OULU 2012
A 605

ACTA
UNIVERSITATIS OULUENSIS

Maarit Laanti

AGILE METHODS IN LARGE-SCALE SOFTWARE DEVELOPMENT ORGANIZATIONS

APPLICABILITY AND MODEL FOR ADOPTION

UNIVERSITY OF OULU GRADUATE SCHOOL;
UNIVERSITY OF OULU,
FACULTY OF SCIENCE,
DEPARTMENT OF INFORMATION PROCESSING SCIENCE

ACTA UNIVERSITATIS OULUENSIS
A Scientiae Rerum Naturalium 605

MAARIT LAANTI

AGILE METHODS IN LARGE-SCALE SOFTWARE DEVELOPMENT ORGANIZATIONS

Applicability and model for adoption

Academic dissertation to be presented with the assent of the Doctoral Training Committee of Technology and Natural Sciences of the University of Oulu for public defence in Auditorium GO101, Linnanmaa, on 18 January 2013, at 12 noon

UNIVERSITY OF OULU, OULU 2012

Acta Univ. Oul. A 605, 2012

Supervised by
Professor Pekka Abrahamsson
Professor Jouni Similä

Reviewed by
Professor Michele Marchesi
Professor Tommi Mikkonen

ISBN 978-952-62-0033-0 (Paperback)
ISBN 978-952-62-0034-7 (PDF)

ISSN 0355-3191 (Printed)
ISSN 1796-220X (Online)

Cover Design
Raimo Ahonen

JUVENES PRINT
TAMPERE 2012

Laanti, Maarit, Agile Methods in large-scale software development organizations. Applicability and model for adoption
University of Oulu Graduate School; University of Oulu, Faculty of Science, Department of Information Processing Science, P.O. Box 3000, FI-90014 University of Oulu, Finland
Acta Univ. Oul. A 605, 2012
Oulu, Finland

Abstract

Agile methods have proven to be beneficial in small organizations, and there has also been growing interest in using these methods in large organizations. This dissertation analyzes what agility and agile development are and creates a framework for using those methods in large organizations.

The work starts with a Concept Analysis of *Agile Software Development* and agile-in-large. The theoretical part also reflects the necessary background of *Complex Adaptive Systems*, *Lean Thinking*, and *Learning Organizations*. Then a model of an *Agile Enterprise* is defined and a *Framework for Organizational Development* and putting *Agile Methods* into use in large software development organizations is presented.

Large development organizations consist of many levels. It is not enough to use *Agile Methods* on a certain level only, e.g., on the lowest level, but all levels need to change and adapt to the new way of working. Failure to do so leads to several unwanted consequences, which are described. One possible large-scale *Agile Framework* is described and analyzed. The usage of *Agile Methods* on a large scale is validated by quantitative studies.

The level of success of using an *Agile Framework* on a large scale is dictated by how much the same framework for operation is shared within the organization, as partial transformation leads to confusion. But smaller successes can lead into organizational learning. The framework that is proposed can be used to further enhance agility. In this way large-scale agility can be seen as a never-ending series of systematic improvements of the enterprises' *Agile Aspects*.

Keywords: agile methods, lean thinking, organization, process, software development

Laanti, Maarit, Ketterät menetelmät isoissa ohjelmistonkehitysorganisaatioissa – soveltuvuus ja malli käyttöönotolle.
Oulun yliopiston tutkijakoulu; Oulun yliopisto, Luonnontieteellinen tiedekunta, Tietojenkäsittelytieteiden laitos, PL 3000, 90014 Oulun yliopisto
Acta Univ. Oul. A 605, 2012
Oulu

Tiivistelmä

Tämä väitöskirja tarkastelee suurten organisaatioiden tarpeisiin sopivien ketterien prosessimallien mallinnusta ja käyttöä.

Ketterät menetelmät on todettu hyödyllisiksi pienissä ohjelmistoyrityksissä, joten myös isoissa yrityksissä on herännyt kiinnostus ketteriä menetelmiä kohtaan.

Työ alkaa ketterien menetelmien käsiteanalyysillä, ja jatkuu määrittelemällä mitä ketteryys laajassa mittakaavassa on. Teoriaosuus käsittelee taustatiedot kompleksisista sopeutuvista järjestelmistä, lean-ajattelusta ja oppivista organisaatioista tarvittavin osin. Tämän jälkeen määritellään ketterän yrityksen käsite ja esitetään malli laajamittaiselle ketteryydelle.

Suurissa kehitysorganisaatioissa on monta tasoa. Ei riitä, että ketteriä menetelmiä käytetään vain jollakin (yleensä alimmalla) tasolla, vaan kaikkien organisaation tasojen täytyy sopeutua uuteen toimintatapaan. Mikäli näin ei tapahdu, saattaa tuloksena olla joukko ei-toivottavia seurauksia, jotka on myös kuvattu tässä työssä. Työssä on esitetty ja analysoitu mahdollinen malli suuren yrityksen ketteryyden toteuttamiseksi. Ketterien menetelmien käyttö isossa yrityksessä on validoitu kvantitatiivisin menetelmin.

Isoissa yrityksissä ketteristä menetelmistä saatu hyöty on sidottu siihen miten hyvin koko organisaatio pystyy noudattamaan samaa ketterää toiminnan mallia – osittainen toimintatavan muutos johtaa toimintatapojen konflikteihin. Kuitenkin myös osittaiset onnistumiset voivat johtaa organisaation oppimiseen. Esitettyä mallia voidaan käyttää kehitettäessä toimintatapaa entisestään ketterämpään suuntaan. Tällä tavalla suuren organisaation ketteryys voidaan nähdä jatkuvana sarjana systemaattisia toimintatavan parannuksia, joista jokainen johtaa entistä ketterämpään toimintatapaan.

Asiasanat: ketterät menetelmät, lean-ajattelu, ohjelmistonkehitys, organisaatio, prosessit

Dedicated to my mother

Who dedicated her life to education and children
and who is capable, like all really great moms,
of reading the minds of their daughters

Mom, I know where I have inherited my perseverance from…

First they ignore you,
then they laugh at you,
then they fight you,
then you win.
-- Gandhi

Preface

From the year 2000 to 2002 I led a WCDMA base station software project at Nokia Networks. I had recently read Cockburn's (1998) book Surviving Object Oriented Projects, and, encouraged by the book, I changed the project to use incremental development.

That turned out to be a success.

I scheduled requirements with the smallest risk to schedule to the first increment (such as new hardware that was already prototype-ready) and requirements with the highest risk to the last, fifth, increment (such as a new version of 3GPP specifications). That was a smart move; the hardware was released early, but the 3GPP specifications were released one year behind the original schedule! Because of incremental releases we were able to win back some market share from our competitors. The first increment – which my superior at the time claimed would make no sense because "who would like to buy new hardware with the old software that does not support it?" – also turned out to be a smart decision as the new hardware was cheaper to produce and we could sell it with remote software update capabilities. Our customers were also happier because this saved an extra visit to the base station site as they could start to use the new hardware version from the beginning.

My motivation to change the project into incremental mode came from the idea of organizing the work better and being able to succeed with the project. I simply moved the more risky elements to the later increments. Despite some counterarguments, I knew in my heart that testing all releases would not become a bottleneck (some claimed it would but it never did), but the fact that incremental releasing led to better business success was a pleasant surprise.

These experiences motivated me to study and understand better what happened in that base station project. At first Petri Kettunen, who was the Quality Manager in the project, and I called it "Turbulent Project Management". Then in 2004 we understood that what we had studied was actually very close to agility, and decided to study whether *Agile Methods* could be used in very large projects. This thesis would not exist without Petri Kettunen, with whom I wrote my first scientific papers. Petri is a long-distance runner, and a true scientist. Petri, thanks for preaching endurance to me.

My desire to see how 3G technologies impact on phone development made me seek and apply for a job in Nokia's phone R&D organization in January 2005. However, I continued my study of *Agile Methods* during the next two years, until the unit where I worked at the time decided to boost productivity by applying *Agile Methods*. It was then, in January 2007, that I applied for and got the job of leading that transition. I would like to thank Dr. Jan Bosch for his trust in me when hiring me into this position in January 2007.

At roughly the same time I met Professor Pekka Abrahamsson for the first time. I attended one of his many presentations of agile development and I was fascinated to find a researcher with such passion and common sense about productivity and good project leadership that I immediately went to introduce myself and asked him to be the supervisor of my doctoral dissertation-to-be. Since then Professor Pekka Abrahamsson has been most supportive during this long journey. I would especially like to thank him for the many mind-boggling discussions that we have had on *Agile Methods*.

Although I had published one research paper stating that agile development methods need to be implemented organization-wide, permission was given to deploy agility only in the context of Scrum teams. I then wrote an open letter to our R&D management team and stated that improving the performance on the team level is not sufficient alone but that *Agile Methods* also need to be used on a project level in our organization. In the second half of 2007 permission was granted to "Implement Program mode with Agility", an initiative that made the organization argue for half a year whether what we should deploy was the old waterfall-style program mode or some sort of Scrum-of-Scrums.

There are many friends with whom I have been working side-by-side on adoption of the agile ways of working. Juha-Markus Aalto and I have fought many wars together and have had countless discussions on *Agile Methods* and *Lean Thinking*. Juha-Markus also encouraged me to start reading more books on *Lean Product Development*. Thank you. I would also like to thank Vasco Duarte for our many inspiring discussions, as well as for introducing me to the Agile Finland community, where he has been a very active member and is a former chairman. I also have many other colleagues to thank: Rauno Kosamo, Pertti Kontio, Dr. Timo Kaltio, Antti Pesonen, Petri Voltti, Phil Purvis, Timo J. Kokkonen, Petri Taavila, and Hannu Kokko. I would also like to thank my agile friends at Nokia Siemens Networks whose experiences of agile deployment we were able to utilize and get a jump-start for our own efforts. In this respect I would like to thank especially Kati Vilkki, Bas Vodde, and Dr. Kirsi Korhonen, whose broad knowledge of statistical methods I can only admire. There are also many other friends and colleagues with whom I have had memorable conversations and discussions but whom I have failed to name here. I would also like to thank all my colleagues and friends who have challenged me with real-life cases and those who have helped me to solve these cases. Every challenge in our lives helps us grow. Every problem we encounter strengthens our mind and soul. And – as martial arts practitioners state – makes you stronger and tougher.

The agile transformation of Nokia that was kicked off in 2007 is still continuing today. The more we practice *Agile Methods*, the better we understand it and we find new and better ways to work. Likewise, my own understanding of agility and *Agile Methods* continues to evolve.

The need to understand our current status of agile deployment led me to carry out a survey of practitioners in April 2008. I prepared the survey questionnaire together with Dr. Outi Salo, and we also published a research paper together that was based on the results. I would like to thank Outi for willingly contributing her free time to our joint paper and her always cheerful attitude and also her good advice about combining working, a dissertation, and family life. Outi – thanks for being my friend.

The 2008 survey revealed a lot about the organization's understanding of agility and helped the organization to continue on its agile journey. That convinced me that these kinds of surveys can be used as instruments in organizational development, which led me to repeat the survey shortly before the next big organizational change in April 2009.

In all these deployment efforts I have always followed lean principles; the reasons behind the method should be understood and communicated to the people in the transforming organization. The people doing the actual work are the best ones to say how the model should be deployed in their area of discipline. However, any such organizational change clearly needs a vision – and that vision needs to arise from understanding both the operative business environment and how R&D actually works. Neither is possible without long experience in the context.

This work would not have been possible without the support of many people along the way. I would like to thank the Nokia Foundation for its Scholarship Grant. I used the money to fund my study leave and to finish this dissertation. The Grant is well-appreciated recognition both for my work and the work that agile coaches have done to help the adoption of *Agile Methods* at Nokia.

I would also like to thank the LESS2010 conference committee for the recognition for the Best Paper Award granted to me for the best scientific article and presentation at the conference.

I would like to thank Dr. Morten Elvang-Gøransson for hiring me to his current organization in July 2009, and also for supporting me throughout the whole agile transformation and also for his supervision of this thesis work.

It is not common to find a manager with the sensitivity to give you a nudge just when you need it but otherwise give you support, freedom, and trust. I am lucky to have two such managers, Sirkku Raussi and Dieter Grüber. Thanks for your support. Markku Halonen and I have had many discussions about what agile means in the product context – thank you, Markku, for sharing the wide knowledge and first-hand experience you have of product development with me.

I would like to thank my friend and agile coach Martin von Weissenberg for his corrections to the first draft of this dissertation, and helping to improve its contents. We have been working side by side for a few years, transforming the organization, and that transformation, I feel, has also changed us.

This journey would have not been the same without Tomi Tarvainen, who helped me during the darkest moments of agile adoption and also introduced the concept of water scrumming to me and the whole organization. I will be forever thankful; true friends are those who help you when the times are darkest.

There have been many agile gurus that I have been honored to meet during these years and from whom I have learned a lot. Thank you, Dean Leffingwell, Dr. Alistair Cockburn, Ken Rubin, Mary Poppendieck, Tom Poppendieck, Mike Cohn, Craig Larman, Karl Scott, Dr. Jeffrey Liker, and many others for contributing to agile and lean methods. I would also like to thank the many coaches and consultants I have met during these years who have willingly shared their knowledge. I have learned a lot from you all!

Danielle Arvanitis has continued her long-time friendship with me despite the physical distance and the fact that I have brutally abused that friendship and begged her to do quick language checks of my research papers. Somehow, despite the distance, we have managed to do some memorable traveling together. Now I am already looking forward to our next trip!

I would also like to thank my friends for their kind words and encouragement during this journey. Sorry for being absent and occupied with writing. I would like to thank Dr. Anne Kaikkonen for her encouragement, friendship, and the goji berries she gave me for encouragement. This world needs nice and wise persons like you!

Our home cleaning lady Lea Pulkki has done the important job of keeping our house clean while I have been writing this dissertation. Without that support this work would not exist. I would also like to thank my Facebook friends; just as

I have been systematically reporting the progress of this project, equally systematically I have also been given a few lines of encouragement! Thank you!

My special thanks go to my spouse, Dr. Harri Kiljander, who has always challenged me to live boldly and helped me in practice, giving me time for writing by doing more than his share of the so-called “soccer mom duties” and also by proofreading my work. Special thanks also go to my son Kristian, who has helped me by willingly doing his part of the domestic chores in order to support me in writing this thesis, and to my daughter Julia for the inspiration that comes from her music. Remember – only music can be perfect, people never are. A lot of practice helps us to grow to be better performers, closer to perfection, but never there. Keep your passion for music, and keep filling us with astonishment and joy! Those perfect pitches raise our spirits and make us better beings!

Abbreviations and Terminology

This section presents a list of abbreviations, terms, and *Agile Methods* referred to in this dissertation. Terminology is presented only to the extent that is necessary for the interpretation of this thesis; it is suggested that the reader will also become familiar with the research topics with the help of the References section of this thesis.

The list of *Agile Methods* presented here is not comprehensive, but consists only of the most commonly used *Agile Methods*, based on Forrester's and the Version One studies (West and Grant 2010, Version One 2011). [1] A more comprehensive list is presented by Professor Abrahamsson, Oza, and Siponen (2010). The creation of a comprehensive list of *Agile Methods* is challenging, as new *Agile Methods* emerge frequently.

3G	Third-generation cellular system. Different analog technologies like NMT (Nordic Mobile Telephone) are called 1st-generation systems; digital synchronous modulation technologies such as GSM (Global System for Mobile Telecommunications) form the 2nd-generation systems. The third generation supports higher data rates, as well as asynchronous and synchronous schemes with different CDMA-based (Code Division Multiple Access) modulation schemes. (Ojanperä, Prasad 1998)
AgileUP	Agile Unified Process (AUP) is a simplified version of the IBM Rational Unified Process (RUP) developed by Scott Ambler. The AUP applies agile techniques, including test-driven development (TDD), Agile Modeling, agile change management, and database refactoring to improve productivity.[2]
AM	Agile Modeling. A practice-based methodology for the modeling and documentation of software-based systems in a more flexible manner than traditional modeling methods.[3]
ADM	The Agile Data Method developed by Scott Ambler. The Agile Data (AD) method is a collection of techniques and philosophies that will enable IT professionals within your organization to work together effectively when it comes to the data aspects of software-based systems. The Agile Data Method is especially focused on how to

[1] AgileUP, AM, ADM, ASD, BDD, Chrystal, DSDM, FDD, Kanban, Lean, Scrum, Six Sigma, TDD and XP.

[2] Wikipedia, accessed 03 May 2011.

work with databases in a highly collaborational and evolutionary way. (Ambler 2010 B)

Adoption (of *Agile Methods*)

Putting the *Agile Methods* into use on the basis of the user group's own will and decision-making, assuming benefits.

Agile

The terms Agile and *Agile Software Development* and *Agile Methods* are often used interchangeably in this thesis, as the viewpoint is one of applying certain *Agile Methods* such as Scrum or some enterprise-level agile model (commonly known or in-house) to the organization under research. See Section 3.1 for a definition of Agile.

Agile Framework(s)

A framework is understood here as a basic conceptional structure (as of ideas). An Agile Framework is a conceptional structure for how to adopt *Agile Methods* into use in an organization.

Agile Methods

Agile Methods are often called lightweight methods in order to differentiate them from the more heavyweight or traditional waterfall-based methods. Early implementations of lightweight methods include Scrum (Schwaber 2004), Crystal Clear (Cockburn 2001), Extreme Programming (Beck 2004), Adaptive Software Development (Highsmith 1999), Feature-Driven Development, and the Dynamic Systems Development Method (DSDM). These are now typically referred to as agile methodologies, after the *Agile Manifesto* was published in 2001.[3] Methodology is how your organization repeatedly produces and delivers systems and a method is a "systematic procedure", similar to a technique. (Cockburn 2006) Therefore, putting *Agile Methods* as such into use in large software development organizations is often not enough, but a complete Methodology with role definitions, skills, techniques, tools etc. is necessary. In this thesis the term *Agile Methods* has been used less strictly, and also covers the latest additions to *Agile Methods*, methods developed for large-scale agility, and new methods developed on the basis of *Lean Thinking*, namely Lean Software Development and *Kanban* for software development. It should be noted that although many *Agile Methods* consider themselves Methodologies, they might not be sufficient as such for the needs of large software development organizations.

Agile Practices	*Agile Practices* are concrete (daily) actions that implement *Agile Methods*; e.g., continuous integration practices include rules to be followed in daily work, such as “integrate early and integrate often” or “integrate at least daily”. *Agile Practices* may also be very generic, such as “use an on-side customer to refine product requirements”.
Agile Software Development	*Agile software development* is an approach to software development, utilizing *Agile Software Development* methods, such as Scrum and Extreme Programming, that are based on iterative and incremental development, where requirements and solutions evolve through collaboration between self-organizing, cross-functional teams. It promotes adaptive planning, evolutionary development and delivery, a time-boxed iterative approach, and encourages a rapid and flexible response to change. It is a conceptual framework that promotes foreseen interactions throughout the development cycle. The *Agile Manifesto* (Agile Manifesto 2001) introduced the term in 2001.[3] In this dissertation the term is seen as also including agile frameworks developed both in-house and externally for large-scale agile adoption.
Agility	In this dissertation agility is understood as an ability that is achieved with the use of *Agile Methods*. See Section 3.1 for a definition of Agility.
ASD	Adaptive Software Development. Adaptive Software Development is a software development process that grew out of rapid application development work by Jim Highsmith and Sam Bayer. ASD embodies the principle that continuous adaptation of the process to the work at hand is the normal state of affairs. ASD replaces the traditional waterfall cycle with a repeating series of *speculate*, *collaborate*, and *learn* cycles. This dynamic cycle provides for continuous learning and adaptation to the emergent state of the project. The characteristics of an ASD life cycle are that it is mission-focused, feature-based, iterative,

[3] Wikipedia, accessed 10 Oct 2012.

	timeboxed, risk-driven, and change-tolerant. (Wikipedia[3], Highsmith 1999)
BDD	Behavior-Driven Development. BDD is a second-generation, outside-in, pull-based, multiple-stakeholder, multiple-scale, high-automation, agile methodology. It describes a cycle of interactions with well-defined outputs, resulting in the delivery of working, tested software that matters. BDD extends TDD by writing test cases in a natural language that non-programmers can read. This allows the developers to focus on why the code should be created, rather than the technical details.
CAS	*Complex Adaptive Systems*. Adaptive Social Systems are composed of interacting and thoughtful (but perhaps not brilliant) agents. *Complex Systems* challenge the notion that by understanding the behavior of each component part of the system perfectly we will then understand the system as a whole. The study of *Complex Systems* is the interest of various fields, such as biology, economics, physics, and computer science. The study of adaptability focuses on when and why productive systems emerge and how they can persist. (Miller and Page 207)
Crystal	The Crystal methodology is one of the most lightweight, adaptable approaches to software development. Crystal is actually comprised of a family of methodologies (Crystal Clear, Crystal Yellow, Crystal Orange, etc.) whose unique characteristics are driven by several factors such as team size, system criticality, and project priorities. This Crystal family addresses the realization that each project may require a slightly tailored set of policies, practices, and processes in order to meet the project’s unique characteristics. Several of the key tenets of Crystal include teamwork, communication, and simplicity, as well as reflection in order to adjust and improve the process frequently. Like other agile methodologies, Crystal promotes the early, frequent delivery of working software, a high level of user involvement, adaptability, and the removal of bureaucracy or distractions. Alistair Cockburn, the originator of Crystal, has released a book, “Crystal Clear: A Human-Powered Methodology for Small Teams”. (Version One 2011 B)
CMMI	*Capability Maturity Model Integration*; is a process improvement approach whose goal is to help organizations to improve

	performance. CMMI is the successor to the capability maturity model (CMM) or Software CMM. CMMI has been registered at the U.S. Patent and Trademark Office by Carnegie Mellon University. The underlying assumption is that with standardized processes an organization could become more mature (developing from level 1 to level 5) and that level 5 organizations would have improved performance.
COMPSAC	Computer Software and Applications Conference by IEEE; see http://compsac.cs.iastate.edu/
Deployment (of *Agile Methods*)	Putting the agile methods into use on the basis of the management's will and decision-making, assuming benefits. The users may be willing users of *Agile Methods* or they might not be, and benefits deriving from their usage could be assumed or not.
DOI	Declaration of Interdependence. A manifesto of values from agile project leaders from the year 2005. (DOI 2005)
DSDM	Dynamic Systems Development Method. Primarily a software development method originally based upon the Rapid Application Development (RAD) method. In 2007 DSDM became a generic approach to project management and solution delivery. DSDM is an iterative and incremental approach that emphasizes continuous user/customer involvement.[3]
IEEE	Institute of Electrical and Electronics Engineers. The world's largest professional association dedicated to advancing technological innovation and excellence for the benefit of humanity. (IEEE 2011)
ISO 9000	The ISO 9000 family of standards relate to quality management systems and are designed to help organizations ensure they meet the needs of customers and other stakeholders. The standards are published by ISO, the International Organization for Standardization, and are available through national standards bodies. Organizations that would like to get the ISO 9000 certificate are audited. Under the 1994 standard, the auditing process could be adequately addressed by performing "compliance auditing", i.e., 1. Tell me what you do *(describe the business process)*; 2. Show me where it says that *(reference the procedure manuals)*, and 3. Prove that this is what happened *(exhibit evidence in documented records)*.

FDD	Feature-Driven Development. The project starts with the development of the overall model and building the feature list. Each feature is then planned, designed, and built separately.[3]
Kanban	Kanban, also spelled *kamban*, literally meaning "signboard" or "billboard", is a concept related to *Lean Manufacturing*. Kanban is a signaling system based on Kanban cards (or signs) used in *Lean Manufacturing* (or TPS) to schedule what to produce, when to produce it, and how much to produce. In software development Kanban means a physical or virtual whiteboard and tasks with a way to limit the work-in-progress (WIP) in order to create a limited pull system that exposes system operation (or process) problems and stimulates collaboration so that the system gets continuously improved. The Kanban Method is an incremental and evolutionary approach to process change for organizations.[3]
Kaizen	Kaizen, Japanese for "improvement" or "change for the better", refers to philosophies or practices that focus upon the continuous improvement of processes in manufacturing, engineering, and business management.[4]
LAI	Lean Advancement Initiative (formerly known as Lean Aerospace Initiative). The Lean Advancement Initiative at MIT, together with its international Educational Network (EdNet), offers organizational members from industry, government, and academia the newest thinking, products, and tools related to lean enterprise transformation; see http://lean.mit.edu/
Lean, Lean Thinking	*Lean Thinking* is based on the *Toyota Production System*, which can be seen as a set of principles and practices that can increase value by the right pull-based processes, work leveling (even work flow), and the reduction of waste, i.e., unprofitable or unnecessary work, such as faulty products. *Lean Thinking* can be implemented as *Lean Manufacturing*, *Lean Product Development,* or *Lean Software Development*.
MIT	Massachusetts Institute of Technology. A private research university located in Cambridge, Massachusetts. MIT has five schools and one

[4] Wikipedia. Accessed October 2012.

	college, containing a total of 32 academic departments, with a strong emphasis on scientific and technological research.[5]
NPD	New Product Development. A term used to describe the complete process of bringing a new product or service to market. There are two parallel paths involved in the NPD process: one involves the idea generation, product design, and detail engineering; the other involves market research and marketing analysis.[2]
ROI	Return On Investment. Rate or profit of the ratio of money gained or lost of an investment relative to the amount of money invested.
SPI	Software Process Improvement. Software Process Improvement tries to improve the organization's processes or process model. SPI activities are defined, e.g., by ISO/IEC 15504, also known as SPICE (Software Process Improvement and Capability Determination). ISO/IEC 15504 is the reference model for the maturity models (consisting of capability levels, which in turn consist of the process attributes and further consist of generic practices) against which the assessors can place the evidence that they collect during their assessment, so that the assessors can give an overall determination of the organization's capabilities for delivering products (software, systems, and IT services).[2]
Scrum	An agile project management method. Scrum has a specific terminology, and a set of rules, three roles, and three artifacts defining how a team or a project should work. The three roles are *Scrum Master*, *Product Owner,* and *Scrum team member*. The three artifacts are *Product Backlog*, *Sprint Backlog,* and *a burn-down chart*. The rules dictate how team or project planning meetings, team meetings, iterations, reviews, and lessons learned (retrospectives) should be collected and run. (Schwaber 2004) Notice that the word Scrum means both the model and a daily meeting where the team gathers together to discuss progress. In this thesis the former is written with a capital letter (Scrum) to distinguish it from the daily scrum meeting.
Six Sigma	Six Sigma is a business management strategy originally developed by Motorola, USA, in 1986. Six Sigma seeks to improve the quality of process outputs by identifying and removing the causes of defects

[5] Wikipedia. Accessed October 2012.

