaluation

Like stated in the research design (Sect. 3.4), it is important to scrutinize the research work both in terms of the industrial relevance (Sect. 5.3.1) as well as scientific validity and rigor (Sect. 5.3.2). This evaluation makes it also possible to reason about the generalizability of the results (Sect. 5.3.3), taking into account the specific constraints and limitations imposed by the organizational environment of this thesis work (Sect. 5.3.4).

5.3.1 Relevance

The research relevance is of particular importance in this kind of an industrial research effort (Benbasat and Zmud 1999). It is important to convince that the research question is significant for the organization(s) and/or theory, and to demonstrate that the existing research either does not address the research question at all, or does so inadequately (Eisenhardt and Graebner 2007). Moreover, the (management) research should be innovative, challenging established theories even counterintuitively (Bartunek et al. 2006).

The real world organizational drivers and interests for the research problem of this thesis are described in Sect. 3.2. That justifies the topical relevance of the research in the case organization context. Furthermore, agile software development is of considerable topical relevance in the industry in more general; see for example (Schwaber 2005; Gottesman and Takas 2007). There are also many recognized open

avenues for further academic research in this field (Dingsøyr et al. 2008; Dybå and Dingsøyr 2008).

The theoretical background and prior research are reviewed in Ch. 2. The consequent research needs are summarized in Sect. 2.4. In particular, the cross-functional synthesis of general-purpose NPD research and agile software product development models is inadequately covered, suggesting open research questions.

Stemming from the practical industrial context, this research work has consciously strived for addressing ñreal worldò problems suggesting usable pragmatic solutions propositions and inferences. Consequently, there are also many relevant practical implications as discussed in Sect. 5.4.2.

The interdisciplinary, explorative approach of the research is not necessarily totally groundbreaking, but it is nevertheless a fresh ï if not counterintuitive ï choice, not often taken in the existing research literature of the field. This research path has certainly opened up many possibilities for further relevant work like outlined in Sect. 6.2.

5.3.2 Rigor and Validity

The overall research work plan is first appraised (5.3.2.1). The research realization is then evaluated against scientific soundness (5.3.2.2) and methodological fitness criteria (5.3.2.3). Finally, the presentation of the research papers and this summary altogether is judged (5.3.2.4).

5.3.2.1 *Design Rationale*

This research work is constructive with some supporting empirical evidence like described in Ch. 4. Eisenhardt and Graebner set the following general criteria for sound empirical research (Eisenhardt and Graebner 2007):

- Begins with strong grounding in related literature.
- Identifies a research gap.
- Proposes research questions that address the gap.

The related literature is reviewed extensively in Ch. 2. Based on that, the research and knowledge gaps are highlighted in Sect. 2.4. This leads to the research problem and the research questions stated in Sect. 1.2.

Eisenhardt and Graebner (2007) require furthermore a justification of why the research problem is better addressed by theory-building rather than theory-testing. Like discussed in Sect. 3.2, the starting point and contextual drivers for this research work originated at the software team level. The primary aims have been in pragmatic proposition-building for real-life SPI problems. Moreover, since the field of software product development in general and agile software development models in particular is mostly in a nascent state, possibly applicable theories are still merely emerging. Therefore, this research thread is closer to theory-building than theory-testing.

5.3.2.2 *Soundness*

Dybå and Dingsøyr (2008) have composed a rigorous overall qualification criterion while surveying published empirical research of agile software development. Table 21 assesses this thesis research work against that criterion.

Table 21. Evaluating the soundness of the research work

Criteria (Dybå and Dingsøyr 2008)	**Addressing in This Research**
Is the paper based on research (or is it merely a "lessons learned" report based on expert opinion)?	Although the research work is grounded to industrial practice, the guiding ambition has been to follow sound academic research conduct from the beginning – subject to practical time and resource constraints, though. The research steps are published in peer-reviewed journals and conference proceedings rather than in less-formal practitioner magazines.
Is there a clear statement of the aims of the research?	The aims of the research are defined in Sect. 1.2. There are both academic research objectives as well as practical industrial goals.
Is there an adequate description of the context in which the research was carried out?	The context of the research work is described in Sect. 3.2, presenting the industrial case environment.
Was the research design appropriate to address the aims of the research?	According to Eisenhardt and Graebner inductive theory-building research using cases answers particularly well research questions that addresses "how" and "why" in unexplored research areas (Eisenhardt and Graebner 2007).
Was the recruitment strategy appropriate to the aims of the research?	The selection of the participants and cases has been limited by the organizational constraints of the research environment. The descriptive cases – although limited in number and scope – are expected to be representative in this particular industrial context, however.
Was there a control group with which to compare treatments?	There was none. The usefulness of the propositions remains thus mostly for further studies. In general, the relative empirical importance of the proposed constructs is not a primary concern in theory-building research.
Was the data collected in a way that addressed the research issue?	The data collection procedures have been described in the individual Research Papers (Ch. 4). All in all, the empirical data used is limited to certain descriptive cases (see Sect. 5.3.4).
Was the data analysis sufficiently rigorous?	The data analysis procedures have been described in the individual Research Papers (Ch. 4). Due to the descriptive nature of the supporting data, the rigor is not considered to be a serious threat for this thesis research (see Sect. 5.3.4).
Has the relationship between researcher and participants been considered to an adequate degree?	The relationship is defined in Sect. 3.2.2. This consideration is taken into account like described in Sect. 3.4.
Is there a clear statement of findings?	The findings are presented in detail by the separate Research Papers. Ch. 4 summarizes the key items.
Is the study of value for research or practice?	Like stated in Sect. 3.2, this research effort is by nature practice-oriented due to its industrial context. However, there is a deliberate ambition to conduct the work in a scientifically rigorous manner. Arguably, not all the publications are in top-tier research forums, though. The ultimate value of the study remains to be seen (Barley 2006).

Table 21 indicates that the weakest areas of this research work are in the empirical data collection and analysis. However, the role of empirical data is not critical in this research process due to the exploratory nature of the research in a nascent field with not much prior established theory.

5.3.2.3 *Methodological Fit*

Edmondson and McManus (2007) define ómethodological fitô as a coherent alignment between the research question, related prior research, research design, and the contribution aims of the new research. They suggest an overall framework for selecting appropriate research methods based on the maturity level of the research area (at the time of the study) like presented in Table 22. Inconsistencies (ñmisfitsò) are likely to lead to difficulties and less compelling results.

Table 22. Methodologically fit new research designs (after (Edmondson and McManus 2007))

State of Prior Research	**TYPICAL New Research Design**	**COMMON Flaws, Threats with Evidence**
MATURE	• focused research questions • relying on existing constructs and theories • quantitative data, statistical data analysis • SUPPORTED THEORY	• ñreinventingò obvious and well-known findings • qualitative evidence used unevenly
INTERMEDIATE	• proposed relationships between old and new constructs • some new constructs • both qualitative and quantitative data (hybrid) • PROVISIONAL THEORY	• lack of supporting qualitative data for convincing new constructs • quantitative data missed for even preliminary hypothesis testing
NASCENT	• inquiry about an interesting phenomenon • new constructs • qualitative data, initially open-ended • SUGGESTIVE THEORY	• quantitative data used with uncertain relationships on the unknown phenomenon • ñfishing expeditionsò with qualitative data

Table 23 evaluates this thesis research work against the general recommendations given in Table 22. Notably the scale for the state of the prior research is a continuum, merely suggesting the broadbrush archetype.

Table 23. Evaluating the methodological fitness of this research design

Research Step / Question (Paper) (Ch. 4)	**State of Prior Research** (Sect. 2.4)	**Methodological FIT**
1. What are the typical problems of large-scale NPD embedded software projects?		
I ñTroubleshooting large-scale New Product Development embedded software projectsò	*MATURE*	• mostly recombining and rejoining prior constructs and findings • some quantitative evidence for demonstrating the proposed artefacts

Research Step / Question (Paper) (Ch. 4)	**State of Prior Research** (Sect. 2.4)	**Methodological FIT**
II ñManaging embedded software project team knowledgeò	*INTERMEDIATE*	• primarily relying on existing theories • some new constructs proposed, linked to well-known ones • some retrospective (qualitative) evidence for demonstrating the efficacy of the proposed new constructs
III ñHow to steer an embedded software project: tactics for selecting the software process modelò	*INTERMEDIATE*	• combining existing models with a new lens • no new theory-building, some new constructs • no quantitative evidence • some descriptive evidence for provisional demonstration of the proposed constructs
2. How can agile software development benefit?		
IV ñHow to steer an embedded software project: tactics for selecting agile software process modelsò	*INTERMEDIATE* to *NASCENT*	• Like III.
V ñCost Modeling Agile Software Developmentò	*NASCENT*	• applying existing theories from a different discipline • some provisional quantitative data used for illustrative purposes (scenarios)
3. How to realize agile software development in large-scale NPD context?		
VI ñExtending Software Project Agility with New Product Development Enterprise Agilityò	*NASCENT*	• expanding existing constructs • new contextual viewpoints • some provisional qualitative data for demonstrating the enhanced constructs and viewpoints

Contrasting to the general guidelines in Table 22, the evaluation in Table 23 suggests that there are no major methodological flaws in the research design of this thesis work. Notably case studies are advocated in general for explanatory and illustrative purposes when the boundaries between the phenomenon under study and the real-life context are not clear (Yin 1994). Edmondson and McManus (2007) propose an iterative research design model (including developing the research question) in particular for nascent research areas. This has essentially been the situation with the present research problem.

One area requiring further strengthening in this research is the uneven empirical evidence, in particular lack of quantitative data for statistical analysis. However, like underlined in Table 22, premature utilization of quantitative data for drawing firm conclusions can be a threat in nascent research areas. The emphasis of this research work has so far been in provisional understanding of the phenomenon.

5.3.2.4 *Presentation*

The presentation of empirical research should be understandable and even enjoyable to the professionals (Benbasat and Zmud 1999). Eisenhardt and Graebner (2007) instruct to rationalize the format of the theory-building presentation with the following guidelines:

- developing the emergent theory in sections, each supported by empirical evidence
- demonstrating each part of the theory by evidence (from at least some of the cases)
- complementing the selective story descriptions with rich presentation of evidence
- summarizing the evidence for each theoretical construct
- visual theory summary

In this compendium, following the research path illustrated in Fig. 11, the findings are presented in consequent research steps in Ch. 4. Each step (Research Paper) presents certain supporting evidence accordingly, demonstrating the propositions and constructs with (descriptive) cases. However, because of the limited availability of current empirical data, the case evidence is not extensively tabulated. This limits also the theoretical argumentation of the constructs (see Sect. 5.3.4). Nevertheless, Fig. 12 presents some visual summarizing of the overall concept building and positioning.

5.3.3 Generalization

Stemming from the industrial context of the research environment described in Sect. 3.2, the generalization of the results and inferences of this research work are constrained by the following two major factors:

- The research work has been conducted in one particular product line scope within this one particular company. Because of the large size and diversity of the particular case organization, the results of this research cannot be argued to be fully representative even in this one particular organization.
- limited empirical data even within this particular local scope

However, recent studies by for example Börjesson (2006) and Suikki (2007) suggest that the research problem setting and the findings are not extraordinary at least within this field of industry. This suggests that the key results and inferences of this thesis research are more generally applicable at least to some extent.

Considering the limited empirical data availability in this research, there is an issue of how the theory can generalize if the cases are not representative (Eisenhardt and Graebner 2007). In general, theory building from multiple cases produces more generalizable and testable theory than single-case research (Yin 1994). However, like defined in Sect. 1.2, the overarching goal of this research is to propose new framework constructs for systematic understanding, and not to attempt to build comprehensive, testable theories due to the nascent nature of the research field. The lack of extensive, multiple-case study data is therefore not considered to be a serious threat in the present state of this research work. Indeed, generalizability is not necessarily limited by single-case designs (Flyvbjerg 2006; Lee and Baskerville 2003).

5.3.4 Limitations

In this thesis research work, the biggest obstacle has been to collect sufficient empirical evidence for evaluating the propositions. Having more independent empirical evidence (both quantitative and qualitative) would strengthen the individual cases. This remains largely tentative, suggesting future studies (Sect. 6.2). However, in general, inductive theory-building research using case studies does not address well

the questions of ñhow oftenò and ñhow manyò, nor the questions about the relative empirical importance of the proposed constructs (Eisenhardt and Graebner 2007).

Overall, it is important to understand that all the topics investigated in these particular research steps are major research areas as such. For example, knowledge management (Sect. 4.1.2) has become ever more significant due to the growing complexity of many software-intensive products as well as more complex organizational setups (e.g., distributed development, outsourcing). The individual constructs and propositions presented in Sect. 4.1-4.3 are thus not necessarily comprehensive solutions to all those specific areas, and more alternatives could possibly have been explored ï like discussed in the respective research papers. However, for the purposes of this thesis research, their key value is in the synthesis (Sect. 4.4) with respect to the main research problem.

Virtually in any industrial organization environment in practice, the need of collecting rigorous validation data over a longer period of time tends to require some balancing due to limited resourcing. The key is then to make this clear for all the parties (researchers and practitioners), and to assess the research findings accordingly (Lee and Baskerville 2003).

Another practical constraint of this research realization is that the action research has mostly been limited to planning the improvements and giving recommendations, but not much implementing the practical changes. That is, the author has been acting more like a professional normative researcher than an intercepting change agent. This is by and large due to the non-managerial role of the author during the research period, i.e., the author has been working merely in a participatory observer role making recommendations but not really controlling the organizational course of the actions (Järvinen 2004). Consequently, the proposed constructs and inferences are mostly conceptual and the validation remains partially speculative. However, by and large the research goals (Sect. 1.2) are attained, although the longer-term impacts and effects in the case organization remain to be confirmed.

The validation is also limited to a single case organization context (Sect. 3.2). Like scoped in Sect. 1.3, this thesis work does not specifically intent to provide substantial evidence of how beneficial agile software development can be or how exactly it can be implemented (deployed) in large-scale organizations. Moreover, the purpose of the research was not to benchmark different companies. However, public research literature suggests similar results in other comparable organizations (e.g., (Börjesson 2006)).

Conclusively, the research scoping leaves many potentially significant areas open for further investigations. For instance collaboration and interactions between people in teams (team dynamics) are frequently underlined as some of the key factors in agile software development. Also many large-scale organization social elements (e.g., self-organization, culture changes) remain for further study (Sect. 6.2). The research thread approaches to that direction, but the present research questions do not cover those areas.

Finally, an overarching difficulty with this kind of a multi-issue research journey is in maintaining a consistent progressive research thread over the years. However, such narrow, situational and context-based emergence of research questions is intrinsic with action research in general (Susman and Evered 1978). Furthermore, changes and

even organizational jolts in the industrial case environment have had to be taken into consideration during this long-lasting research period.

5.4 Inferences and Implications

This thesis research has progressed towards getting deeper understanding of the problems and prospective solutions of agile software development in industrial real-life settings. The results and findings as aggregated in Sect. 4.4.2 imply multidisciplinary concerns both for the academic inquiry (Sect. 5.4.1) and in particular for the practitioners in industrial NPD organizations (Sect. 5.4.2). While the propositions and constructs built in this thesis are mostly provisional, a set of practice-oriented recommendations can be suggested.

5.4.1 Theoretical Inferences

For rational inferring, it is important to first understand the positioning of agile software development in the overall NPD problem space (Sect. 5.4.1.1). This makes it possible to see the potential impacts (Sect. 5.4.1.2), and target the organizational changes (Sect. 5.4.1.3).

5.4.1.1 *Positioning Agile Software Development*

Conceptually, this thesis maintains the following overall standpoint with agile software development in NPD context:

- Software projects have certain stated and possibly also implicit objectives (organizational goal space).
- In practice, there are typically many problems and obstacles hindering or even preventing the projects from reaching those goals (situational problem space).
- On the other hand, there is a range of potential solution alternatives for solving or mitigating the problems (solution space). Different solutions (e.g., process models) support achieving different project goals. There are some known solutions / remedies for most of practical project problems.
- Agile software development is a subset of this space. Agile software development models advance a subset of the goal-settings in certain project environments, providing solutions for a subset of the problems while avoiding some problems.

Moreover, it is fundamental to understand how software project management in NPD ï and indeed the whole mode of operation ï should dynamically be adjusted according to the level of uncertainties and changes of the organizational environment (Collyer 2008; Crawford and Pollack 2004; De Meyer et al. 2002). The appropriate choice of flexible project management tactics depends much on the level of uncertainties. Agile software development models address much of the most uncertain side (Nerur and Balijepally 2007).

Agile software development can thus be seen as one contributing element of the overall capabilities of the flexible NPD functions of the company (c.f., Fig. 6). In this respect, it can be regarded as a strategic option.

5.4.1.2 *Effects of Agile Software Development*

The externally visible strategic agility of the company is a total effect of all the agile capabilities. Consequently, the NPD organization can express agility externally even without using agile software development models internally.

Taking a holistic view of the entire NPD value-creation network and the company performance entails integrating the company strategy goals, the software production system, and the resulting performance effects. The following theory-building questions should then be considered:

- What are the business goals and the operational performance targets of the NPD company, what are their interrelationships?
- How does the software production system support them?
- What are the key contributing elements of the software production system, and what are their effects? In particular, what is the impact of agile software development on each of them?

It is not obvious how much for instance the overall NPD productivity can be improved with agile software development models. For example Mäkelä (2008) studies the question of how the management of the software process and that of general product development operations interact to effect software product development performance, and subsequently firm performance, and how the firm strategy and dynamic capabilities ï including agile software development ï influence this. They suggest that the performance relations are indirect. Nevertheless, even coarse-grained analytical reasoning, such as for example with the Value-Price-Cost (VPC) framework, can provide valuable insights about the main directions of the improvements (Hoopes et al. 2003). Vehtari (2006) investigates in a similar vein, how manufacturing capabilities and performance change in different phases of business life cycles, and how manufacturing could provide competitive advantages.