	(errors) and minimizing variability in manufacturing and business processes. A Six Sigma process is one in which 99.99966% of the products manufactured are statistically expected to be free of defects (3.4 defects per million). (Wikipedia, accessed 18.5.2011)
TDD	Test-Driven Development. A software development process that relies on the repetition of a very short development cycle: first the developer writes a failing automated test case that defines a desired improvement or new function, then produces code to pass that test and finally refactors the new code to acceptable standards.[3] Two types of TDD exist, namely A-TDD for acceptance-test-driven development and U-TDD for unit-test-driven development.
TPS	*Toyota Production System*. The philosophy which organizes manufacturing and logistics at Toyota. The system is a major precursor of the more generic *Lean Manufacturing.* Taiichi Ohno, Shigeo Shingo, and Eiji Toyoda developed the system between 1948 and 1975.[2]
Transformation (in relation to *Agile Methods*)	How the use of *Agile Methods* changes how organizations work and think (Methodologies). An example of this is how traditionally separated requirements management and project management can unite: agile projects could be managed via tracking the number of completed requirements and estimating the amount of functionality needed in a minimum marketable release.
WCDMA	Wideband Code Division Multiple Access, a 3G-modulation technology that is used to pack the signal with the carrier band. Typical data rates are 144 and 384 Kbps with 5-MHz bandwidth, although a 2-Mbps-peak rate can be provided under limited conditions. (Ojanperä, Prasad 1998)
XP	Extreme Programming is a lightweight methodology and one of the first *Agile Methods*. Extreme Programming consists of a set of rules and values. The rules concern the planning, management, design, coding, and testing of software and the values promote simplicity, communication, feedback, respect, and courage. (Wells 2009) XP is also described as a discipline of software development. (Beck 2004)

List of Publications

I Kettunen P & Laanti M (2005) How to Steer an Embedded Software Project: Tactics for Selecting the Software Process Model. Information & Software Technology 47(9): 587–608.

II Kettunen 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.

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

IV Laanti M (2008) Implementing Program Model with Agile Principles in a Large Software Development Organization. COMPSAC '08 Proceedings of the 2008 32nd Annual IEEE International Computer Software and Applications Conference. DOI: 10.1109/COMPSAC.2008.116.

V Laanti M, Salo O & Abrahamsson P (2010) Agile Methods Rapidly Replacing the Traditional Methods at Nokia: A Survey of Opinions on Agile Transformation. Information and Software Technology. DOI: 10.1016/j.infsof.2010.11.010.

VI Laanti M (2010) Agile Transformation Study at Nokia – One Year After. Lean Enterprise Software and Systems. Lecture Notes in Business Information Processing 65(1): 3–19. DOI: 10.1007/978-3-642-16416-3_2.

Table of Contents

1 Introduction

In a modern economy, intelligent products and services, as well as the digitalization of information and content, cause an increasing need for software. While we have witnessed the capabilities of microprocessors growing exponentially, the productivity of software development is only growing linearly. This phenomenon, making software projects larger, longer, and more complex, was first identified in 1968, and is called the Software Crisis. As it is defined, software systems take too long to develop, cost too much, and do not work very well (Feller and Fitzgerald 2000). There have been many attempts to solve this problem, varying from technical solutions such as 3rd-generation programming languages that raise the level of abstraction to various processes and methods, such as object-oriented languages and even open source development.

From the product development point of view, as products become more complex, the project complexity increases, making projects subject to more and more complex problems. Companies try to fight this complexity by hiring experienced managers (personal competence), as well as building knowledge inside the organization (such as building detailed process models). However, we should not underestimate other aspects impacting on success, such as the value of skilled personnel. Several studies indicate a magnitude of difference in performance between the most and least skilled programmers (Blackburn, Scudder, and Wassenhove 1988).

One of the major project management decisions, then, is the selection of the project's software process model. An appropriate agile process model could help in coping with the challenges, and prevent many potential project risks and problems.

The software business has become highly competitive (Highsmith and Cockburn 2001). Thus the ability to implement and test the right features quickly has become a key competence. If changes to software requirements happen at a constant rate and probability, during a long development program there will be more pressure to change the course of the program, i.e., a longer development program will actually need the ability to change course effortlessly even more than shorter ones. Traditional requirements freeze and change management will become too costly and slow. Companies must resolve the gap of productivity with some means, for example with process models that enable them to become more responsive to the inevitable changes.

In the '90s, several new lightweight software process models such as XP, Scrum, and Feature-Driven Development (FDD) emerged. These were later known as *Agile Methods*. The origin of the term “agile” dates back to 2001, when several of the inventors of these new methods met and discussed these new phenomena, as well as the current state of software development at the time (Salo 2007). The essence of these models was the ability to embrace change, but there is also more in these methods, such as the focus on people and communication.

Schwaber *et al.* (2007), among many others, argue that *Agile Methods* yield several benefits. These include reduced time-to-market, increased quality, reduced waste, better predictability, and better morale. Among other suggested benefits is, for instance, an increased ability to respond to dynamic market changes (Lycett, Macredie, Patel, and Paul 2003).

It is notable that the two most widely used *Agile Methods*, namely Extreme Programming and Scrum (Version One[6] 2011 A), both originated from the urge to solve the productivity problem, a.k.a. the Software Crisis.

Scrum (Schwaber and Beedle 2002) can be considered to have been originated at the Easel Corporation in 1993 (Sutherland and Schwaber 2007). It is based on the studies on high-performance teams in the report by Borland, Takeuchi, and Nonaka (1986) in the Harvard Business Review, which suggests that projects using small, cross-functional teams produce the best results (Sutherland and Schwaber 2007). Sutherland’s (2004) experiences at the iRobot Corporation impacted on how these key persons were thinking, i.e., there existed an understanding and experience of Complex Adaptive Systems and the emergent behavior behind Scrum.

Extreme Programming (XP) was born at DaimlerChrysler when Kent Beck tried to find a better way of doing software development around 1990 (Beck 2004). The background of Extreme Programming lay in the need to increase performance: Beck (2004) simply abandoned all practices that did not seem to work and started to do more of those that did seem to prove useful.

Both Scrum and XP were applied first to single projects. When the Patient Keeper project was reported to have 45 workers it was considered large (Sutherland and Schwaber 2007). Sutherland (2005) stated in 2005 that in order to use Scrum organization-wide, the model itself must change, i.e., convert to so-

[6] According to the Version One Survey, Scrum, XP and their hybrids were the most commonly adopted agile methods, with 68% usage, with the rest being shared between various other methods: Custom Hybrid, Lean, Kanban, Scrumban, FDD, Agile UP, Agile Modeling, and DSDM.

called type C Scrum, with overlapping project phases that have tightly integrated builds several times per day, tightly coupled teams, and a reduced amount of administrative overhead.

This thesis is born from the urge to understand how *Agile Software Development* could be done on a large scale. When the work was started in 2004 *Agile Methods* were mostly used in small software teams and small software organizations. Even at that time there were already hints that the application of *Agile Methods* would bring benefits for its users (Barnett 2003, Larman 2003, Barnett 2004). The logical next question was that if we were able to use these methods in a larger context, whether and how (i.e., under which circumstances) we could enjoy these benefits in large-scale software development.

Failing to scale agile beyond team practices to the organizational level may result in a battle between the traditional way of working and the agile way of working, as Borland's case proves: the framework of Scrum does not cover all organizational aspects, and thus traditional business processes cause friction and when the agile processes were inserted into a larger value chain a problem was caused (Maples 2009).

According to Maples's (2009) research, agile development can clearly be seen as a disruptive force: the challenges he listed included e.g.:

- flooding number of requirements,
- scaling and arranging testing,
- challenges for the traditional requirements documents for marketing, and
- making product road-mapping and product releasing processes agile.

These disruptions were so serious that even *Agile Principles* were questioned and many managers reverted to authoritarian behavior. Interestingly, Borland was one of the companies where Scrum was originally developed, before it even had a name. Apparently, at that time, the Borland Quattro Pro project team was able to produce one million lines of C++ code in 31 months with a 4-person staff, that later grew to 8 people, with a defect rate of less than 3%. They also had intensive interaction in daily meetings with project management, product management, developers, documenters, and quality assurance staff. (Sutherland and Schwaber 2007)

The question of merging all doings intensively with each other actually poses a challenge to traditional process thinking. Traditionally, processes are separated from each other, and different processes are executed by different people. There is also a handoff between the different groups of people and processes. *Agile*

Methods call for a more integrated way of working, thus challenging the old process thinking. Maybe that is why Sutherland (2007) describes *Agile Methods* as an organizational pattern rather than a process. Cockburn and Highsmith (2001) go even further by stating that an agile process requires responsive people and organizations. They claim that rigorous processes are designed to standardize people to the organization, while agile processes are designed to capitalize on each individual and each team's unique strengths. With agile every process must be selected, tailored, and adapted to the individuals on a particular project team.[7] So there are more dimensions in the question of how *Agile Methods* could be used on a large scale than just answering the question of what a large-scale agile process is. We must also understand what kind of a balance we need to have with the common process framework and each team tailoring their own processes and if the large-scale development could be avoided in the first place by replacing quantity by quality.

1.1 Research Questions

Järvinen and Järvinen (2004) explain that three phases can often be recognized when defining the *area of research:*

- a set of original questions, i.e., what the researcher would like to know,
- reasoning the research (why this knowledge is important), and
- clarifying questions, i.e., what questions the researcher wants to study in order to create the answers to the original questions.

Any large-scale software development organization may be dazzled by *Agile Methods*. The typical case in many organizations may be that some software teams have tried *Agile Methods* on their own and are excited about the emerging possibilities. But there are some primary questions to answer before any organization can put *Agile Methods* into use on a large scale. The organization may ask how they could use *Agile Methods* in a large-scale development. In order to have sound results, a systematic plan for usage might be needed. How should

[7] Cockburn and Highsmith also state that software engineering and rigorous process adherents use processes to compensate for the lack of competence. According to them, a process can provide a useful framework for groups of individuals to work together, but a process per se cannot overcome a lack of competence, while competence can surely overcome the vagaries of a process — again, "people trump process". And when you look under the cover of agile methods, the emphasis on people and their talent, skill, and knowledge becomes evident. (Cockburn and Highsmith 2001)

an organization understand agile on a large scale? Agile on a large scale might be different to what agile on a team level is. What are the potential benefits of the usage of *Agile Methods* on a large scale? Is it worth trying to pursue these benefits? Are there any reasons why an organization should not seek to apply *Agile Methods*? How could the organization gain the maximum benefits and avoid the pitfalls?

Any real reason why any large-scale organization might want to seek to use *Agile Methods* on a large scale must be linked to benefits to its business. For this it would be essential to understand why *Agile Methods* have gained such popularity on a small scale and in small businesses, and why the small-scale businesses seem to benefit from the usage of these methods. But the business models in small and large enterprises are different. Small companies are by nature more agile, being better able to detect unfulfilled product needs and segments and provide tailored or customized services, while large companies have typically played with quantity and mass markets rather than customized or specialized solutions. This situation is rapidly changing with automated service provision because with electronic distribution channels the final productization cost of software, including the marketing and releasing of new versions, may become only a fragment of the total product cost. This is enabling even large companies to operate with long-tail business models (Osterwalder and Pigneur 2010). An obvious question is how these changes impact on large enterprises, and how the large enterprises can benefit from *Agile Methods* on a large scale.

An enterprise may decide to start the agile deployment from a selected set of agile practices. This may not be a straightforward activity. First, the enterprise should make a selection and decide which practices to put into use. Before even going that far, the enterprise would need to know if it makes any sense to even try the deployment of these practices on a large scale and if these practices will provide the same or similar benefits on a large scale as is achieved when these practices are used on a team level, and on a smaller scale. How well can we understand the relationship between different agile practices and the perceived benefits? How universal are these relations? How could the same benefits be achieved on a large scale if these practices are removed from their original context and applied in a different context? Are there any limitations? Can the benefits be achieved in all circumstances or are there some constraints or rules for the application of these practices before the benefits can be gained? And then, if the practices do scale, do the benefits also scale, meaning that if a certain practice gives some benefit in a small enterprise, is the benefit in a large enterprise proportional to its size? If only the same-sized benefit is achieved in a large enterprise, putting the practice into use may not be worth the effort.

To answer these questions requires an understanding of the complex relations of practices and benefits, the special characteristics and constraints of large-scale environments, and how all these are interlinked.

We may not yet be ready to answer some of these questions. Thus we need to select an appropriately scoped set of questions for this research.

Assuming that using *Agile Methods* on a large scale would provide a benefit compared to traditional large-scale development, we now set the research question as:

> Q. How can *Agile Methods* be put into use in large-scale software development organizations?

Thus our question can be sub-divided into two sub-questions:

> Q1. On what level(s) of the organization should *Agile Methods* be used in order to maximize the benefits to the enterprise?

and

> Q2. How can these methods be applied in large-scale software development organizations?

1.2 The Scope of the Research

This thesis focuses on the applicability of *Agile Methods*, such as Scrum and Extreme Programming, in large-scale software development organizations, and on developing both a model for an *Agile Enterprise* and a model by which the adoption of the above-mentioned methods can happen on a large scale. In this context, a large scale means several thousand software developers. The selected viewpoint is how to create organizational abilities for the application of these selected *Agile Methods* and not specifically the impacts obtained. The most important *Agile Methods* applied were Scrum (Schwaber 2004 and 2007), scaled Scrum, as defined by Leffingwell (2007 and 2011), along with Continuous Integration (Humble and Farley 2011), and elements of Lean Software Development (Poppendieck 2003 and 2010), Agile Portfolio Management (Vähäniitty and Rautiainen 2008), and Practices for Scaling Lean & Agile Development (Larman and Vodde (2009 and 2010). Some teams were also using

Kanban for software development (2010), Extreme Programming (2004), and Agile Estimation and Planning (Cohn 2005). The models applied vary over the research period. The *Framework for Organizational Development* that is presented (Fig. 19) was used in practice several times in different cases and in different organizations to reach agreement on and communicate the changes needed.

Table 1 presents a summary of which topics lie within the scope of this thesis and which do not.

Table 1. Scope of this thesis.

Topic	In Scope	Not in scope
Product development	Software development	Hardware development, embedded software, outsourced software
Methods	*Agile Methods* in general; scaling *Agile Methods* using the principles of *Lean Thinking* and the *Complex Adaptive Systems* Framework	Applying *Lean Thinking* principles as such to product development; research into *Complex Adaptive Systems*
Validation	General principles for scaling *Agile Software Development*	Validation is based on a model that was in use in the subject organization during 2008
Adoption of *Agile Methods*	How to adopt *Agile Methods*	Change management, if agile adoption is a paradigm change or not
Organizational development	Results reflected in change management theories and organizational development theories, link from *Agile Enterprise* model to organizational development theory	Organizational development as such
Viewpoint	Internal to organization, one organization at a time	External stakeholders, ecosystem(s), supply chains, subcontractors

Note that in complex product systems such as mobile phones there are often many profoundly different types of embedded software subsystems, ranging from real-time hardware drivers to sophisticated man-machine interfaces. The most widely recognized software models are pure models in the sense that only the software is under focus. The Models used in embedded software development are often variations on these. However, most existing process models can to some extent be tuned to real-time embedded software projects by taking into account the systems engineering and hardware dependencies.

This thesis is focused on *Agile Software Development* on a large scale. Although Paper III[8] handles the scaling of *Agile Methods* from the entire product development perspective, the model that is presented (Paper IV) and its validation are scoped only around software development, and thus the focus is limited specifically to the software development side of agility, although on a large scale.

It is common to refer to *Lean Thinking* and lean methods when *Agile Methods* are scaled. In this thesis the focus is on *Agile Methods* and references to *Lean Thinking* have been kept to a minimum; that is, when there was a choice as to whether to refer to lean or agile references the latter were selected. Thus questions of how lean product development can be applied in either product development or large-scale software development are beyond the scope of this thesis.

Complex Adaptive Systems have been used in this thesis as a source of inspiration. As such, CAS is not the focus or area of research here, and nor does this dissertation contribute to studies on CAS. CAS and complexity studies, though, provide one relevant framework that can be used for *reflection-in-action*.

Although the principles presented for scaling *Agile Methods* into a large software development organization are general, the validation of these principles is based on and limited to the model that that was in use in 2008 and also presented in Paper IV. In the subject organization the model has been developed further – and the model is still developing – but validating many models would bring only little extra value as the principles for scaling agility can already be validated with one specific model.

The adoption of a new method requires the usage of different change management practices. Even though these are highly interesting topics on their own they fall outside the scope of this thesis. The area of change management is investigated only when it potentially impacts on the results; i.e., from the *Internal Reliability* point of view. That is why this thesis investigates only the use of *Agile Methods* on a large scale, not the adoption itself.

Some people claim that *Agile Software Development* is a paradigm change (Rajlich 2006). Answering this question is beyond the scope of this work.

[8] References to the author's own work are marked like this: e.g., Paper III refers to (Kettunen and Laanti 2008). See the list of publications for clarification.

Although the focus in this research is on how to adopt *Agile Methods* in large software development organizations, familiarity with change management or organizational development theories is not necessary for understanding the framework developed here. Organizational change and organizational development belong to the areas of *Agile Strategy* (Doz and Kosonen 2008), *Business Agility* (Evans 2002, Hugos 2009), and the *Agile Organization* (Atkinson and Moffat 2005). These topics are not comprehensively discussed in this work since the focus is primarily on the adoption of agile on a large scale. However, the results of this research are reflected against organizational development, and change management theories are of relevance from the viewpoint of *Internal Validity* in this dissertation.

The scope of this work is limited to one organization only. This leaves the external environment, software business ecosystem, and supply chain and development of subcontractors beyond the scope of this work.

1.3 Overview of the Thesis

This dissertation is based on six original research papers (cited as Papers I, II, III, IV, V, and VI) that are refereed international journal and conference publications. Each research paper has contributed to an increased understanding of the phenomena under study, i.e., *Agile Software Development* on a large scale. During the time of the preparation of these research papers the researcher was acting as a *Reflective Practitioner* (Schön 1991).

1.4 Motivation and Learning

The motivation to write Paper I, entitled "*How to steer an embedded software project: tactics for selecting the software process model*" was to gain a better understanding of what kinds of problems can be solved with the use of *Agile Methods*, and what kinds of problems are immune to these or other methods. The research question set in this paper was "*How can the project manager*[9] *avoid typical project problems by selecting an appropriate software process model, based on the situational factors of the project?*" The learning in Paper I was

[9] Many people claim that being a project manager is not a scrum or agile role. Although the terms used here (in this position) vary (between teams and across the organizations), the term 'project manager' is well established, unlike, e.g., scrum of scrum of scrum manager (Paper VI).

theory building, i.e., understanding what problems we solve with different process models. The conclusion in Paper I is that *Agile Methods* try to solve the problems detected in traditional processes – but we do not know if *Agile Methods* cause some other, additional new kinds of problems.

The urge to understand different *Agile Methods* better was the reason for writing Paper II, entitled "*How to Steer an Embedded Software Project: Tactics for Selecting Agile Software Process Models*", which compares different *Agile Methods* against each other from the point of view of what the home ground of each of these methods is; i.e., which problems the methods are especially good at solving. The research question set in Paper II was "*How do different agile process models respond to different project problems faced in turbulent environments (if at all)?*" The learning of Paper II was the historical perspective on *Agile Methods*: a continuum exists in developing these methods and the newer ones solve some problems that the older ones are not able to solve, except for Extreme Programming, which really is unique in its approach to team-level problems. Logical deduction was used to build a fuller theoretical background.

A takeaway from Paper II was that teams would really work better with *Agile Methods*. Trying to imagine what kind of an environment these teams would need established the rationale for seeking information on agile on a large scale, from manufacturing and military sources. In Paper III, entitled "*Combining agile software projects and large-scale organizational agility*" a thinking framework was created for this kind of enterprise-wide agility. Two research questions were set in Paper III.

1. How does software project agility relate to NPD enterprise agility?
2. What are the implications for SPI (improving agility) in large-scale NPD organizations?

Paper III extended the agility of the software team or project to the organizational level. It used logical deduction and tested the model in practice in the organization under research. This all led to concept formation.

After writing Paper III the researcher started to deploy *Agile Methods* in practice. Paper IV, entitled "*Implementing Program Model with Agile Principles in a Large Software Development Organization*" is a description of how agility was brought into product development programs, and lists actual experiences of scaling agility. There were no actual research question set in this paper but it rather described the *deployment of the agile methods with the program model.* It presents an interpretation of scaling agile along with observations.

In real life, deploying *Agile Methods* turned out to be a much harder task than had been thought. The motivation to conduct a survey in the subject organization was to seek real information about how people felt and thought about *Agile Methods*. There had been some loud voices in the subject organization which questioned the sense of the whole transformation. The researcher also wanted to search for variables impacting on the deployment. The learning in Paper V, entitled "*Agile Methods Rapidly Replacing the Traditional Methods at Nokia: A Survey of Opinions on Agile Transformation*", was that people really prefer these methods compared to traditional methods and that the actual experience of using these methods impacts on their deployment.

There were two research questions that were set in Paper V, namely these.

1. How does the length of practitioners' experience of using *Agile Methods* affect their opinions of *Agile Methods*?
2. How does the length of practitioners' experience of using traditional methods affect their opinions of *Agile Methods*?

The motivation to repeat the survey after one year was to see which way the opinions were developing in: whether people were still happy using *Agile Methods* or not. The survey results from the previous year were also validated against the new data. The learning from Paper VI, entitled "*Agile Transformation Study at Nokia – One Year After*", was that the opinions about *Agile Methods* were still very positive, and the number of people favoring *Agile Methods* had grown as more people had been using these methods in practice.

The research questions set in Paper VI were these.

1. How does the respondents' length of experience of using *Agile Methods* in practice impact on their attitudes and opinions about *Agile Methods*?
2. How have the respondents' attitudes developed during the one-year follow-up period?

Papers V and VI contributed to a deeper understanding of the problems of using agile on a large scale through observations and measurements.

1.5 This Dissertation

This dissertation aims to build a theory about the deployment of *Agile Methods* on a large scale. It is based on literature research, information and theories available about similar cases, and empirical experience of applying *Agile Methods* on a large scale. It is mostly based on the principles and philosophy of *Agile Methods* and the many discussions that searching for sense-making in real organizational settings.

The researcher has been working as a deployment project manager, as well as a coach in the subject organization. There has been resistance to change in the organization and the motivation and beliefs of this change have been challenged on numerous occasions. The researcher has prepared numerous vision documents that explain the targeted state of agile deployment in the organization and why agile practices are a mechanism for getting there. Understanding and using the *Agile Framework* prepared in Paper III was essential to success: none of the agile practices provide value as such, but are typically *Enablers* for *Means* to reach certain *Goals*. This is very different to a traditional setting, where there is typically just one straightforward link between the action and the outcome.

We start Chapter 2 by analyzing different definitions of *Agile Software Development* and see how those definitions have varied over time and scale. Chapter 2 also builds a theoretical background for the research by providing a background to the currently existing *Agile Enterprise* models, *Complex Adaptive Systems*, *Learning Organizations,* and *Lean Thinking*. Analysis is also performed of what *Aspects* a large-scale *Agile Enterprise* model should have. At the end of Chapter 2 we introduce and discuss different *Agile Frameworks*.

Chapter 3 starts with a Concept Analysis of *Agile Software Development* based on an analysis of the different emphases studied in the previous chapter. It then presents a model of an *Agile Enterprise*, based on the *Aspects* found in the previous chapter. It then continues with another Concept Analysis, this time for an *Agile Enterprise*. Then experiences of the deployment of an *Agile Enterprise* model(s) are discussed.

It discusses the levels we need to apply *Agile Methods* on and presents a model for transformation to an *Agile Enterprise*. The *Agile Enterprise* Transformation model that is presented is reflected against different theories presented in Chapter 2. Chapter 2 ends by studying the transformation via surveys and quantitative analysis.

Chapter 4 goes through the research methods used in this thesis. The work is based on reflection-in-action, where the researcher has been working as a change agent in the organization under study. Concept Analysis, used earlier, in Chapter 3, is also briefly described.

Chapter 5 explains the contributions of the individual research papers to this research. Chapter 6 contains discussion presenting the key findings of this research, and implications for both research and practice. It also discusses the validity of this work from the external and internal points of view and how the research was constructed.

Chapter 7 consists of conclusions that answer each of the research questions. It also presents the limitations of the work and suggestions for future research. The thesis ends with references. The appendices provide a few *Agile Enterprise* models that were too large to include within the text. It also provides a summary table of the emphases in different definitions of *Agile Software Development*.

2 Agile Software Development

This chapter explains the background to the research questions. In order to scale agile, we first need to understand what this ‘agile’ is that we want to scale. *Agile Software Development* is often considered to have been born when the *Agile Manifesto* (Agile Manifesto 2001) was written, although many of the *Agile Methods* were actually introduced earlier. Section 2.1 introduces the *Agile Manifesto*, how it was born, and how it related to some other theoretical frameworks of that period.

The *Agile Manifesto* and *Agile Principles* are typically referred to as the definitions of "agile" and "agility". However, many other definitions exist in the literature. Section 2.2 goes through the other definitions of agile (software development) and compares these definitions to the *Agile Principles*. For each definition we examine where their emphasis is and compare that to the emphases found in the *Agile Principles*. The emphases may change because people have different views or different understandings of the phenomena. The emphases may also change because people learn to appreciate different key aspects because of what they learn when applying *Agile Methods* in practice or when applying them in a different way than originally anticipated, e.g., when applying *Agile Methods* to projects instead of teams. When the *Agile Manifesto* and *Agile Principles* were defined, most of the experience originated from applying *Agile Methods* to small teams and small projects.

Sub-section 2.2.1 examines how the emphases of the *Agile Manifesto* were changed when the *Agile Principles* were reviewed in 2011. At this time, the attendees of the review board not only probably had more experience of using *Agile Methods* but had also been applying those methods in larger projects and organizations. Sub-section 2.2.1 examines other common definitions of *Agile Software Development* that originated roughly around the time when the *Agile Manifesto* was written and what emphases these definitions have, and compares the original *Agile Manifesto* to the emphasis in the original *Agile Principles*. Sub-section 2.2.3 studies the emphases of the *Declaration of Interdependence* (DOI 2005). DOI is an attempt to scale up the use of *Agile Methods* to the project level, and provides an interesting view on how the emphases on the project level may be different than those on the team level. Sub-section 2.2.4 studies recent definitions of *Agile Software Development* and agility in the literature, providing a view of how the understanding of the concept of *Agile Software Development* and agility has evolved over time. Sub-section 2.2.4 ends with a classification of different definitions. Different emphases are used for Concept Analysis in the thesis.

Now, with a better understanding of the different definitions of *Agile Software Development*, the focus can be moved to different existing models of large-scale *Agile Software Development* and agility. Section 2.3 examines these existing models of an *Agile Enterprise*. All of these models have a different viewpoint. Leffingwell's (2007) *Agile Enterprise* Big Picture is a holistic model, although it lacks some integration and releasing aspects. Fowler (2009) presented a model of continuous integration, and another model by Leffingwell (2011) presents the Architecture Kanban Board for managing the development of new architecture solutions. Vähäniitty and Rautiainen (2008) presented a model for product and business planning to be linked with agile development. Hugos (2010 and 2006) presents models for a responsive organization and real-time business dynamics. All these models can be seen as complementing each other.

Sections 2.4 and 2.5 present related conceptual frameworks either for the comparison of different agile models or that can be used as sources of inspiration when creating new such models of large-scale agility. Section 2.4 presents *Complex Adaptive Systems*, as many people state that CAS models were the original source of inspiration for *Agile Methods*. Sub-section 2.4.1 introduces the Cynefin Framework, which is a management framework for decision making that is often referred to in the agile literature. Sub-section 2.4.2 discusses the (general) organizational aspects of CAS and sub-section 2.4.3 discusses to what extent the CAS frameworks can be used to model real software development organizations.

Section 2.5 provides background information on *Lean Thinking*. Even though *Lean Thinking* can also be used as a source of inspiration when creating models of large-scale agility, here it is viewed from the point of view of making changes in an organization. Section 2.5 introduces the LAI model (Nightingale 2002) for making lean improvements. As *Lean Organizations* are one manifestation of *Learning Organizations*, some theory about single and double-loop learning (Argyris and Schön 1978) is presented in Sub-section 2.5.1. Sub-section 2.5.2 compares the similarities and differences between *Complex Adaptive Systems*, *Learning Organizations*, and *Lean Organizations*.

After familiarization with all the existing large-scale agile models has been achieved, the focus can be moved to the adoption of these methods. Section 2.6 discusses the adoption of *Agile Methods* on the basis of the existing research. An organization's first choice is often to adopt some *Agile Practices*, as discussed in Sub-section 2.6.1. If an organization decides to embrace agility further, often the next step is to adopt a specific agile model, which is discussed in Sub-section 2.6.2. The success of the adoption can be measured in various ways; this is discussed in Sub-section 2.6.3.

This thesis defines a model of large-scale *Agile Software Development* and agility. Section 2.7 presents the currently known *Aspects of Agility* found in the literature. These Aspects of Agility provide a theoretical background for creating a new model of large-scale *Agile Software Development*.

Section 2.8 discusses *Agile Frameworks* from the point of view of the adoption of *Agile Methods*. Subsection 2.8.1 presents the Agile Organizing Framework of Vidgen and Wang (2009) and Sub-section 2.8.2 presents a holistic *Agile Framework* created by the author. Later in this thesis these *Agile Frameworks* are used for creating a model of how to put large-scale agile models into use.

2.1 The Agile Manifesto as a Definition of Agile Software Development

Agile Software Development is most typically defined via the "Manifesto for Agile Software Development" (Agile Manifesto 2001, Cockburn 2005). The *Agile Manifesto* states:

> *We are uncovering better ways of developing software by doing it and helping others do it.*
> *Through this work we have come to value:*
>
> *Individuals and interactions over processes and tools*
> *Working software over comprehensive documentation*
> *Customer collaboration over contract negotiation*
> *Responding to change over following a plan*
>
> *That is, while there is value in the items on the right, we value the items on the left more.*

The *Agile Manifesto* was formulated in 2001, when Cockburn invited a group of respected software engineering professionals to ski and to discuss topical issues in software engineering.[10] Cockburn (2005) defines *Agile Methods* as techniques

[10] Based on the author's discussions with Dr. Cockburn in 2008.

that allow a team to track rapid changes in people, technology, and business. He further explains that although the ideas of agile development are based to some extent on the theory of constraints and *Lean Thinking*, the agile way of working was born separately.

The *Agile Manifesto* is further backed up by twelve *Agile Principles*.

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference for the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity – the art of maximizing the amount of work not done – is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly. (Agile Manifesto 2001, Conboy 2004)

The *Agile Manifesto* is a document that is discussed and argued about a lot. The argumentation is partially related to how the *Agile Manifesto* is understood (or not understood) and whether people agree or do not agree about it. Partially because of this, a non-profit organization called the Agile Alliance was formed in 2001 to promote the principles and values listed in the *Agile Manifesto* (Fowler 2001).

Because of the controversy some people have an urge to explain the manifesto and even some agile experts such as Ambler (2011), would like to make some changes or updates to it. The *Agile Manifesto* has also been criticized; for example, Coplien has stated that the Manifesto should talk about *Usable* *software* rather than *Working software* in order better to take the usability aspect into account.[11]

There has also been critiques of the *Agile Manifesto*, stating that it is “too vague” to be used as a basis for scientific work. It has been claimed that it lacks a proper grounding in management theory and philosophy (Conboy and Fitzgerald 2004). It has also been stated that even though there are some methods that are called *Agile Methods* (such as Scrum and Extreme Programming), these methods focus heavily on some of the *Agile Principles*, but not evenly on all of those. As an alternative, Conboy and Fitzgerald (2004) proposed a conceptual framework of *Agile Methods*, explaining agility as flexibility which reflects the robust, proactive, reactive, and temporal dimensions:

> *“the ability of an entity to proactively, reactively, or inherently embrace change in a timely manner, through its internal components and its relationships with its environment.”*

The problem of the framework proposed by Conboy and Fitzgerald is that it covers only one aspect of the *Agile Manifesto*, i.e., the ability to embrace change, while it leaves the simplicity of processes, incremental deliveries, and the people aspect all uncovered and unrecognized.

Conboy and Fitzgerald also compare *Agile Software Development* with *Lean Thinking* and the *Toyota Production System*. Zaninotto was apparently the first one to make this connection in a keynote talk at the XP2002 conference[12] that discussed the tie-ins between *Agile Methods* and *Lean Manufacturing* (Poppendieck 2003, Fowler 2004). However, we know from the testimony of Cockburn (2005) that the *Agile Manifesto* was born separately and independently from *Lean Thinking*, *Lean Manufacturing*, and *Agile Manufacturing*, although Cockburn admitted in his keynote speech at the ICAM 2005 conference that some of the signatories of the *Agile Manifesto* may have known about these methods, and that might have had an impact on why they decided to call the new paradigm

[11] James (Jim) Coplien’s lecture in 2007 with the author attending.

[12] Zaninotto explained that the key aspects of both approaches are that they tackle complexity by reducing the irreversibility in the process. If you can easily change your decisions, this means it's less important to get them right — which makes your life much simpler. The consequence for evolutionary design is that designers need to think about how they can avoid irreversibility in their decisions. Rather than trying to get the right decision now, look for a way to either put off the decision until later (when you'll have more information) or make the decision in such a way that you'll be able to reverse it later on without too much difficulty. (Poppendieck 2003, Fowler 2004) Zaninotto’s explanation clearly includes both the iterative and economic aspects of agility.

"*Agile Software Development*" and not "adaptive software development", which was a name suggested earlier by Highsmith (1999).

2.2 What is Emphasized in Different Agile Definitions

Table 2 contains an analysis of the *Agile Principles* and where the emphasis is put in each principle.

Table 2. Agile Principles and what they emphasize.

	Agile Principle	Emphasis
1	Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.	Customer satisfaction, Continuous delivery, value, early deliveries
2	Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.	Adaptability, competitiveness, customer benefit
3	Deliver working software frequently, from a couple of weeks to a couple of months, with a preference for the shorter timescale.	Frequent deliveries
4	Business people and developers must work together daily throughout the project.	Collaboration
5	Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.	Motivated individuals, good environment, support, trust
6	The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.	Efficiency, communication
7	Working software is the primary measure of progress.	Measure progress via deliverables
8	Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.	Sustainability, people
9	Continuous attention to technical excellence and good design enhances agility.	Focus on technical excellence, Good design as enabler of agility
10	Simplicity – the art of maximizing the amount of work not done – is essential.	Simplicity, optimize work
11	The best architectures, requirements, and designs emerge from self-organizing teams.	Self-organization
12	At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.	Built-in improvement of efficiency and behavior

The first principle places an emphasis on: 1. *customer satisfaction*, 2. *continuous delivery*, 3. *value*, and 4. *early deliveries*. The second principle places an emphasis on 5. *adaptability*, 6. *competitiveness*, and *customer benefit*. *Customer benefit* (or the *customer's competitive advantage*) is close to *customer satisfaction*, so those two can be combined into 1. *customer satisfaction/benefit*.

The third principle places an emphasis on *frequent deliveries*. This is close to *early deliveries*, so these two can be combined into 4. *early/frequent deliveries*. The fourth principle emphasizes 7. *collaboration*.

The fifth principle places an emphasis on 8. *motivated individuals*, 9. *good environment*, 10. *support*, and 11. *trust*. The sixth principle places an emphasis on 12. *efficiency* and 13. *communication*. The seventh principle is the only one on metrics, and places an emphasis on 14. *measure progress via deliverables*. The eighth principle emphasizes 15. *sustainability* and 16. *people*. The ninth principle emphasizes 17. *focus on technical excellence* and 18. *good design as an enabler of agility*.

The tenth principle emphasizes 19. *simplicity* and 20. *optimize work* and the eleventh 21. *self-organization*. The twelfth, and last, principle emphasizes 22. *built-in improvement* of *efficiency and behavior*.

Given that the *Agile Principles* are widely agreed on as defining *Agile Software Development*, it would be interesting to compare any other definitions of *Agile Software Development* and agility against the *Agile Principles*. The analysis of the emphasis of the *Agile Principles* is further used as a basis for such a systematic analysis in this thesis.

2.2.1 Tenth Anniversary of the Manifesto

In February 2011 30 leading agile thinkers convened to discuss the manifesto, focusing on the questions:

- What problems in software or product development have we solved?
- What problems are fundamentally unsolvable?
- What problems can we sensibly address — problems that we can mitigate with money, effort, or innovation?
- What are the problems we don't want to try and solve?

Out of the discussions came a consensus that the agile community should:

- demand technical excellence

– promote individual change and lead organizational change
– oganize knowledge and promote education
– maximize value creation across the entire process (Stevens 2011)

When this list of four items is compared with the emphasis of the twelve *Agile Principles* we can see that a lot of the original emphasis has been lost: there is no mention of *customer satisfaction or benefit*, *competitiveness*, *motivated individuals*, a *good environment*, *support*, *trust*, *communication*, *collaboration*, *sustainability*, *people*, *simplicity*, or *self-organization*. Other areas of emphasis are now seen as being even more important, especially the *demand for technical excellence*.

It should be noted that while the *promotion of individual change* and *leading organizational change* is about *people*, these areas of emphasis are not seen in the same way as was originally the case in the *Agile Principles*: the new area of emphasis focuses on the dynamics of the organization (i.e., change), whereas the *Agile Principles* are "taking what is given"; building around motivated individuals. The former is a view of someone developing the organization; the latter (in the *Agile Principles*) is of a project manager's or entrepreneur's view on selecting the people to be on the team. The viewpoint on Adaptability is different: where the *Agile Principles* focus on the Adaptability of the Requirements, the view in *promoting individual change* and *leading organizational change* is on the adaptability of the organization. Thus it can be seen that this meeting raised *promotion of individual change* and *leading organizational change* as a new area. Similarly, the *organization of knowledge and promotion of education* can also be seen as a new area.

Continuous delivery, value, early or frequent deliveries, efficiency, measuring progress via deliverables, and optimizing the work, as well as *built-in improvement of efficiency and behavior*, can all be seen as being included in the *maximization of value creation across the entire process*.

As a summary, the view (or emphasis) is changed from team focus and self-emergent behavior into a more controlled, organized, and managed direction.

2.2.2 More Definitions of Agile Software Development and Agility

In 1990 the US Congress became concerned about American industrial capability not matching its competitors, especially Japan. That is why a special technology advisory board was set up to study how US industry should be developed. *Agile*

Manufacturing, an *Agile Competitive Environment,* and the *Agile Enterprise* were proposed by this group as answers to the question of how to raise competitiveness. (Preiss, 2005) In 1994 some of the group members, namely Goldman, Naegel, and Preiss, published a book called *Agile Competitors and Virtual Organizations* that was based on this work.

Goldman defines agility as:

> *"a 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. It is dynamic, context-specific, aggressively change-embracing, and growth-oriented. It is not about improving efficiency, cutting costs, or battening down the business hatches to ride out fearsome competitive "storms", it is about succeeding and about winning: about succeeding in emerging competitive arenas, and about winning profits, market share, and customers in the very center of the competitive storms many companies now fear."* (Goldman 1994)

Interestingly, Cockburn (2006) describes this as *"the best description he has found for agility"*.

Kettunen (2009) claims that there is no uniform definition of "*Agile Software Development*" but provides a comprehensive list of different definitions of *Agile Software Development*, in chronological order. A copy of that table is presented in Table 3, but a third column is now added in this dissertation work. The third is an analysis of each agile definition, explaining where the emphasis is in the corresponding definition.

Table 3. Definitions of agile software development (adapted from Kettunen 2009, third column added).

Source	Definition of agile	Emphasis of the corresponding definition
Cockburn 2001	Being effective and maneuverable. Use of light-but-sufficient rules of project behavior and the use of human and communication-oriented rules.	Effective, steerable, rule-based, people, communication
Highsmith 2002	Ability to both create and respond to change in order to profit in a turbulent business environment.	Profitability, steerable, adaptability
Anderson 2003	Ability to expedite.	Speed
Larman 2003	Rapid and flexible response to change.	Speed, flexibility, responsiveness
Schuh 2004	Building software by empowering and trusting people. Acknowledging change as a norm, and promoting constant feedback. Producing more valuable functionality faster.	People, empowerment, change, feedback, value, speed
Lyytinen 2006	Discovery and adoption of multiple types of Information Systems Development innovations through garnering and utilizing agile sensing and response capabilities.	Delivery, innovations, responsiveness
Subramaniam 2005	Uses feedback to make constant adjustments in a highly collaborative environment.	Feedback, adaptability, collaboration
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.	Iterative, incremental, self-organizing, less process-driven, collaborative, cost-conscious, speed, customer-driven
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).	Iterative, responsive
Wikipedia 2007	Conceptual framework for software engineering that promotes development iterations throughout the life-cycle of the project.	Iterative, conceptual framework

Cockburn's (2001) definition of *Agile Software Development* places the emphasis in such a way that agile is defined as 1) *effective*, 2) *steerable*, 3) *rule-based*, 4) (about) *people*, and 5) *communication*. Only the latter two are the same as can be found in the *Agile Principles*; see Table 3.

Anderson (2003) places the emphasis only on 1) *speed*. Besides *speed*, Larman (2003) also places the emphasis on 2) *flexibility* and 3) *responsiveness*. Schuh (2004) also places the emphasis on 1) *speed*, but also on 2) *people*, 3) *empowerment*, 4) *change*, 5) *feedback*, and 6) *value*.

Lyytinen (2006) places the emphasis on 1) *feedback*, 2) *adaptability*, and 3) *collaboration*. Subramaniam (2005) emphasizes 1) *feedback*, 2) *adaptability*, and 3) *collaboration*. Ambler (2007) states that agile is 1) *iterative*, 2) *incremental*, 3) *self-organizing*, 4) *less process-driven*, 5) *collaborative*, 6) *cost-conscious*, 7) (about) *speed*, and 8) *customer-driven*.

IEEE (2007) states that agile is 1) *iterative* and 2) *responsive*, whereas Wikipedia (2007) states that agile is 1) *iterative* and 2) a *conceptual framework*.

Those definitions that name only one or two points of emphasis can be considered narrow. The other definitions partially cover the same points of emphasis as the *Agile Principles* (see Table 2) but use slightly different terms or viewpoints on agility. When Cockburn states that agile is about *communication* – which is also one point of emphasis in the *Agile Principles* – Ambler states it is about *collaboration*. These kinds of nuances might seem irrelevant, but they can cause confusion in a large organization when *Agile Methods* are being used. In practice, people will come and ask what it is that the organization is aiming to do – and is *collaboration* better than *communication*?

Obviously, the length of the definition is not the only measure of its goodness, although the longer the definition is, the more points that are emphasized it can cover. When the *Agile Principles* are compared to Goldman's definition, it can be seen that there is no single point that is emphasized that can be stated as being the same. Instead of customer satisfaction or benefit Goldman talks about "customer-configured goods and services". Even the contrary is true: while one aspect found in the *Agile Principles* is "*efficiency*", Goldman (1994) states that being agile "is **not** about efficiency".

One explanation of this considerable difference in definitions is that Goldman's definition of agility is performed from an organizational and business perspective, whereas the *Agile Principles* clearly focus on process, software development, and people. Thus the perspective from which *Agile Software Development* and agility are defined — business, organization, process, software development project, or team level, just to mention some possible perspectives —

will impact on the definition and how *Agile Software Development* and agility are seen.

2.2.3 Agile Software Development on the Project Level

In order to understand how we can scale *Agile Software Development* and achieve agility we can take a look at the further definitions that are available, especially from sources that look into agility from a perspective that is wider than that of just one team.

One of these attempts to define agility in a larger context took place in 2005, when Cockburn gathered a group of project managers together to discuss agility within a project context and from a project management viewpoint. This gathering resulted in the Declaration of Interdependence (DOI), which links people, projects, and value with agile and adaptive approaches. The Declaration of Interdependence states: (DOI 2005)

We are a community of project leaders that are highly successful at delivering results. To achieve these results:

- We increase return on investment by making continuous flow of value our focus.
- We deliver reliable results by engaging customers in frequent interactions and shared ownership.
- We expect uncertainty and manage for it through iterations, anticipation, and adaptation.
- We unleash creativity and innovation by recognizing that individuals are the ultimate source of value, and creating an environment where they can make a difference.
- We boost performance through group accountability for results and shared responsibility for team effectiveness.
- We improve effectiveness and reliability through situationally specific strategies, processes, and practices.

Analyzing the Declaration of Independence gives hints as to how the emphasis of *Agile Methods* may change when the viewpoint is changed from a team perspective to a project perspective. Table 4 contains an analysis of the DOI and states the emphasis of each statement.

Table 4. Emphasis in agility definition in the Declaration of Interdependence.

DOI statement		Emphasis
1	We increase return on investment by making continuous flow of value our focus.	Maximizing return on investment, flow of value
2	We deliver reliable results by engaging customers in frequent interactions and shared ownership.	Customer engagement, delivery accuracy, collaboration
3	We expect uncertainty and manage for it through iterations, anticipation, and adaptation.	adaptability, proactivity, anticipation
4	We unleash creativity and innovation by recognizing that individuals are the ultimate source of value, and creating an environment where they can make a difference.	Innovativeness, people, environment
5	We boost performance through group accountability for results and shared responsibility for team effectiveness.	Shared responsibility, effectiveness
6	We improve effectiveness and reliability through situationally specific strategies, processes, and practices.	Reliability, situationality

The first statement in the DOI places the emphasis on 1) *maximizing ROI* and 2) *flow of value*. The second statement in the DOI emphasizes 3) *customer engagement*, 4) *delivery accuracy*, and 5) *collaboration*. The third statement emphasizes 6) *adaptability*, 7) *proactivity*, and 8) *anticipation*. The fourth statement emphasizes 9) *innovativeness*, 10) *people*, and 11) *environment*. The fifth principle emphasizes 12) *shared responsibility* and 13) *effectiveness*. The sixth principle emphasizes 14) *reliability* and 15) *situationality*.

When Table 4 is compared with Table 2 and the emphasis in the *Agile Principles* it can be seen that five of these aspects are the same (*value*, *collaboration*, *adaptability*, *people*, and *environment*) but the majority are new (*maximizing ROI*, *delivery accuracy*, *proactivity*, *anticipation*, *innovativeness*, *reliability*, and *situationality*) or the viewpoint is slightly different (*customer engagement* rather than *customer satisfaction*, *shared responsibility* rather than *self-organization*).

This raises the question of whether these differences can be explained just with a different viewpoint, or if the understanding of *Agile Methods* has evolved, as the DOI was written four years after the *Agile Manifesto*. New points of emphasis (*maximizing ROI*, *delivery accuracy*, *proactivity*, *anticipation*, *innovativeness*, *reliability*, and *situationality*) can also be considered as requirements from the project management level for *Agile Methods*.

2.2.4 Recent Definitions of Agile Software Development and Agility

Research papers seem to avoid defining *Agile Software Development* and agility, or define it via references to a few existing sources, e.g., the definition of Conboy and Fitzgerald (2004), or define agility via the methods researched. Assuming that the understanding of *Agile Methods* increases as people practice these methods more, it is interesting to see how the definitions have evolved and how the emphasis differs from the *Agile Manifesto* and *Agile Principles*. Since research provides little help in this respect, the latest agile literature is researched. Advancing *Agile Methods* seems to be primarily driven by industry practitioners, not by academic researchers.

Table 5 presents the newest definitions of *Agile Software Development* and the emphases of those definitions.

Table 5. Newest definitions of agile software development with analysis of emphases in those definitions.

Source	Definition	Emphasis
Wikipedia 2012[13]	Group of software development methodologies based on iterative and incremental development, where requirements and solutions evolve through collaboration between self-organizing, cross-functional teams. The *Agile Manifesto* introduced the term in 2001.	Group of software development methodologies, iterative, incremental, self-organizing, cross-functional, evolutionary
Poppendieck 2010	Aspects … that reside in the project management frame are likely to change over time and are therefore a matter of choice. On the other hand, agile development has extended many practices from the system development frame. These technical practices have withstood the test of time and they will continue to be highly effective far into the future.	System development frame, technical practices, effectiveness
Larman & Vodde 2009	Be agile …. the Merriam-Webster dictionary defines agile as a ready ability to move with quick easy grace …. Agile **does not mean delivering faster**. Agile **does not mean fewer defects or higher quality**. Agile **does not mean higher productivity**. Agile means agile — the ability to move with quick easy grace, to be nimble and adaptable. To embrace change and become masters of change — to complete through adaptability by being able to change faster than your competition can.	Ability to change more quickly and easily than the competition

[13] Accessed in October 2012.

Source	Definition	Emphasis
Larman & Vodde 2010	Not defined *per se;* refers to system development and lists different *Agile Practices* “to try”. Looks at agility from the “systems improvement” point of view. House of Lean, see Appendix A.	Systems improvement using *Agile Practices* (House of Lean)
Appelo 2011	Agility is about staying successful in ever-changing environments (page 376). Agile has never been some specific set of practices (page 377) but rather has its roots in complexity theory (page 11) — and solutions depend on the problem’s context.	Agile is context-specific, having its roots in complexity theory
Leffingwell 2011	(*Agile Software Development* defined via the *Agile Manifesto*; also House of Lean — see Appendix B; agile also defined context-dependent several times, e.g., page 432, enterprise agility; team agility; focus on product development flow too; also defined as what agile is not about) Adaptive (agile) processes … assumed that — with the right development tools and practices — it was simply more effective to write the code quickly, have it evaluated by customers in actual use, be “wrong” (if necessary), and quickly refactor it than it was to try to anticipate and document the requirements up front.	Agile seen as context-specific processes including a set of practices bringing business benefits
Cohn 2009	(Asks the reader to refer to earlier books for definition.) The opposites of agile are sequential, traditional, and even non-agile. Successful agile teams produce higher-quality software that better meets user needs more quickly and at lower cost than traditional teams … organizations that take becoming agile seriously by adopting a process … are seeing significant gains in productivity with corresponding decreases in cost. They are able to bring products to market much faster and with a greater degree of customer satisfaction. They experience better visibility into the development process, leading to greater predictability. And for them, out-of-control, will-it-ever-be-done projects have become a thing of the past.	Non-sequential, non-traditional, high-quality, speed, meets user needs, low-cost, process, productivity, visibility, predictability, in-control

Wikipedia (2012) defines agile as 1) *a group of software development methodologies*, 2) *iterative*, 3) *incremental*, 4) *self-organizing*, 5) *cross-functional*, and 6) *evolutionary*. Poppendieck (2010) defines agile as 1) *a system development frame*, 2) *technical practices*, and 3) *effectiveness*. Larman and Vodde (2009) place the emphasis in agility as 1) *ability to change quicker and easier than the* *competition*. In 2010 Larman and Vodde define agile via House of Lean, i.e., as *systems improvement* using *Agile Practices*. Appelo states that Agile is context-specific, having its roots in complexity theory. Leffingwell (2011) sees Agile as context-specific processes including a set of practices bringing business benefits. Cohn (2009) does not give a definition of *Agile Software Development* but asks the reader to refer to his earlier books. Here, agile is seen as 1) *non-sequential*, 2) *non-traditional*, 3) *high-quality*, 4) *speed*, 5) *meets user needs*, 6) *low-cost*, 7) *process*, 8) *productivity*, 9) *visibility*, 10) *predictability*, and 11) *in-control*.

From Table 5 it can be seen that even these recognized gurus do not have a unified vision of what *Agile Software Development* is and are struggling with the definition. If the Wikipedia definition is omitted as practical (and thus as an outlier), the rest of the definitions can be categorized as follows:

1. *A null definition* (*agile is agile*[14]; differing from all the previous and succeeding definitions)

   Larman's 2009 recursive definition that *agile is agile* and a list of negative statements about what it is *not*, which contradicts almost all of the succeeding definitions of *Agile Software Development* and agility.

2. *Traditional*

   Cohn's 2009 definition as a negative statement of *what it is not* (Cohn 2009; Leffingwell 2011) also aligned himself with the previously discussed (positive) aspects.

3. *Set of practices*

   Agile as a set of practices that should be used together with *Systems Thinking* (Senge 2006, Larman 2010, Poppendieck 2010).

4. *Context-specific*

   Agile as a context-specific set of processes and practices (Leffingwell 2011) which may vary as the problem that we are solving varies, rooted in complexity theories (Appelo 2011).

[14] The sentence "*A rose is a rose is a rose*" is probably the most famous quotation by Gertrude Stein, often interpreted as meaning "things are what they are," a statement of the law of identity, "A is A" (Wikipedia, accessed 26 April 2011) However, this kind of "agile is agile" definition leaves very little room for the *purpose* – why, then. should we be agile and what would the benefit be? This type of definition is simply odd, as the reason for agility must – first and foremost – be a business benefit!