Concluding, in theory, agile software development should thus be seen as one moderator in the total value-creation network of the NPD company. The effects should consequently be modelled from the global systems perspective, rather than just in the micro-context of isolated project teams.

5.4.1.3 *NPD Organizational Development*

Many companies are nowadays contemplating agile software development and its potential benefits in particular for time-to-market and productivity gains (Schwaber 2005). A strategic fit should be gained between the overall business goals, software product development goals, and the operations of the software production system (Heikkilä and Ketokivi 2005). The software product development competence can then be viewed from strategic, organizational, and technological points of view. In high-velocity competitive environments the key strategic organizational elements are frequent environment scanning and consequent adaptation (Brown and Eisenhardt 1998). The same traits should be mapped down to the agile software development projects (Nerur and Balijepally 2007).

Agility can thus be considered as a strategic capability, i.e. what Johnson et al. (2006) define as ñadequacy and suitability of the resources and competences of an organization for it to survive and prosperò. Agile software development can then be

considered from two different strategic points of view within NPD organization context (c.f., Fig. 6):

- Competition-driven view: required agility of the software production function (outside-in)
- Resource view: software production agility as an enabling capability (inside-out)

The former depends much on the business network (both competitors and partners) of the company (Oosterhout et al. 2006). The latter view includes the dynamic capability of reconfiguring the software development resources in new ways, and using time as a source of competitive advantages (Santalainen 2005). Notably NPD is a first-order dynamic capability, but it is possible to deal with changes without explicit dynamic capabilities (ôad hoc problem-solvingõ) (Winter 2003). Investments to dynamic capabilities may not even pay off in some cases unless the environment drives frequent changes.

The agile software development capabilities spread over multiple dimensions in the organizational context. Hence, agile capability development should be considered in different mutually interdependent dimensions, in particular:

- people / organization (management)
- technology / product (software)
- process (engineering)

Current agile software development models alone cannot necessarily provide effective external agility in otherwise non-agile organization context. This implies that software engineering may not be the primary area of concern in all cases in particular in larger organization business competence. Other typical key development areas are strategy development, marketing and organizational design (OD), and competence development (HRD). There are then theoretical linkages to other reference disciplines, such as information systems (IT), production and operations management, organization theory (OT), and also general business management (e.g., financing and accounting) (Nambisan 2003; Nerur and Balijepally 2007).

Considering the total space of agile realization in (large) organizations, interestingly, agile software development models address to some extent both traditional structural and process areas as well as more intangible people (HR) development and cultural elements. However, they are mostly silent about (large-scale) organizational development aspects (e.g., organizational learning, extended enterprise). For instance Dehoff and Loehr (2007) emphasize the role of the less-visible organizational and cultural bases.

5.4.2 Managerial Implications

By combining the results and findings of the research work (Sect. 4.4.2) and the theoretical implications discussed in Sect. 5.4.1, we infer the following practice-oriented recommendations. They relate to strategic (Sect. 5.4.2.1), tactical (Sect. 5.4.2.2), and operational (Sect. 5.4.2.3) concerns of the software production.

5.4.2.1 *Strategic Goal-Orientation*

In all, the NPD company should first make clear strategic choices about how much agility (e.g., in terms of responsiveness) and on which areas it strives for (for example considering the product variety). Such decisions concern not just the R&D strategy but also the business and marketing strategies (Tyrväinen et al. 2004).

When agility is recognized to be a critical success factor (CSF), the strategic impacts should be analyzed throughout the organization. Contextual assessments with appropriate measurements, taking into account the competitive environment situational factors (agility drivers), could be devised to guide this (c.f., Table 5 and Table 9). This may then put totally different requirements for the different product development projects within the company.

5.4.2.2 *Agile Capability Development*

To begin with, it is important to understand the key strategic goals (e.g., productivity), and the intents and coverage of the suggested solutions (software construction, project management, organizational factors) offered by agile software development models (Sect. 2.3.2.2). If and when agile software development is deemed to be the appropriate solution considering the context-specific problems and goals of the organization (unit), strategic agile capability development can be approached from multiple different points of view. While bottom-up emergence from the individual software teams can create the foundations, larger organizations typically need in addition more top-down cross-functional coordination, enabling supports, and executive sponsorship to succeed profoundly.

Agile software development capabilities could become strategic assets for sustainable competitive advantages. The capability development strategy should support it (Lecklin 2002). Indeed, for example Näsi and Neilimo (2006) underline that systems thinking is the key to strategic business competence development.

5.4.2.3 *Complementary Organizational Improvements*

Expanding the team-level perspective of this research work (Sect. 4.3), other (organizational) improvement approaches such as CMMi or Lean Thinking could be combined with agile software development models to address those larger-scale organizational areas, which are beyond the scope of current the software models (c.f., Table 30). However, care should be taken not to pursuit divergent or even conflicting goals (Ngwenyama and Nielsen 2003; Paulk 1996). The key is to understand the goals on the one hand, and the underlying assumptions and means on the other hand. Different approaches may use different means towards the same ends. The cost factors may differ significantly (ROI).

On the whole, the performance dimensions may span the entire business competence space of the organization such as along the outcome lines of the Baldrige framework (Baldrige 2008). Ultimately all the improvements ï agile or otherwise ï should be goal-oriented and aligned with the strategic vision of the company.

6 Conclusions

This final section brings together the entire research work of the thesis. It concludes this compendium while bringing up some potential avenues for future work.

The contributions of the thesis (Sect. 6.1) build on the results and findings of the individual research papers presented in Ch. 4. Following the research design in Fig. 11, several ideas of future research have emerged during this thesis research process. These are outlined in Sect. 6.2.

6.1 Key Contributions

Like illustrated in Fig. 11, the overarching aim of this research work is to develop deeper understanding of agile software product development advantages and limitations in large-scale NPD organization environments. The main contribution of this thesis is such a holistic, capability-oriented realization of what agile software development entails in large-scale NPD context. Coupled with the organizational business and NPD process competence, agile software product development models can ï within supporting organizational and environmental conditions ï provide competitive benefits in particular in turbulent business environments.

This understanding builds on the stepwise research and development summarized in Table 24. The research steps contribute to this overall knowledge by introducing new constructs and propositions emerging from the selected embedded software team improvement viewpoints.

Table 24. Research contributions by research steps

Research Step / Question (Paper)	**CONTRIBUTIONS**
1. What are the typical problems of large-scale NPD embedded software projects?	
I ñTroubleshooting large-scale New Product Development embedded software projectsò	Project problems / uncertainties: • problem/uncertainty profiling matrix
II ñManaging embedded software project team knowledgeò	Managing information uncertainty: • systematic methods for capturing and managing embedded software project team knowledge / information
III ñHow to steer an embedded software project: tactics for selecting the software process modelò	Managing and avoiding different project problems with software process models: • process model comparison/selection matrix
2. What problems and goals does agile software development address?	
IV ñHow to steer an embedded software project: tactics for selecting agile software process modelsò	Responding to different project problems with agile software process models: • agile software process model comparison/selection matrix
V ñCost Modeling Agile Software Developmentò	Cost-based evaluation and justification of agile software development: • interdisciplinary application of agility cost theory

Research Step / Question (Paper)	**CONTRIBUTIONS**
3. How can typical large-scale NPD problems be tackled with agile software development methods?	
VI ñExtending Software Project Agility with New Product Development Enterprise Agilityò	Extending software project team agility within larger scale organization context: • software agility extension framework

In large-scale NPD environments, there are typically a host of different uncertainties and consequently potential project problems (Research Question 1). The key to comprehensive agility-based improvement is then to realize the competitive drivers and critical improvement needs of the entire NPD value network, and how agile software development teams contribute in it (Research Question 2). Beneficial agile software development requires a holistic view considering not only the embedded software engineering function, but the entire product development value-creation chain within the overall business model of the organization. In particular in large organizations, software production cannot necessarily be improved effectively within the software development function and software engineering discipline alone (Research Question 3).

The essential contribution of this thesis work is the overall synthesis of those research cycles (Table 24), putting agile software development into wider organizational context (Sect. 4.4.2). This inferring includes disentangling the current agile software method assumptions, focus areas, and scoping. They have considerable limitations and incomplete coverage for large-scale NPD environments. However, their benefits in terms of flexibility and software product development value creation are achievable by understanding and consequently complementing them with other (software) process improvements (SPI) and organizational development (OD) approaches.

Scaling up agile software development is currently an active research topic in general, and this thesis contributes to that from the NPD perspective, although the starting point is at the software team level. The novelty here is to be able to step out and see above the organizational ñboxò, and thus realize how the external outcomes of the integrated software development teams contribute to the NPD offerings and responses of the organization. It is important to realize how agility affects at the different organizational levels (Fig. 1). This is fundamentally about how agility in software production can support the NPD and ultimately the business strategy of the firm.

A methodological contribution of this research is the holistic conceptual framing and partitioning of the problem space of software development agility in the NPD context. In such a complex interdisciplinary problem domain, one of the key research issues is to capture the essential research questions, and to understand their interrelationships in the context. This discovery is a contribution, too, indicating that comprehensive understanding and consequent agility-based improvements in large-scale NPD requires complementary, spanning viewpoints. With this respect, the research thread and consequent inferring (Sect. 5.1) can be considered as emergent provisional theory-building. In alignment, the background review part of the thesis

has provided provisional clarification of the currently confusing terminology and incomplete conceptualization of agility in NPD and software production (Sect. 2.2).

Concluding, one can contemplate whether the overall contribution of this thesis makes a striking difference in the research field or not (Barley 2006). This remains mostly for further studies to determine, but for instance Mäkelä (2008) emphasizes that the combination of software production and new product development in general is a fresh viewpoint, opening up new research avenues.

6.2 Future Research

Like noted in Sect. 3.3.2, more work could have been done with most of the research steps. There are several specific further work areas recognized during the research studies. The Research Papers I-VI discuss them in detail. In addition, notably, the excluded areas of this research effort set in Sect. 1.3 (e.g., cultural and human resource aspects) are potential areas for fruitful further research.

All in all, this exploratory thesis research work has brought up perhaps more further questions than definite answers. Consequently, it opens up new research avenues, and sets pointers for future research. The following outlines certain key ideas for further work:

1. Considering NPD improvement in general and agile adoption in particular, the following elements are important to understand: drivers, goals and needs, and means. Also the enablers and potential impediments are concerns for implementing the changes (deployment). While there are no universal answers to all these major questions, we hypothesize that there are common characteristics facing many NPD software development organizations in particular within the same competitive environments. Therefore, a profiling model including a database of for example typical impediments like surveyed in Table 29 could be composed in order to guide (although not prescribe) the improvement work (Sect. 2.3.3).
2. Like discussed in Sect. 2.2.2, agility is not unique to software development. In fact, many of the origins and principles can be traced to manufacturing developments in the early 1990s (Preiss 2005). This leads to an idea of investigating more systematically the similarities and differences of software production and agile manufacturing in order to understand the possibilities for cross-functional learning (Gunasekaran 1998; Gore 2008). Can strategic software production capabilities be treated like manufacturing capabilities (Vehtari 2006)? Moreover, does the inherent flexibility of software enable even more strategic agility since there are fewer constraints than with physical manufacturing systems? We have tentatively addressed this research avenue elsewhere (Kettunen 2009).
3. Supply chain agility is currently an active research area (Christopher 2000; Hofman and Cecere 2005; Zsifkovits and Engelhardt-Nowitzki 2007; Naylor et al. 1999). There are new flexibility requirements faced in many markets (Collin 2003; Collin and Lorenzin 2006; Kaipia 2007). The trend is towards more holistic approaching of entire demand-supply networks and total value-chain management. The question is then how all this recent development relates ï if at all ï to software production. Furthermore, how do the software product development functions link to the total demand-supply network of a NPD company, even crossing internal

(site) and external company boundaries? New modeling approaches are needed here (Agarwal et al. 2007). Indeed, there are some propositions to explicate software supply chains (Oberhauser and Schmidt 2007; Underseth 2007).

Finally, there is a general problem with the lack of common definitions of agile software development concepts ï and NPD agility in more general. This leads to misconceptions and even confusion in practice, and makes scientific work troublesome for instance due to lack of common comparable metrics. It is important to realize this for example when reviewing the related literature. There is even some skepticism about the whole concept of distinguishingly ógileô software development (post-agilism); see for example (Ågerfalk and Fitzgerald 2006; Levine 2005; Northover et al. 2007).

This thesis advocates the scientific approach followed in Sect. 2.2, i.e., that one must first define the concept rigorously in relation to the existing literature, and only after then judge the uniqueness and the need for specific new terms (Näsi and Neilimo 2006). This in turn makes it possible to define valid comparable performance metrics, for instance.

Having defined the concepts and terminology in a rigorous way, it is possible to link agile software development and agility in general into the larger context. In particular, business competence is the key macroconcept here. Strategic systems thinking is then an inherent trait ï including the underlying management principles and assumptions.

y

References

Abrahamsson, P., 2002. The Role of Commitment in Software Process Improvement. Dissertation, University of Oulu, Finland.

Abrahamsson, P., 2006. Pealing the hype into pieces: What do we really know about agile in research & practice? In: Bi-annual OO-days, Tampere University of Technology, Finland http://www.cs.tut.fi/tapahtumat/olio2006/abrahamsson.pdf (accessed 20.11.2007)

Abrahamsson, P., 2007. Speeding up embedded software development. ITEA Innovation Report

Abrahamsson, P., Salo, O., Ronkainen, J., Warsta, J., 2002. Agile software development methods ï Review and analysis. http://www.vtt.fi/inf/pdf/publications/2002/P478.pdf

Abrahamsson, P., Warsta, J., Siponen, M.T., Ronkainen, J., 2003. New directions on agile methods: a comparative analysis. In: Proc. 25th International Conference on Software Engineering (ICSE), pp. 244-254.

Adeleye, E.O., Yusuf, Y.Y., 2006. Towards agile manufacturing: models of competition and performance outcomes. International Journal of Agile Systems and Management 1(1), 93-110.

Agarwal, A., Shankar, R., Tiwari, M.K., 2007. Modeling agility of supply chain. Industrial Marketing Management 36, 443-457.

Agile DOI, 2005. http://pmdoi.org/ (accessed 28.3.2008)

Agile Manifesto, 2001. http://www.agilemanifesto.org/ (accessed 13.11.2007)

Aken, van J.E., 2004. Management Research Based on the Paradigm of the Design Sciences: The Quest for Field-Tested and Grounded Technological Rules. Journal of Management Studies 41(2), 219-246.

Akgün, A.E., Lynn, G.S., Byrne, J.C., 2004. Taking the guesswork out of new product development: how successful high-tech companies get the way. Journal of Business Strategy 25(4), 41-46.

Ambler, S.W., 1998. Process Patterns: Building Large-Scale Systems Using Object Technology. Cambridge University Press

Ambler, S.W., 2007. Agile Software Development: Definition. http://www.agilemodeling.com/essays/agileSoftwareDevelopment.htm (accessed 21.11.2007)

Ambler, S.W., Kroll, P., 2007. Lean development governance. White paper, IBM/Rational

Ancona, D.G., Caldwell, D., 1990. Improving the Performance of New Product Teams. Research • Technology Management 33(2), 25-29.

Anderson, D.J., 2004. Agile Management for Software Engineering. Prentice Hall, Upper Saddle River, NJ, USA.

Anderson, D.M., 1997. Agile Product Development for Mass Customization. Irwin Professional Publishing, Chicago, IL, USA.

Ante, S.E., 2007. Back From the Dead. Business Week, June 25, 48-56.

Aoyama, M., 1998a. Agile Software Process and Its Experience. In: Proceedings of the 20th International Conference on Software Engineering, pp. 3-12.

Aoyama, M., 1998b. Web-Based Agile Software Development. IEEE Software 15(6), 56-65.

Aramand, M., 2006. Developing Dynamic Design Capabilities in Software Product Development Companies. Dissertation, Tampere University of Technology, Finland.

Aramand, M., 2008. Software products and services are high tech? New product development strategy for software products and services. Technovation 28(3),154-160.

ARTEMIS, 2006. Strategic Research Agenda: Design Methods & Tools.

Atkinson, R., 1999. Project management: cost, time and quality, two best guesses and a phenomenon, its time to accept other success criteria. International Journal of Project Management 17(6), 337-442.

Atkinson, R., Crawford, L., Ward, S., 2006. Fundamental uncertainties in projects and the scope of project management. International Journal of Project Management 24, 687-698.

Atkinson, S.R., Moffat, J., 2005. The Agile Organization. http://www.dodccrp.org/files/Atkinson_Agile.pdf (accessed 15.11.2007)

Atwater, J.B., Pittman, P., 2008. We Want To Be TOYOTA: Understanding why the Toyota production system is successful. APICS magazine (March/April), 32-35.

Augier, M., Teece, D.J., 2007. Perspectives on Research and Development: Organizing and Managing Innovation in a (Global) Knowledge-Based Economy. In: Knowledge Creation and Management: New Challenges for Managers (Eds. Ichijo, K., Nonaka, I.). Oxford University Press.

Avison, D., Lau, F., Myers, M., Nielsen, P.A., 1999. Action Research. Communications of the ACM 42(1), 94-97.

Aydin, M.N., Harmsen, F., Slooten van K., Stegwee, R.A., 2005. On the Adaptation of an Agile Information Systems Development Method. Journal of Database Management 16(4), 24-40.

Baldrige, 2008. Criteria for Performance Excellence. http://www.baldrige.nist.gov/Business_Criteria.htm (accessed 13.2.2008)

Barczak, G., 1995. New product strategy, structure, process, and performance in the telecommunications industry. Journal of Product Innovation Management, 12(3), 224-234.

Barley, S.R., 2006. When I Write My Masterpiece: Thoughts on What Makes a Paper Interesting. Academy of Management Journal 49(1), 16-20.

Bartunek, J.M., Rynes, S.L., Ireland, R.D., 2006. What Makes Management Research Interesting, and Why does It Matter? Academy of Management Journal 49(1), 9-15.

Baskerville, R., Balasubramaniam, R., Levine, L., Pries-Heje, J., 2006. High-Speed Software Development Practices: What Works, What Doesn't. IEEE ITProfessional 8(4), 29-36.

Benbasat, I., Zmud, R.W., 1999. Empirical Research in Information Systems: The Practice of Relevance. MIS Quarterly 23(1), 3-16.

Benefield, G., 2008. Rolling Out Agile at a Large Enterprise. In: Proc. Hawaii International Conference on Software Systems (HICSS), pp. 461

Benko, C., McFarlan, F.W., 2003. Connecting the Dots: Aligning Projects with Objectives in Unpredictable Times. Harvard Business School Press.

Boehm, B., 1991. Software Risk Management: Principles and Practices. IEEE Software 8(1), 32-41.

Boehm, B., 2002. Get Ready for Agile Methods, with Care. IEEE Computer 35(1), 64-69.

Boehm, B., 2005. Software Process Disruptors, Opportunity Areas, and Strategies. USC-CSE-2005-500

Boehm, B., Turner, R., 2004. Balancing Agility and Discipline ï A Guide for the Perplexed. Addison-Wesley, Boston, MA, USA.

Boehm, B., Turner, R., 2005. Management Challenges to Implementing Agile Processes in Traditional Development Organizations. IEEE Software 14(5), 30-39.

Bosch, T., 2007. Medical Software Development ï Going Agile. Medical Device & Diagnostic Industry (October), pp. 42-47.

Breu, K., Hemingway, C.J., Strathern, M., Bridger, D., 2001. Workforce agility: the new employee strategy for the knowledge economy. Journal of Information Technology 17, 21-31.

Brooks, F.P. Jr., 1995. The Mythical Man-Month: Essays on Software Engineering (20th Anniversary Edition). Addison-Wesley

Brown, S.L., Eisenhardt, K.M., 1995. Product Development: Past Research, Present Findings, and Future Directions. Academy of Management Review 20(2), 343-378.

Brown, S.L., Eisenhardt, K.M., 1998. Competing on the Edge: Strategy as Structured Chaos. Harvard Business School Press

Brown, W.J., McCormick III, H.W., Thomas, S.W., 2000. AntiPatterns in Project Management. New York, NY, USA: John Wiley & Sons.

Büchel, B., 2007. Knowledge Creation and Transfer: From Teams to Whole Organization. In: Knowledge Creation and Management: New Challenges for Managers (Eds. Ichijo, K., Nonaka, I.). Oxford University Press.

Börjesson, A., 2006. Making software process improvement happen. Dissertation, IT University of Gothenburg, Sweden.

Capgemini, 2007. Four Out of Ten CIOs at Multi-Billion Euro Enterprises Believe Their IT Function Cannot Deliver Enough Agility in an Environment of Accelerating Business Change. Press release, http://www.capgemini.com/m/en/n/pdf_CIO_Survey_2007.pdf (accessed 16.11.2007)

Caswell, N.S., Nigam, A., 2005. Agility = Change + Coordination. Proc. 7th IEEE International Conference on E-Commerce Technology Workshops (CECW), pp. 131-139.

Ceschi, M., Sillitti, A., Succi, G., De Panfilis, S., 2005. Project Management in Plan-Based and Agile Companies. IEEE Software 22(3), 21-27.

Chakravarthy, B., 1997. A New Strategy Framework for Coping with Turbulence. Sloan Management Review 38(2) (Winter), 69-82.

Charette, R.N., 1996. Large-scale project management is risk management. IEEE Software 13(4), 110-117.

Chillarege, R., 2002. The Marriage of Business Dynamics and Software Engineering. IEEE Software 19(6), 43-49.

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

Christensen, C.M., Kaufman, S.P., Shih, W.C., 2008. Innovation Killers: How Financial Tools Destroy Your Capacity to Do New Things. Harvard Business Review 86(1), 98-105.

Christopher, M., 2000. The Agile Supply Chain ï Competing in Volatile Markets. Industrial Marketing Management 29, 37-44.

Cockburn, A., 2002. Agile Software Development. Addison-Wesley / Pearson

Cockburn, A., 2007. What Engineering Has in Common With Manufacturing and Why It Matters. CrossTalk 20(4), 4-7.

Cognizant, 2006. Best Practices In Global Agile Development For Product Innovation. White paper, http://www.cognizant.com

Cohan, P.S., Unger, B., 2006. Sources of Advantage. Business Strategy Review (Spring), 9-14.

Cohen, D., Lindvall, M., Costa, P., 2004. An Introduction to Agile Methods. Advances in Computers 62, 1-66.

Collin, J., 2003. Selecting the Right Supply Chain for a Customer in Project Business ï An Action Research Study in the Mobile Communications Infrastructure Industry. Dissertation, Helsinki University of Technology, Finland.

Collin, J., Lorenzin, D., 2006. Plan for supply chain agility at Nokia: Lessons from mobile infrastructure industry. Intôl Journal of Physical Distribution and Logistics Management 36(6), 418-430.

Collyer, S., 2008. Project management approaches for dynamic environments. International Journal of Project Management (in press)

Conboy, K., Fitzgerald, B., 2004. Toward a conceptual framework for agile methods: a study of agility in different disciplines. In: Proc. ACM workshop on Interdisciplinary software engineering research (WISER), pp. 37-44.

Cooper, R.G., 2006. Formula for Success in New Product Development. MM, 18-http://www.stage-gate.com/downloads/working_papers/wp_23.pdf

Cooper, R.G., Kleinschmidt, E.J., 1996. Winning businesses in product development: The critical success factors. Research ÅTechnology Management 39(4), 18-29.

Cooper, R.G., Kleinschmidt, E.J., 2007. ñWinning businesses in product development: The critical success factorsò Revisited. Research ÅTechnology Management 50(3), 60-61.
Coplien, J.O., 2007. Seven Subtle Stumbling Blocks of Agile. Expo-C,
Coplien, J.O., Harrison, N.B., 2004. Organizational Patterns of Agile Software Development. Prentice Hall
Crawford, D.P., Mason, P., Melenovsky, M., Turner, V., Waxman, J., 2003. Enabling Business Agility: Hewlett-Packardôs Adaptive Enterprise Strategy. White paper, IDC
Crawford, L., Pollack, J., 2004. Hard and soft projects: a framework for analysis. International Journal of Project Management 22, 645ï 653.
Curtis, B., Krasner, H., Iscoe, N., 1988. A Field Study of the Software Design Process for Large Systems. Communications of the ACM 31(11), 1268-1287.
Dagnino, A., 2001. Coordination of Hardware Manufacturing and Software Development Lifecycles for Integrated Systems Development. In: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, pp. 1850-1855.
Dagnino, A., 2002. An Evolutionary Lifecycle Model with Agile Practices for Software Development at ABB. In: Proc. 8th Intôl Conf. Engineering of Complex Computer Systems (ICECCS), pp. 215-223.
Dagnino, A., Smiley, K., Srikanth, H., Antón, A., Williams, L., 2004. Experiences in Applying Agile Software Development Practices in New Product Development. International Association of Science and Technology for Development (IASTED) Software Engineering and Applications (SEA)
Davis, C.R., 2002. Calculated Risk: A Framework for Evaluating Product Development. MIT Sloan Management Review 43(4) (Summer), 71-77.
Day, G.S., 1994. The Capabilities of Market-Driven Organizations. Journal of Marketing 58, 37-52.
De Meyer, A., Loch, C.H., Pich, M.T., 2002. Managing Project Uncertainty: From Variation to Chaos. MIT Sloan Management Review 43(2) (Winter), 60-67.
Dehoff, K., Loehr, J., 2007. Innovation Agility. strategy+business 47 (Summer)
Delaney, J.D., Mitchell, G.G., Delaney, S., 2003. Software Engineering Meets Problem-Based Learning. The Engineers Journal 57(6).
Deloitte, 2007. Innovation Benchmark Study
Dingsøyr, T., Dybå, T., Abrahamsson, P., 2008. A Preliminary Roadmap For Research On Agile Software Development Research. In: Proc. Agile Conference, pp. 83-96.
Dove, R., 1997. The Meaning of Life & The Meaning of Agility. http://www.parshift.com/Essays/essay001.htm
Dove, R., 2004. Enterprise agility ï What is it and what fuels it? http://www.parshift.com/Essays/essay065.htm
Dove, R., Hartman, S., Benson, S., 1996. An Agile Enterprise Reference Model. http://www.parshift.com/docs/aermodA0.htm
Doz, Y.L., Kosonen, M., 2008. Fast Strategy: How Strategic Agility Will Help You Stay Ahead of the Game. Pearson Education Ltd.
DSDM, 2003. DSDM and Process Improvement. White paper, DSDM Consortium
DSDM, 2004. Organization Suitability Risk List. White paper, DSDM Consortium
Dutton, J.L., McCabe, R.S., 2006. Agile/Lean Development and CMMI. In: SEPG
Dybå, T., Dingsøyr, T., 2008. Empirical Studies of Agile Software Development: A Systematic Review. Information and Software Technology 50(9-10), 833-859.
Edmondson, A.C., McManus, S.E., 2007. Methodological Fit in Management Field Research. Academy of Management Review 32(4), 1155-1179.
Eisenhardt, K.M., 1989. Building Theories from Case Study Research. Academy of Management Review 14(4), 532-550.
Eisenhardt, K.M., Brown, S.L., 1998. Time Pacing: Competing in Markets That Wonôt Stand Still. Harvard Business Review 76(2), 59-69.

Eisenhardt, K.M., Graebner, M.E., 2007. Theory Building from Cases: Opportunities and Challenges. Academy of Management Journal 50(1), 25-32.

Eisenhardt, K.M., Jeffrey, M.A., 2000. Dynamic Capabilities: What are they? Strategic Management Journal 21, 1105-1121.

Eisenhardt, K.M., Tabrizi, B.N., 1995. Accelerating Adaptive Processes: Product Innovation in the Global Computer Industry. Administrative Science Quarterly 30, 84-110.

Ernst, H., 2002. Success factors of new product development: a review of the empirical literature. Int. Journal of Management Reviews 4(1), 1-40.

Fairley, R.E., Willshire, M.J., 2003. Why the Vasa Sank: 10 Problems and Some Antidotes for Software Projects. IEEE Software 20(2), 18-25.

Ferrarini, E.M., 2008. Getting Technology Agility Right. Inside BTM (February), 16-18.

Fitzgerald, B., Hartnett, G., Conboy, K., 2006. Customising agile methods to software practices at Intel Shannon. European Journal of Information Systems 15, 200-213.

Fitzgerald, B., Russo, N., OôKane, T., 2003. Software Development Method Tailoring at Motorola. Communications of the ACM 46(4), 64-70.

Flyvbjerg, B., 2006. Five Misunderstandings About Case-Study Research. Qualitative Inquiry 12(2), 219-245.

Fugetta, A., 2000. Software Process: A Roadmap. In: Proceedings of the Conference on The Future of Software Engineering (ICSE), 27-34.

Gilb, T., 1988. Principles of Software Engineering Management. Addison-Wesley

Gilb, T., 2006. Fundamental Principles of Evolutionary Project Management. http://www.gilb.com/community/tiki-list_file_gallery.php?galleryId=15

Glass, R.L., 1994. The Software-Research Crisis. IEEE Software 11(6), 42-47.

Glass, R.L., 2004. Matching Methodology to Problem Domain. Communications of the ACM 47(5), 19-21.

Glass, R.L., Ramesh, V., Vessey, I., 2004. An Analysis of Research in Computing Disciplines. Communications of the ACM 47 (6), 89-94.

Glazer, H., Dalton, J., Anderson, D., Konrad, M., Shrum, S., 2008. CMMI or Agile: Why Not Embrace Both! Technical Note 2008-TN-003, CMU Software Engineering Institute.

Goldman, S.L., Nagel, R.N., Preiss, K., 1995. Agile competitors and virtual organizations: strategies for enriching the customer. Van Nostrand Reinhold, New York, NY, USA.

Gore, A., 2008. Exploring the competitive advantage through ERP systems: From implementation to applications in agile networks. Dissertation, University of Oulu, Finland.

Gottesman, E., Takas, A., 2007. Agile is Here to Stay é Now What? Agile Journal (December).

Gould, P., 1997. What is agility? IEE Manufacturing Engineer 76(1), 28-31.

Graaf, B., Lormans, M., Toetenel, H., 2003. Embedded Software Engineering: The State of the Practice. IEEE Software 20(6), 61-69.

Greene, B., 2004. Agile Methods Applied to Embedded Firmware Development. In: Proc. Agile Development Conference (ADG), pp. 71-77.

Guckenheimer, S., Perez, J.J., 2006. Software Engineering with Microsoft Visual Studio Team System. Addison-Wesley

Gunasekaran, A., 1998. Agile manufacturing: enablers and an implementation framework. International Journal of Production Research 36(5), 1223-1247.

Gunasekaran, A., Yusuf, Y.Y., 2002. Agile manufacturing: a taxonomy of strategic and technological imperatives. International Journal of Production Research 40(6), 1357-1385.

Haeckel, S.H., 1999. Adaptive Enterprise: Creating and Leading Sense-And-Respond Organizations. Harvard Business School Press.

Hansson, C., Dittrich, Y., Gustafsson, B., Zarnak, S., 2006. How agile are industrial software development practices? Journal of Systems and Software 79(9), 1295-1311.

Hartmann, D., Dymond, R., 2006. Appropriate Agile Measurement: Using Metrics and Diagnostics to Deliver Business Value. In: Proc. Agile Conference, pp. 126-131.

Heikkilä, J., 2002. From supply to demand chain management: efficiency and customer satisfaction. Journal of Operations Management 20, 747-767.

Heikkilä, J., Holmström, J., 2005. Agility and cost efficiency in supply chain management: mutually exclusive or mutually reinforcing objectives? In: Proc. ICAM, pp. 7-11.

Heikkilä, J., Ketokivi, M., 2005. Tuotanto murroksessa: strategisen johtamisen uusi haaste. Talentum, Helsinki, Finland (in Finnish).

Heikkinen, H.L.T., Jyrkämä, J., 1999. Mitä on toimintatutkimus? In: Heikkinen, H.L.T., Huttunen, R., Moilanen, P. (Eds.). Siinä tutkija missa tekijä: Toimintatutkimuksen perusteita ja näköaloja. Atena, Finland (in Finnish)

Heiskanen, A., Newman, M., 1997. Bridging the gap between information systems research and practice: the reflective practitioner as a researcher. In: Proceedings of the eighteenth international conference on Information systems (ICIS), 121-131.

Henderson-Sellers, B., Serour, M.K., 2005. Creating a Dual-Agility Method: The Value of Method Engineering. Journal of Database Management 16(4), 1-23.

Henzinger, T.A., Sifakis, J., 2006. The Embedded Systems Design Challenge. In: Proc. 14th International Symposium on Formal Methods (FM), pp. 1-15.

Hevner, A.R., March, S.T., Jinsoo, P., Ram, S., 2004. Design Science in Information Systems Research. MIS Quarterly 28(1), 75-105.

Hietala, J., Kontio, J., Jokinen, J.-P., Pyysiainen, J., 2004. Challenges of software product companies: results of a national survey in Finland. In: Proc. 10th Intõl Symp. Software Metrics (METRICS), pp. 232-243.

Highsmith, J., 2002. Agile Software Development Ecosystems. Addison-Wesley / Pearson

Highsmith, J., 2004. Agile Project Management: Creating Innovative Products. Addison-Wesley / Pearson, Boston, MA, USA.

Highsmith, J., 2005. Agile for the enterprise: From agile teams to agile organizations. http://www.cutter.com/project/fulltext/reports/2005/01/index.html

Highsmith, J., 2007. Agile Transitions, Part 1. http://www.cutter.com/project/fulltext/advisor/2007/apm071108.html

Highsmith, J., Cockburn, A., 2001. Agile Software Development: The Business of Innovation. IEEE Computer 34(9), 120-127.

Highsmith, J., Wysocki, K., 2006. How Agile Are Organizations Today? Cutter Agile Project Management Executive Report 7(12).

Hinkin, T., Holtom, B.C., Klag, M., 2007. Collaborative Research: Developing Mutually Beneficial Relationships Between Researchers and Organizations. Organizational Dynamics 36(1), 105-118.

Hoda, R., Noble, J., Marshall, S., 2008. A for Agile: Issues with Awareness and Adoption. In: Proceedings of the Research-In-Progress Track at Agile Conf.

Hofman, D., Cecere, L., 2005. The Agile Supply Chain. Supply Chain Management Review 9(8), 18-19.

Holmberg, L., Mathiassen, L., 2001. Survival Patterns in Fast-Moving Software Organizations. IEEE Software 18(6), 51-55.

Hoopes, D.G., Madsen, T.L., Walker, G., 2003. Guest Editorõs Introduction to the Special Issue: Why Is There a Resource-Based View? Toward a Theory of Competitive Heterogeneity. Strategic Management Journal 24(10), 889-902.

Hoque, F., Sambamurthy, V., Zmud, R., Trainer, T., Wilson, C., 2008. BTM Capability Investment Mix. Inside BTM (February), 20-21.

HP, 2003. Meeting the agility challenge: Increasing the time, range, and ease of implementing change. 5981-8004EN

HP, 2005. Measuring and benchmarking the agility of your business. White paper, 4AA0-1790ENW

http://www.gbg.expo-c.se/JamesOCoplienMonday234.htm

Humphrey, W.S., Over, J.W., Konrad, M.C., Peterson, W.C., 2007. Future Directions in Process Improvement. CrossTalk 20(2), 17-22.