Most of the newest definitions of *Agile Software Development* have stopped talking about effectiveness, but describe agile rather as a set of practices that you can try when doing systems improvement. But agile must be more than just a set of practices that are applied: while the first attempts at putting agile into use in large organizations were about trying out some practices (Kähkönen 2004), there was a lot of complaining that agile must mean a lot more than some teams (or even some individuals) following some *Agile Practices* only: for example, you could well do pair coding and still follow a traditional process.[15]

An organization needs to know when it has become agile. The agile literature defines no point after which an organization has adopted enough practices to be called agile. Rather, the literature presents various operational models but little guidance as how to get to that dream state. It has also been stated that agility is rather the mindset with which to approach the problems at hand, but an organization cannot simply change to an agile mode by simply stating that it has done so. A large organization would need something it can develop, deploy, and measure.

Appelo (2011) states that most people have got agility wrong, because they have not understood that agile originates from complexity theories — or rather that they do not understand complexity theories. Thus any simplistic, linear model (or attempt to create one) is bound to fail, and we should rather focus on the adaptability, not the predictability. In fact, conflict is a natural aspect of *Complex Systems* and a pre-requisite for creativity and innovation (Appelo, 2011). This provides an additional challenge for developing large-scale agile models.

A comparison matrix for different definitions of *Agile Software Development* is presented in Appendix C.

2.3 Agile Enterprise Models

Several authors have presented different models of an *Agile Enterprise*. Leffingwell (2007 and 2011) presented The *Agile Enterprise* Big Picture; see Fig. 1.

[15] Interestingly, *Extreme Programming* is defined as a set of practices, each of which you need to follow.

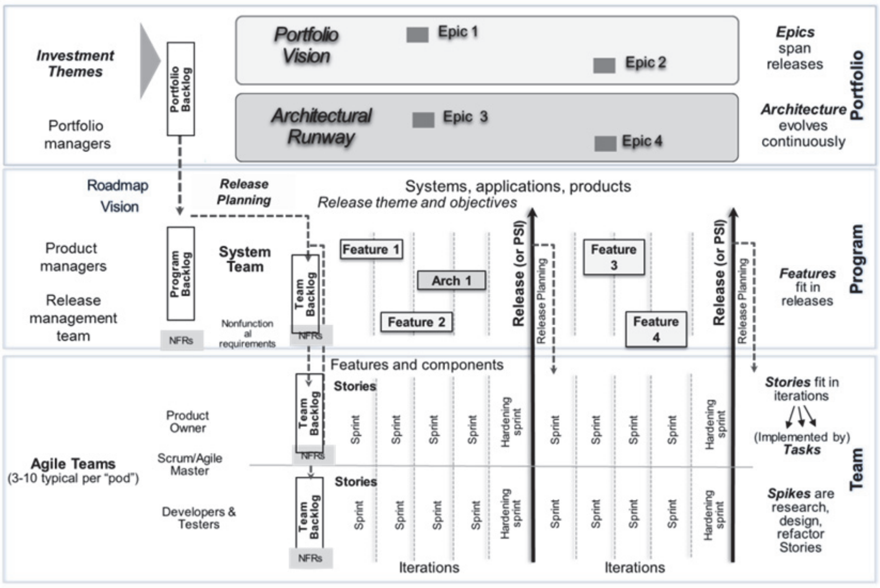


Fig. 1. The Agile Enterprise Big Picture. (Leffingwell 2007 and 2011).

Leffingwell's model of an *Agile Enterprise* is holistic, but not sufficient for making the whole organization agile. The picture is a view on how to plan and release software but it is only one view (the "requirements view"). It is missing, e.g., the release chain. Fig. 2 shows an example of what this kind of continuous integration view could look like. What is different in this diagram, compared to traditional integration, is that there is more merging happening; each change is immediately merged with all the other branches and the mainline. Leffingwell (2007), though, lists continuous integration as one *Agile Practice* that scales up. In fact, delivery after each increment and iteration is not possible if the integration takes longer or even the same amount of time as the defined length of the iteration or increment.

Depending on the subject organization, there might also be other needs in the organization that Leffingwell's model does not cover. The views that are missing may be, e.g., how to combine hardware development, if the subject organization is developing embedded devices. Neither does Leffingwell's model cover combining user interface design with the development. Nor does it say how to do localization and internationalization, or how to handle product variants.

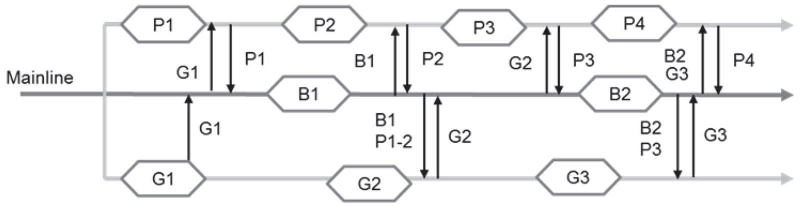


Fig. 2. Continuous Integration Diagram. (Fowler 2009)[16].

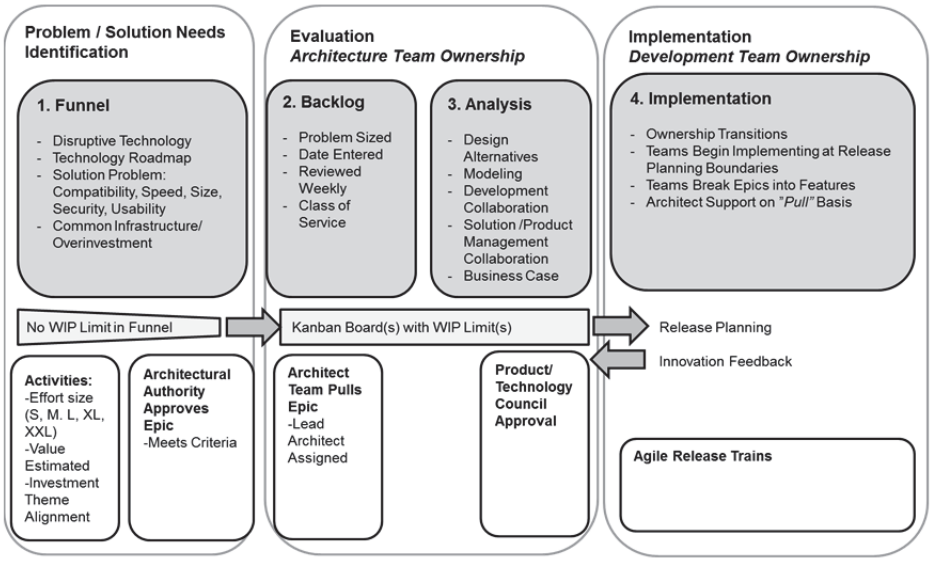
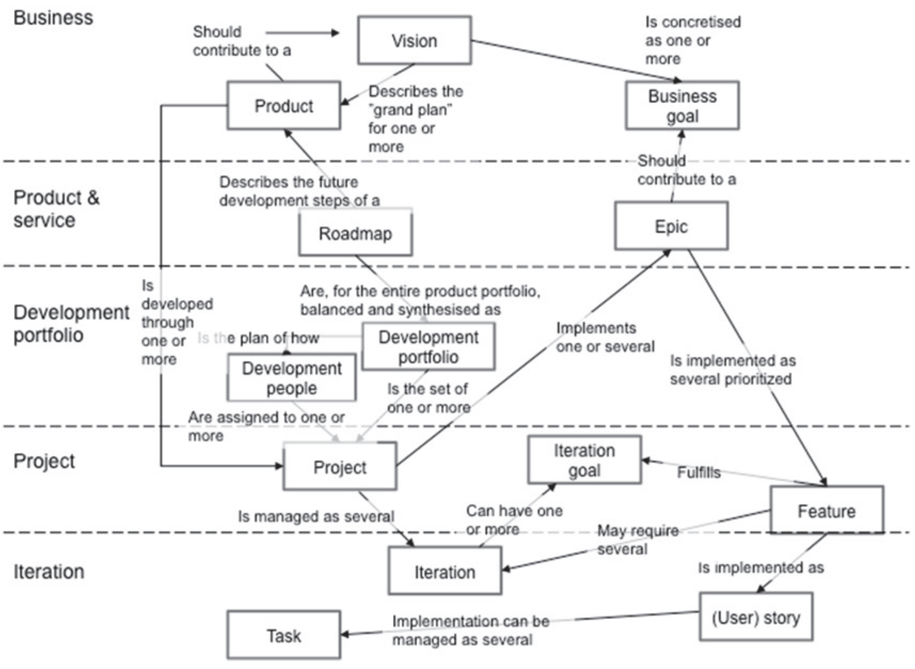

Problem / Solution Needs Identification
Evaluation
Architecture Team Ownership
Implementation
Development Team Ownership
1. Funnel
- Disruptive Technology
- Technology Roadmap
- Solution Problem: Compatibility, Speed, Size, Security, Usability
- Common Infrastructure/ Overinvestment
2. Backlog
- Problem Sized
- Date Entered
- Reviewed Weekly
- Class of Service
3. Analysis
- Design Alternatives
- Modeling
- Development Collaboration
- Solution /Product Management Collaboration
- Business Case
4. Implementation
- Ownership Transitions
- Teams Begin Implementing at Release Planning Boundaries
- Teams Break Epics into Features
- Architect Support on "Pull" Basis
No WIP Limit in Funnel
Kanban Board(s) with WIP Limit(s)
Release Planning
Innovation Feedback
Activities: -Effort size (S, M. L, XL, XXL) -Value Estimated -Investment Theme Alignment
Architectural Authority Approves Epic -Meets Criteria
Architect Team Pulls Epic -Lead Architect Assigned
Product/ Technology Council Approval
Agile Release Trains

Fig. 3. Architectural Epic Kanban Board. (Leffingwell 2011).

[16] In Figure 2.2 the first developer makes four contributions to the code, marked P1-P4. The contributions of the second developer are marked G1-G3. B1 and B2 are builds done on the mainline. The arrows represent the replication of changes to development branches that *Distributed Version Control* (Humble and Farley 2011) tools handle automatically.



Business
Should contribute to a
Vision
Is concretised as one or more
Product
Describes the "grand plan" for one or more
Business goal
Product & service
Describes the future development steps of a
Should contribute to a
Roadmap
Epic
Development portfolio
Is developed through one or more
Are, for the entire product portfolio, balanced and synthesised as
Is the plan of how
Development portfolio
Implements one or several
Development people
Is implemented as several prioritized
Is the set of one or more
Are assigned to one or more
Project
Project
Iteration goal
Fulfills
Is managed as several
Can have one or more
May require several
Feature
Iteration
Iteration
Is implemented as
Task
Implementation can be managed as several
(User) story

Fig. 4. Linking Product and Business Planning with Agile Development. Adapted from Vähäniitty and Rautiainen (2008).

Often we can also hear a complaint that one should understand better what is actually happening inside the *Architectural Runway* and *Portfolio Management* boxes. Leffingwell has actually submitted an additional model called the *Epic Architecture Kanban Board*; see Fig. 3. Vähäniitty and Rautiainen (2008) studied the latter, i.e., *Portfolio Management*, and presented the model in Fig. 4. In this model a *product* is a generic term used for an offering that may be a piece of software or service that the company is developing. *Products* should contribute to the *vision,* which describes the "grand plan" for one or more *products* and is concretized as one or more business goals.

Leffingwell's model can be compared to the *Sashimi Model*, presented by Takeuchi and Nonaka (1986). Fig. 5 presents the sequential vs. overlapping phases of development. Here, Type C represents the *Sashimi Model*. Leffingwell's model is closer to Type B development.

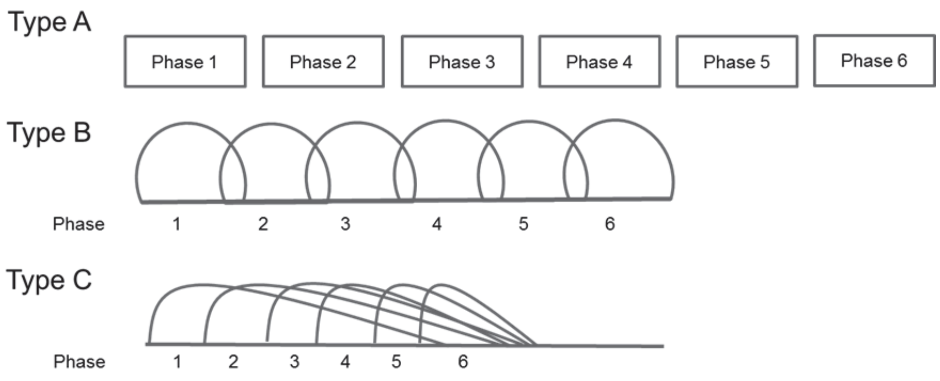


Fig. 5. Sequential (Type A) vs. Overlapping (Type B and C) Phases of Development. (Takeuchi and Nonaka 1986).

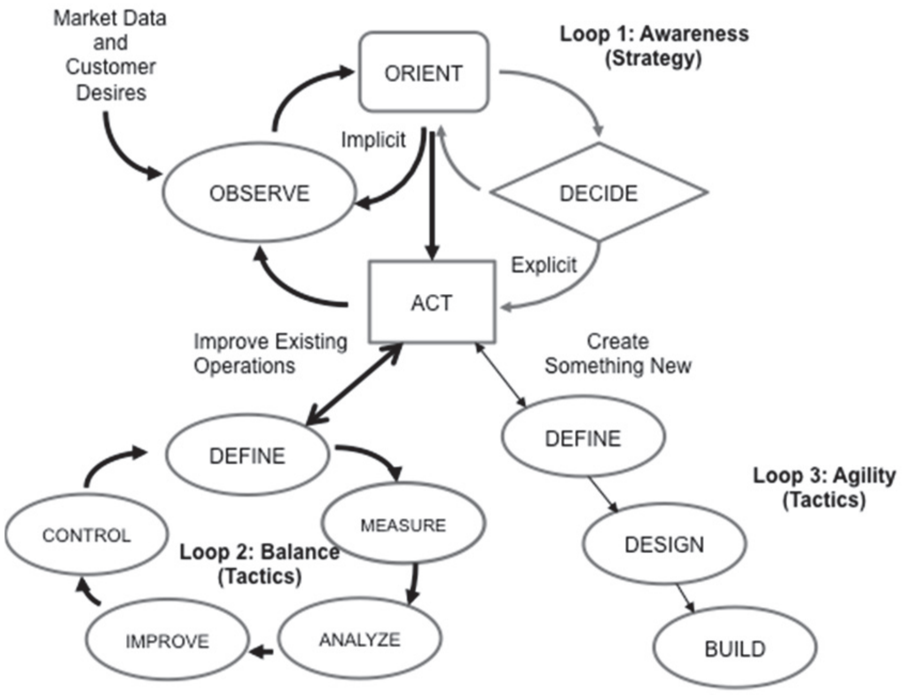

Market Data and Customer Desires
ORIENT
Loop 1: Awareness (Strategy)
Implicit
OBSERVE
DECIDE
Explicit
ACT
Improve Existing Operations
Create Something New
DEFINE
DEFINE
CONTROL
MEASURE
Loop 2: Balance (Tactics)
DESIGN
Loop 3: Agility (Tactics)
IMPROVE
ANALYZE
BUILD

Fig. 6. Process Model of a Responsive Organization. (Hugos 2010).

Hugos (2010) presents yet another model of an *Agile Enterprise*, representing a responsive organization, as illustrated in Fig. 6. The process model consists of three process loops. While Leffingwell's model is static and focuses on how to pace the development and deliveries (how an enterprise can implement iterative and incremental development), the viewpoint here is dynamic and how an enterprise should respond to real-time changing business dynamics. Here iterations and increments are a mechanism to implement the dynamic response.

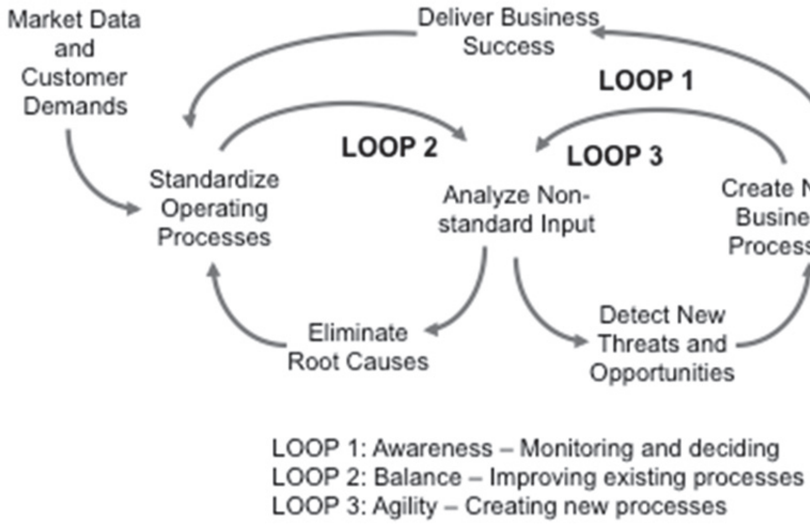

Market Data and Customer Demands
Deliver Business Success
LOOP 1
LOOP 2
LOOP 3
Standardize Operating Processes
Analyze Non-standard Input
Create New Business Processes
Eliminate Root Causes
Detect New Threats and Opportunities
LOOP 1: Awareness – Monitoring and deciding
LOOP 2: Balance – Improving existing processes
LOOP 3: Agility – Creating new processes

Fig. 7. Real-Time Business Dynamics. (Hugos 2006).

Fig. 7 defines management's role in an *Agile Enterprise*. As the balancing (loop 2) and agility loops (loop 3) are self-empowered and self-regulating, management's role changes to monitoring and deciding. The right decisions are guided by knowledge of internal operations, strategy, and knowledge of the business environment.

2.4 Complex Adaptive Systems

Even if the definitions of *Agile Software Development* and agility may vary, and these may cover different aspects of agility, most agile leaders tend to agree that a software development team (or organization) is a *Complex Adaptive System*

(CAS). (Highsmith 1999, Schwaber and Beedle 2002, Larman 2003, Anderson 2003, and Appelo 2011)

However, the obvious question remains: assuming that a software development organization is a *Complex Adaptive System*, which we have previously managed with a frame belonging to the "simple" category, i.e., formerly "slicing the problem into pieces of a manageable size"[17] and "leading with simple structures", an obvious challenge remains: new structures and leadership tools are needed to bring the management into the 21st century, i.e., to create a software management frame that takes into account the fact that software creation happens in a complicated area (Snowden and Boone 2007). We need to replace the former reductionistic leadership frame used so far. This means that we also need new ways to scale software development.

2.4.1 Cynefin Framework

The *Cynefin Framework* is used to explain how software development falls into the area of complexity and thus simple (mechanistic) process models (like the traditional waterfall model) fail. This approach is promoted at least by David Snowden, Joseph Pelrine, and Jurgen Appelo, and recently seems to have been adopted by some agile tool vendors such as Rally Software (Zach 2011).

Snowden and Boone (2007) presented a *Cynefin Framework* for the leader's decision making; see Fig. 8. It is based on the understanding of complexity, i.e., while there are areas that are simple, with a link between cause and consequence (and where you can use simple fact-based management), there also exist *Complicated*, *Complex,* and *Chaotic* areas, where we should resist the urge to make complex problems simple. We should learn to manage in *Complicated* areas by analyzing the situation before responding, as there might be many possible responses, whereas in *Complex* areas we should learn first to probe, then sense and respond.[18]

[17] Note that architectural componentization and clearly defined interfaces have been one manifestation of this analytical approach to managing complexity by slicing the problem into parts. The agile literature nowadays advocates cross-functional feature teams with "full responsibility" in place of component teams and clearly defined responsibility boundaries as a faster alternative. (Larman and Vodde 2009)

[18] Scrum certainly has its roots in studies of complexity, because of understanding the set of defined rules as the basis of how to organize. However, as Snowden points out, whether rules-based flocking behavior can be applied to human complex systems is arguable, as humans have intelligent

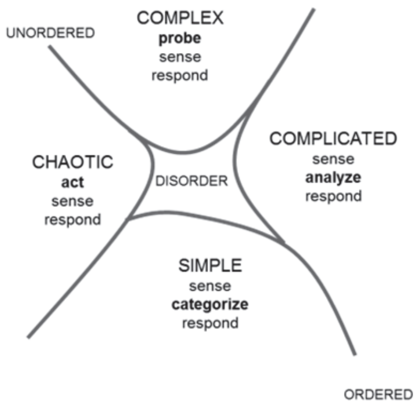


Fig. 8. Leader's Framework for Decision Making. (Snowden and Boone 2007).

While software development itself is complex (as there are many implementation alternatives) and self-organizing software development teams may also be, it is necessary to understand more about *Complex Adaptive Systems* in order to understand what this means from the point of view of scaling software development or agile models. The former "slicing into chewable chunks" needs to be replaced with a more synergistic software management frame.

2.4.2 Complex Adaptive Systems and Software Development Organizations

Complex Adaptive Systems are special cases of *Complex Systems*. They are complex in that they are dynamic networks of interactions and relationships instead of aggregations of static entities. They are adaptive in that their individual and collective behavior changes as a result of experience. *Complex Adaptive Social Systems* are composed of interacting and thoughtful agents. (Miller and Page 2007)

Miller and Page (2007) propose a framework for such a system, consisting of eight components, as presented in Table 6.

capabilities beyond those of birds and bees, being able to adapt their behavior in a far more complex manner. (Snowden and Boone 2007)

Table 6. Eightfold Mapping of Agent-Based Object Models (Miller and Page 2007).

Path	Focus
View	Information and connections
Intention	Goals
Speech	Communication among the agents
Action	Interaction
Livelihood	Payoffs
Effort	Strategies and actions
Mindfulness	Cognition
Concentration	Model focus and heterogeneity

The first characteristic of such a system is how much the agents 1) share the same *view*, i.e., have the same *information and connections*. The second characteristic is if 2) the agents also have the same *intention,* i.e., share the same *goals.* The third, fourth, fifth, and sixth characteristics include 3) the amount of *communication among the agents, 4)* the *interaction,* and *5)* the *payoffs that* exist, and if *6)* their *strategies and actions* are somewhat unified. 7) *Cognition* defines how intelligent the agents that are acting are. Lastly, 8) the *model focus and heterogeneity* defines whether the model is sufficient to capture the phenomenon of interest. Heterogeneity defines that in the model there can be substantial heterogeneity across agents; e.g., many economic models investigate the worlds of multiple agents via a single "representative" agent. Models always have context, and what works well in one context may fail in another. Models of *Complex Systems* phenomena should be simple, not complicated. The target is to simplify the otherwise overly complex world.

What distinguishes a CAS from a *pure multi-agent system* (MAS) is the focus on top-level properties and features such as self-similarity, complexity, emergence, and self-organization. Some people claim that traditional software development organizations are not *Complex Adaptive Systems* as they lack emergency and self-organization at the team level. But all human organizations are self-organizing — even when some agents dictate the shape and behavior of the whole organization (Schwaber and Beedle 2002).

Miller and Page (2007) demonstrate how different levels of adaptation can impact on behavior. They built a CAS system that first started with a fixed-rule model and then allowed for successive degrees of adaptation, i.e., *Agents* first followed a homogeneous model, then a heterogeneous model, and finally individual models. This level of adaptation impacted on the detected behavior.

In this thesis CAS has been used as a source of inspiration. The phenomena detected in human systems are compared to a CAS framework. The human *agents* differ from those in nature because humans have attributes that *agents* in nature do not possess. Humans have a multitude of identities and intelligence, and they do things intentionally. Humans are also messy. The branch of science studying *Cognitive Science* together with *Complexity Theories* is called *Cognitive Complexity*. (Snowden 2012)

2.4.3 Software Development Organizations as Complex Adaptive Systems

Dooley (1997) studied how *Complex Adaptive Systems* can be used to model organizational change. This is based on the assumptions that:

- organizations are potentially chaotic;
- organizations move from one dynamic state to another through a discrete bifurcation process (second-order change);
- forecasting is impossible, especially on a global scale and in the long term (unpredictability);
- When in a chaotic state, organizations are attracted to an identifiable configuration (order out of randomness);
- when in a chaotic state, similar structural patterns are found on the organizational, unit, group, and individual levels (the fractal nature of chaotic attractors).

Similar measures taken by organizations in a chaotic state will never lead to the same result. (Dooley 1997)

This view is very far from ideas of organizations that prevailed earlier and held reductionism, determinism, and equilibrium to be core principles — seeing organizations as machines. Organizational direction was embedded in plans which were then deployed via planning, budgeting, and management-by-objectives systems. A centralized, bureaucratic structure was seen as key in helping organizations' leaders determine the proper measures and deploy instructions to the workforce. (Dooley 1997)

Johnson (2007) defines *Complex Adaptive Systems* as systems that have some or all of the following attributes:

– the number of parts (and types of parts) in the system and the number of relations between the parts is non-trivial — however, there is no general rule to separate what is "trivial" from "non-trivial",
– the system has memory or includes feedback,
– the system can adapt itself according to its history or feedback,
– the relations between the system and its environment are non-trivial or non-linear,
– the system can be influenced by, or can adapt itself to, its environment, and
– the system is highly sensitive to initial conditions.

It should be noticed that the aspects of emergency and self-organization are present but are not overly emphasized in this definition by Johnson.

2.5 Lean Thinking

When scaling *Agile Software Development* in large organizations, many people search for support from *Lean Thinking*, i.e., from lean concepts, practices, and understanding (Vilkki 2010). However, lean product development will not provide a complete answer or solution. As Vilkki (2010) states, *Lean Thinking* gives better tools to understand and address the underlying problems. Lean and agile approaches complement each other in many areas but there are also challenges in combining the two.

The reason why *Lean Thinking* aligns better with *Agile Software Development* is that *Lean Thinking* also looks at systems holistically, and not via reductionism. That is why the lean management principles fit managing agile development better than reductionist management methods.

Massachusetts Institute of Technology has developed the LAI model (Lean Advancement Initiative) over the last ten years, resulting in an extensive amount of information on how to perform a large-scale lean transformation. The LAI model of lean transformation is presented in Fig. 9.