Huttunen, R., Kakkori, L., Heikkinen, H.L.T., 1999. Toiminta, tutkimus ja totuus. In: Heikkinen, H.L.T., Huttunen, R., Moilanen, P. (Eds.), 1999. Siinä tutkija missa tekijä: Toimintatutkimuksen perusteita ja näköaloja. Atena, Finland (in Finnish)

Iansiti, M., 1995. Shooting the Rapids: Managing Product Development in Turbulent Environments. California Management Review 38(1) (Fall), 37-58.

IEEE, 2007. Draft Recommended Practice for the Customer-Supplier Relationship in Agile Software Projects. P1648/D5

Irani, Z., Love P.E.D., 2002. Developing a frame of reference for *ex-ante* IT/IS investment evaluation. European Journal of Information Systems 11, 74-82.

Ismail, H.S., Snowden, S.P., Poolton, J., Reid, I.R., Arokiam. I.C., 2006. Agile manufacturing framework and practice. International Journal of Agile Systems and Management 1(1), 11-28.

ITEA, 2004. Technology Roadmap for Software-Intensive Systems (2nd Ed.)

ITEA-AGILE, 2007a. http://www.agile-itea.org/public/publications.php (accessed 6.8.2007)

ITEA-AGILE, 2007b. Project Results (October 2007).

ITID, 2008. Product Development Capabilities (FY2007 Edition). White Paper, iTiD Consulting Ltd.

Itkonen, J., Rautiainen, K., Lassenius, C., 2005. Towards Understanding Quality Assurance in Agile Software Development. In: Proc. ICAM, pp. 201-207.

James, T., 2005. Stepping back from lean. IEE Manufacturing Engineer 84(1), 16-21.

Jankovic, A.D., 2005. Business Research Projects. Thomson Learning, London, England.

Johnson, G., Scholes, K., Whittington, R., 2006. Exploring Corporate Strategy: Text and Cases (7th Enhanced Media Ed.), Pearson Education Ltd

Judy, K.H., Krumins-Beens, I., 2007. Using Agile Practices to Spark Innovation in a Small to Medium Sized Business. In: Proc. HICSS, pp. 275b

Järvinen, P., 2004. On research methods. Opinpajan kirja, Tampere, Finland.

Kahn, K.B., Franzak, F., Griffin, A., Kohn, S., Miller, C.W., 2003. Editorial: Identification and Consideration of Emerging Research Questions. Journal of Product Innovation Management 20, 193-201.

Kaipia, R., 2007. Supply chain coordination ï Studies on planning and information sharing mechanisms. Dissertation, Helsinki University of Technology, Finland.

Kane, D., Ornburn, S., 2002. Agile Development: Weed or Wildflower? CrossTalk 15(10), 30.

Kanter, R.M., 2008. Transforming Giants. Harvard Business Review 86(1), 43-52.

Karlström, D., Runeson, P., 2005. Combining Agile Methods with Stage-Gate Project Management. IEEE Software 22(3), 43-49.

Karlström, D., Runeson, P., 2006. Integrating agile software development into stage-gate managed product development. Empirical Software Engineering 11, 203-225.

Katayama, H., Bennett, D., 1999. Agility, adaptability and leanness: A comparison of concepts and a study of practice. International Journal of Production Economics 60-61, 43-51.

Kauppinen, T.J., 1999. Navigoiva johtaminen. Otava, Helsinki, Finland (in Finnish).

Keaveney, S., Conboy, K., 2006. Cost Estimation in Agile Development Projects. In: Proc. 14th European Conf. Information Systems (ECIS).

Kettunen, P., 2000. Software Requirements Engineering for an Embedded System Development. Licentiate Thesis, Helsinki University of Technology, Finland.

Kettunen, P., 2001. Towards Rapid Development of Embedded Telecommunications System Software Products. Working paper (unpublished), Helsinki University of Technology, Finland.

Kettunen, P., 2009. Adopting Key Lessons from Agile Manufacturing to Agile Software Product Development ï A Comparative Study. Technovation 29, 408-422.

Kettunen, P., Laanti, M., 2008. Combining Agile Software Projects and Large-scale Organizational Agility. Software Process: Improvement and Practice 13(2), 183-193.

Khan, A., Balbo, S., 2005. Agile versus Heavyweight Web Development: An Australian Survey.
http://ausweb.scu.edu.au/aw05/papers/edited/khan/paper.html (accessed 27.12.2007)

Kidd, P.T., 1997. Agile Enterprise Strategy: A Next Generation Manufacturing Concept. IEEE

Kirby, J., 2005. Toward a Theory of High Performance. Harvard Business Review 83(7/8), 30-39.

Kirjavainen, P., Laakso-Manninen, R., 2001. Strategisen osaamisen johtaminen: Yrityksen tieto ja osaaminen kilpailuedun lähteeksi. Edita, Helsinki, Finland (in Finnish).

Kivelä, M., 2007. Dynamic Capabilities in Small Software Firms. Dissertation, Helsinki School of Economics, Finland.

Kiviniemi, K., 1999. Toimintatutkimus yhteisöllisenä prosessina. In: Heikkinen, H.L.T., Huttunen, R., Moilanen, P. (Eds.). Siinä tutkija missa tekijä: Toimintatutkimuksen perusteita ja näköaloja. Atena, Finland (in Finnish)

Koskela, L., Howell, G., 2002. The Underlying Theory of Project Management is Obsolote. In: Proc. PMI Research Conference, pp. 293-302.

Kotler, P., 1994. Marketing Management: Analysis, Planning, Implementation, and Control (8th Ed.). Prentice-Hall

Kotler, P., Armstrong, G., Saunders, J., Wong, V., 1996. Principles of Marketing: The European Edition. Prentice Hall Europe

Krisnan, V., Ulrich, K.T., 2001. Product Development Decisions: A Review of the Literature. Management Science 47(1), 1-21.

Kujala, S., 2007. IHTE-1800: Scientific research methods in human-centered technology ï Introduction. Tampere University of Technology, Finland.
http://www.cs.tut.fi/~kujala/tutkimusmenetelma/Materiaali/Introduction.pdf (accessed 11.10.2009)

Kähkönen, T., 2004. Agile methods for large organizations ï building communities of practice. In: Proc. Agile Development Conf. (ADC), pp. 2-10.

Laanti, M., 2008. Implementing Program Model with Agile Principles in a Large Software Development Organization. In: Proc. Annual IEEE International Computer Software and Applications Conference, pp. 1385-1387.

Lappo, P., Andrew, H.C.T., 2004. Assessing Agility. In: Proc. Extreme Programming and Agile Processes in Software Engineering (LNCS 3092), pp. 331-338.

Larman, C., 2004. Agile and Iterative Development: A Managerôs Guide. Addison-Wesley / Pearson

Larman, C., Basili, V.R., 2003. Iterative and Incremental Development: A Brief History. IEEE Computer 36(6), 47-56.

Lassenius, C., 2006. Software Development Control Panels: Concepts, a Toolset and Experiences. Dissertation, Helsinki University of Technology, Finland.

Lecklin, O., 2002. Laatu yrityksen menestystekijänä. Kauppakaari, Helsinki, Finland (in Finnish).

Lee, A., Baskerville, R., 2003. Generalising generalsibility in information systems research. Information Systems Research 14, 221-243.

Leffingwell, D., 2007. Scaling Software Agility: Best Practices for Large Enterprises. Addison-Wesley.

Leishman, T., 2001. Extreme Methodologies for an Extreme World. CrossTalk 14(6), 15-18.

Leppälä, K., Kääriäinen, J., Takalo, J., Savolainen, P., 2005. Challenging global competition: tune up your product development. http://www.vtt.fi/inf/pdf/workingpapers/2005/W34.pdf

Levine, L., 2005. Reflections on Software Agility and Agile Methods: Challenges, Dilemmas, and the Way Ahead. CMU Software Engineering Institute

Levinson, M., 2004. How to Build an Agile IT Department. CIO Magazine (10 September)

Liker, J.K., 2004. The Toyota Way: 14 Management Principles from the Worldôs Greatest Manufacturer. McGraw-Hill, New York, NY, USA.

Lin, C.-H., Chiu, H., Chu, P.-Y., 2006. Agility index in the supply chain. Int. J. Production Economics 100, 285-299.

Lindstedt, P., Burenius, J., 2003. The Value Model: How to Master Product Development and Create Unrivalled Customer Value. Nimba AB, Sweden.

Lindvall, M., Muthig, D., Dagnino, A., Wallin, C., Stupperich, M., Kiefer, D., May, J., Kähkönen, T., 2004. Agile software development in large organizations. IEEE Computer 37(12), 26-34.

Little, T., 2005. Context-adaptive agility: managing complexity and uncertainty. IEEE Software 22(3), 28-35.

Liu, X., Sun, Y., Kane, G., Kyoya, Y., Noguchi, K., 2006. Business-Oriented Software Process Improvement Based on CMM using QFD. Software Process: Improvement and Practice 11, 573-589.

Lovén, E., 2006. New Technology Initiatives from Within and Outside of a Company Express Different Agility Abilities. International Journal of Agile Manufacturing 9(1), 109-113.

Lyytinen, K., Rose, G.M., 2004. How Agile is Agile Enough? Towards A Theory of Agility in Software Development. Sprouts: Working Papers in Information Environments, Systems and Organizations 4(4), 169-192.

Lyytinen, K., Rose, G.M., 2006. Information system development agility as organizational learning. European Journal of Information Systems 15, 183-199.

MacCormack, A., 2001. Product Development Practices That Work: How Internet Companies Build Software. MIT Sloan Management Review 42(2) (Winter), 75-84.

MacCormack, A., Verganti, R., 2003. Managing Sources of Uncertainty: Matching Process and Context in Software Development. Journal of Product Innovation Management 20, 217-232.

MacCormack, A., Verganti, R., Iansiti, M., 2001. Developing Products on ñInternet Timeò: The Anatomy of a Flexible Development Process. Management Science 47(1), 133-150.

Manhart, P., Schneider, K., 2004. Breaking the Ice for Agile Development of Embedded Software: An Industry Experience Report. In: Proc. 26th Intôl Conf. Software Engineering (ICSE), pp. 378-386.

Mar, K., 2006a. An Enterprise Strategy for Introducing Agile. White paper, http://danube.com/whitepapers/enterprise-strategy

Mar, K., 2006b. Impediments to Enterprise Agile. White paper, http://danube.com/whitepaper/impediments-enterprise

Mascitelli, R., 2006. The Lean Product Development Guidebook: Everything Your Design Team Needs to Improve Efficiency and Slash Time to Market. Technology Perspectives

Mathiassen, L., Vainio, A.M., 2007. Dynamic Capabilities in Small Software Firms: A Sense-and-Response Approach. IEEE Trans. Engineering Management 54(3), 522-538.

McConnell, S., 1996. Rapid Development: Taming Wild Software Schedules. Microsoft Press, Redmond

McConnell, S., 1998. Software Project Survival Guide. Microsoft Press, Redmond

McGrath, M.E., 2004. Next Generation Product Development: How to Increase Productivity, Cut Costs, and Reduce Cycle Times. McGraw-Hill

McGrath, R.G., MacMillan, I.C., 1995. Discovery-Driven Planning. Harvard Business Review 73(4), 44-54.

McInerney, P., Maurer, F., 2005. UCD in Agile Projects: Dream Team or Odd Couple? ACM Interactions, 12(6), 19-23.

McMahon, P., 2002. Integrating Systems and Software Engineering: What Can Large Organizations Learn From Small Start-Ups? CrossTalk 15(10), 22-25.

McMahon, P., 2005. Extending agile methods: a distributed project and organizational improvement perspective. CrossTalk 18(5), 16-19.

Mikkonen, T., Pruuden, P., 2001. Flexibility as a design driver. IEEE Computer 34(11), 52-56.

Mingers, J., 2001. Combining IS Research Methods: Towards a Pluralist Methodology. Information Systems Research 12(3), 240-259.

Mintzberg, H., 2007. Opinion: Productivity Is Killing American Enterprise. Harvard Business Review 85(7/8), 25-25.

Mirakaj, D., 2008. The Building Blocks of Agility. Inside BTM (February), 8-10.

Moløkken-Østvold, K., Jørgensen, M., 2005. A Comparison of Software Project Overruns ï Flexible versus Sequential Development Models. IEEE Trans. Software Engineering 31(9), 754-766.

Morgan, J.M, Liker, J.K., 2006. The Toyota Product Development System: Integrating People, Processes, and Technology. Productivity Press

Myers, M.D., 2007. Qualitative Research in Information Systems. http://www.qual.auckland.ac.nz/index.htm (accessed 7.8.2007)

Mäkelä, M., 2008. Essays on Software Product Development. A Strategic Management Viewpoint. Dissertation, Helsinki School of Economics, Finland

Männistö, T., 2000. A Conceptual Modelling Approach to Product Families and their Evolution. Dissertation, Helsinki University of Technology, Finland.

Naik, B., Chakravarty, A.K., 1992. Strategic acquisition of new manufacturing technology: a review and research framework. Int. J. Prod. Res. 30(7), 1575-1601.

Nakano, M., 2007. Corporate Finance: Intangible Assetsô Effect on Stakeholdersô Value. In: Knowledge Creation and Management: New Challenges for Managers (Eds. Ichijo, K., Nonaka, I.). Oxford University Press

Nambisan, S., 2003. Information Systems as a Reference Discipline for New Product Development. MIS Quarterly 27(1), pp. 1-18.

Nambisan, S., Wilemon, D., 2000. Software Development and New Product Development: Potentials for Cross-Domain Knowledge Sharing. IEEE Trans. Engineering Management 47 (2), 211-220.

Narasimhan, R., Swink, M., Kim, S.W., 2006. Disentangling leanness and agility: An empirical investigation. Journal of Operations Management 24(5), 440-457.

Naylor, J.B., Naim, M.M., Berry, D., 1999. Leagility: Integrating the lean and agile manufacturing paradigms in the total supply chain. International Journal of Production Economics 62, 107-118.

Nerur, S., Balijepally, V., 2007. Theoretical Reflections on Agile Software Development Methodologies. Communications of the ACM 50(3), 79-83.

Nerur, S., Mahapatra, R., Mangalaraj, G., 2005. Challenges of migrating to agile methodologies. Communications of the ACM 48(5), 73-78.

Ngwenyama, O., Nielsen, P.A., 2003. Competing Values in Software Process Improvement: An Assumption Analysis of CMM From an Organizational Culture Perspective. IEEE Trans. Engineering Management 50(1), 100-112.

Nidiffer, K.E., Dolan, D., 2005. Evolving Distributed Project Management. IEEE Software 22(5), 63-72.

Nies, Z., 2008. Continuous Quality ï Shortening the Development and Test Feedback Loop Beyond Continuous Build. Agile Journal (June).

Nobelius, D., 2004. Towards the sixth generation of R&D management. International Journal of Project Management 22(5), 369-375.

Nobelius, D., Trygg, L., 2002. Stop chasing the Front End process ï management of the early phases in product development projects. International Journal of Project Management 20(5), 331-340.

Northover, M., Northover, A., Gruner, S., Gruner, S., Kourie, D.G., Boake, A., 2007. Agile Software Development: A Contemporary Philosophical Perspective. In: Proc. SAICSIT, pp. 106-115.

Näsi, J., Neilimo, K., 2006. Mitä on liiketoimintaosaaminen? WSOYpro, Helsinki, Finland (in Finnish).

Oberhauser, R., Schmidt, R., 2007. Improving the Integration of the Software Supply Chain via the Semantic Web. In: Proc. International Conference on Software Engineering Advances (ICSEA), pp. 79-79.

Oiva, A., 2007. Strategy-focused capability management model and organizational strategic readiness. Dissertation, University of Oulu, Finland.

Ojala, P., 2006. Implementing a value-based approach to software assessment and improvement. Dissertation, University of Oulu, Finland.

Oosterhout, van M., Waarts, E., Hillegersberg, van J., 2006. Change factors requiring agility and implications for IT. European Journal of Information Systems 15, 132-145.

Overby, E., Bharadwaj, A., Sambamurthy, V., 2006. Enterprise agility and the enabling role of information technology. European Journal of Information Systems 15, 120-131.

Paulk, M.C., 1996. Effective CMM-Based Process Improvement. CMU

Pavlou, P.A., El Sawy, O.A., 2006. From IT Leveraging Competence to Competitive Advantage in Turbulent Environments: The Case of New Product Development. Information Systems Research 17(3), 198-227.

Pettit, R., 2006. An ñAgile Maturity Model?ò Agile Journal, http://www.agilejournal.com/content/view/52/

Phillips, M., 2008. CMMI with Agile, Lean, Six Sigma, and Everything Else. http://sei.cmu.edu/news-at-sei/columns/2008/01/ (accessed 19.2.2009)

Pikkarainen, M., 2008. Towards a Framework for Improving Software Development Process Mediated with CMMI Goals and Agile Practices. Dissertation, University of Oulu, Finland.

Pikkarainen, M., Mäntyniemi, A., 2006. An Approach for Using CMMI in Agile Software Development Assessments: Experiences from Three Case Studies. In: Proc. SPICE Conf.

PMBOK, 2004. A Guide to the Project Management Body of Knowledge. http://www.pmi.org

Poppendieck, M., Poppendieck, T., 2004. Lean Software Development: An Agile Toolkit. Addison-Wesley / Pearson: Upper Saddle River, NJ, USA.

Porter, M.E., 1993. Strategia kilpailutilanteessa: toimialojen ja kilpailijoiden analysointitekniikat. Rastor: Helsinki, Finland (in Finnish).

Preiss, K., 2005. Agility ï the origins, the vision and the reality. In: Proc. Intõl Conf. Agility (ICAM), pp. 13-21.