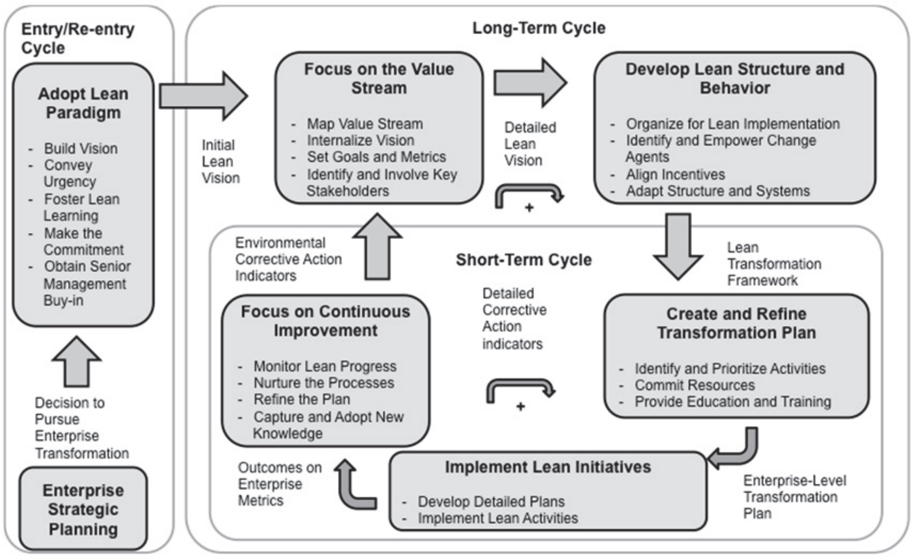

Entry/Re-entry Cycle
Adopt Lean Paradigm
- Build Vision
- Convey Urgency
- Foster Lean Learning
- Make the Commitment
- Obtain Senior Management Buy-in
Decision to Pursue Enterprise Transformation
Enterprise Strategic Planning
Long-Term Cycle
Initial Lean Vision
Focus on the Value Stream
- Map Value Stream
- Internalize Vision
- Set Goals and Metrics
- Identify and Involve Key Stakeholders
Detailed Lean Vision
+
Develop Lean Structure and Behavior
- Organize for Lean Implementation
- Identify and Empower Change Agents
- Align Incentives
- Adapt Structure and Systems
Lean Transformation Framework
Short-Term Cycle
Environmental Corrective Action Indicators
Focus on Continuous Improvement
- Monitor Lean Progress
- Nurture the Processes
- Refine the Plan
- Capture and Adopt New Knowledge
Detailed Corrective Action indicators
+
Create and Refine Transformation Plan
- Identify and Prioritize Activities
- Commit Resources
- Provide Education and Training
Outcomes on Enterprise Metrics
Implement Lean Initiatives
- Develop Detailed Plans
- Implement Lean Activities
Enterprise-Level Transformation Plan

Fig. 9. LAI: Enterprise-level Roadmap to Lean Enterprise. (Nightingale 2002).

The LAI model is primarily based on the Toyota Motor Company's *Toyota Production System*. The Toyota Motor Company is widely recognized for developing and successfully implementing many of the original concepts that underlie the lean framework (Womack and Jones 1996). *The Toyota Production System* focuses primarily on the factory floor level (Ohno 1988). From these initial concepts an array of researchers, universities, companies, and industries have developed an expanded vision of the values, behaviors, and practices within enterprises that constitute a new and emerging expression of what it means to be a "lean enterprise" (Womack and Jones 1996).

In the LAI model *what* is being implemented seems to be in a constant state of flux, affected by what is already in place. Nightingale (2002), who has been developing the LAI model for the last ten years, states that *"No company, not even Toyota, has completely mastered this expanding version of lean. This will always be the case, since the framework is evolving continuously, including enhancements from non-automotive and non-Japanese firms as well."* This indicates that lean organizations are *Learning Organizations*.

2.5.1 Learning Organizations

There are four steps by which the organization acquires knowledge and learns: *knowledge acquisition, information distribution, information interpretation*, and *organizational memory*. Learning and change may often be considered dual. Argyris and Schön (1978) differentiate two types of learning: first (or single-loop) and second (or double-loop) order; see Fig. 10. (Dooley 1997)

Most learning happens in single-loop order, i.e., the governing variables or underlying assumptions are not known or communicated.

Argyris (1993) divides organizations into two types: Model I and Model II organizations; see Fig. 11. Model I organizations have single-loop learning change action strategies that are consequences of four governing values.

1. Achieve your intended purpose.
2. Maximize winning and minimize losing.
3. Suppress negative feelings.
4. Behave according to what you consider rational.

The most prevalent action strategies that arise from Model I are the following:

- advocate your position;
- evaluate the thoughts and actions of others (and your own thoughts and actions);
- attribute causes for whatever you are trying to understand.

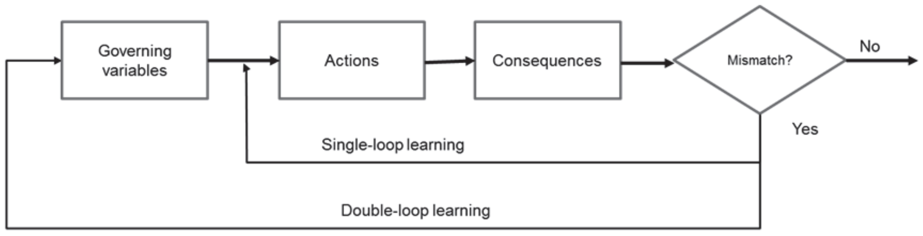


Fig. 10. Single and Double-loop Learning. (Argyris and Schön 1978).

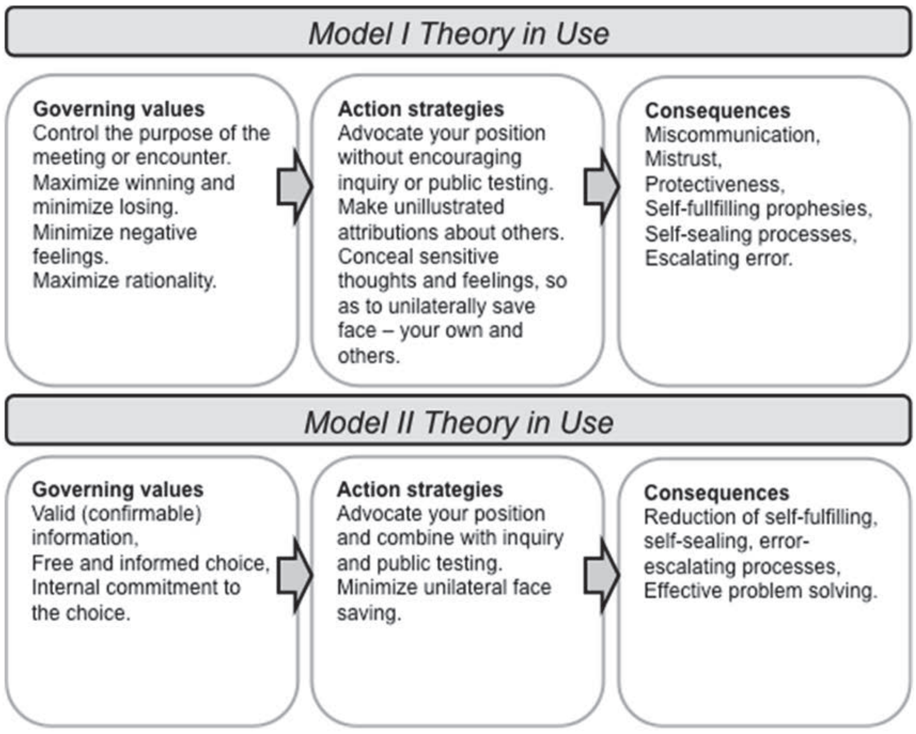

Model I Theory in Use
Governing values
Control the purpose of the meeting or encounter.
Maximize winning and minimize losing.
Minimize negative feelings.
Maximize rationality.
Action strategies
Advocate your position without encouraging inquiry or public testing.
Make unillustrated attributions about others.
Conceal sensitive thoughts and feelings, so as to unilaterally save face – your own and others.
Consequences
Miscommunication,
Mistrust,
Protectiveness,
Self-fullfilling prophesies,
Self-sealing processes,
Escalating error.
Model II Theory in Use
Governing values
Valid (confirmable) information,
Free and informed choice,
Internal commitment to the choice.
Action strategies
Advocate your position and combine with inquiry and public testing.
Minimize unilateral face saving.
Consequences
Reduction of self-fulfilling, self-sealing, error-escalating processes,
Effective problem solving.

Fig. 11. Model I and Model II Organizations. (Argyris and Schön 1978).

The consequences of Model I strategies are likely to be 1) *defensiveness, 2) misunderstanding,* and 3) *self-fulfilling and self-sealing processes.* (Argyris 1993)

Model II organizations differ from Model I organizations in that the *governing values*, or *master program*, is also communicated. The governing values are 1) *valid information*, 2) *informed choice*, and 3) *vigilant monitoring of the implementation of the choice in order to detect and correct error*. Model II behaviors are crafted into action strategies that openly illustrate how the actors reached their evaluations or attributions and how they created them to encourage inquiry and testing by others. As a consequence, defensive routines that are anti-learning are minimized and double-loop learning is facilitated. (Argyris 1993)

2.5.2 Lean Organizations

Complex Adaptive Systems are *Learning Organizations*, but *Complex Adaptive Systems* also have other qualifications beyond *Learning Organizations*.

In a *Learning Organization* activities are steered from the top. Argyris (1993) explains that organizational learning is inhibited by *Defensive Routines in Business*, as well as the *Gap between Knowledge and Action.* Effective organizations exhibit an appropriate balance between differentiation and integration, and the exact point of balance is heavily influenced by environmental demands. A key to implementing the correct balance is the behavior of the managers (Argyris 1993). A constructive confrontation is needed by managers to originate such improvement measures. Argyris (1993) states that any organization that is capable of double-loop learning must learn to interrupt defensive routines by action strategies that illustrate advocating, evaluating, and attributing the most prominent behaviors. Argyris (1993) also suggests that the biggest leverage to initiate double-loop learning has been, and continues to be, made at the top. This is clearly different to how some *Complex Adaptive Systems* are led: in many swarms the learning and adaptation happens at all levels, i.e., the swarms have only temporary leaders (Fisher 2009).

The statement of many agile advocates is that traditional management methods are less efficient, because *Complex Adaptive Systems* are led with a simple (mechanistic and reductionistic) model.

A frequently heard statement at agile conferences is that team-level self-organization and more emergent behavior would lead to better business results.

Appelo (2010) compared *Lean Thinking* and *Complexity thinking.* Appelo (2010) states that he believes a value chain is an example of linear thinking, which usually leads to sub-optimization of the whole organization. But in any *Lean Manufacturing* site the value chain actually consists of multiple value chains, each interlinked with each other; *Lean Manufacturing* optimizes the end-to-end value chain in order to avoid any sub-optimizations. This has a certain logic as the actual product delivery happens only once, as a single compound delivery. In a software company providing software to the internet, which is a *Complex Adaptive System* itself, a different structure with multiple value chains each having separate deliveries may provide a better alternative. So the optimal structure for a company may be dependent on the business environment around the organization.

According to Holling (2001), all human ecosystems traverse the different phases of *Adaptive Cycles*. All human ecosystems and organizations are also *Complex Adaptive Systems*. This is true for any organization, whether it is a *Lean Organization* or traditional reductionism-based organization managed with scientific management methods (Schwaber and Beedle 2002). Swarms are *Complex Adaptive Systems* but not all swarms are alike. Ants, for example, have quite developed hierarchies. Some *Complex Adaptive Systems* involve more organizational levels in decision making when the organization is re-aligning or reorganizing itself, some less. (Fisher 2009)

All software development organizations are *Complex Adaptive Systems*. So the question is merely about management behavior; whether more probing (as suggested in the *Cynefin Framework* decision-making model) before all decision-making would lead to better business results. But part of *Lean Thinking* is to perform *Kaizen events* for continuous improvement. In a Kaizen event, first the delivery capabilities and the amount of inventory are measured. Then the leadership team makes suggestions for improvements, which are immediately implemented and the performance is measured to see if there is an improvement or not. The *Lean Manufacturing* sites typically have far more initiatives and most of the proposed initiatives are already implemented before even being written as proposals. Kaizen events are actually based on theories of iterative learning. Fig. 12 presents another version of the Action Research Cycle. Deming taught this cycle in Japan, and it is now used as the basis of the *Kaizen* improvement events in the Toyota Production System (Lazko and Sounders 1995, Liker 2004).

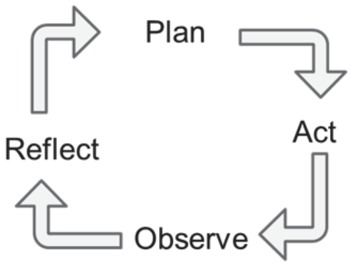


Fig. 12. (Simplified) Action Research Cycle. (O'Brien 1998).

Lean organizations are *Learning Organizations*, but the examples of *Kaizen improvement* events and many improvement suggestions are all examples of management behavior based on *probing*. Therefore the management of a *Lean Organization* has at least some elements of complex decision-making built into it.

Lean Thinking does not provide a complete answer to the question of how to scale *Agile Software Development*. *Lean Thinking* covers Lean Manufacturing and Lean Product Development (mainly from the physical product development point of view), but it does not produce direct answers that can be used to create software. Poppendieck (2003 and 2010) provides some answers to this question. However, there is also room for new discoveries, such as the Kanban method for software development (Anderson 2010, Ikonen, Kettunen, and Abrahamsson 2010, Kniberg and Skarin 2010) has proved. The Kanban method takes the principles used in manufacturing and applies those to software development. The result is something new; Kanban is a known concept in *Lean Manufacturing*, and those concepts have now been applied in software development in a new way, resulting in a method that did not exist before.

2.6 Adoption of Agile Methods

Over the last ten years *Agile Methods* have reached the mainstream: according to a Forrester study, 35% of the respondents stated that agile most closely reflects their development methods (West and Grant 2010). This makes *Agile Methods* probably the most widely applied systematic software engineering methods in the world.

The growth in the usage of *Agile Methods* from 17% in 2007 (Schwaber, Laganza, and D'Silva 2007) to 30.6% in 2010 (West and Grant 2010) has been remarkable. Besides those who stated that they did not use any method at all (30.6%), different *Agile Methods* (Scrum, Agile Modeling, Feature-Driven Development, Test-Driven Development, Extreme Programming, Lean Development, the Microsoft Solutions Framework for Agile, the Agile Data Method, Adaptive Software Development, Six Sigma, Crystal, Behavior-Driven Development, and Dynamic Systems Development Method) are used more (35%) than traditional software engineering methods (13% reported they used CMMI,[19] waterfall, or ISO 9000 as their development method). (West and Grant 2010)

The adoption of *Agile Methods* has proven to be a challenging task (Svensson and Höst 2005). It has also been claimed that agile adopters are often unaware of

[19] The relation of CMMI to Agile Methods is beyond the scope of this thesis. In the Forrester survey, some respondents stated that CMMI is the software development method they use.

what agile adoption really means, and how broad a change is actually required (Schwaber, Laganza, and D'Silva 2007).

Next, we examine the level of agile adoption. The minimum level of adoption is on the level of adopting some *Agile Practices* to support development with traditional methods. The next level is the adoption of a specific agile method. Finally, the whole organization can become agile and lean.

2.6.1 Adoption of Some Agile Practices

We know from several studies that many organizations have adopted individual *Agile Practices* or certain fundamentals of *Agile Software Development* in order to complement the organization's existing processes (Manhart and Schneider 2004). Often, the adoption of existing *Agile Methods* may require their radical modification to fit the operating context (Grenning 2001). Most large-scale adopters have also had to mix agile with their currently existing methods, and compromise the agile orthodoxy (West and Grant 2010). This is in line with the thinking that just having shorter iterations resulting in better quality may not be sufficient for large companies producing complex software; a more holistic view of agility may be needed (Paper III).

2.6.2 Adoption of a Specific Agile Process Model

There are many different *Agile Software Development* methodologies available, e.g., Extreme Programming (XP), Scrum, and Adaptive Software Development (ASD). The problem for a project manager is the selection of an appropriate process model from among the many alternatives (Paper II).

The usefulness of each model depends on the actual project context and the respective challenges. The key is to recognize the unique goals and problems in the particular project environment (Paper II).

There are also many other ways to direct the course of the project than just selecting and adjusting the software process. The project management tools form a very wide arsenal in this sense. Some of these belong to the area of organizational psychology; some belong to the area of financial control. Our purpose here is not to cover all the different aspects of effective project management. In general there is a wide range of software development project types, ranging from large contract-driven IT/IS systems to small in-house developments. While they share many common characteristics, each project type and context induces particular considerations. (Paper I)

2.6.3 On the Success of the Adoption

Various success and failure factors have been proposed as being significant in the agile adoption phase (Chow and Cao 2008, Norton 2008). For example, it has been suggested (Chow and Cao 2008) that for various reasons *Agile Methods* seem to polarize people and stakeholder groups into opponents and supporters who have very different standpoints regarding the usefulness of *Agile Methods* for the organization. To overcome this, a fundamental change in philosophy and the development of new behaviors are claimed to be required across the organization.

A recent study proposes that the appreciation of *Agile Methods* seems to increase once they have been adopted and applied in practice (Norton 2008), which indicates that while there is likely to be resistance among the agile adopters in the organization, time and experience of applying the methods will have a positive effect on this resistance.

The level of success in the improvement of the software process can be characterized in terms of personnel satisfaction and whether the new process is actually used (Abrahamsson 2000). An efficient process may be disliked by personnel, but such a process would not be compliant with the agile value of team empowerment (Svensson and Höst 2005) and the proper balance of the centralization and decentralization of decision making (Reinertsen 2009). Thus, it is of major importance to study and understand the underlying factors affecting the satisfaction of stakeholders with agility in organizations adopting *Agile Methods*.

However, the ultimate measure of the success of the adoption of agile needs to be the business benefits, i.e., the successful adoption of agile needs to be defined (and measured) with the business parameters.

2.7 Aspects of Agility

The fact that the definitions of *Agile Software Development* and agility vary considerably indicates that the concept really is complex and multidimensional (i.e., it is not simply about responsiveness to changes). The customer/market interface is important (Katayama and Bennett 1999). The ultimate goal is profitable business (Gould 1997).

While most *Agile Methods* do have a focus on business, these methods do not take any position on how to run the business processes such as business model refactoring or finding and nurturing new emerging businesses. Strictly speaking, these methods are more focused on how to deliver rather than how to come up with a good business strategy. Extreme Programming and Scrum merely define how software teams (and integration and delivery) should be working. When compared to the overall traditional process charter, one can see that many aspects are missing, such as, e.g., product (or portfolio) management and people processes (such as reviewing and rewarding performance). On the other hand, when properly applied, agile will transform many such old processes — e.g., project management and requirement management will become the same. There will be no need for separate progress tracking as in traditional project management, when done items in the backlog reveal the status more accurately. Or vice versa — if project management is retained as an overlapping system, there is a big risk that the agile style of backlog grooming and scope management will be left undone and the information in backlogs and project management reports will not be the same. Besides, from a lean perspective, overlapping project management reporting is wasteful. Testing will not be a phase at the end of the development, but rather an essential part of what gets done.

It can be seen that agility is not only a dimension or an aspect of a project, but the aspects of agility are applicable in the entire organization. From these dimensions we can identify at least the following.

1. *Strategic Agility*, i.e., the ability to continuously redirect and reinvent the core businesses without losing momentum (in contrast to traditional portfolio restructuring) by maintaining balance with strategic sensitivity (awareness and attention), leadership unity (collective commitment), and resource fluidity (people rotation and organizational structures), working as an integrated real-time system (Doz and Kosonen 2008).
2. *Business Agility*, i.e., the marriage of strategy (awareness) and agility (tactics) in order to create a responsive organization for business benefit (Hugos 2009) or a sum of process agility and technical agility or a sum of speed and flexibility that we can then, e.g., use to enable mobile business solutions[20] (Evans 2002). Hugos (2009) states that all products have two components: the actual product and an information component that adds value to a

[20] The underlying statement here is that according to Evans, organizations can increase the flexibility of their business models by adding mobility to their data solutions.

customer. Widely understood, the information component can be understood as covering all the immaterial benefits that the user gets when purchasing the specific product in question. A product ecosystem provides similar (immaterial) added value to the customer; thus in this dissertation the additional value provided by a product ecosystem is included in the Business Agility aspect.

3. *Agile Organization*, i.e., the well-working combination of Informal Networks and the Formal Organizational structure, for which agility is key and pervading trust a necessity (Atkinson and Moffat 2005).
4. *People Agility*, i.e., the ability to shuffle work around the organization when the priorities or focuses change — this is roughly similar to the "Resource Fluidity" of Doz and Kosonen (2008).
5. *Tools Agility*, i.e., the ability to have tools that support the agile way of working and that can easily be modified for a new purpose as the process changes (West and Hammond 2010).
6. *Organizational Culture*, i.e., the competing different organizational values and cultures related to agile values and agile culture (Iivari 2010).
7. *Agility of the Product that is Built*, i.e., the ability to modify, version, personalize, configure, or refresh the product to reach new customer groups or please the existing user (Grant 2010).

One could argue that the first three (Strategic Agility, Business Agility, and Agile Organization) are actually the same aspects, only observed from different angles or viewpoints. The list could also contain *Agile Innovation* (Oza and Abrahamsson 2009) as one agile aspect. Oza and Abrahamsson (2009) define *Agile Innovation* as an aspect that combines innovation processes with agile processes. Here it is omitted because creativity and innovation are seen as an outcome of a *Complex System* tolerating internal conflicts (Appelo 2011). Creativity and innovation can thus be understood as intrinsic qualities of such an Agile Organization, and they could thus be derived from other Agile Aspects as a result.

2.8 Agile Frameworks

Agile Methods are mostly characterized by a set of principles and practices. Most people refer to these principles and practices when discussing how *Agile Methods* could be scaled, but these principles and practices actually do not indicate how to scale *Agile Software Development*.

If we want the whole company to work in an agile manner, it is not enough to focus on the team and project-level dimensions alone, and run agile adoption like any typical *Software Process Improvement* (SPI) initiative, focusing only on process-level changes. For example, Sharifi and Zhang (1999) have proposed a methodology and framework for achieving agility in manufacturing enterprises. The company has first to understand how agile it currently is, and how agile it needs to become (drivers of agility). The necessary capabilities of agility (e.g., responsiveness and flexibility) can then be achieved with different providers stemming from the organization, technology, people, and innovation (Paper III).

In large-scale product development projects, the original assumptions of agile methodologies do not necessarily hold, and additional prerequisites and organizational conditions have to be satisfied in order to achieve the full benefits of agility. The successful usage of *Agile Methods* in such environments requires a wider understanding of *Agile Organizations* and their enabling factors, as well as of the factors that prevent large companies from achieving agility. (Paper III)

The current trend in the development of software process models advocates more adaptable and flexible ways of working, i.e., moving from rigid all-defining huge organizational processes towards sketched, tailorable, agile processes. A typical way of working is to add only the few most essential practices to the project — more like a process skeleton — to which the practices can gradually be added. The model here is to let the project decide about the practices whenever it is ready to put them into use rather than establishing a well-set rigid model to be followed. (Highsmith 1999, Aaen 2003, Gnatz 2003, Keenan 2004)

2.8.1 Agile Organizing Framework

Vidgen and Wang (2009) propose an *Agile Organizing Framework* consisting of Enablers, Inhibitors, and Emergent Capabilities. Via careful analysis of two organizations — one agile and one non-agile — they came to propose three co-evolving, self-renewing principles and six agile team capabilities that are supported by multiple detected agile enablers (characteristics of fostering behavior in an *Agile Organization*) and multiple detected agile inhibitors (characteristics of the suppression of agile behavior in an organization). They suggest three *Agile Principles*, each one backed up by two agile team capabilities, as follows:

- Principle 1: matching co-evolutionary change rate, backed up with:
  - co-evolution of IT team and customer to create business value, and
  - sustainable working with rhythm.
- Principle 2: optimize self-organization, backed up with:
  - collective mindfulness, and
  - sharing and team learning.
- Principle 3: synchronize exploitation and exploration, backed up with:
  - process adaption and improvement, and
  - product innovation.

An example of an agile enabler for collective mindfulness is shared responsibility of project management and team discipline through peer and self-observation. An example of an agile inhibitor would be a centralized project management that is external to the team members.

All the agile inhibitors presented by Vidgen and Wang seem to be more or less the results of management based on reductionism and determinism. If a traditional software development organization can be seen as an organization that splits the tasks into items of a manageable size with development targets that are then integrated and delivered by a pre-planned schedule, an *Agile Organization* can be seen as an organization that is trying to bridge the actions with the reasons, e.g., by connecting the customer with the product development requirements and integration with emergence — i.e., what used to be split into separate blocks is no longer separated.[21]

The inhibitors described by Vidgen and Wang are called *water-scrumming* (West 2011) problems — problems that originate from the attempt to manage agile Scrum teams with traditional, Waterfall-based management methods. The enablers provide examples where the organization has successfully been able to make the change. But the Vidgen and Wang model lacks the capability to describe how to make the full transition from water-scrumming to an *Agile Organization* — apart from removing impediments one by one.

[21] The former can be illustrated with a block architecture and hierarchy, the latter with fractals and iteration/integrating circles.

2.8.2 Agile Framework

Paper III presents a framework for large-scale agility; see Fig. 13. This framework consists of hundreds of *Aspects of Agility*, presented in a huge spreadsheet. Paper III brings structure to these different aspects, categorizing those into *Goals* (why this is done), *Means* (how one can implement agility), and *Enablers* (what is needed to be agile).

The arrows between *Goals*, *Means,* and *Enablers* can be seen as force vectors, each vector trying to stretch one aspect of agility. However, at the time of writing, the linkage between the different components was unclear.

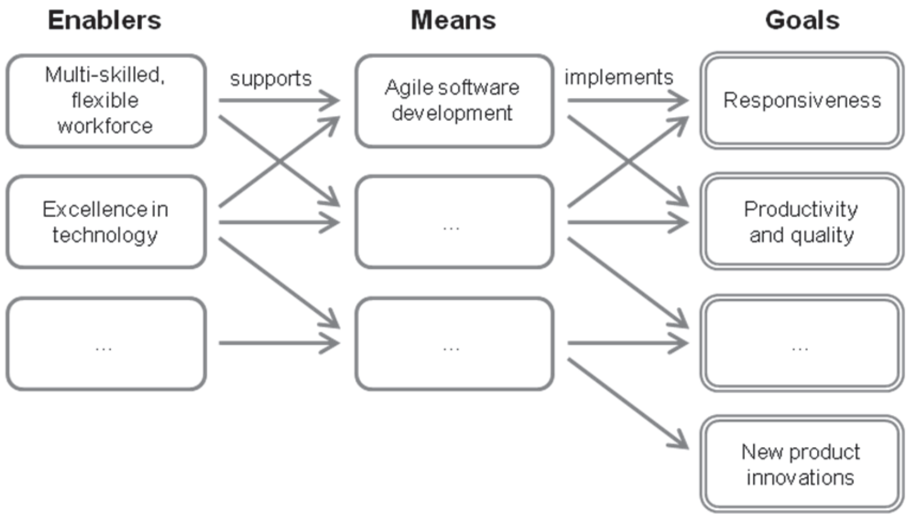


Fig. 13. Components of Achieving Enterprise Agility doing Product Development. (Paper III)[22].