Prewitt, E., 2004. The Agile 100 Honoree Survey. CIO Magazine (15 August)

Pyhäjärvi, M., 2006. Going Agile at F-Secure. In: Bi-annual OO-days, Tampere University of Technology, Finland http://www.cs.tut.fi/tapahtumat/olio2006/pyhajarvi.pdf

Qumer, A., Henderson-Sellers, B., 2008. A framework to support the evaluation, adoption and improvement of agile methods in practice. Journal of Systems and Software 81, 1899-1919.

Raafat, F., 2002. A comprehensive bibliography on justification of advanced manufacturing systems. International Journal of Production Economics 79(3), 197-208.

Raelin, J.A., 1997. Action Learning and Action Science: Are They Different? Organizational Dynamics 26(1) (Summer), 21-33.

Raivio, T., Syrjänen, M., Halonen, M., 2006. Tekes elektroniikan moottorina: Tekesin elektroniikka-alan ohjelmatoiminnan arviointi. Tekes, Finland (in Finnish).

Ramesh, B., Cao, L., Mohan, K., Xu, P., 2006. Can Distributed Software Development Be Agile? Communications of the ACM 49(10), 41-46.

Ramos, C., Gravendeel, E., 2008. Top 9 Challenges of Adopting Scrum. Agile Journal (August).

Rauscher, T.G., Smith, P.G., 1995. Time-Driven Development of Software in Manufactured Goods. Journal of Product Innovation Management 12, 186-199.

Rautiainen, K., Lassenius, C., Nihtilä, J.. Sulonen, R., 1999. Key Issues in New Product Development Controllability Improvement ï Lessons Learned from European High-Tech Industries. In: Proc. Portland International Conference on Management of Engineering and Technology, Technology and Innovation Management (PICMET). 1, pp. 177.

Raynor, M.E., Leroux, X., 2004. Strategic Flexibility in R&D. Research Å Technology Management 47(3), 27-32.

Ronkainen, J., Abrahamsson, P., 2003. Software development under stringent hardware constraints: Do agile methods have a chance? In: Proc. 4th Intõl Conf. Extreme Programming and Agile Processes in Software Engineering, pp. 73-79.

Ropponen, J., Lyytinen, K., 2000. Components of Software Development Risk: How to Address Them? A Project Manager Survey. IEEE Trans. Software Engineering 26(2), 98-111.

Routio, P., 2007. Arteology, the science of products and professions. http://www2.uiah.fi/projects/metodi/e00.htm (accessed 2.8.2007)

Royce, W., 1998. Software Project Management. Addison-Wesley

Royce, W., 2005. Successful software management style: Steering and balance. IEEE Software 22(5), 40-47.

Salem, S.R., 2001. New product development balanced scorecard. In: Proceedings of Change Management and the New Industrial Revolution, pp. 110-112

Salo, O., 2007. Enabling Software Process Improvement in Agile Software Development Teams and Organizations. Dissertation, University of Oulu, Finland.

Salo, O., Abrahamsson, P., 2008. Agile methods in European embedded software development organizations: a survey study of Extreme Programming and Scrum. IET Software 2(1), 58-64.

Sanchez, R., 1995. Strategic Flexibility in Product Competition. Strategic Management Journal 16, 135-159.

Sanchez, R., Mahoney, J.T., 1996. Modularity, Flexibility, and Knowledge Management in Product and Organization Design. Strategic Management Journal 17 (Winter), 63-76.

Santalainen, T., 2005. Strateginen ajattelu. Talentum, Helsinki, Finland (in Finnish).

SAS-050, 2006. Exploring New Command and Control Concepts and Capabilities. http://www.dodccrp.org/files/SAS-050%20Final%20Report.pdf (accessed 14.11.2007)

Schein, E.H., 1999. The Corporate Culture Survival Guide ï Sense and Nonsense about Culture Change. Jossey-Bass

Schuh, P., 2005. Integrating Agile Development in the Real World. Charles River Media, Inc.

Schmidt, R., Lyytinen, K., Keil, M., Cule, P., 2001. Identifying Software Project Risks: An International Delphi Study. Journal of Management Information Systems 17(4) (Spring), 5-36.

Schoonenderwoert, van N., 2007. The Four Change Initiatives of Agile Adoption. In: Deep Agile Seminar, MIT

Schoonenderwoert, van N., 2008. The Four Pillars of Agile Adoption. Agile Journal (June).

Schwaber, C., 2005.Corporate IT Leads The Second Wave Of Agile Adoption. Forrester Research, Inc.

Schwaber, C., 2007. The Truth About Agile Processes. Forrester Research, Inc.

Schwaber, K., 2001. Will the real agile processes please stand up? http://www.cutter.com/project/ fulltext/reports/2001/08/index.html

Schön, D.A., 1983. The Reflective Practitioner: How Professionals Think in Action. Ashgate Publishing, England.

Scinta, J., 2008. Industrial Research Instituteõs R&D Trends Forecast for 2008. Research Å Technology Management 51(1), 19-23.

Sharifi, H., Zhang, Z., 1999. A methodology for achieving agility in manufacturing organisations: An introduction. International Journal of Production Economics 62, 7-22.

Shum, P., Lin G., 2007. A world class new product development best practices model. International Journal of Production Research 45(7), 1609-1629.

Sidky, A., 2007. A Structured Approach to Adopting Agile Practices: The Agile Adoption Framework. Dissertation, Virginia Tech, USA.

Sifakis, J., 2007. Embedded Systems: Research Challenges and Work Directions. ARTEMIS Annual Conference.
http://www.artemis-office.org/DotNetNuke/Portals/0/Conference%202007/AC_J.SIFAKIS.pdf

Skaistis, B., 2006. A Four-Step Plan for Measuring Enterprise IT Agility.
http://esj.com/enterprise/article.aspx?EditorialsID=2135 (accessed 15.11.2007)

Skowronski, V., 2004. Do Agile Methods Marginalize Problem Solvers? IEEE Computer 37(10), 118-120.

Smith, J.M., 2001. Troubled IT Projects ï prevention and turnaround. IEE

Smith, P.G., 2007. Flexible Product Development: Building Agility for Changing Markets. Jossey-Bass.

Smith, P.G., 2008. Change: Embrace It, Donôt Deny It. Research • Technology Management 51(4), 34-40.

Smith, P.G., Reinertsen, D.G., 1998. Developing Products in Half the Time: New Rules, New Tools. John Wiley & Sons.

Sobek II, D.K., Liker, J.K., Ward, A.C., 1998. Another Look at How Toyota Integrates Product Development. Harvard Business Review 76(4), 36-49.

Solingen, van R., 2002. Integrating Software Engineering Technologies for Embedded Systems Development. In: Proc. PROFES, pp. 466-474.

Sommerville, I., 2001. Software Engineering (6th Ed.), Addison-Wesley.

Souder, W.E., Moenaert, R.K., 1992. Integrating Marketing and R&D Project Personnel within Innovation Projects: An Information Uncertainty Model. Journal of Management Studies 29(4), 485-512.

Subramaniam, V., Hunt, A., 2006. Practices of an Agile Developer ï Working in the Real World. The Pragmatic Bookshelf, USA.

Suikki, R., 2007. Changing business environment ï effects of continuous innovations and disruptive technologies. Dissertation, University of Oulu, Finland.

Susman, G.I., Evered, R.D., 1978. An Assessment of the Scientific Merits of Action Research. Administrative Science Quartely 23, 582-603.

Sutherland, J., 2001. Inventing and Reinventing SCRUM in Five Companies. http://www.agilealliance.org/system/article/file/888/file.pdf (accessed 16.2.2009)

Sutherland, J., Viktorov, A., Blount, J., Puntikov, N., 2007. Distributed Scrum: Agile Project Management with Outsourced Development Teams. In: Proc. HICSS, pp. 274a

Syed-Abdullah, S., Holcombe, M., Gheorge, M., 2006. The Impact of an Agile Methodology on the Well Being of Development Teams. Empirical Software Engineering 11, 143-167.

Tabaka, J., 2007. 11 Ways Agile Adoptions Fail.
StickyMinds.com Column, (6/4/2007).

Tabaka, J., Martens, R., 2008. Leaning IT: Applying the Principle of ñPullò to Scale Agile Teams. White paper, Rally

Takeuchi, H., Nonaka, I., 1986. The new new product development game. Harvard Business Review 64(1), 137-146.

Tanskanen, M., 2008. Organizing of Faster Release Cycle ï Software Development Aspect. Masterôs thesis, Helsinki University of Technology, Finland.

Taramaa, J., Khurana, M., Kuvaja, P., Lehtonen, J., Oivo, M., Seppänen, V., 1998. Product-based software process improvement for embedded systems. In: Proc. 24th Euromicro Conf. (2), pp. 905-912.

Tate, K., 2006. Sustainable Software Development ï An Agile Perspective. Addison-Wesley

TEKES, 2006. Liito ï Innovative Business Competence and Management 2006-2010.
http://akseli.tekes.fi/opencms/opencms/OhjelmaPortaali/ohjelmat/Luotsi/en/etusivu.html

Thomke, S., Reinertsen, D., 1998. Agile Product Development: Managing Development Flexibility in Uncertain Environments. California Management Review 41(1) (Fall), 8-30.

TNO/IDATE, 2005. Software intensive systems in the future, Final report. http://www.itea2.org/attachments/150/ITEA_SIS_in__the_future__Final_Report.pdf

Treacy, M., Wiersema, F., 1995. The Discipline of Market Leaders: Choose Your Customers, Narrow Your Focus, Dominate Your Market. Addison-Wesley.

Trott, P., 2005. Innovation Management and New Product Development. Pearson Education, England.

Tsourveloudis, N., Valavanis, K., Gracanin, D., Matijasevic, M., 1999. On the Measurement of Agility in Manufacturing Systems. In: European Symposium on Intelligent Techniques (ESIT). http://www.erudit.de/erudit/events/esit99/12607_p.pdf

Turk, D., France, R., Rumpe, B., 2002. Limitations of Agile Software Processes. In: Proc. XP, pp. 43-46.

Turk, D., France, R., Rumpe, B., 2005. Assumptions Underlying Agile Software-Development Processes. Journal of Database Management 16(4), 62-87.

Turkulainen, V., 2008. Managing Cross-functional Interdependencies ï The Contingent Value of Integration. Dissertation, Helsinki University of Technology, Finland.

Tuormaa, J., 2009. Autoteollisuuden kriisi ei vielä iske tuotekehitykseen. Tekniikka & Talous 49(4), 24 (in Finnish).

Turner, R., 2007. Toward Agile Systems Engineering Processes. CrossTalk 20(4), 11-15.

Turner, R., Boehm, B., 2003. People Factors in Software Management: Lessons From Comparing Agile and Plan-Driven Methods. CrossTalk 16(12), 4-8.

Tyrväinen, P., Varsta, J., Seppänen, V., 2004. Toimialakehitys ohjelmistoteollisuuden vauhdittajana: Uutta liiketoimintaa lähialoilta. Tekes, Finland (in Finnish).

Ulrich, K.T., Eppinger, S.D., 2000. Product Design and Development. McGraw-Hill.

Underseth, M., 2007. Optimizing the Embedded Software Supply Chain. ESE Magazine / Application Software Development 15.8, 48.

Upton, D.M., 1994. The Management of Manufacturing Flexibility. California Management Review 36(2), 72-89.

Vainio, A.M., Tuunanen, T., Abrahamsson, P., 2005. Developing Software Products for Mobile Markets: Need for Rethinking Development Models and Practices. In: Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS), pp. 189b-189b.

Vanhanen, J., Jartti, J., Kähkönen, T., 2003. Practical Experiences of Agility in the Telecom Industry. In: Proc. 4th Intõl Conf. Extreme Programming and Agile Processes in Software Engineering, pp. 279-287.

Vázquez-Bustelo, D., Avella, L., 2006. Agile manufacturing: Industrial case studies in Spain. Technovation 26, 1147-1161.

Vehtari, S., 2006. The Dynamics Involved with Manufacturing Capabilities Towards a Competitive Advantage. Dissertation, Helsinki University of Technology, Finland.

Verganti, R., 1999. Planned Flexibility: Linking Anticipation and Reaction in Product Development Projects. Journal of Product Innovation Management 16, 363-376.

Version One, 2008. 3rd Annual ñState of Agile Developmentò Survey. http://www.versionone.com/agilesurvey/

Vilkki, K., 2007. Experience with the use Agile Methods in Systems Development. In: EuroSPI, Keynote

Vodde, B., 2006. Nokia Networks and Agile Development. http://www.odd-e.com/articles/2006/nokia_agile.pdf (accessed 17.11.2006)

Vodde, B., 2007. Stories from the Flexible Company. http://www.odd-e.com/articles/2007/JAOO_flex_company.pdf (accessed 31.12.2007)

VTT, 2008. Mobile-D. http://agile.vtt.fi/mobiled.html (accessed 22.9.2008)

Välimäki, A., Kääriäinen, J., 2008. Patterns for Distributed Scrum ï a Case Study. In: Enterprise Interoperability III, pp. 85-97.

Wallin, C., Ekdahl, F., Larsson, S., 2002. Integrating Business and Software Development Models. IEEE Software 19(6), 28-33.

Wang, C.L., Ahmed, P.K., 2007. Dynamic capabilities: A review and research agenda. International Journal of Management Reviews 9(1), 31-51.

Ward, A.C., 2007. Lean Product and Process Development. The Lean Enterprise Institute: Cambridge, MA, USA.

Wheelwright, S.C., Clark, K.B., 1992. Revolutionizing Product Development: Quantum Leaps in Speed, Efficiency, and Quality. The Free Press.

Wils, A., Van Baelen, S., Holvoet, T., De Vlaminck, K., 2006. Agility in the avionics software world. In: Proc. 7th Intõl Conf. Extreme Programming and Agile Processes in Software Engineering (XP), pp. 123-132.

Wind, J., Mahajan, V., 1997. Issues and Opportunities in New Product Development: An Introduction to the Special Issue. Journal of Marketing Research XXXIV, 1-12.

Winter, S.G., 2003. Understanding Dynamic Capabilities. Strategic Management Journal 24(10), 991-995.

Worley, C.G., Lawler, E.E., 2006. Designing Organizations That Are Built to Change. MIT Sloan Management Review 48(1) (Fall), 19-23.

Yin, R.K., 1994. Case study research: design and methods. Sage: Thousand Oaks, CA, USA.

Yoffie, D.B., Cusumano, M.A., 1999. Building a Company on Internet Time: Lessons from Netscape. California Management Review 41(3), 8-28.

Yourdon, E., 2002. Managing High-Intensity Internet Projects. Prentice Hall: Upper Saddle River, NJ.

Yusuf, Y.Y., Sarhadi, M., Gunasekaran, A., 1999. Agile manufacturing: The drivers, concepts and attributes. International Journal of Production Economics 62, 33-43.

Zsifkovits, H.E., Engelhardt-Nowitzki, C., 2007. An Analysis of Frameworks for Measuring Supply Chain Agility. In: Proc. ICAM, pp. 87-95.

Ågerfalk, P.J., Fitzgerald, B., 2006. Flexible and Distributed Software Processes: Old Petunias in New Bowls? Communications of the ACM 49(10), 27-34.

Appendix

Table 25. Representative NPD success/failure factor findings (alphabetical order)

Publication	Influencing Factors	Success Criteria
(Akgün, Lynn and Byrne 2004)	POSITIVE: • clear, stable, and supported project visioning based on understanding the customer needs and goals • executive management support • process profiency (execution, tracking) • teamwork (unified goal, joint responsibility, knowledge sharing, experienced people) • documentation systems (rich and efficient information processing) • communication (balanced formal and informal) • project deadline setting (ambitious yet doable)	meeting (or exceeding) project cost, profit, technical and market expectations
(Ancona and Caldwell 1990)	POSITIVE: • efficient product development team external interactions in the organizational context (boundary management): coordination, information sharing	speeding up the product development process
(Atwater and Pittman 2008)	POSITIVE: • parallel developmentof new alternative design options with fall-back solutions	risk reduction, shorter release cycle-times
(Barczak 1995)	POSITIVE: • right mix of NPD strategy, organization, process • fit between the firm NPD strategic goals and the capabilities (skills, resources) • functional R&D teams (for new-to-market products) • cross-functional project teams • product champions • market activities (idea generation, screening)	NPD performance: • % of sales spent on R&D • % of profits and sales accounted for products introduced within the last 5 years • perceived profit, sales and market share goals • overall satisfaction with firmsô NPD efforts
(Brown and Eisenhardt 1995)	• R&D management: commitment, control • project lead: vision, power, skills • project team: composition, work allocation, team process • customers: involvement • suppliers: involvement • process performance (leadtime, productivity) • product effectiveness (fit with market needs, firm capabilities)	financial performance (profits, revenues, market share)

Publication	Influencing Factors	Success Criteria
(Cohan and Unger 2006)	POSITIVE: • entrepreneurial leadership (hiring and motivating people with both technical and business skills) • open technology (using the fastest and most effective sources to satisfy the customer needs) • boundary-free product development (cross-functional teams, rapid prototyping, fast feedback) • disciplined resource allocation (systematic analysis, terminating projects that are unlikely to succeed)	more profitable, faster growth, better stock market performance than the peer competitors
(Cooper and Kleinschmidt 1996, 2007)	POSITIVE: 1. a high-quality new product process 2. a defined new product strategy for the business unit 3. adequate resources of people and money 4. R&D spending for new product development 5. high-quality new product teams 6. senior management commitment 7. an innovative climate and culture 8. the use of cross-functional project teams 9. senior management accountability	Profitability: How profitable the businessõs total new product efforts are (e.g., the profitability vs. competitors)? Impact that the total new product efforts have on the business (percentages of sales by new products, impact of new products on sales and profits)
(Cooper 2006)	POSITIVE: • holistic (effective cross-functional teams) • lean, scalable and adaptive process • customer focused • heavy front-end homework before development begins • metrics, accountable teams, profit/loss reporting for continuous learning • spiral development (loops with users throughout development) • focus and portfolio management	Productivity: output (new product sales, profits) vs. input (research and development, NPD costs and time)
(Dehoff and Loehr 2007)	POSITIVE: • setting ñstretchò new product goals	New-to-world product introductions
(Ernst 2002)	POSITIVE: • NPD process (formal or informal): quality of the planning before beginning the actual development (preparatory work); continuous evaluation of the projects; all process steps aligned with the market requirements; customer integration (possibly); • Organization: dedicated project organization, people specifically assigned to the NPD team, cross-functional project teams with interfunctional communication and co-operation, strong project leader, substantial autonomy for the NPD team, team having the responsibility for the whole NPD process • Senior management recognition of the value of new products: e.g., resource allocation • Organizational culture and NPD strategy: champions, strategic framework, long-term thrust	Following the surveyed publications; Advocates profitability as the ultimate success measure.