The first step for the organization to improve its software production agility is to understand exactly what agility is and what that company would actually like to

[22] At the time this picture was drawn the authors did not have a solid understanding of what else was needed in the "*Means*" category, except that something was needed to fill the gaps besides *agile software development*. Now the author would fill the empty bubbles with *adaptive control for finance and human resources*, as well as *collaborative leadership styles supporting agility* — i.e., various elements from the Agile Aspects (not the aspects themselves, though).

focus on. Agility is really a multidimensional concept. What kind of agility do we need? For example, do we want to:

- reduce the *Lead Time* of producing new product features,
- release new product variants more often,
- increase people's satisfaction with the methods in daily use,
- introduce totally new product concepts more frequently, and/or
- improve the cost efficiency of the software production?

The basic questions when studying the dimensions of agility in large organizations are thus these.

- What is the competitive edge of the company?
- Why is an agile (or lean) approach taken in certain operations?
- How is it achieved (i.e., techniques to master, decisions to make, questions to answer)? (Paper III)

A strategic choice is first to select the kind of agility that the company needs. That dictates the overall *Agile Framework* that the company needs to take into use. All agile models aim at fast and responsive operations in order to achieve high development productivity and quality. Each organization needs a specific *Agile Framework* for its own purpose that must be created on the basis of the business strategy, business models, and understanding the best way to organize operations.

The remaining activities must be defined around this agile frame. In a large product development organization, the software product development is just one element contributing to the overall agility of the company.

Considering the 'How' question from the software production point of view, we realize that the right choice to increase agility is business-dependent. As the ultimate goal for each company may differ from one to another, the means to achieve the goals may also differ. (Paper III)

3 Agile Models on Large Scale

Originally, the agile methodologies evolved to fulfill the needs of small flexible software teams. In large-scale product development projects, the original assumptions of the agile methodologies do not necessarily hold, and additional prerequisites and organizational conditions have to be satisfied in order to achieve the full benefits of agility. The successful use of *Agile Methods* in such large-scale environments requires a wider understanding of *Agile Organizations* and their enabling factors, as well as the factors that prevent large companies from achieving agility. (Paper III)

3.1 Agile and Agility in this Dissertation

Previously in this dissertation many different definitions of agility have been viewed. Some of those were broad — such as Appelo's definition of agility — to stay successful in ever-changing environments — but insufficient for a scientific work like this. Leffingwell's definition — i.e., House of Lean; see Appendix B — is better, but too large. That is why yet another definition is needed. Based on the analysis of the different points of emphasis in the *Agile Principles* and the several definitions of what agile means, agile and agility are defined in the context of this dissertation as:

Agile	*Agile Methods* are context- and case-sensitive, situational, value-focused processes and practices that are used in order to improve efficiency, performance, quality, and throughput. *Agile Methods* are iterative and incremental by nature and they need to be supported by tools, organizational structure, and people. *Agile Methods* are characterized by a common set of values and principles. (Laanti 2011)[23]
Agility	Agility means flexibility of organizational structures, business model, and product that is implemented by flexible operations, situation-specific processes, flexible tools, and innovative people guided by fast opportunity-driven and new-creating business strategies in order to

[23] The author was asked to add a reference that would clarify that the definition here is new, created by the author for this dissertation on the basis of the earlier analysis of the emphasis of the *Agile Principles* to cover as many of the original points of emphasis as possible.

> create benefit and new differentiating value for customers and the organization itself. (Laanti 2011)

3.1.1 Similar Concepts

Flexibility is a concept related to agility, and is quite often used almost as a synonym. Flexibility is an older concept than agility. Organizations have sought flexibility with methods other than agility, e.g., by outsourcing some of the development effort. An organization can be flexible even though it is not agile. An *Agile Organization* needs to reshape itself for a purpose, so it needs flexibility. So in this work agility is defined as flexibility that is achieved with *Agile Methods*; an organization can, however, be flexible without being agile. In this sense, in this work flexibility is considered as a larger concept than agility, containing all the possible means (also methods not depending on *Agile Methods*) used to extend flexibility.

Adaptability is a concept that is related to agility, and is considered in this dissertation as a synonym. Adaptability was the word Highsmith suggested in place of agile when the *Agile Manifesto* was written.[24]

3.1.2 Model Case

A model case is a "real-life" example of a concept and contains all of the critical attributes and no attributes of any other concept (Walker and Avant 1995). Here, the critical attributes are the emphasis found in the *Agile Manifesto*.

> *He woke up early, motivated to get the day's work started. Just yesterday the organization had delivered a bigger new incremental release to thousands of happy customers, and he was eager to see the first feedback prior to planning the next increment and starting a new iteration to adapt the functionality better to fulfill customer needs. The throughput of his organization had increased, as they had put new continuous integration tools into use, and as everybody was working with aligned, prioritized backlogs.*
>
> *As his team went through the happy feedback, he already knew what he was going to put into today's release. An Easter egg! The customers loved those,*

[24] Based on conversation with Alistair Cockburn.

and having fun was an important part of the work and business. A surprise like that might win a few teenagers over to be more loyal customers.

A friend from a neighboring team came by at noon, and asked if he or his team needed any help, but he answered that they were fine, so the friend decided to check with some other team. There was no reporting to do, as reporting was based on deliveries and done automatically. But he had noticed that their boss frequently checked with them and kept up to date on the technical details of what was happening, even though the dashboards were visible to him and everybody else every day. He spent the rest of the day updating the on-line documentation of their code, and providing a suggestion to be added to their backlog at some point.

Tomorrow they would be discussing the content of the next bigger release, and what the simplest way of implementing it would be. He knew the architects were busy today with model designs that would ensure future flexibility. He trusted those guys, as they had a proven track record of excellent designs. As he left the office on time, he decided to check with them in case some of the architects had a tip where to go tonight — or maybe they could even go together.

This is a model example of demonstrating agility. All the critical attributes from the *Agile Manifesto* are embodied in this example. The subject in question feels totally happy and empowered working with *Agile Methods*, and the business is flourishing, with happy customers, frequent feedback, and fast deliveries.

3.1.3 Borderline Case

A borderline case contains some, but not all, of the critical attributes of the concept. The inconsistency of the borderline case demonstrates the perfection of the model case and clarifies the qualities of the defining attributes (Walker and Avant 1995):

The day in the office started in the usual way. They had their Scrum meeting, and the project manager was assigning tasks to everybody. His job today was to fix the errors found in the latest release. Even though they were trying to follow Scrum, they had to stop new development to fix the errors because final testing was done at the end of the project. Luckily, since they decided to go agile a few months ago they had also decided that everybody is allocated *only to one project. Dropping the constant context switching enables me to focus on one task at a time, which relieves stress and improves efficiency, he thought, and that he actually liked this new way of working.*

In this example, some *Agile Principles* are followed up, but also many are violated. The person in question is obviously not empowered, and testing is done waterfall-style at the end of the project.

3.1.4 Contrary Case

A contrary case presents a scenario that is clearly not an example of the concept being investigated (Walker and Avant 1995). This illustrates the differences between the concept being analyzed and the contrary case and clarifies the attributes that have been defined.

When he went to work, there was an e-mail waiting from his boss in which the boss asked him to file last month's report, which was again late. As he reluctantly started to write the report, his project manager came by and asked him to do some ad hoc testing for him. When he started to file the error reports, he noticed that the team in Bangalore had already run similar tests, and got exactly the same results. He filed the errors anyway, because there was no advice on what else to do. His boss was not in the office when he tried to reach him for further guidance, and nobody knew where he had gone. He tried not to get frustrated, as he knew his boss was nervous because the latest release was already late before they got it into their hands and could start testing.

This example lacks all the critical attributes that *Agile Software Development* comprises. Whatever "agile" is, this is clearly not an example of the concept.

3.2 Model of an Agile Enterprise

An organization with all these aspects of agile that have been identified can be defined as an *Agile Enterprise*. Fig. 14 explains these concepts and their relations.

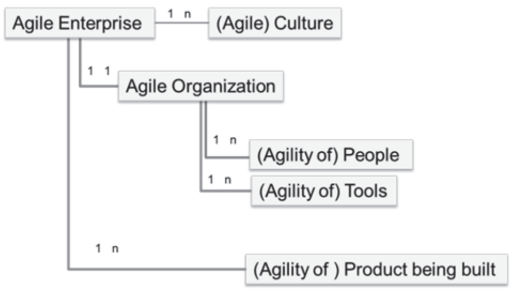


Fig. 14. Agile Enterprise Concept Model.

There are many ways to implement an *Agile Organization,* i.e., to have organizational flexibility in an enterprise. The Scrum literature advocates cross-functional and self-organizing Scrum teams. One question is if this self-organization and flexibility should be restricted to inside one architecture domain or whether the flexibility should be allowed across the enterprise. An organization can increase its flexibility if it allows the transfer of one Scrum team from one architecture domain to another when necessary. An enterprise can still be defined as an *Agile Enterprise* even though it does not allow this much flexibility.

The organization's processes, the way of working, the organization itself, and its policies define how much such flexibility exists in the organization. The organization's process model provides one viewpoint (but not the only one) on these aspects. The capabilities of the members of the organization are another one. An organization's process framework partially defines the level of flexibility, but is not the only definition.

How stretched each *Agile Aspect* is defines how much the organization can benefit from its flexibility. If the organization is implementing *Agile Methods* only on the team level, the potential benefits are also seen on the team level. In order to see an enterprise-level benefit, the organization must have aspects of agility that are fully stretched. A well-working Scrum team has resource fluidity when Scrum team members sign up for high-priority tasks. An architecture domain can have resource fluidity if it has some emergency troops to help Scrum teams that are showing poor progress. An *Agile Enterprise* can have resource fluidity as a result of an organizational change, as suggested by Doz and Kosonen

(2008), or if it has generalist Scrum teams that can operate like emergency troops, i.e., they can be called for help when needed.[25]

The number one reason for adopting *Agile Methods*, according to Forrester (2007), is time-to-market benefit. A large software development organization may not have a significant enough improvement of its time-to-market if *Agile Methods* are put into use on the team level only. Even though the teams may produce working code more frequently and more rapidly, the overall product development time is not improved. This is a typically known problem in *Lean Manufacturing*: optimizing the speed of delivery in parts of the system will not increase the output from the whole system. In fact, such sub-optimization can lead to increased waste.

3.2.1 Agile Enterprise and Lean Thinking

An ideal *Agile Enterprise* would have all its aspects stretched — i.e., agility brought up to the enterprise level. Such an ideal *Agile Enterprise* might be like Ohno's (1998) ideal factory and impossible to achieve in practice. Nevertheless, just as there is no limit to how far a lean enterprise can optimize its operations, there probably is no limit to how agile an enterprise can become. As a *Complex Adaptive System* or a *Learning Organization* an organization can learn from its best practices, and find new improved ways to stretch its agility even further.

However, many of the *Lean Thinking* tools are beneficial when defining the optimal implementation of an *Agile Enterprise*. Thinking tools such as *Value Chain Analysis* are helpful when thinking about how to get the benefit from agile operations on the organizational level.

An enterprise with many fast Scrum teams is like a machine with its inner parts spinning rapidly, with transmission to the system level failing. In practice, the rest of the organization may be unable to react to the rapidly produced code and fail in integration and putting all the code that has been produced into use. From the lean perspective, excessive code that cannot be integrated and used is waste.

An optimal value chain, in the organizational respect, is an optimized end-to-end delivery chain. Like in a lean factory, where there are many material flows that can each be optimized, in the *Agile Enterprise* there are also many information and delivery flows that can be optimized.

[25] These are typically called *Tiger teams* in the agile literature.

3.2.2 Agile Aspects as a Definition of a Complex Adaptive System

If we define an *Agile Enterprise* via the *Complex Adaptive System Framework* (Miller and Page 2007), we need to make sure that *Agile Aspects* cover all the dimensions of such a framework. If some dimension is not covered, the model of an *Agile Enterprise* cannot work, since it is not complete enough. This would be an indication of some *Agile Aspect* potentially being missing. Table 7 maps the CAS dimensions into *Agile Aspects*.

Table 7. CAS dimensions and Agile Aspects.

	CAS Dimension	Agile Aspects
1	Information and connections	This is mostly defined by *Agile Organization*.
2	Goals	What kinds of goals are set is mostly defined by *Business Agility*.
3	Communication among the agents	Communication mechanisms are mostly defined by *Agile Tools*, *Processes,* and *Organizational Culture*.
4	Interaction	This is defined by *Agile Organization*, *Organizational Culture*, and *Agile People*.
5	Payoffs	The payoff model is defined by the business model but also by the organization. *Business Agility* is looking for ways to improve payoffs. Are the payoffs even or not? This is an area that is not fully covered by Agile Aspects.
6	Strategies and actions	*Strategic Agility* defines this area.
7	Cognition	Cognition is affected by *People Agility* — people having sufficient training and understanding on all organizational levels. The *Organization Culture* also defines how an organization sees and understands itself.
8	Model focus and heterogeneity	Strategic Agility and Business Agility contribute to the model; heterogeneity is achieved through People Agility and Organizational Culture.

The CAS dimension of *Information and Connections* is mostly defined by *Agile Organization*, as organizational structures define how people interact with each other and how flexibility is built into the organization. The CAS dimension of *Goals* is defined by *Business Agility*, i.e., by the organizational strategy, which needs to recognize the responsiveness of an *Agile Enterprise* as a key enabler in the organizational strategy. The CAS dimensions *Communication among the agents* and *Interaction* are defined by *Agile Tools*, *Processes,* and *Organizational Culture*. The CAS dimension of *Payoffs* is defined by the business model but also by the organization, i.e., how the payoffs are divided between people. This area is not fully covered by *Agile Aspects*. The *Agile Aspect* of *Business Agility* looks for ways to improve payoffs for the whole organization, but there is no aspect that seeks flexibility in the payoffs or tries to balance payoffs in an optimal way.

The CAS dimension of *Strategies and actions* is defined by the *Agile Aspect* of *Strategic Agility.* The CAS dimension of *Cognition* — that people share the same understanding — is dependent on *People Agility. People Agility* covers the need to have sufficient training for all the people in the organization. The *Organizational Culture* also defines how the organization behaves. The CAS dimension of *Model focus and heterogeneity* is defined by aspects of *Strategic Agility, Business Agility, People Agility,* and *Organizational Culture. Strategic Agility* and *Business Agility* give focus onto the CAS model, and heterogeneity is achieved through *People Agility,* i.e., that people are properly trained, and by *Organizational Culture* that demands that they work in a flexible manner.

On the basis of the analysis above, it can be concluded that these aspects lack *Agile Payoffs*. *Agile Payoffs* mean the optimization of the payoff function — i.e., how the investments are made and how people get paid for their work.[26] This explains the interest in *Beyond Budgeting* (Bogsnes 2008) at agile and lean conferences, e.g., at the LESS 2010 conference there was a separate track for *Beyond Budgeting.*

3.3 Defining an Agile Enterprise

The definitions of agile and agility are necessary but not sufficient to understand what an *Agile Enterprise* is. In Chapter 2.7.4 *Aspects of Agility* were discussed. This chapter builds upon that basis a Model of an *Agile Enterprise* using Concept Analysis.

3.3.1 Model Case — Agile Enterprise

A model case is a “real-life” example of a concept and contains all of the critical attributes and no attributes of any other concept (Walker and Avant 1995). Here, the critical attributes are the *Aspects* of an *Agile Enterprise*.

[26] Note that optimization as the CAS framework defines it does not mean paying less to the workers but balancing the results with investments. An example could be a decision to purchase ready-made software instead of developing it in-house.

The daily synchronization call was canceled today, as he was going to meet with the leadership team. When he reached the office, he checked the dashboard and found out that the Program Zeta release was late. He drilled down the release plans, deliveries, and quality and testing data and saw that the problem was with the team in Milan: they were still not getting their build green, and thus they were not delivering the functionality they had estimated, which blocked the whole release. He decided to call them this afternoon to discuss how they could solve the problem: maybe he could send the Tiger team there to help now that they were free from initiating the systems for a new Beta product. Since all the reporting had been replaced by automated dashboards, he had had so much more time for real discussions with people on subjects that really mattered.

The leadership meeting was about investments. They decided to invest a bit more in a new product that had promising initial customer feedback but was not competitive enough. They also decided to initiate a study to find a partner with whom to develop the system together. Another product was late; they decided to withdraw the rest of the investments and focus somewhere else, as they knew the product would hit the market late if it had not solved the major technical problems by now. Luckily, the company's primary flywheel was doing well; they kept the investment on a steady level, as well as not allowing technical debt to accumulate. Fortunately, people were still enthusiastic about that product and kept on filing more innovations to improve it even more. In fact, most of those proposals were already done before hitting his desk. His job was to reward the best ideas for the money saved or future revenue created for the Enterprise.

During lunch he was discussing with his colleagues how their work had changed for the better. In the old times they had to struggle in the information jungle, not knowing what piece of information to trust. Nowadays everything was made visible and people communicated more about their intentions and viewpoints: not just about what actions are needed but why they think these actions are good. Everybody's opinion was heard as the company understood that different views and different backgrounds complement each other. But the operations were really fast because the company did not hesitate to start new initiatives. Only the initiatives that proved promising got permission to continue. That was another thing that was changed: in the past the management was looked to for guidance: nowadays the advice was taken *from the crowd. There was more accountability — even though people communicated more with each other, they only committed themselves to things that they could do themselves. Even he would not commit on behalf of his programs.*

In the afternoon there was a workshop to improve the continuous delivery system. A team representing various roles and various levels of the organization was called in to make proposals for how delivery models could be improved even further. The current metrics already looked competitive — but he knew the group would come up with some ideas to try out. The current quality metrics from each delivery were the existing bottleneck of the system — and he already had some ideas about how distributed computing could be used to speed things up. They could also improve the pre-processing of the automatic delivery feedback.

While he was on his way home, he got a call from the CEO of a partner enterprise. They had problems fulfilling their current order, and they were briefly brainstorming together how they could use the responsiveness of the Enterprise to find a way to help the partner in trouble. They finally concluded that if the partner were to use product Zeta in a flexible manner and add some more functionality to it, they might speed up their delivery. It was decided to ask for expert opinions about the applicability of this solution. The partner CEO was so happy he promised to provide the people and tools needed.

This is a model example of an *Agile Enterprise*. It covers all *Agile Aspects*, and is also a description of a *Learning Organization* that is also lean and complex. This *Agile Enterprise* demonstrates agility not just on the Enterprise but also on the project and team levels and also extends the concept of agility to partners, ecosystem, and product. It also demonstrates the importance of proper tools, understanding among people, and situation-based leadership.

3.3.2 Borderline Case — Agile Enterprise

It was evident that enabling agility on the team and project levels had been a success. Although the manager was slightly worried about these requirements for empowerment — how could they deliver if the teams would not obey orders and do what had been promised to the customer but engaged in *something else? — he could clearly see that there was a new kind of energy in the teams and they were making progress.*

Lately, there had been some criticism of how programs are managed. He could not understand this criticism. He had signed the program contract and he expected everyone else to fulfill their commitments too. He had to admit, though, that he was quite worried. Not all the features were ready and errors were starting to pile up. He wondered whether the new improved way of working and fixing errors in teams would help them to get the final project delivery milestone approved as promised.

This example shows a team-/project-level implementation of agile, but although all the teams might be agile, the example is not a description of an *Agile Enterprise*.

3.3.3 Contrary Case — Agile Enterprise

He started his day by checking the reports from his subordinates. Tom's report was again late, so he had to send him a reminder. Sometimes he had the feeling nothing would happen in his team if he didn't monitor and check all the time. He must have been given a team of complete idiots!

In his inbox there was an e-mail waiting from operations in Brazil. They stated that there was an error in the software release preventing the delivery. This must be some sort of power game again, he thought; the operations ramp-up is probably just running late again, and they want to put the blame on us. Thus he decided to call Tom — even though Tom's report was late he was the only one whose competence could be trusted — and asked him to run some test cases for him. Shouldn't they know by now how to make these products? he thought. This is already the thirteenth release.

During lunchtime he had an appointment with the product manager, who had some interesting ideas about what features could be put into the next releases. Unfortunately he had to turn him down by stating that they had already fixed the features to be developed for the next 18 months, and no new features could be fitted in.

In the afternoon they had a big milestone review of the product concepting that the guys had been preparing for months. This concept was part of the *ideas of renewing the roadmap that popped up while they were reviewing the company's five-year plan. There was an idea for a new type of product and they had to think if the business model was feasible or not. Some people objected to the new idea, stating that it would never fly, but he thought it might be promising. Because of the conflicting opinions they decided to return the concept to the drawing board and asked for more details in order to make the concept foolproof.*

This example lacks all the critical attributes that comprise an *Agile Enterprise*. Whatever an *Agile Enterprise* is, this clearly is not an example of the concept.

3.4 Deployment of Agile Enterprise

One needs to notice that no process model can ever resolve all the possible challenges in product creation. For some of the issues, tailoring the process might simply be the wrong measure to use. (Paper I)

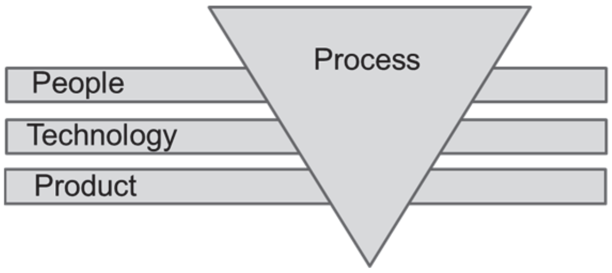


Fig. 15. Product Process Provides only a One-angled View of the Problems. (Paper I).

The reason why *Process* is presented as a triangle over *People*, *Technology*, and *Product* is because of the amount of influence or impact one can have on these with process means: *Processes* mostly change how *People* behave. Fig. 15 illustrates the fact that processes are just one way to solve the problems there might be. It would not help, for example, to tailor the process if the selected technology is too new and immature for the project at hand. Respectively, with process methodologies only people management issues can be tackled, while leadership issues typically remain outside the scope of the process. Product issues belong in the domain of business strategies and are developed separately from processes. Such implications for software process models have been raised, for example, by Curtis et al (1998).

Processes have some influence on the organization's capability to exploit new emerging technologies (e.g., a slow purchasing process might hinder the incorporation of the latest technology) but have little impact when it comes to the flexibility of the end product (e.g., processes may even prevent technologies from being combined in new, innovative ways by decoupling the development of product parts).

In software development agility is mostly concentrated only on the project-level software process, but in manufacturing the concept of agility is really much wider and also covers:

- (agile) product,
- (agile competitive) environment, and
- (agile) enterprise.

Each of the three (product, environment, and enterprise) can be made agile independently or together, thus the parentheses. In this thesis the environment is scoped out as defined in Chapter 1.2, and *agility of product* is considered as one *Aspect* of an *Agile Enterprise*, as defined in Chapter 2.7.4.

It is noticeable that in *Agile Manufacturing* the very idea of making only the process agile without considering the business and product would make little sense. Reversing this thought leads to a conclusion that considering the business and organizational enablers, as well as the agility of the product, would be greatly beneficial to any agile software company. (Paper III)

This viewpoint is backed up by the "4P model" that Liker (2004) has developed on the basis of his experience of lean transformations. "4P" is a model for continuous improvement; see Fig. 16. Liker states that most companies are dabbling with change on one level — the "Process" level. Without adopting the other 3 P's,[27] companies will do little more than dabble because the improvements and intelligence behind the process improvement will remain largely unknown to the rest of the organization, and thus the change will not be sustainable throughout the company.

The 4P Model explains that the organization undergoing change must be able to develop and restructure itself. This requires double-loop learning, i.e., the

[27] The other 3 P's are 1) Problem Solving, 2) People and Partners, and 3) Philosophy. These three, together with "Processes", form "the 4 P's".

organization must understand why the actors behind the process changes have reached their list of measures that are needed. [28]

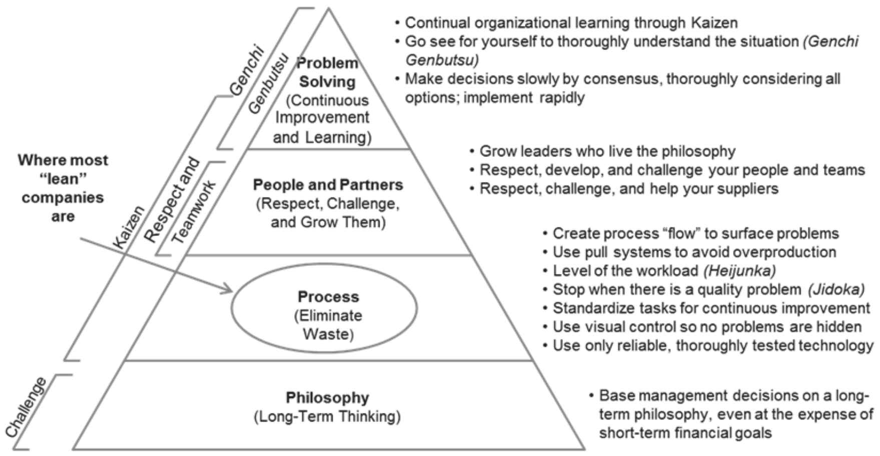

Problem Solving (Continuous Improvement and Learning)
People and Partners (Respect, Challenge, and Grow Them)
Process (Eliminate Waste)
Philosophy (Long-Term Thinking)
Genchi
Genbutsu
Respect and
Teamwork
Kaizen
Challenge
Where most "lean" companies are
• Continual organizational learning through Kaizen
• Go see for yourself to thoroughly understand the situation (Genchi Genbutsu)
• Make decisions slowly by consensus, thoroughly considering all options; implement rapidly
• Grow leaders who live the philosophy
• Respect, develop, and challenge your people and teams
• Respect, challenge, and help your suppliers
• Create process "flow" to surface problems
• Use pull systems to avoid overproduction
• Level of the workload (Heijunka)
• Stop when there is a quality problem (Jidoka)
• Standardize tasks for continuous improvement
• Use visual control so no problems are hidden
• Use only reliable, thoroughly tested technology
• Base management decisions on a long-term philosophy, even at the expense of short-term financial goals

Fig. 16. 4P Model of Lean Transformations. (Liker 2004).