Publication	Influencing Factors	Success Criteria
(ITID 2008)	POSITIVE: • appropriate resource allocation following the product development plan • effective, tailored software design methodologies • viewpoint analysis of product functionality allocation, specification for different disciplines • clear and comprehensive evaluation (testing) • key project elements planned up-front • prompt processing of critical risks and problems • project management with necessary authorities until the finish NEGATIVE: • product (hardware) development started without the software developers participating • product schedule set without clear software requirements • software requirements not clearly allocated • software development started without sufficient requirements or concept design • ignorance in upstream processes • organizational boundaries (ñbureaucracyò) between functional departments	product functionality, quality, cost, resources and development lead time achieved as expected (or exceeded)
(Kotler et al. 1996)	POSITIVE: • a unique superior product (new features, high quality) offering clearly better value • well-defined product concept considering the target market and value-offering benefits (prior to actual development) • New product meets the market needs better than the existing ones. • market attractiveness • technological and marketing synergy • quality of execution in all stages • effective organizational structures for new product innovation and realization • top management commitment to new products (e.g., resource allocation) • investment in people (hiring, development) • open and rich information/knowledge sharing across the organization • entrepreneurial company culture (ñclimateò) NEGATIVE: • market size overestimated • new product not clearly better than the ones already available in the marketplace • incorrect market positioning, pricing • higher than budgeted product development cost • unexpected competitor responses	meeting target returns on investment

Publication	Influencing Factors	Success Criteria
(Liker 2004)	POSITIVE: • cross-functional product development teams with strong, visionary leaders (Chief Engineer) • co-location with visual management tools (ñwar roomò) • concurrent engineering of product designs and production (manufacturing) processes	achieving ñstretchò new product goals, product creation cycle-time acceleration
(Rauscher and Smith 1995)	POSITIVE: • resourcing for software growth • knowledge spread (application domain and software engineering management) • sound software engineering economics (scheduling) • appropriate progress measures (# of components completed vs. LOC) • hardware/software codevelopment (no ñsilosò) • early user feedback (prototyping) • effective reviewing throughout the development • understanding the project objectives and scope • flexible design (independent modules) • change cost trade-offs (hardware vs. software) NEGATIVE: • incomplete integration of software engineering functions with the other disciplines • accumulating (late) changes to software (vs. hardware; poor time-money-features trade-offs) • underestimated learning curves (new personnel)	time-to-market (embedded software development time acceleration)
(Trott 2005)	NEGATIVE: • The product offers nothing new or no improved performance. • inadequate budget to develop ideas or market the product • poor market research, positioning, misunderstanding consumer needs • lack of top management support • Did not involve customer. • exceptional factors such as government decision • market too small, either forecasting error with sales of insufficient demand • poor match with company capabilities • inadequate support from channel (supply chain) • Competitive response was strong and competitors were able to move quickly to face the challenge of the new product. • internal organizational problems, often associated with poor communication • poor return on investment forcing company to abandon project • unexpected changes in consumer tastes/fashion	
(Yoffie and Cusumano 1999)	POSITIVE: • clear business vision driving the new product releases • managerial experience in organizational scaling • leveraging external resources to complement in-house capabilities NEGATIVE: • underestimating the time needed to change the customer behavior • revolutionary business strategies stretching the technological capabilities • lack of long-term systematic strategy planning • short-sighted partnership relations management	

Table 26. Representative approaches to flexible NPD (chronological order)

Publication	Approaches	Potential Benefits Costs and Problems
(Takeuchi and Nonaka 1986)	PROCESS: • built-in instability (broad goals with considerable freedom to realize) • overlapping development phases (rhythms and synchronization) ORGANIZATION: • self-organizing project teams (autonomy, self-transcendence, cross-functional fertilization) • ñsubtleò control (enough controls to prevent instability while not impairing creativity) • ñmultilearningò (new information across multiple levels and functions) • organizational transfer of learning (to new projects and other units)	Enable a fast and flexible dynamic process for new development when combined; Support organizational transformations. Requires an appropriate mind-setting and organizational culture of shared responsibility of the outcomes. An intensive process may be more difficult to manage. Finding the right rhythms for the different development cycles may be complicated.
(Eisenhardt and Tabrizi 1995)	PROCESS: • multiple design iterations allowing adjustments • extensive testing providing early feedback • frequent milestones for assessing the progress and current circumstances • powerful project leader maintaining a disciplining vision ORGANIZATION: • multifunctional teams	experiential strategy supporting quick learning and understanding the uncertainties, guiding vision; Need to really understand the project uncertainties. Avoid ñchaoticò processes while allowing adaptation. Organizational processes must be aligned.
(Iansiti 1995)	PROCESS: • overlapping product concept development and implementation (iterative) • system-focused	Need to really understand the project uncertainties. Avoid ñchaoticò processes while allowing adaptation. Organizational processes must be aligned. Requires new skills, management processes and engineering methodologies.
(Sanchez 1995)	ORGANIZATION: • resource flexibility • coordination flexibility • modular organizations PROCESS: • IT tools for rapid designing, testing, and prototyping with efficient market information acquisition and knowledge sharing (even inter-organizational) • modular product designs (standardized interfaces) • concurrent product creation	real-time exploitation of technological and market opportunities, dynamic efficiency in (re)deploying resources, diversification through competence leveraging (networking), flexibility in strategic decision making (options); achievement of dynamic balance in turbulent environments Need to identify and acquire necessary flexible resources. Requires non-traditional coordination mechanisms, organization structures, and even new mind-setting.
(Sanchez and Mahoney 1996)	ORGANIZATION: • modular organizations based on modular product architectures	Allows concurrent and possibly networked development of loosely coupled components; enhanced architectural learning

Publication	Approaches	Potential Benefits Costs and Problems
	• embedded coordination structures	Requires early standardized component interfaces.
(Eisenhardt and Brown 1998)	PROCESS: • Time Pacing: setting the right (natural) rhythms for change (e.g., product releases) and synchronizing those rhythms both with external influences and with the internal capabilities	staying aligned with the markets and technology developments, even influencing the pace of the competition; Need to be proactive. Must avoid too frequent changes (overreaction).
(Smith and Reinertsen 1998)	PROCESS: • incremental innovation	earlier and more reliable feedback from customers and technology, shorter planning horizon and lower technical complexity; earlier profits increased cost of making more (interim) releases, overloading the upstream channel and even customers; avoidance of making radical technology breakthroughs
(Thomke and Reinertsen 1998)	PROCESS: • management processes to increase flexibility: progressive product concept and requirements locking (piecewise commitment), alternative designs (set-based), rapid systematic tradeoff decisions, considering design interdependencies while organizing the work • product architecture solutions: modularity, isolating volatility, low coupling TECHNOLOGY: • adopting inherently flexible technologies allowing low cost design iterations	Requires new skills, management processes and engineering methodologies. Investing in flexibility on the areas where it is most beneficial (economic trade-off).
(Verganti 1999)	PROCESS: • anticipation capabilities: systemic project learning, teamworking and communication, proaction (e.g., early prototyping) • reaction capabilities: flexible resources and design technologies, intense communication for accelerated problem solving, overlapping activities, redundancies (alternative designs), flexible product technologies and modular architectures	Planned flexibility, i.e., the combination of the anticipation and reaction, makes it possible to postpone design decisions about critical uncertainties in a managed way. The early phase of the product development must be capable of anticipating the critical uncertainties and preparing the reaction capabilities accordingly. The realization of such capabilities may require radically new competence, organizational routines, technology, and even culture.
(Nobelius and Trygg 2002)	PROCESS: • adaptation of the front-end activities according to the project-specific needs and circumstances • different alternative Front End ñroutesò	Focusing and prioritizing the activities on the current most important ones taking into account the overall (company) situation. Requires careful assessment of the situational factors. Skilled Front End teams expected. There may be a risk of neclecting important activities.

Publication	Approaches	Potential Benefits Costs and Problems
(McGrath 2004)	ORGANIZATION: • integrated enterprise-wide product/project/resource management • large, complex product development programs managed as many small (independent yet synchronized) projects • collaborative product development PROCESS: • networked (cross-functional, virtual) project teams • self-organizing TECHNOLOGY: • IT-based knowledge sharing and real-time collaboration	Enables dynamic (real-time) allocation and control of the projects/products and resources based on current needs and competences. Promotes efficient and effective knowledge sharing across functions and disciplines. Requires enterprise-level holistic thinking (culture) and efficient supporting infrastructure (Design Chain Management System).
(Morgan and Liker 2006)	PROCESS: • set-based design approach during the front-end phase (concurrent) • standardized (lower-level) activities and skill sets	Exploring prospective design alternatives early prior to committing to the solution. Standardization enables efficient higher-level flexibility (e.g., staff allocation). Managing the (concurrent) set-based exploration requires firm leading for converging and conflict resolution (Chief Engineer); Standardization should be based on thorough understanding of what can really be standardized and when it is best to deviate from / revise the standards.
(Dehoff and Loehr 2007)	PROCESS: • rapid product release cycle	Provides faster market feedback about new products thus reducing the need for accurate long-range forecasting. Accelerates innovation. Requires comprehensive understanding of the market dynamics. The product development chain must be prepared to leverage the fast feedback.
(Christensen, Kaufman and Shih 2008)	ORGANIZATION: • financial tools for justifying long-term investments to build future competitive capabilities (e.g., with new disruptive technologies) • new strategy processes integrating uncertainty management and financial performance (e.g., discovery-driven planning)	Future competitive capabilities are available when the current capabilities become obsolete (even disruptively). Hard to predict future needs in turbulent environments with rapid changes; Short-term business goals may lead to underinvestments to new capability options. May require completely revisiting the current company strategy assumptions.
(Smith 2008)	• anticipating changes based on continuous customer monitoring • uncertainty-driven specification • encapsulating change-prone product components (with modularity) • experimentation • exploration of the design space (set-based design)	Accommodating change by reducing the cost of making changes throughout the product development process. Enhancing the ability to change, thus encouraging new product innovation.

Publication	Approaches	Potential Benefits Costs and Problems
	• strong (small, co-located) eams • deferred commitments (last responsible moment decisions) • rolling-wave planning, loose-tight planning • intrinsic risk management • reconfigurable high-level processes (with standardized lower-level)	Allowing more flexibility introduces some costs (monetary or otherwise), making it necessary to be able to identify and anticipate the projects and product areas of most likely changes. There may contradicting organizational goals (e.g., co-located teams vs. outsourcing).

Table 27. Definitions of agility (chronological order)

Publication	Definition / Characterization
(Goldman et al. 1995)	Comprehensive response to the business challenges of profiting from rapidly changing, continually fragmenting, global markets for high-quality, high-performance, customer-configured goods and services; Processes that support the creation, production, and distribution of goods and services be centered on the customer-perceived value of products
(Gould 1997)	Ability of an enterprise to thrive in an environment of rapid and unpredictable change
(Gunasekaran 1998)	Capability to survive and prosper in a competitive environment of continuous and unpredictable change by reacting quickly and effectively to changing markets, driven by customer-designed products and services
(Katayama and Bennett 1999)	Cope with demand volatility by allowing changes to be made in an economically viable and timely manner; Abilities for meeting widely varied customer requirements in terms of price, specification, quality, quantity and delivery
(Sharifi and Zhang 1999)	Ability to cope with unexpected changes, to survive unprecedented threats of business environment, and to take advantage of changes as opportunities
(Tsourveloudis et al. 1999)	Ability of an enterprise to operate profitably in a rapidly changing and continuously fragmenting global market environment by producing high-quality, high-performance, customer-configured goods and services
(Yusuf et al. 1999)	Successful exploration of competitive bases (speed, flexibility, innovation proactivity, quality and profitability) through the integration of reconfigurable resources and best practices in a knowledge-rich environment to provide customer-driven products and services in a fast changing market environments
(Christopher 2000)	Ability of an organization to respond rapidly to changes in demand, both in terms of volume and variety; A business-wide capability that embraces organizational structures, information systems, logistics processes, and mindsets
(Vanhanen, Jartti and Kähkönen 2003)	Ability to adapt to changing situations appropriately, quickly and effectively; Early noticing of relevant changes, prompt and effective action planning and reorientation accordingly
(Conboy and Fitzgerald 2004)	Continual readiness of an entity to rapidly or inherently, proactively or reactively, embrace change, through high quality, simplistic, economical components and relationships with its environment
(Dove 2004)	Ability of an organization to adapt proficiently (thrive) in a continuously changing, unpredictable business environment; Characteristic quality of nimbleness, ability to remain viable in a changing environment; Proficiency at change; Having a controlled response ability to deal effectively with things that are beyond control ï whether internal or external, whether opportunity or necessity;
(Caswell and Nigam 2005)	Ability to smooth and dexterous performance in response to the unexpected
(Adeleye and Yusuf 2006)	Systematic response to pressures imposed by the highest levels of market instability and product complexity; Simultaneous emphasis on a wide range of competitive capabilities

Publication	Definition / Characterization
(Ismail et al. 2006)	Ability to respond to, and create new windows of opportunity in a turbulent market environment driven by individual (bespoke) customer requirements cost effectively and rapidly
(Lovén 2006)	Ability of a system to adapt itself to rapid and unexpected changes that are good
(SAS-050 2006)	Ability to be robust (maintaining effectiveness), flexible (employing multiple ways of succeeding), responsive (reacting to changes in a timely way), innovative, resilient (recovering from perturbations), and adaptive (changing work processes and the organization); Characterized by quickness, lightness, ease of movement, nimble;
(Baldrige 2008)	Capacity for rapid change and flexibility; Ability to adapt quickly, flexibly, and effectively to changing requirements (depending on the nature of the organizationôs strategy and markets);

Table 28. Empirical evidence of applying agile software development (chronological order)

Publication	Method(s) Application Area	Results, Findings
(Aoyama 1998a)	Agile Software Process (ASP): • time-boxed incremental delivery, iterative design • concurrent distributed development • people-centered modular process architecture (macroprocesses / microprocesses) with network-centric just-in-time information management large-scale telecommunication software systems	• faster development cycle-times • increased flexibility to plan changes • higher quality • higher productivity • smoother work-load balancing
(Dagnino 2002)	Agile Development in Evolutionary Prototyping Technique (ADEPT): • incorporating selected practices from XP, Scrum, and DSDM methods • incremental and iterative development model • emergent plans and design • feature-driven • intense customer involvement Technology development of ñalphaò systems for new industrial products	In large organizations with established ñtraditionalò development models, there may be small-team projects with evolutionary requirements and volatile technology, favoring agile development practices.
(Vanhanen, Jartti and Kähkönen 2003)	XP and other typical agile practices Telecommunication software product development projects in a large company	• The most widely adopted practice is the measuring of progress by working software code. • low adaptation of agile testing practices

Publication	Method(s) Application Area	Results, Findings
		(possibly due to lack of education) • A technically competent project manager (understanding the product architecture design) is a success factor. • Skilled, stable project teams can successfully work on tacit knowledge. • Focusing on working software (frequent delivery and instant customer feedback) increases the developer satisfaction. • Methodology (XP) not fully and systematically adopted. Process patterns could help.
(Dagnino et al. 2004)	Evolutionary with agile practices (ADEPT) vs. ñtraditionalò incremental (IDM) Technology development projects in an industrial NPD organization (product prototypes)	• degreased documentation time • increased customer satisfaction • ability to incorporate (late) requirements changes with smaller impacts • ability to deliver business value earlier and added value more often • higher risk awareness and proactive risk management reducing the impacts
(Lindvall et al. 2004)	XP applied Pilot projects in large industrial organizations	• more flexibility and speed in implementing change requests (agility) • improvements in customer satisfaction, product quality, productivity, cost • increased developer satisfaction • Even safety-critical software can be developed with agile methods. • Agile methods and practices could influence and be combined with other (traditional) software process approaches (hybrid).
(Khan and Balbo 2005)	XP, FDD, Scrum vs. ñheavyweightò methodologies Web systems development (low, medium, high level categories)	• decrease in costs • increase in quality • shorter delivery schedule • inherently user-centric design • long-term maintenance improved by acceptance test suites • For high-level development (large complex systems with multiple organizations) more in favor of ñheavyweightò methodologies. Customer collaboration is the key advantage of agile development models.
(Fitzgerald Hartnett and Conboy 2006)	XP and Scrum combined Industrial network processor microcode software development	• increase in quality (reduced defect density) • delivery ahead of schedule • better developer motivation • Certain XP, Scrum practices were not used following conscious project-specific considerations. The complementary combination of the XP and Scrum practices is applicable.