3.5 Transformation and Organization Levels

In large-scale product development organizations it is important to have a system-wide view of the whole organization (Rand and Eckfeld 2004). Fig. 17 is a high-level model of a large product development organization with different levels of operation. An organization consists of business units and/or development areas. The (embedded) software project teams are connected to many other parts of the organization, i.e., many development areas, and even to parts outside the business unit.

In embedded systems development, more efficient workflow can possibly be achieved, for example, simply by co-locating the related software and hardware developers (Waters and Bevan 2005).

[28] The "4P" model also fulfills all the attributes that Johnson (2007) has defined for a *Complex Adaptive System* (see Chapter 2.6.5.).

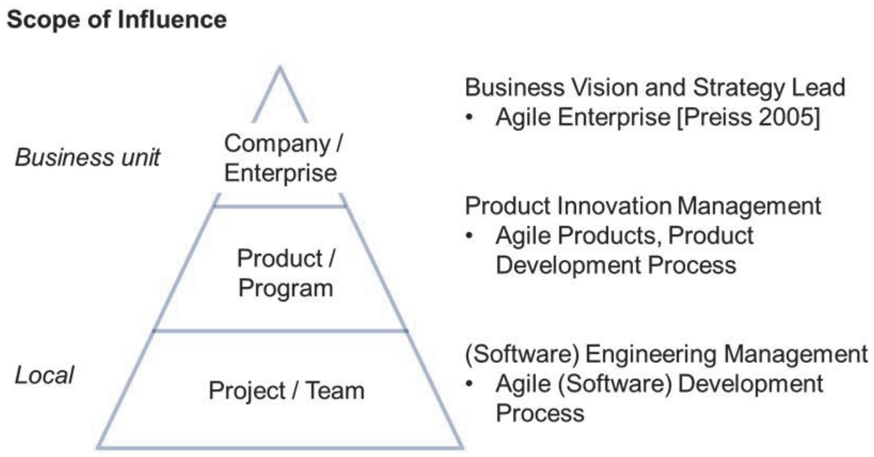

Scope of Influence
Business unit
Local
Company / Enterprise
Product / Program
Project / Team
Business Vision and Strategy Lead
• Agile Enterprise [Preiss 2005]
Product Innovation Management
• Agile Products, Product Development Process
(Software) Engineering Management
• Agile (Software) Development Process

Fig. 17. Levels of Agility of a Product Development Organization, adapted from Yourdon (2002).[29] (Paper III).

Fig. 17 illustrates the different levels of an *Agile Enterprise* where agility needs to be deployed. The reason why we choose to have a triangle in this picture is that a triangle is a typical way of representing an organization: organizations have more people on the Project/Team level and fewer on the Company/Enterprise level.

Failure to transfer the complete organization leads to problems such as water-scrumming. Whether there are actually three levels that need transformation or more may depend on the subject organization and its size. Below we discuss all these levels: the Team/Project level, Product/Program level, and Company/Enterprise level.

3.5.1 Deployment Only on Team Level

Before agile deployment in our organization, some teams had applied Scrum on the respective team level, despite the surrounding organization, which was working according to a waterfall software engineering model. According to some informal interviews we carried out in early 2007, the deployment of Scrum only on the team level had led to some problems because the overall program decision making was not synchronized with the team decision making. As an example, teams were making up priorities since no absolute prioritized list of requirements

[29] Yordon's original figure illustrates Business Process Re-engineering, listing Top, Middle, and Bottom. (Business process re-engineering redefines the question "What business are we in?")

was available for them, they were not able to drop less important features, and they did not actually have a *Product Owner*. In brief, the Scrum teams were lacking external program-level guidance that would have supported the teams' operation in Scrum mode. In Scrum, the *Product Owner* makes the decisions (Schwaber and Beedle 2002). In a large organization, especially when the use of Scrum has originated within the development team itself, the *Product Owner* may be hard to find. In practice this often results in the use of a proxy instead of a real *Product Owner*. Sometimes there has been a solution that the program manager has a dual role, also acting as a *Product Owner*. Experiences with this kind of solution have not been good (Paper IV). In these cases, the program manager was pushing the deadlines to the Scrum teams and not prioritizing the backlogs so the Scrum teams had to invent the priorities themselves.

Even in cases where the teams were able to find a good proxy as a product owner, such as a former lead developer, the lack of formal authority sometimes became a problem. The proxy product owner could set the priorities and do the scoping, but when projects started to request traditional deliveries, requiring a certain functionality by a certain date, the limits of team-level agility became very clear.

3.5.2 Deployment Only on Software Program Level

Few published experiences on scaling agile existed at the time when we decided to deploy agile within the program model in spring 2007. Eckstein (2004) describes how communication can be solved in a large agile program. Leffingwell describes a practical approach to the question in his book *Scaling Software Agility* (Leffingwell 2007). Few research papers exist on this subject (Lindvall 2004), with some more journal articles on experiences (McMahon 2005, Highsmith 2005), but, according to our experience, every company or organization needs to find out its own adaptation that is based on its own business, operating model, and needs. Shortly after our decision was made, Ken Schwaber published his third book, *Enterprise and Scrum* (Schwaber 2007), which made us believe more strongly that we were on the right track. (Paper IV)

The model we used to scale Scrum to the program level is presented in Appendix E. It is based on the understanding that if Scrum is a control loop (Schwaber and Beedle 2002) then one can scale agile by having *nested control loops*, just like in the *Sashimi Model*. In our pilot case we had 500 developers and five different levels of nested loops, i.e., scrum-of-scrum-of-scrum-of-scrum-of-scrums. The pilot case was a success, so the approach was extended to the whole organization.

We sometimes hear claims that scaling agile is the last thing to do (Kähkönen 2004). When Scrum is scaled up, the practical advice is to use *Scrum-of-Scrum Masters* instead of program managers, a single *Product Owner*, and a single backlog for the whole program (Schwaber 2004).

Despite all this we decided to take a different approach. We saw Scrum as a software team management method affecting mostly the team level, although deploying agile would probably have had an impact on almost everything we do. In our organization we had teams serving many programs, and programs consisting of many teams, so we decided to use two sets of backlogs on the team level and two sets of backlogs on the program level (Fig. 18). The teams or projects serving several programs were picking up their user stories from several sources and prioritizing those.

We decided to call the topmost backlog on the program level a *Program Content Backlog*, containing the highest level of requirements. The *Program Content Backlog* would contain all the requirements defined for a program when a program was started in priority order. During the program, the requirements could be changed and re-prioritized, if necessary. For each program release, a *Program Backlog* based on the requirements would be created, containing all the user stories needed in the next release. The *Program Backlog* would be different from the *Program Content Backlog* in the sense that it would contain all the program tasks (such as test environment creation and documentation) that would be needed in accomplishing the program mission. No Gantt chart would be needed any longer in our programs (Fig. 18). (Paper IV)

On a team level there would also be two sets of backlogs, namely the *Scrum Team Backlog*, containing user stories for that team (just like the *Product Backlog* in Scrum), and the *Sprint Backlog*, containing tasks (as in Scrum) (Schwaber 2004). The *Scrum Team Backlog* was needed since we had some teams that were working for multiple programs (although we decided that we would aim at dedicated resources for programs in the future). All these backlogs needed prioritization, so a hierarchy of *Product Owners* was needed to support a single program. We decided to call them the *Program Product Owner*, who had the ultimate responsibility for a program, and *Team Product Owners*, who were following orders from the *Program Product Owner*. And we decided to have a *Program Manager*, responsible for the program execution and playing the *Scrum-of-Scrum Masters* role in the (team) *Scrum Masters*.

We tried to maintain and multiply the Scrum ceremonies (Schwaber 2004) as much as possible: each team would have its daily scrums, sprint planning events, retrospectives, reviews, and demos. Each program would have a program scrum once or twice a week. We decided to repeat the other Scrum ceremonies on the program level too: release planning would be participated in by all *Scrum Masters*, as well as the *Program Manager* and a *Program Product Owner*. A program release planning meeting would always precede the team-level sprint planning meeting so that the team *Scrum Masters* could familiarize them with the user stories in that program release before the team-level sprint planning. After the release there would also be a review and retrospective sessions on the program level, as well as a demonstration of the working software.

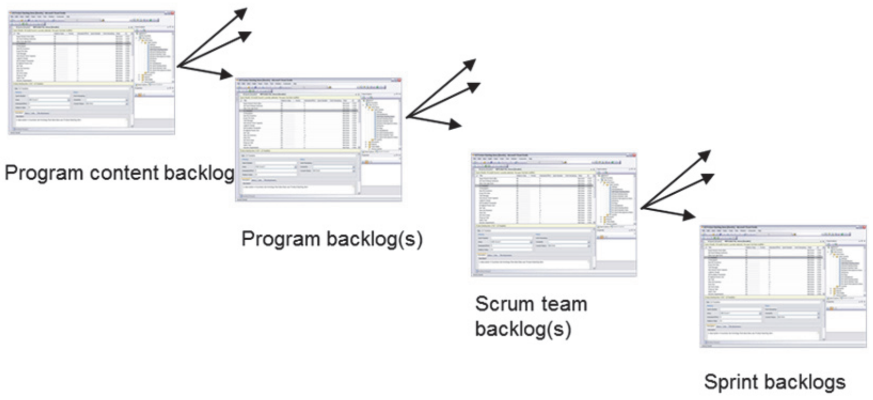


Fig. 18. Program and Team Backlog Structure Consisting of Four Levels of Backlogs. (Paper IV).

Agile software projects need active content steering, which is different compared to traditional program management, and thus many of the old traditional program management methods become inapplicable — or worse — lead to the wrong outcomes. One outcome of this was the water-scrumming problem: the incompatibility of traditional ways of managing programs that were agile.

Water-scrumming was seen in practice as:

– *traditional programs* and portfolio management have expectations for agile projects to give waterfall-like commitments,

- *traditional programs* push aggressive targets and manage dependencies through plans with a command and control mentality,
- a flat set of MUST requirements with a change request process do not allow scoping as it should happen in agile,
- working with an old requirements database with a heavy process considered anti-agile: in agile requirement management, project management, and tracking are inseparable,
- immature agile teams, in turn, may interpret agility incorrectly as "*no* commitments, no plans" beyond the next sprint,
- true agile teams are self-organizing, pull the work, and manage their own dependencies,
- traditional programs expected only one completion date for a requirement and thus gave no means for agile projects to communicate release iterations by releasing one requirement several times for different configurations or to identify the level of completeness (basic/advanced functionality released) or to communicate the level of ambition (minimum/basic/advanced) the release was targeted at.

Scaling agile from teams up to the program level solved some problems but created some new ones too. If *Agile Methods* are used only on the team level the dependencies that teams have between each other when developing large software systems are left unsolved. The cadence of development in parallel teams may be different; top-level backlogs help to get into the same cadence. The software engineering teams also need input into their decision-making (or actual decisions) and need to be synchronized with other activities. (Paper IV) Additionally, projects need to make decisions on reducing the scope i.e. dropping the features planned originally to be developed in that project — agile does not give you "change for free" in the sense that you are able to add content to programs without any penalty but expects strict content scoping and backlog grooming to happen systematically throughout the whole program in order to keep to the due date.

3.5.3 Deployment on Enterprise Level

The author of this thesis had already proposed a model for scaling agile at Nokia to the enterprise level in 2007 (see Appendix D), but this model was not accepted. The proposal was challenged, e.g., because of the lack of an error management process. While the organization was adopting Scrum on the team level in 2007, the researcher was having discussions on how to scale agile with the process team of the organization. Eventually it took until 2010 to gain a sufficient understanding of *Agile Methods* in the organization for large-scale agile transformation to take place. At that time, Leffingwell's book on Scaling Software Agility was already available, which helped the deployment tremendously.

There is little information available about what *Enterprise Agility* should be. The preceding chapters have defined and discussed the emphases of different definitions of agile that cover the assumed benefits of *Agile Methods*. The *Agile Aspects* that an ideal *Agile Enterprise* could possess were also described.

Table 8 discusses the *Agile Aspects*, and compares them with emerging actual experiences of large-scale agile. Emerging actual experiences are described for each aspect.

Table 8. Reflections from actual experience[30] of Agile Aspects.

	Agile Aspects	Actual Experience
1	Strategic Agility	How agility is successfully used to compete in existing markets and to transfer to new areas. Example: resource/work fluidity. How to make a balance between agile programs concerning what is truly important. Example: not everything that was implemented was put into the final product. Higher-level prioritization and decision making would have prevented the implementation of this kind of functionality in the first place.
2	Business Agility	How business benefits from agility. Example: improved coordination between product and portfolio management and development needed. Feedback from progress in development is immediately reflected in product and portfolio decision making. Practical example: discussion of priorities by portfolio decision board.
3	Agile Organization	How flexible organization structures are. Example: cross-functional teams vs. traditional component teams. Organizational silos preventing co-operation where it is needed; two programs developing exactly the same thing.
4	People Agility	New skills needed from people to work and deliver code in increments. Teamwork capabilities emphasized. Example: people confessing that after ten years they feel that they are working as a team for the first time.

[30] Based on experiences in the year 2008 of the agile mode described earlier.

	Agile Aspects	Actual Experience
5	Tools Agility	New types of tools needed that are capable of managing work done in iterations and increments: one must be able to prioritize, allocate work for an iteration and increment, and also mark work that has been completed. The best tools generate reports as by-products. In agile, requirements management, project management, and tracking all melt together as one discipline. Example: Excel-based tools developed to fill the gap.
6	Organization Culture	Rapid decision making essential. Attention to what actually gets done, and improvement happening at the lowest level. Agile requires a new kind of visibility and trust. Example: visibility was created via one general web portal for all programs.
7	Agility of the product being built	Custom-tailorable or modifiable products. Ability to reverse decisions already made if better solutions discovered. Product that is suitable for many purposes. Extendability. Example: downloadable services/extra software.
8	Agility of payoff functions	No balance of investments. People were working for designated areas. Example: new functionality in niche areas when primary functions were lacking performance.

The *Agile Aspect* of *Strategic Agility* refers to the organization's ability to re-invent its core business. In any business situation it is crucial to focus scarce resources correctly. Without proper high-level prioritization and decision-making resources could be used to develop something that is discarded in the end: this is a form of waste. An *Agile Enterprise* that implements its *Strategic Agility* correctly focuses its effort on delivering products that are innovative and bring positive attention from the market.

The *Agile Aspect* of *Business Agility* can be summarized as meaning how a business benefits from agility. In practice there is more and improved feedback on product and portfolio management. High-level backlog priorities are discussed by decision boards.

The *Agile Organization* means how flexible the organization structures are. In practice organizational silos can hinder cooperation where it is needed. In the worst scenario people in one organizational silo do not even know what is done in another organizational silo, and may end up developing exactly the same thing. Joint release planning, improved visibility, and improved communication in the *Agile Organization* should prevent these kinds of problems. In some development areas a special domain or other knowledge may be needed that does not allow full organizational flexibility. Depending on the development team type, it may be a very specialized team or a more generalist team.

People Agility means the new skills and knowledge that people need in order to work in an *Agile Enterprise.* In practice this can be validated by studying whether people work in an agile manner or not. Whether they feel that they are really working as one team or not is an example of such measure.

Tools Agility means all the new capabilities that are needed to support the agile way of working. When agile deployment was brought to the program level in 2008 a set of Excel-based backlog tools was developed to fill the gap.

Organization Culture means that agile values and principles are accepted as being part of the way the organization works. An example of this is the value of visibility that *Agile Methods* promote. A web portal (or dashboard) can be created to have such visibility organization-wide. Note that there needs to be one common shared *Agile Organization* culture, but the sub-organizations (e.g., Scrum teams) may have their own flavor added.

Agility of the product being built means more custom-tailorable or more modifiable products. After-sales deliveries or software updates are examples of such flexibility.

Agility of payoff functions means how flexible our investment-making and reward systems are. An example of lacking this kind of flexibility in investment planning can result in the allocation of resources to new projects in the same way that those were allocated in the past. Thus areas that have grown in interest may lack resources while some other areas may, in practice, have too many resources allocated to them.

The *Agile Aspects* cover the agility of an *Agile Enterprise*, but the focus is internal to the enterprise and does not take the external environment into account. The model can be extended so as also to include the end-to-end value chain, including subcontractors, third parties, and ecosystems. However, that falls outside the scope of this dissertation.

3.6 Model of Agile Transformation

Some people claim that in order to succeed with agile adoption, freedom and decision-making power at the lowest levels of the organization need to be increased (Benefield 2008). Having more freedom and decision-making power means changes in the surrounding structures.

If an *Agile Method* such as Scrum is successfully applied in only one team, that team only can benefit from faster time-to-market. Any larger-scale improvement would need some synchronization between the parts of the organization, and speeding up time-to-market for the whole complex networked system such as a large software organization requires the improvement of the whole. But at the same time the question of how to empower the software teams sufficiently and lead them in an agile-compatible way remains.

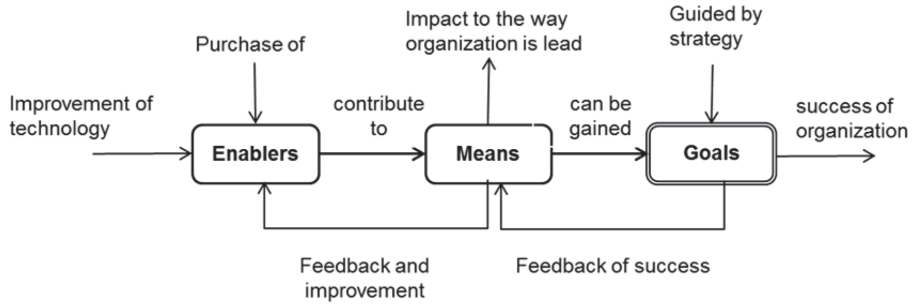


Fig. 19. Goals, Means, and Enablers as Model of Organizational Development.

Goals, *Means*, and *Enablers* provide a framework for such discussion in the transforming organization. Fig. 19 presents a generic framework for organizational development. Agile development is based on a set of technical improvements (such as an increase in computing power that enables continuous integration and deployment to take place or new tools for better test automation) as well as practices (*Means*), e.g., how fast feedback, increments, and iterations can benefit development (as listed earlier in Chapter 2). When commonly shared these tools and principles can help the organization to reach the *Goals* it pursues with agility.

The *Enablers* may be not only tools, but also other enabling factors. One example of this kind of enabling factor could be the co-location that is widely promoted by the agile development literature (e.g., the Scrum Room in Scrum, the XP Facilities Strategy, or the *Agile Principle* concerning face-to-face conversations). (Lehtonen 2009, Schwaber and Beedle 2002, Beck 2004)

Typically, these *Enablers* have such a nature that they need investment and work to be present. In Nokia, we have had several change projects to put these *Enablers* in place — or to further improve the *Enablers*.

The *Means* describe how our *Agile Enterprise* should work. The idea of how an organization should work is the foundation for all leadership, i.e., the understanding of the relations between causes and consequences impacts on the conclusions drawn and the actions the leaders make. Without a unity of leadership there is no way to transform the organization into agile mode.

The importance of leadership in driving the agile transformation may lead to an assumption that a better way to drive agile deployment is top-down rather than bottom-up.[31] This statement would, however, need further validation, and is thus left for later research. Nevertheless, it should be obvious that systematic deployment results need alignment at all levels of the organization — clear and frequent communication, transparency of what gets deployed, and visibility of the current status related to deployment *Goals*.[32]

Goals, *Means,* and *Enablers* form a model for how an *Agile Organization* works. This model may be shared, or it might be different for every member of the organization. *Complex Adaptive Systems* behave differently, depending on how much the same models are shared. Thus from the CAS frame (Miller and Page 2007) point of view it makes sense to study how much the same mental models of how the organization works are shared within the organization, and how the different understanding of how the organization actually works impacts on people's behavior and the behavior of the whole system, i.e., the organization. This topic is discussed more thoroughly in Chapter 3.8.

3.6.1 Comparing the Agile Transformation Model to the Learning Organization Model

Goals, Means, and *Enablers* can be compared with the single- and double-loop model of Argyris and Schön (1978); see Fig. 5. Here the *Goals* and *Means* are comparable to *Governing variables.* The difference here is that *Means* cannot be implemented without *Enablers*: if the underlying technological capabilities are missing the *Means* will not make sense. A practical example of this is the idea of continuous integration. Here, the *Goal* is the improved quality and faster feedback. *Means* is the idea of continuous integration itself: that the integration happens seamlessly, and every time some piece of code is available. But this requires an

[31] Notice that lean transformations are typically driven top-down.

[32] This means change projects that put *Enablers* in place must be managed like any other (agile) projects.

Enabler, which is the continuous integration machinery: without the proper tools the idea (*Means*) becomes invalid.

The theory of *Learning Organizations* states that the *Governing variables* or *Underlying assumptions* must be known for double-loop learning to happen. This means that there needs to be a certain level of agreement of *Goals*, *Means,* and *Enablers* in the agile context for *Agile Methods* to be used. The *Agile Framework* proposed can be used for a discussion, where different opinions about how the organization should work can be aligned.

Learning is based on loops. It would be natural if putting *Agile Methods* into use also followed such loops. An agile (incremental and iterative) approach to agile deployment is thus suggested. As an organization learns from one model based on *Goals, Means,* and *Enablers* it can replace that with another model. Salo (2007) proposes that organizations that value continuous organizational learning would alter their traditional *Software Process Improvement* initiatives in this more continuous direction. Fig. 19 presents two separate feedback loops in order for such learning and development to take place.

3.6.2 Comparing the Agile Transformation Model to the LAI Model

The model presented in Fig. 19 can also be compared with the LAI model for large-scale lean transformation (Fig. 4). It is noticeable that there are no *Enablers* in the LAI model. Stories about how the Toyota Production System was developed (Ohno 1998) also include stories on how new types of molding systems and other physical equipment allowing new operational modes were created. Building such new equipment can be compared to building *Enablers* for agile transformation.

The LAI model consists of series of nested iterations (*short-term cycle* and *long-term cycle*). Focus on the *value stream* suggests that the model is optimizing cycle time (or time-to-market). But lean implementations may also optimize for several other parameters. Reinertsen (2009) suggests optimization for economic factors such as cycle time (time-to-market), product cost, development expense, product value and risk.

Goals, *Means*, and *Enablers* expand in the enterprise setting into a complex network of relations. Learning happens in iterations, as presented in the LAI model, but the focus with *Goals*, *Means*, and *Enablers* is not on these learning loops but on the relations between the *Goals*, *Means*, and *Enablers*. Maybe when *Agile Methods* become more common these relations will become more evident, and will not need to be communicated like today. Then the solutions and options will be narrower, and a more straightforward approach — like in the LAI model — from vision to deployment might be sufficient. But that is not the case today — *Agile Methods* on a large scale cause a lot of questions and confusion. In particular, the role of *Enablers* is not understood, and needs to be communicated. *Goals*, *Means*, and *Enablers* provide a thinking framework for setting a vision for the *Agile Enterprise* that can then be implemented via learning iterations.

3.7 Applying the Agile Transformation Model to Transforming the Enterprise

It is typical that in an *Agile Enterprise* many *Goals, Means, and Enablers* can be found. It is also typical that *Goals, Means,* and *Enablers* have complex relations between each other. But for a successful transformation we must have unity of understanding about what we want to put in place, i.e., there must be leadership unity on how we want to change and operate and what the *Enablers* are.

That is, while there may be many *Enablers*, and many *Means* in achieving organization-level agility, the *Goals*, *Means*, and *Enablers* should be commonly agreed and driven in order to achieve full-scale change. It can also be learned from different *Complex Adaptive Systems* adaptation models (Miller and Page 2007) that a certain unity and synchronization in what needs to be achieved on all levels of the organization must be in place. Paper VI, entitled "*Agile Transformation Study at Nokia — One Year After*", confirms how the opinions in the organization under study are becoming more unified in one year's time. This can be compared to any *Complex Adaptive System* where *Agents* are learning from each other and becoming more aligned with their *Intention*.

The generic model presented in Fig. 19 needs to be adapted for each organization, as it is organization- and situation-specific. Fig. 20 presents one possible use of this model. For the sake of simplicity, the feedback loops have been removed from this figure.

The *Goals* in Fig. 20 are retrieved from Forrester's study of the top five reasons for organizations to conduct an agile transformation (Schwaber 2007). The framework presented here is based on actual experience, and thus the *Enablers* on the left-hand side are different agile (Scrum and XP) practices that have been put into practice at Nokia. These are all practices that teams are using. The list of *Enablers* is sufficient but not comprehensive. There could be, e.g., *Pair programming. Pair programming* is not listed, as that has not been conducted on a large scale at Nokia, or at least, not yet. The *Means* have no value as such, but they are mechanisms to gain value from different *Agile Practices*. These are common for the whole organization.

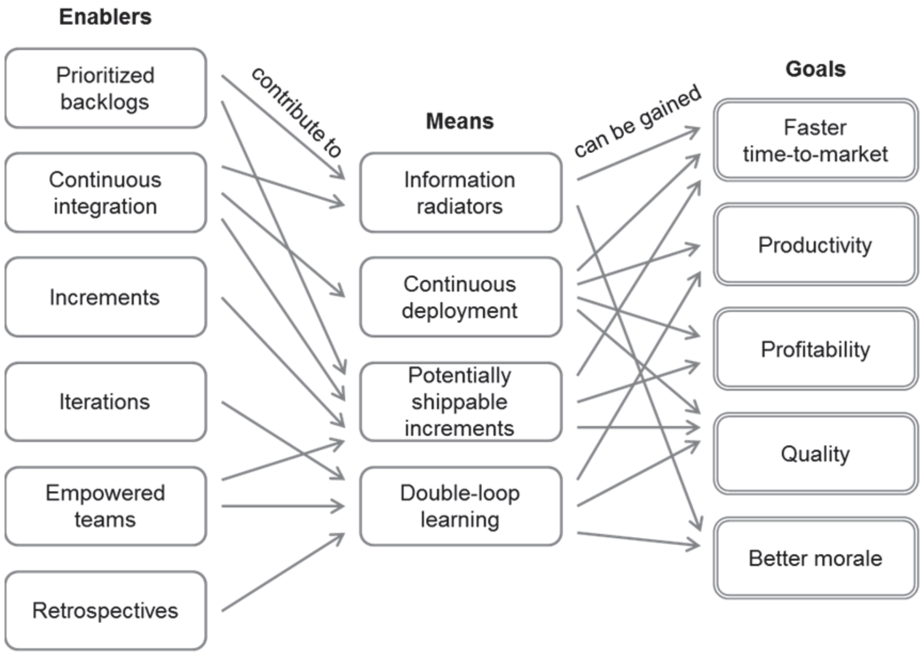


Fig. 20. Example of Goals, Means, and Enablers explaining how an Agile Organization can be deployed. For the sake of simplicity the feedback loops are not shown.