Publication	Method(s) Application Area	Results, Findings
(Karlström and Runeson 2006)	XP with traditional gate-based NPD management models Embedded software development for industrial systems	• The product quality increases, the schedule shortens, and the software developers have better controllability. • The intense communication and collaboration of agile development models clarify the customer needs, highlight the problems early, and thus help reducing the project risks. • The interfaces between the agile teams and the surrounding organization and other (non-agile) teams must be clearly defined (mapping the documents and other software artefacts with the governance and quality models, synchronizing the timing differences). A clearly defined customer interface (role) is a key success factor. • The line-management may have to rethink its attitudes towards software project control (e.g., change management, confronting problems early, micro/macro planning needs). • All and all it is feasible and beneficial to combine and augment agile software development with a gate-based management model.
(Keaveney and Conboy 2006)	various Cost estimation practices in agile software projects (case study of four different industrial organizations)	• Frequent, short iterations makes it possible to begin with coarse-grained initial estimates, which are incrementally made more accurate. • experience-based expert estimation preferred to formal estimation models • fixed-price project budgets applicable with some agreed variable dimension • Weak customer/user relationships, lack of past project data, and overreliance to personal memories may make accurate estimation more difficult.
(Syed-Abdullah et al. 2006)	XP vs. design-based methodology Projects of different levels of stability (student projects with external industrial clients)	• Agile teams experience a higher level of well-being (anxiety, depression, enthusiasm) compared to the design-based teams in unpredictable projects: Ability to proceed with only partial requirements, frequent testing and feedback, pair programming, constant communication and other teamwork-supporting practices contribute. • The number of XP practices adopted is associated with a higher level of work related well-being experienced.
(Vodde 2006)	Scrum (primarily)	• People mostly (very) satisfied with the

Publication	Method(s) Application Area	Results, Findings
	Telecommunications systems agile software product development piloting in a large company environment (projects ranging from small to very large ones)	new agile, iterative development mode. • Only a minority would like to return to the old way of working. • A majority see agile, iterative development (very) important in the future.
(Wils et al. 2006)	XP practices application aimed for shortening the development cycle times and coping with changing requirements: • software development phase • target hardware embedding phase • final product certification phase Mission-critical avionics software development (embedded) with regulatory certification requirements	• Certain general agile principles must be refined: The software creates value only when certified in the final product operation context; Much information must be documented explicitly with formal traceability. • The basic software development phase can utilize all agile practices. • The hardware embedding phase introduces additional constraints for coordinating the different functions. • Agile practices are not especially beneficial in the formal certification phase, and some practices (e.g., refactoring) may be even counterproductive.
(Abrahamsson 2007)	New agile method development trials various industrial (embedded) systems and products	• significant cost-savings • high employee satisfaction • higher quality (less defects) Critical success factors (for embedded software development): • appropriate adaptation of the methods and practices • fit-for-purpose tools
(Judy and Krumins-Beens 2007)	Scrum/XP practices new product concept creation	• enhanced organizational innovation (knowledge creation) with agile software development practices
(Sutherland et al. 2007)	Scrum (applied for distributed and outsourced teams) large complex library data management system (platform) development	• high productivity (comparable to collocated teams) • higher quality Critical success factors (for distribution and outsourcing): • set of teams integrated (Integrated Scrum) • shared global build repository • common tracking and reporting tool • daily meetings (geographic transparency) • highly skilled outsourced teams
(Chow and Cao 2008)	XP and Scrum mostly	Critical success factors (in terms of quality, scope, time, cost):

Publication	Method(s) Application Area	Results, Findings
	a survey of some 100 projects	• correct delivery strategy (regular, most important features first) • proper use of agile software engineering techniques (e.g., integration testing) • strong team capability (competence, motivation, knowledgeable managers with adaptive management style, technical training)
(Laanti 2008)	Scrum scaled up very large, complex product development (multiprograms)	• better visibility and continuous steering with hierarchical backlogs • increased productivity and quality perceived by the developers
(Qumer and Henderson-Sellers 2008)	Agile Product-Enhancement Process (APEP): • agile (iterative) practices for the front-end phase (new features) • traditional (incremental) production phase large, complex product development	• Agile practices in new feature development enhance quick and adaptable delivery of business value. • Agile team accountability improves productivity, but requires a different mindset than in the traditional culture. • An executable model (prototype) facilitates smooth transition to the formal software production phase.
(Salo and Abrahamsson 2008)	XP and Scrum a survey of some 35 industrial projects applying (or considering) agile methods for embedded software development	• XP practices applied more often than Scrum (54% vs. 27%) • ~90% of the respondents using XP practices considers them useful (vs. 77% using Scrum practices). • not all practices applied systematically (e.g., TDD in only 18% cases)
(Tabaka and Martens 2008)	Scrum (scaled up) large-scale software product development (programs)	• faster delivery (4.5 times) • fewer defects (11%)
(Välimäki and Kääriäinen 2008)	Scrum (distributed) automation systems development	• clearer visibility of the project status • better management of product features • improved team communication • stronger commitment to the project goals
(Version One 2008)	Scrum (primarily), Scrum/XP hybrid a global survey of various software organizations using agile software methods	• increased productivity • accelerated time-to-market • reduced software defects • reduced costs

Table 29. Typical barriers and impediments of agile software development (chronological order)

Publication	Factors
(Aoyama 1998a)	• May require fundamental rethinking of the entire software development function in the organization including management techniques. • Necessitates efficient tooling to support for instance timely information sharing in particular in dispersed environments. • May require years (even a decade) to nurture (in large organizations).

Publication	Factors
(Constantine 2001)	• Relies on highly skilled, disciplined, and motivated developers. • Requires exceptionally good management and leadership skills. • Iterative development with small increments may not applicable with every product/project type. • scaling tightly coordinated teamwork • integrating advanced user-interface design techniques requiring complex modeling (UCD)
(Chillarege 2002)	• predictability • scalability, distributed development • multiproduct integration
(Turk, France and Rumpe 2002)	• distributed (even) global software development (face-to-face communication vs. documentation) • subcontracted (outsourced) software development (fixed contracts) • building reusable and generalized solutions • managing large development teams (management-in-the-large) • managing large software products (complexity and size requiring systematic architectural design, possibly not feasible to implement in increments, long-lasting maintenance needs) • developing safety-critical software (formal quality control)
(Wallin, Ekdahl, and Larsson 2002)	• mismatches between organizational business decision models and technical software development life-cycle models (timing and necessary information for decision-making points)
(DSDM 2003)	• existing organizational culture (resistance to change) • organizational hierarchy (restrictive management practices) • current roles and responsibilities incompatible with new or changed ones • existing process enforcement mechanisms • outside influences and restrictions (e.g., external customer requiring other ways of working)
(Ronkainen and Abrahamsson 2003)	EMBEDDED SOFTWARE DEVELOPMENT: • system architectural performance requirements, refactoring risks (e.g., timing subtleties) • gradually more rigidity and documented formalism needs • complex (and possibly expensive) target test environments • various different cross-functional stakeholder expectations and coordination
(Vanhanen, Jartti and Kähkönen 2003)	• lack of knowledge about new agile practices (difficult to deploy without relevant education and experience) • ad hoc (emergent) adoption neglecting some key practices • difficulties in implementing fully new methodologies in large organizations
(Glass 2004)	• large projects requiring more formalities than smaller ones • application domain differences (e.g., business applications vs. engineering systems) • differences in project/product criticality levels • project innovation-level variation (unprecedented)
(Kähkönen 2004)	• multiteam collaboration and coordination (in large organizations)
(Lindvall et al. 2004)	• integrating agile software development with existing organizational process definitions and quality systems (e.g., Change Control Boards) • interfacing agile software teams with dependent (non-agile) teams • Agile software development models may require cultural changes in large established organizations with traditionally defined operating models and structures.
(Skowronski 2004)	• Agile software development models may not support well research-oriented (analytic) problem-solving development tasks. • People-centric work practices may not suit to all individuals.

Publication	Factors
(Boehm and Turner 2005)	• integrating agile and ñtraditionalò software development teams following different life-cycle models (in large organizations) • up-front planning, milestone, and predictability requirements of the ñtraditionalò governance models (vs. experimentation and exploration of agile process models) • formal requirements engineering and systems engineering needs • new progress measurement models (activities and cost vs. business value) • collocation and other intensive communication needs of agile teams • new skill needs of software developers and team leaders, project manager roles • legacy systems development and maintenance • external maturity and quality systems assessments
(Ceschi et al. 2005)	• lack of general understanding of agile software development (superficial knowledge) • company internal inertia and customer resistance to adopt totally new concepts • large geographically separated teams
(Henderson-Sellers and Serour 2005)	• lack of practitioner participation in selecting and adapting the projectôs method (method ownership)
(Highsmith 2005)	ORGANIZATION: • governance models not aligned with agile software development teams
(Itkonen et al. 2005)	PROCESS (QA): • Quality Assurance (QA) activities possibly seen unnecessary and unproductive • testing time limited by time-boxing • complete (formal) specifications not necessarily available for testing • independent testers with specific testing skills not necessarily used • relying heavily on automated tests, developers concentrating on constructive QA • few QA practices on the iteration time horizon, hardly any on the release time horizon in the current agile software development models
(Karlström and Runeson 2005)	• market-driven (even global) product development and evolution considerations • supporting the customer interface role at different levels and areas in large organizations (marketing, product management, program management, etc.) • changing ongoing development projects under daily delivery time pressures
(Khan and Balbo 2005)	• risks of low planning, lack of project structure • complexity of large-scale high-level development
(McInerney and Maurer 2005)	• incorporating UCD expertise (e.g., hi-fi prototyping, usability testing)

Publication	Factors
(Nerur et al. 2005)	ORGANIZATION: • culture and mindsets (e.g., decision-making, information filtering, negotiations) • management style (from command-and-control to leading-and-collaboration) • organizational forms (supporting co-operative self-organization) • customer relationships (commitment, trust, proximity, knowledge-sharing) • knowledge management (tacit) • effective teamworking (trust-based social co-operation) • competence and skill levels (above ñaverageò) • reward systems (teamwork result-orientation) PROCESS: • life-cycle model supporting iterative feature-driven development (people-centric vs. process-centric) • appropriate method selection (no unified approach) • managing large-scale projects TECHNOLOGY: • tools and techniques supporting rapid change cycles and tacit knowledge • training new software development skills (e.g., refactoring)
(Turk, France and Rumpe 2005)	ORGANIZATION: • distributed development environments • subcontracting (fixed contracts) • large teams PROCESS: • building reusable artifacts (generalized solutions) • safety-critical software • large, complex software
(Schuh 2005)	ORGANIZATION: • large distributed project teams • imposed waterfall-style planning, control, progress measurements • unavailability of collaborative customers (representative) • ñpretendingò to be agile (either because of masquerading or misconceptions) • not willing to invest in learning and deploying new practices, tools (e.g., test automation) PROCESS: • imposed (heavyweight) processes, tools • fixed project cost and scope • legacy input documentation • not really understanding (or ignoring) the underlying premises and goals PEOPLE: • some individuals not willing to change their ways of working • not establishing a communicative (talkative), collaborative team culture
(Cognizant 2006)	• global distributed (offshore) development • projects with a tight budget and schedule (cost control) • projects that require a high level of process adherence (e.g., regulations) • products in the mature phase of their life-cycles (stable requirements)
(Fitzgerald, Hartnett and Conboy 2006)	• finding a suitable context-specific palette between different general-purpose agile methods and practices (method engineering) • complementing the selected method(s) according to the project-specific needs (e.g., documentation requirements) • taking into account the product technology specialties (e.g., target hardware manufacturing in case of embedded software development)
(Hansson et al. 2006)	• External customers mandate a certain development process model to be followed.

Publication	Factors
(Karlström and Runeson 2006)	• rigid organizational milestone management models (e.g., Stage Gate) • conflicting organizational governance and engineering attitudes and expectations • lack of management support for introducing new methodologies
(Mar 2006b)	PEOPLE: • overly specialized (restricting) skill-sets • lack of ownership (result-orientation) in teams • dysfunctional teams with specialists (e.g., architects) • senior management not clearly supporting agile development PROCESS: • no single customer can be identified (barriers between the business functions and the product development, many different stakeholders). • management mandates to combine agile software development with (incompatible) traditional models and practices (e.g., reporting) • incomplete realization of key practices and roles (e.g., Scrum Master) TECHNOLOGY: • weak SCM and software build systems • neglecting QA issues • ineffective tools, organizational mandatory tool selection ORGANIZATION: • friction between different interdependent teams (large organization) • lack of understanding of governing agile software development (e.g., metrics) • cultural incompatibilities (e.g., fixed job roles, reward systems)
(Pyhäjärvi 2006)	• coordinating multiple component teams in product line development set-ups • long-term planning of multiple (concurrent) releases of multiple products of a product line • leaving room for free-form innovation in a timeboxed cyclic development mode
(Ramesh et al. 2006)	• communication need vs. communication impedance • fixed vs. evolving quality requirements • people vs. process-oriented control • formal vs. informal agreements • lack of team cohesion
(Syed-Abdullah et al. 2006)	• Learning a new methodology during the project may increase the (initial) anxiety level.
(Tate 2006)	• lack of understanding of what is valuable to the customers and why • lack of understanding of the economics of the markets • conflicting multiproject and product portfolio management • Teams donôt pay attention to their software development process.
(Bosch 2007)	• regulatory requirements (medical devices) • waterfall-based organizational tradition
(Highsmith 2007)	• applying an incoherent subset of mutually enforcing agile practices
(Nerur and Balijepally 2007)	• organizational fostering: learning, teamwork, self-organization, empowerment
(Sutherland et al. 2007)	• large, distributed, outsourced teams developing complex systems (e.g., synchronizing the work, effective communication)
(Vodde 2006, 2007)	• lack of resources for supporting all projects adopting new agile development models • Some (legacy) tools do not support well agile iterative development. • Certain existing company-wide metrics may give wrong indicators with agile development models (e.g., progress and performance measurements).
(Wils et al. 2006)	• software project external constraints (e.g., regulatory standards) and dependencies (e.g., target hardware development)

Publication	Factors
(Abrahamsson 2007)	• certification requirements • hardware dependencies • cultural barriers • lack of fit-for-purpose tools
(Coplien 2007)	• not really understanding the problems that agile software development addresses • neglecting necessary planning • overhead of overly granular unit testing • procedural testing as a design tool, leading to hierarchical procedural designs • overemphasis on customer-centricity (vs. task- and context-centricity) • narrow focusing on a single customer (instead of planning many customers / larger markets) • ignoring larger-scale adaptation capabilities
(Gottesman and Takas 2007)	• lack of executive support and lead for organizational change management • immature company culture, geographical diversity • inadequate enterprise-level tools and infrastructure • integrating large-scale organizational functions (e.g., governance, quality management) with agile software development functions • dependencies with non-agile third parties
(IEEE 2007)	PROCESS: • not responding appropriately to customer priorities and needs • production-level quality code not being demonstrated / delivered on a regular basis • supplier not communicating with the customer frequently • lack of reflection and process improvement • limited (or no) commitment to automating tasks to improve productivity and agility • too much documentation or heavyweight processes ORGANIZATION: • lack of commitment to meeting promised deadlines • unwillingness of the customer to share decision-making power with the supplier in the matter of negotiating the builds/iterations • The customer cannot identify relevant stakeholders, who actively participate. • isolating the onsite customer representative from his or her normal social circle • lack of group cohesion • lack of trust in the relationship • Both customer and supplier fail to take responsibility for failures as well as successes.
(Judy and Krumins-Beens 2007)	• relating agile teams to the larger organization (bottom-up agile implementation) • removing impediments arising from outside the software development team
(Leffingwell 2007)	PROCESS: • apparent impediments of the software development methods (small team size, customer integration, collocation, emerging architecture design, informal requirements analysis and documented specifications, culture and physical environment) ORGANIZATION: • process and project management organizations • existing formalized policies and procedures • corporate culture • fixed schedule, fixed functionality mandates • friction between developer departments and user/customer proxy teams • people organized by discipline rather than product line • high degree of distribution

Publication	Factors
(Sidky 2007)	ORGANIZATION: • Adopting agile software development does not add value (inappropriate need for agility). • lack of executive support (buy-in to change to agile software development) • lack of sufficient funding for the adoption PROCESS: • mission/safety-critical software products
(Schoonenderwoert 2007; Schoonenderwoert 2008)	ORGANIZATION: • interconnecting agile software development teams with the rest of the organization (e.g., understanding the production rate of the agile teams) • shifting from the team level up to enterprise-level agility (e.g., lean portfolio management)
(Schwaber 2007)	• organizing large projects and complex product architectures • functional / matrix / project organization conflicts (even ñsilosò) • diverse skill requirements of the developers (more generalists than specialists) • self-management of teams, new roles of project managers • new software development tool requirements (e.g., fast builds, frequent testing) • possibly changes needed outside the actual software development function • lack of buy-in from team members, business customers, line management, or other key stakeholders • large-scale organizational change efforts (ñAgile transformationò)
(Tabaka 2007)	• lack of full stakeholder involvement and consensus (including multiple business customers) • lack of true team self-management authority and decision-making power • weak organizational culture for continuous learning (e.g., ignoring roadblocks)
(Turner 2007)	• traditional hardware-oriented systems engineering processes and mindsets (lack of codevelopment)
(Vilkki 2007)	• contradicting needs and assumptions in organizational process development (common standardized processes vs. team level tailored processes) • lack of human factors considerations (e.g., innovation)
(Aramand 2008)	• not necessarily producing competitive products for the overall marketplace (average customers vs. lead customers) • lack of support for adaptation to the market trends (reviewing similar products)
(Dybå and Dingsøyr 2008)	• complex organizations • complicated integration on agile software teams with the surrounding organization • risk of insufficient software architectural design work
(Glazer et al. 2008)	• weak organizational change management capabilities • (perceived) conflicting organizational process improvement goals and stakeholder interests • mismatching audit requirements
(Hoda, Noble and Marshall 2008)	• lack of customer buy-in (awareness)
(Laanti 2008)	• integrating team-level work (synchronizing requirements and architectural decisions) with the higher levels (product programs) • assigning product owners (in large organizations) • conflicting interests in short- and long-term organizational planning (e.g., resource allocations)
(Nies 2008)	• accelerating the verification and validation feedback loops (test automation)
(Phillips 2008)	• treating agile software development as a óone-size-fits-allô organizational improvement approach

Publication	Factors
(Qumer and Henderson-Sellers 2008)	• full agile transition may take several years (in large organizations) • strategic alignment of business and agility goals (integrating large-scale organizational governance)
(Ramos and Gravendeel 2008)	• weak feedback loops (organizational learning) • lack of trust • addressing wrong problems (unclear goals) • weak Product Owners (Scrum) • failure to adapt the basic methods according to the situational and organizational needs • failure to change the rest of the organization to accommodate agile software teams (enablers)
(Tabaka and Martens 2008)	• lack of business collaboration (product owners) • organizational change barriers (e.g., testing responsibilities) • weak value feedback loops (e.g., product ownerships) • lack of infrastructure investments (e.g., increasing real-time visibility) • incomplete integration with the other parts of the organization (flow rates)
(Välimäki and Kääriäinen 2008)	• overcoming/reducing communication problems • enforcing tool-supported global common processes • knowledge-sharing (transfer) between different sites • ensuring common project visibility (shared view)
(Version One 2008)	• ability to change organizational culture • general resistance to change • personnel with the necessary Agile experience

Table 30. Agility improvement / transition approaches (chronological order)

Publication	Field, Scope	Approach / Principles
(Upton 1994)	Manufacturing (flexibility)	• dimensions (what needs to change and adapt) • time horizon (operational-tactical-strategic) • elements (range, uniformity, mobility)
(Goldman et al. 1995)	Manufacturing / Organizational design	• enriching the customer • cooperating to enhance competitiveness • mastering change and uncertainty • leveraging people and information
(Kotler 1994; Kotler et al. 1996)	Marketing (management)	• market-oriented total-company value-delivery system (strategic network) aimed at profitable customer-satisfaction

Publication	Field, Scope	Approach / Principles
(Paulk 1996)	Software process, organization	Building organizational capabilities by effective SPI: • understanding the business context of SPI • both managerial and technical skills involved in SPI • working relationship with the customer / end-user (open communication, integrity, customer expectations) • converging ñas isò and ñshould beò processes • balancing design-intensive creativity and disciplined work • keeping the processes ñsimpleò in rapidly changing environments • using incremental or evolutionary life cycle models (risk management) • empowering the most crucial asset: people • internal commitment process • developing skills (in particular management training) • management sponsorship and (top-down) coordination for SPI activities
(Dove et al.1996; Dove 2004)	Enterprise	• knowledge management • value propositioning • response ability (system response architecture, change management, culture of reactive and proactive change proficiency)
(Dove 1997)	Enterprise	• business continuation requirements (profitability, adaptability) • transformation strategies (e.g., TQM) • operating strategies (e.g., leanness)
(Haeckel 1999)	Organizational design	• sensing and responding to external signals (open system) • organizational context (purpose and governing principles, adaptive structure) • coordination of capabilities (commitment management)
(Kauppinen 1999)	Enterprise	• vision-based strategic development and management of agility capabilities • integrating strategic, operational, and team levels
(Sharifi and Zhang 1999)	Manufacturing	Total-company agility conceptual model and development methodology: • agility drivers • agility needs (strategic intent to become agile) • agile capabilities, enablers/providers for responsiveness, competency, flexibility, speed • change effects at different levels (current operations, strategy goal changes, new business strategy)
(Holmberg and Mathiassen 2001)	Software process, organization	• agile SPI
(Kirjavainen and Laakso-Manninen 2001)	Organizational design, human resource management	Strategic Competence Management: • knowledge management • intellectual capital management, • competence-based strategic management, • learning organization
(Leishman 2001)	Software process	• adopting appropriate methodologies based on how ñextremeò (rapid, uncertain) the projects really are • accompanying CMM with agile methods
(Atkinson 2002)	Organizational design (military)	• self-synchronization • shared situation awareness

Publication	Field, Scope	Approach / Principles
(Chillarege 2002)	Software process (business)	Process management: • adoptation following the product life-cycle stage • component development of larger, mature products
(Gunasekaran and Yusuf 2002)	Manufacturing	• strategic planning and objectives (virtual enterprise, alliances, core competencies) • market focus (range of markets, products range) • flexible technologies • flexible people
(Benko and McFarlan 2003)	Organizational design (business)	Organizational traits development: • effective collaboration (value networks, ócosystemô) • outside-in view (relationships) • responsiveness, coordination and options (continuous selective sense and respond, alignment) • efficient intra-enterprise operation (enabler)
(DSDM 2003)	Software process	• DSDM incremental development process applied to introducing and/or improving the DSDM software development process itself
(HP 2003)	IT, business processes	1. Profile, prioritize, and plan agility improvements. 2. Architect and build an adaptive enterprise that enables business agility. 3. Manage and measure agility results.
(Anderson 2004)	Software process, organization	• Theory of Constraints • Agile Maturity Model (ñThe Learning Organization Maturity Modelò)
(Coplien and Harrison 2004)	Software process, organization	• situational patterns
(EDUCAUSE 2004)	IT organization (institutional)	• flexible staffing and funding models • collaborative, shared development of IT (possibly Open Source) • flexible IT architectures (allowing cross-functional application integration) • smaller IT projects delivering benefits quickly • rapid new solution deployment (measured in weeks rather than years) • Agile transition is most often incremental (rather than in one radical step), remaking the IT organization.
(Kähkönen 2004)	Software process, organization	• facilitated collaborative cross-team workshops (Communities of Practice)
(Lappo and Andrew 2004)	Software process, organization	• quantifiable benefit/cost assessment of implementing agility (value-stream mapping)
(Levinson 2004)	IT organization	• close collaboration between the business people (users) and the developers • training agile staff • building an agile architecture • agile software development methodologies, selected according to the project-specific needs and constraints • selecting agile vendors • budgeting for agility (businessôs total technology costs, internal profit-loss dynamics, cost transparency)
(Liker 2004)	Enterprise	Lean Thinking applied to product development: • product development process value streaming • long-term continuous improvement • total-quality culture, starting from the top management

Publication	Field, Scope	Approach / Principles
(Poppendieck and Poppendieck 2004; Ward 2007)	Software process, organization	Lean Thinking applied to software production: • eliminating ñwastesò (e.g., unnecessary waiting)
(Prewitt 2004)	IT organization	• a company-wide standard software base • central control and accountability for IT costs • repeatable processes for project management • a flexible software architecture • a standardized development platform • a fluid balance of payroll employees, contract employees, and outside consultants and outsourcers • a flat organizational hierarchy • flexible, short-term provider contracts • flexible, quickly deployable teams (such as SWAT) • an optimal balance of flexible versus fixed IT operating costs
(Aydin et al. 2005)	Software process, organization	Agile method adaptation (DSDM): • forms: static (prescribed tailoring, conceptual) vs. dynamic (during the project, empirical process innovation) • perspectives: engineering vs. socio-organizational • taking into account the project characteristics and situational context factors (Extended Suitability and Risk List) • possibly also modifying the project context (preventive and corrective management measures for enabling the suitability of the method) • coaches
(Boehm 2005)	Software process, organization	• hybrid agile and plan-driven methods • value-based software engineering (VBSE)
(Schuh 2005)	Software process, organization	• making the project environment amenable to agile software teams • gradually adopting individual practices (vs. immediate full method transition), even personally • dynamically discontinuing unsuitable methods and rejecting unnecessary practices • tackling the hardest project problems first (vs. the easiest ones) • keeping a ñlow profileò with new agile teams (initially)
(Henderson-Sellers and Serour 2005)	Software process, organization	• method engineering (both project-level and organizational) based on configurable method fragments (OPF) • incremental adoption with ñsmall-winò pilot projects • continuous method evolution supporting organizational maturity growth and environmental changes
(Levine 2005)	Software process, organization	• agile software development as a part of larger organizational transformations • multidisciplinary (e.g., organizational development, knowledge management, information systems)
(Nerur, Mahapatra and Mangalaraj 2005)	Software process, organization	• management and organization (e.g., management style, organizational culture) • people (customer relationships, competences, teaming) • process (e.g., selecting appropriate people-centric agile methods, scaling large projects) • technology (tools and techniques, new skills)

Publication	Field, Scope	Approach / Principles
(Royce 2005)	Software process	Balancing ranges: • scope management (user needs vs. design assets) • process control (creativity vs. rigor) • progress tracking (experimentation vs. production) • quality control (abstract vs. tangible, testing)
(Börjesson 2006)	NPD software process, organization	• Change Agents
(Dutton and McCabe 2006)	Software (systems) process	Agile / Lean CMMI: • CMMI model mapping to agile/lean development practices with a valuation approach • Disciplined Agility
(Fitzgerald, Hartnett and Conboy 2006)	Software process	• XP and Scrum combined by using a subset of selected practices (rather than a full method) • bottom-up (ñgrassrootsò) adoption strategy • CMM used in parallel (top-down approach).
(Hansson et al. 2006)	Software process	• realizing how much agile-oriented practices are already used (albeit informally) in the organization • combining agile and ñtraditionalò practices
(Ismail et al. 2006)	Manufacturing	Agility Road Map (strategy-driven): • turbulence assessment • agility focus selection (product, process, people, operation, organization) • Agility Capability Indicators • agility tools
(Lovén 2006)	NPD software process, organization	• acquiring new technology competence outside (vs. in-house development) • rapid process change (vs. incremental improvement) • rapid partnership formation (facilitated by external consultants)
(Lyytinen and Rose 2006)	Organization	• ISD process innovation (including agile methods) • different types and needs of agility (exploration and exploitation)
(Oosterhout et al. 2006)	Enterprise, IT	Business Agility: • business networks • change factors requiring agility (external and internal) • Gap analysis: required vs. current agility (difficulty to cope with the required business change) • IT as an enabler (or disabler) of agility
(Overby, Bharadwaj and Sambamurthy 2006)	Enterprise	IT capability as an enabler: • ability to sense IT-based environmental changes • IT-enabled responses • knowledge and processes reach and richness (Digital Options)
(Pikkarainen 2008; Pikkarainen and Mäntyniemi 2006)	Software process, organization	CMMI applied for agile software development: • finding suitable agile practices • understanding the connections between agile software models (practices) and organizational CMMI goals
(Tate 2006)	Software process	• continual refinement of the product and project practices • a working product at all times • continual investment in and emphasis on design • valuing defect prevention over defect detection

Publication	Field, Scope	Approach / Principles
(Vázquez-Bustelo and Avella 2006)	Manufacturing	• human resources • value chain integration • concurrent engineering • advanced technologies • knowledge management
(Vehtari 2006)	Manufacturing	• dynamic manufacturing capabilities • How do the manufacturing capabilities support the product strategy over the life-cycle?
(Vodde 2006, 2007)	NPD software process	• project-specific emergence, no centralized control • no company-wide unified agile methodology enforced • organizational support and ñcoachingò, sharing of experiences and lessons learned (communities)
(Ambler and Kroll 2007)	IT organization, software process	Lean principles applied to software development governance: • mission and principles (e.g., business-driven project porfolio and pipeline management) • organization (HR policies, stakeholder involvement) • development processes (iterative, risk-driven; situational adaptation and continuous improvement) • measurement (real-time project monitoring of value delivered, quality, and cost) • roles and responsibilities (self-organizing teams with appropriate software architecture allocations) • policies and standards (leveraging flexible, reusable, high-value corporate assets)
(Bosch 2007)	Software process, organization	• commitment to organizational change (e.g., overcoming waterfall-based process tradition) • focused and dedicated resources (teams) for establishing the new/changed development model • enough time for settling the model (min. 6 months) • adapting existing organizational assets (e.g., quality standards) when appropriate and useful • educating all organizational stakeholders for the new/changed mode of operation
(Capgemini 2007)	IT, business processes	• investing in improving the capabilities of the IT staff • improving processes that bring business and IT together • embracing a service-oriented business culture
(Dehoff and Loehr 2007)	NPD organizational design	• total focus on the customer value across the product development organization • investing in long-term competence development (people, in particular Chief Engineers) • aligning and coordinating all product development projects/programs for achieving common value goals • emphasizing knowledge processes (creation, learning) • results-driven risk management
(Gottesman and Takas 2007)	NPD organizational design	• enterprise-level (lean) transformation view (vs. team-level focus)
(Highsmith 2007)	Software process, organization development	Agile Transition (key areas): • agile vision (including expected benefits) • organizational roll-out strategy • agile method/practice selection strategy • (methodology) support strategy • integration strategy • software development environments

Publication	Field, Scope	Approach / Principles
(IEEE 2007)	Software process	• customer-developer interfaces (external or internal): collaboration and communication, contracts, requirements, planning, iterative lifecycle, documentation, testing, delivery; • indicators of ineffective application of agile practices
(Judy and Krumins-Beens 2007)	Organization development (knowledge management, innovation)	• bottom-up adoption extended gradually towards the larger product development organization (innovation management, knowledge-creating company) • Software Product Development Manifesto (guiding principles based on core-agile values)
(Leffingwell 2007)	Software process, software engineering management	• Define/Build/Test component teams • two levels of planning and tracking • mastering the iteration • smaller, more frequent releases • concurrent testing • continuous integration • regular reflection and adaptation
(Oiva 2007)	Organization development (strategic agility)	• strategy-focused capability management model • P-CMM extended
(Salo 2007)	Software teams, organization	• SPI process
(Sidky 2007)	Software projects, organization	Agile Adoption Framework: • Agile Measurement Index • 4-Stage Process for Agile Adoption: 1. Identification of discontinuing factors 2. Project-level assessment 3. Organizational readiness assessment 4. Reconciliation • Agile Coach • pre/post-adoption assessments
(Smith 2007)	Organizational design (NPD)	• top-down and bottom-up changes combined • iterative organizational change • situational selection of new method (tool) implementation order and grouping • ambitious yet realizable change goals • demonstrating visible success early (e.g., pilots)
(Turner 2007)	Organizational software process development	• ñagile-friendlyò organizational process assets in other disciplines supporting software development
(Vilkki 2007)	Organizational software process development (large-scale)	• organizational level definition of common concepts, principles, and organization interfaces • team (project) level selection of local implementation tactics and practices • emphasis on interactions, people
(Worley and Lawler 2006)	Organizational design	• investing in talented people proficient at change • learning-oriented reward systems supporting changes • flexible, reconfigurable organization structures with wide exposure to external environment inputs • decentralized decision-making with extensive information sharing and visibility • shared leadership

Publication	Field, Scope	Approach / Principles
(Miers 2007)	Organizational process development	Business Process Management: • developing a spectrum of process capabilities ranging adaptation (flexibility) and standardization (efficiency) • empowering people (knowledge workers) to adjust the processes based on actual situations • continuous, long-term process improvement culture • Business Process Maturity Model
(Aramand 2008)	NPD software process	• Dynamic Design Capabilities: adaptive to changes in markets/needs and creative in superior value delivery for sustainable competitive advantage • involvement of lead users/customers • review of similar product designs • changing, modifying and combining software development techniques and methods during the new product development based on learning
(Doz and Kosonen 2008)	Enterprise	• current state analysis (momentum vs. stagnation/crisis) • strategic goal analysis (repositioning vs. reaffirming) • environment analysis (nature and speed of change) • holistic, systematic capability development • sequencing of complementary capabilities
(Glazer et al. 2008)	Organizational process development (large-scale)	• utilizing the CMMI framework model for overall organizational deployment and continous improvement
(Kanter 2008)	Organizational design (large-scale, even global)	• common shared company values • open-ended standardization allowing quick but coherent local decision-making • governance system based on empowerment (guidance)
(Laanti 2008)	Software projects, organization (large-scale)	• scaling up Scrum-based software development with program management structures and hierarchical backlogs (including practical tool support) Agile Policy: • agreement on organization-wide principles and criteria for implementing the proprietary program management model based on the Agile Manifesto values
(Mäkelä 2008)	Organizational software process development	• strategy-driven software process selection (strategy-first approach) • software engineering capabilities • absorptive capacity • social capital and integration mechanisms
(Phillips 2008)	Organizational software process development	• combining agile software methods with other quality and improvement models (e.g., CMMI)
(Qumer and Henderson-Sellers 2008)	Software process, organization	Agile Software Solution Framework: • selective adoption of agile software development processes and practices based on their business value • method engineering for composing situational process models consisting of agile process fragments Agile Adoption and Improvement Model: • Level 1: speed, flexibility, responsiveness • Level 2: communication-oriented (collaborative) • Level 3: executable artifacts (minimal documentation) • Level 4: people-oriented • Level 5: learning (organization) • Level 6: lean production, keeping agile

Publication	Field, Scope	Approach / Principles
(Schoonenderwoert 2008)	Software projects, organization	• combining agile software development teams with organizational Lean Thinking (e.g., work flow management)
(Salo and Abrahamsson 2008)	Software projects	• likely that embedded software development requires context-specific adaptations of current general-purpose agile method practices (XP, Scrum)
(Tabaka and Martens 2008)	Software projects, organization (large-scale)	Enterprise Agile Adoption: • scaling from (pilot) teams to programs with cross-team, cross-program steering and synchronization practices and tools (ñwhole programò view) • applying Lean principles (pull, flow) • selecting and perfecting agile practices while scaling up (innovation)
(Tanskanen 2008)	Software projects/programs, organization (large-scale)	• constructing a new, hybrid process model for a faster release cycle based on known R&D problems, suggested solutions, and selected agile method practices (XP, Scrum)

ISBN 978-952-248-113-9
ISBN 978-952-248-114-6 (PDF)
ISSN 1795-2239
ISSN 1795-4584 (PDF)