3.7.1 Faster Time-to-Market

Agile Methods state that traditionally project schedules are prolonged, because: 1) scope management is not tight because projects are so long (the *cone of uncertainty* means estimation can be improved by shortening the range of estimation; shorter increments can improve scope management) (Reinertsen 2009, Leffingwell 2011); 2) time is wasted on doing secondary or unnecessary tasks (Larman 2003); 3) time is wasted on waiting for feedback or information (Reinertsen 2009); 4) tasks are done several times, i.e., time is wasted on error corrections (Poppendieck 2003); 5) people work on multiple tasks at the same time (Poppendiek 2003, Reinertsen 2009); 6) projects are run in parallel, rather than in sequential mode (Reinertsen 2009), and 7) people are more than the length of a school bus apart from each other (Cockburn 2006).

Faster time-to-market can be achieved by: 1) having a clear focus or priorities on what gets done; 2) chopping the projects into very small pieces or increments that are close to product quality and that can be individually shipped; 3) by achieving a higher level of test automation and integration and even delivery effort, and 4) by developing capabilities to perform testing parallel to development. The idea here is that instead of one very large project that is getting prolonged we keep the project better at hand by chopping it up into increments that are smaller and less complex so the scope management gets easier and visibility improves. The actual speeding-up of the development comes from eliminating the unnecessary or unwanted tasks by better communication and alignment, and automation of all tasks that can be automated.

That is why in Fig. 20 the time-to-market is mostly contributed by: 1) information radiators (providing full visibility to the actual status of projects: these are similar to Scrum burndown charts and demos for a large organization); 2) continuous deployment (full automation of product testing and shipping), and 3) *Potentially shippable increments* (as mechanisms for better scope and content management in a project — potentially shippable increments are for a large organization what sprint releases are for a Scrum team).

Information radiators are important, because without organization-wide radiators there cannot be full visibility on what projects and Scrum teams are working on. Potentially shippable increments give focus and make possible joint planning and scope management project-wide and organization-wide. Openness and focus are two of the Scrum values (the other three are respect, courage, and commitment) (Schwaber and Beedle 2002). Continuous integration and continuous deployment facilitate fast project- and organization-wide *fast feedback*. According to Reinertsen, one should optimize for development speed — and fast feedback is what allows us to operate processes effectively in a noisy environment (Reinertsen 2009).

An organization can base its information radiators upon continuous integration and a repository of prioritized backlogs. Information radiators are an automation of what used to be done manually for each project in the form of a Gantt chart or project progress report.

Continuous deployment is an automation of product delivery operations. This part of Fig. 20 could also be drawn in different ways. Continuous integration could also be here, listed under *Means* (as they replace schedule-based integration and automatic quality checks also replace manual quality gates), but was put into *Enablers* as something that teams are responsible for doing and because continuous integration is also an essential *Enabler* for potentially shippable increments (one cannot have potentially shippable increments if the integration and testing machine is not able to go through many iterations before shipping so that there will be sufficient time to make the necessary corrections).

Potentially shippable increments need the alignment of development effort within that increment. This is done by backlog prioritization (and backlog grooming) and having increments defined. The role of continuous integration as a key enabler in terms of making the increment shippable (i.e., tested for quality) was explained earlier. Empowered teams (or cross-functional teams) are good, but maybe not the necessary enabler — it is easier if the team does not need to wait for permission, information, or dependent code to make the functionality complete.

3.7.2 Productivity

The automation of development efforts contributes to productivity; that is why continuous deployment contributes to productivity most. Double-loop learning contributes to productivity too, as the more competent organization becomes faster at performing similar tasks.

Double-loop learning grows out of iterations. All human learning is based on our ability to go through iterations (Lehrer 2009). Empowered teams discuss alternatives for solutions and learn why certain approaches are better than others, thus being Model II organizations (Argyris 1993), instead of being just told what to do, like Model I organizations. Reviews enable reflection to take place on what was done and retrospection on how it was done — and improvement happens when people are able to change the how part.

The list of *Enablers* in Fig. 20 represents the current understanding in the organization under research of what agile *Enablers* contribute to productivity. There are other factors that contribute to productivity that need not necessarily be interpreted as agile. One example of these types of factors is *Reuse of code and development effort.* If an *Agile Organization* would like to maximize the productivity aspect, it should focus on the *Agility of the product*, as that would enable wider use of the product to be made and potentially lead to a longer product lifecycle.

3.7.3 Profitability

Profitability is mostly affected by the automation of effort, i.e., continuous deployment. Potentially shippable increments contribute to profitability most if the organization is able to prioritize the most value-added work first.

It should be noticed that most profits come from successful new disruptive innovations that allow organizations to go to new blue oceans (Kim and Mauborgne 2005). For example, Reinertsen (2009) states that product development organizations should take rational risks and exploit the variability for better payoffs, as the payout functions may not be symmetric; that is, the value of success can be much higher than the cost of failure. Traditional working and safe product solutions only give decent profit margins when newer technology or riskier products can lead to breakthrough products.

3.7.4 Quality

Continuous deployment and potentially shippable increments improve quality most, as these enable parallel testing and fast feedback to take place on the organizational level. Double-loop learning improves quality as well, as people will learn better implementations. (Korhonen 2012) The ability to produce code in several iterations helps in reaching better quality.

3.7.5 Better Morale

Full visibility into how teams are doing creates peer pressure for performance. This is another reason why Scrum teams are so effective (daily scrum meetings provide visibility on daily progress). Gary Hamel (2007) has proposed organization-wide visibility for performance as a mechanism for new leadership. Here, visibility is provided with information radiators, both based on work marked as done in backlogs and code that is integrated and tested.

There is also an assumption that double-loop learning would lead to better morale. This assumption is common — all known leadership methods that are based on feedback according to performance include this assumption.

The framework presented in Fig. 20 is by no means complete, but represents the current understanding of the research organization. It would be natural to question, e.g., the missing *Enablers,* such as pair programming. Pair programming would contribute to *Means* of fast feedback, resource fluidity. Fast feedback would support the *Goals* of quality and faster time-to-market and learning would support the *Goal* of resource fluidity.[33]

The arrows that connect different *Enablers* to *Means* and to *Goals* can be interpreted as force vectors. In order for the *Means* to work properly, all contributing *Enablers* must be in place and the practices followed up. The organization cannot have other contradicting *Means* in place, as these will quickly become inhibiting factors. An example of such contradicting *Means* would be, e.g., "Plan well, do once" which would contradict the idea of working with increments and iterations or "Failures shall be punished", which would contradict the *Means* of double-loop learning.

The *Agile Framework* above suggests that large-scale agile adoption should take a comprehensive approach to the development of the organization. That means one should try to implement different *Means* and measure their impact on the system. The *Goals* may be conflicting: e.g., it might be impossible to get both resource fluidity and co-location at the same time. The organization must choose which *Goals* to pursue.

Because of the possibility of having conflicting *Goals*, *Agile Organizations* need to specifically develop one interpretation of agility, implemented as their *way of working*. An organization must select what kind of agility it would like to implement. This *strategic business agility* has not been extensively studied, and nor has the relationship between *Agile Frameworks* and the business strategy.

There needs to be a certain alignment between the *Goals*, *Means,* and *Enablers*. Failure to achieve this alignment between *Goals, Means* and *Enablers* results in the failure to scale the usage of *Agile Methods* to a large scale.

It should be noted that Fig. 20 provides just one view on agile deployment. Fig. 20 lacks, e.g., the need for agile coaches on the team level.

[33] Pair programming has been found to contribute both to cross-organizational learning and quality. In the study of pair programming practice in education, the study of Williams, Wiebe, Yang, Ferzli, & Miller at North Carolina State University in 2003 showed superior results on graded assignments, increased satisfaction/reduced frustration, increased confidence in their project results, and a reduced workload for the supporting technical staff. As well as the superior grades, superior designs and the utilization of more sophisticated programming techniques are also expected. The study focused on 112 students who were taught solo and 87 in pairs in Fall 2001; the groups were formed with equal programming backgrounds. Boehm and Turner (2004) found out that “for well-jelled, stable teams pair programming appears about equivalent to inspections in adding about 10-15% to effort and eliminating about 60% of the defects. But pair programming’s 45% reduction in schedule is extremely valuable whenever software is on the critical path.”

3.8 Comparing the Agile Transformation Model to Other Agile Enterprise Models

In Chapter 2.5 several models of an *Agile Enterprise* were presented. All of the models in Chapter 2.5 are ideal models, i.e., a target state, how the enterprise should ideally operate. The models do not necessarily state how this target state can be achieved.

Goals, Means, and *Enablers* can be used to draw a transformation model to reach each of the described target states, e.g., if an enterprise wants to deploy Leffingwell's *Agile Enterprise* Big Picture, described in Fig. 1.

Potential *Enablers* may include:

- organizing teams as Scrum teams,
- nominating persons to mentioned agile roles,
- a tool to keep team, feature and global backlogs,
- a list of' *Investment Themes*,
- establishing *Portfolio Vision* and *Architectural Runway*.

The *Means* could be:

- asking teams to follow Scrum and deliver in sprints,
- starting agile release planning consisting of four sprints and one hardening sprint,
- trying to deliver potentially shippable increments after each release,
- splitting *Investment Themes* into *Epics*.

The list of *Goals* could include:

- improved visibility on what is being produced with each sprint and release,
- better focus because of prioritized backlogs at all levels,
- improved speed of operations because of better management of internal queues.

3.9 Measuring the Unity of Views of the Followed-up Model in an Enterprise

In order to successfully deploy *Agile Methods* on a large scale, the organization must have a unified understanding of how the system is supposed to work, i.e., share the same *Intention* (as defined in the CAS framework). The *Intention* focuses on the goals of the agents. In this way it is linked to the adaptation of the model, i.e., the *Agents* may have the same or different goals and they may have the same or a different understanding of the model to be followed up. In such cases the micro-motives and macro-behavior fail to align. In social systems, we may want the *Agent* behavior to aggregate in such a way that the system achieves some goal (Miller and Page 2007). *Intention* can be studied by researching whether the understanding of agility is aligned in the subject organization.

This means that the organization must have a shared and communicated model, a description of its agile way of working.

3.9.1 Examples of Misalignment in Agile Understanding

Agile deployment in our organization was hampered by some questionable sidetracks. Some practitioners misunderstood agile as denoting hacking; some associated it with no obligation to document what has been implemented. Some teams were suffering from old habits, like the *Scrum Master* dictating the tasks the team members should be doing. We tried to combat these false understandings by arranging public lectures about agile in all our development centers. (Paper IV)

One of the management concerns when transitioning to agile was the conflict between long-term and short-term planning. One of the threats that was seen was that backlogs and agile planning would give us only short-term visibility. We had many experienced program managers who were able to estimate the size and duration of new programs on the basis of their earlier experiences, and we saw no reason to leave that experience unutilized. Thus we decided to keep both a long-term vision based on experience and a short-term "strategy" based on backlogs. This has resulted in more time spent on actual planning but also in improvements to the plans themselves. (Paper IV)

For practical reasons, some Scrum teams needed to work for several programs.[34] Sharing resources has not been seen as a good option by the programs, as it results in non-steady output levels and makes estimation harder. Programs with fewer dedicated resources have not made very good progress. In fact, the deployment of the agile program model has revealed the overload of tasks on programs and teams and the true velocity and capability of development. (Paper IV)

[34] The term "program" is used to mean a larger group of (interlinked) projects, i.e., an element that is used for scaling up projects.

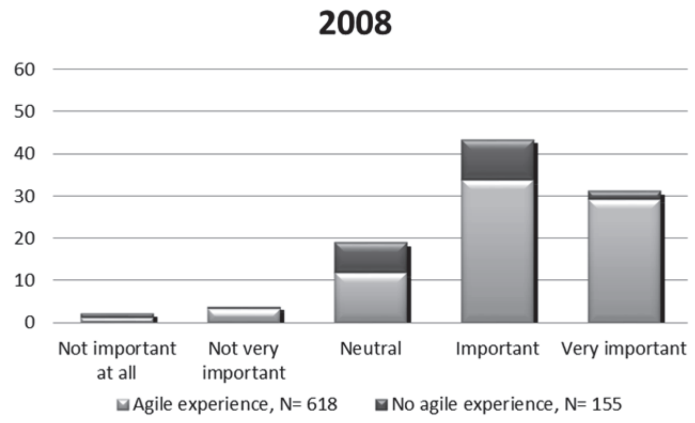
3.9.2 Example of Creating More Aligned Understanding

Scrum teams have a sprint demonstration of working code at the end of each sprint. Programs arrange working code demonstrations of new features too, which has been received with great interest by program customers. Working code demonstrations have also been a visible place to communicate the new agile program mode to our customers and let them really see the change and the fact that they can truly have an influence on the program outcome. Despite the popularity of the program demonstrations, in practice we learned that we should not arrange too many of those but rather do so more sparingly. There should be working code demonstrations no more often than once every three months in a program. That is because arranging a proper demonstration takes time and effort on the part of both the program and the customers who attend. Most programs make enough progress in two months for the new features to really interest the demonstration audience. (Paper IV)

3.9.3 Studying Alignment of Agile Understanding

Two surveys were conducted to study what was happening in the target development organization regarding the perceptions of' *Agile Methods* and their usage in 2008 and 2009.

In 2008 the attitude towards agile development could be considered very positive. For example, 75% of all respondents considered agile as important or very important (Fig. 21, top), and nearly 60% would not like to go back to the old ways of working (Fig. 23, top). The level of satisfaction with agile development within the respondents' own work (Fig. 22, top) was not as positive; only 47% were either "satisfied" or "very satisfied". (Paper VI)



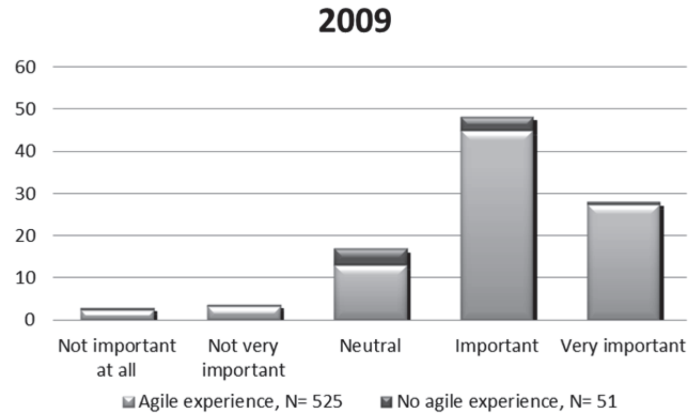
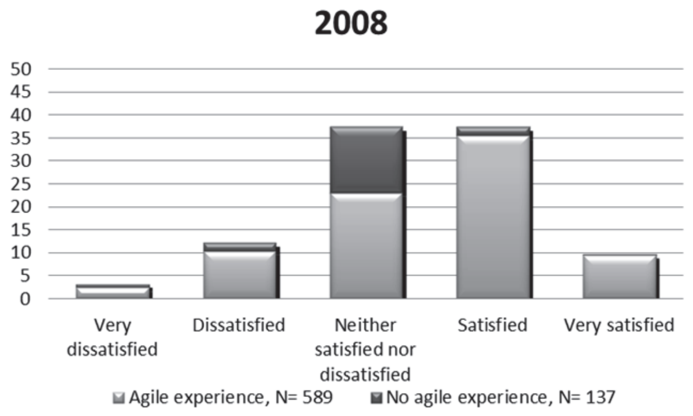


Fig. 21. Statistics for the question, "How important do you consider agile & iterative development in the future?", categorized by agile experience. (Paper VI).

Interestingly, respondents with no agile experience seem to be more neutral in their answers than people with some agile experience. This is clearly visible in Fig. 21, Fig. 22 and Fig. 23.



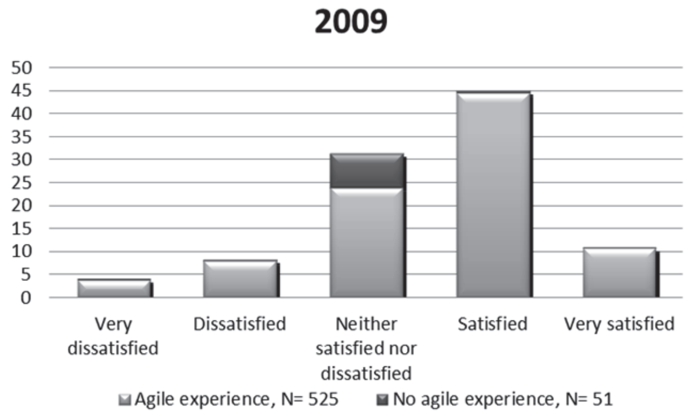
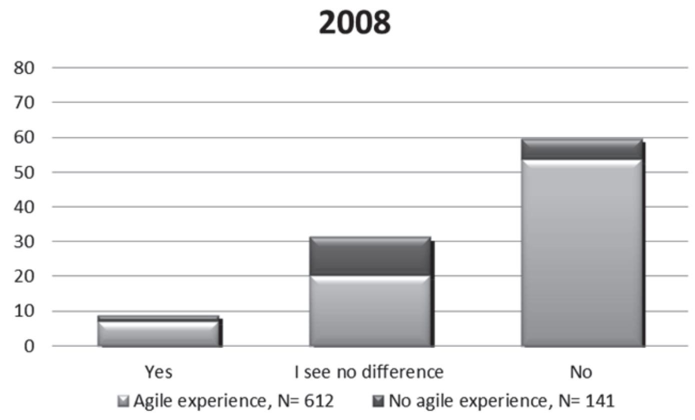


Fig. 22. Statistics for the question, *"How satisfied you are with Agile Methods?"*, categorized by agile experience. (Paper VI).

The result of the survey from the year 2009 shows similar results. As we can see in Fig. 21 (bottom), 76% of respondents think agile is important or very important, and only 7% think it is not. The level of satisfaction with *Agile Methods* seems to have increased slightly: not only are 57% satisfied rather than dissatisfied (year 2008 47%) — see Fig. 22 (bottom) — but also the number of dissatisfied or very dissatisfied persons has decreased and is now 13% (in the year 2008 it was 16%). The number of those who would like to stay agile has increased and is now 67% (in the year 2008 it was 60%) but also the group that would like to go back has increased in size and is now 12% (in the year 2008 it was 9%). This could be explained by the smaller number of neutral responses, as seen in Fig. 23 (bottom).



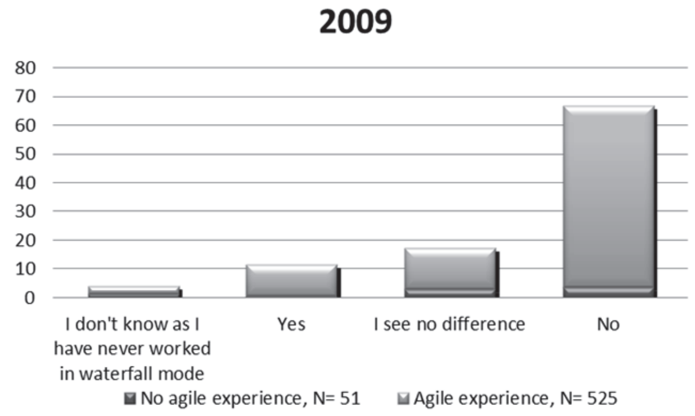


Fig. 23. Overall distribution of responses to the question, *"Would you like to go back to the old way of working?"*, categorized by agile experience. (Paper VI).

The impact of the actual experience of *Agile Methods* on attitudes was studied with the Kruskal-Wallis H test. With the Kruskal-Wallis H test the mean ranks of samples from the populations are expected to be the same, i.e.:

$$H_0: \mu_1 = \mu_2 = \mu_3$$
$$H_A: \mu_i \neq \mu_j$$

The key finding was that all the attitudes (importance of agile development, satisfaction with *Agile Methods*, and whether one would like to go back to the old methods) varied according to how much agile expertise the respondent had in all cases. The Kruskal-Wallis H tests done for the year 2008 and for the year 2009 data gave the same result.

The *independence* of attitude and agile experience variables were then further studied by means of a Chi-Square test and Fisher's exact test (when the sampling distribution was too small for the Chi-Square test). The *null hypothesis* is that there is no relationship between the two variables. This is done by comparing the observed frequencies:

$$H_0 : f_0 = f_A$$
$$H_A : f_0 \neq f_e,$$

where f_0 is the observed and f_e is the expected frequency.

This led to the second key finding, that all three variables that were examined, namely, how important the respondents see agile development as being for the future, how satisfied they are with agile development, and whether they would like to go back to the old way of working, were not independent of the agile expertise.

Attitudes regarding the overall satisfaction with *Agile Methods*, how important *Agile Methods* are considered to be, and whether people would like to go back to the old ways of working did not change dramatically from 2008 to 2009. When the data for the years 2008 and 2009 were compared on a detailed level, it looked as if the opinions of different groups were converging with each other. However, the trend figures showed that the opinions of those who were experienced in *Agile Methods* showed only very little change and maybe were slightly more unified, while the contrast still remained between those who had agile experience and those who did not have. That led to the recommendation that one must try out *Agile Methods* in practice. (Paper VI)

The narrowed opinion-neutral population and significantly smaller response rate in the study from the year 2009 might indicate that those who were facing problems with *Agile Methods* had more probably answered the 2009 survey than those who were content. The fact that the opinions of people with different experience of agility seemed to have converged with each other from 2008 to 2009 was suggesting that opinions changed when people communicated with each other and shared their experiences. The satisfaction with *Agile Methods* might have suggested that the company was making progress on the *New Technology Adoption Curve* (Rogers 2003, Paper VI).

Another analysis was made for the 2008 survey data to see whether *Agile Methods* provided any real improvement from the adopters' viewpoint. Thus, it was first examined how a person's background — i.e., the length of a person's experience of agile and traditional methods — correlated with the variation in the respondents' opinions about *Agile Methods*. From a broader perspective, this provided a way to find out whether agility was making a permanent entrance or not. As a result of these ANOVA calculations, it could be claimed that the longer the experience a practitioner had with *Agile Methods*, the more positively it impacted on their opinions regarding the usefulness of *Agile Methods*. This finding is in line with the results reported earlier. (Salo and Abrahamsson 2008, Paper V)

The detailed studies with data from the year 2008 revealed that the opinions changed, becoming more positive as the respondents' experience of *Agile Methods* grew and this change was large enough to be statistically significant regarding transparency and collaboration. Likewise, it was discovered that if a respondent had more than 3 years of experience of traditional methods this had a negative impact on their attitude towards *Agile Methods*, and that this difference was significant regarding opinions on effectiveness, increased collaboration, work being more organized/planned, and agile enabling errors to be detected earlier. (Paper V)

Although a link was found between longer experience of traditional methods and a negative attitude towards *Agile Methods*, further investigation of the "would you like to go back" attitude with Cluster Analysis revealed that the actual experience of *Agile Methods* overrode any possible prejudices. Overall, actual experience of *Agile Methods* seemed to play a more important role than any earlier experiences of traditional methods. From the industrial viewpoint, the results indicated that once the respondents have sufficient experience of *Agile Methods*, these were perceived as providing added value. (Paper V)

In the 2008 survey of opinions regarding the actual agile deployment it was found that most respondents agreed on all counts with the generally claimed benefits of *Agile Methods*. (Paper V) These benefits include greater satisfaction, a feeling of effectiveness, increased quality and transparency, increased autonomy and happiness, and the earlier detection of defects. It can be assumed that these benefits were the reason why in both the 2008 and 2009 surveys the majority of respondents would not have liked to return to the old way of working (e.g. in the year 2008 survey 60% of the respondents stated that they would like to stay in agile mode).

Analyzing the survey responses from the years 2008 and 2009, we found that while the perception of the impact of *Agile Methods* was predominantly positive, there were also several challenge areas. However, on the basis of the findings of these studies, *Agile Methods* are here to stay. (Paper V, Paper VI)

When we look at the results from the 2008 and 2009 studies on a detailed level, it looks as if the opinions of different groups are converging with each other (a smaller H value in the Kruskal-Wallis H test in the 2009 data compared to the 2008 data).

When we look at the 2008 survey results, we can see that the only opinions that vary according to the agile experience (being statistically significant) were *transparency* and *collaboration*. So, even if the opinions vary regarding the other benefits of *Agile Methods* between the more experienced and less experienced agile practitioners, this difference in opinions was large enough for only *transparency* and *collaboration* to be statistically significant. Assuming that even these opinions would become more similar as the organization itself learns more about *Agile Methods*, it could be possible that in 2009 we could not have found such a statistically significant difference in opinions at all! It could have been just a lucky coincidence that this survey was conducted so early that this kind of statistically significant difference could be spotted while people were still learning about *Agile Methods*!

Assuming that the above-mentioned assumption holds, it can lead us to another astonishing conclusion: if these methods can be learned only by experience, how could you ever reach the next level of agility if you have no experience of it?

Here we must understand the multidimensional structure of scaled-up agility. Assuming we had an organization that has not used some agile practice, e.g., continuous integration, a researcher might possibly find such a statistically significant difference regarding the opinions of continuous integration and the typical benefits reported in the literature. Once the organization has started to explore and use continuous integration these benefits would be more commonly known to all the members of the organization, and thus argued about less.

This can lead us to two further assumptions:

- it is important to analyze how many aspects of agility our understanding covers, and how much is already there, because that dictates how much you can implement — OR the reverse,
- there needs to be a group of people with unified vision or a single mastermind behind every transition for the rest of the organization to learn — OR,
- an organization can try to learn from different random acts of improvement as long as it can measure if improvement has actually happened; however, most probably the likelihood of hitting an actual improvement is higher if it is based on some understanding of what could actually work and the internal structures of the organization, such as agile dimensions.

The basis of these assumptions is that there are many *Agile Aspects* and agile dimensions that can be improved by context-sensitive deployment actions (as concluded in Chapters 2 and 3). Boehm, Turner, Ould, and Sommerville have also stated that for a large, complex project, often no single model is the best one. Instead, a hybrid model blending and balancing the features of different models is often the choice. (Sommerville 1996, Ould 1999, Boehm and Turner 2004)

We can verify these assumptions by reflecting on what has been written about similar transformations. The deployment of *Lean Thinking* in manufacturing has similar characteristics of a multidimensional, large transformation affecting all levels of the organization.

Some of that large-scale agile understanding could be developed as a by-product of such a large-scale transformation. That is dependent on how much *Organizational Learning* happens during the agile transformation.

It is quite plausible that learning agile development needs such double-loop learning, and thus this kind of appreciating, open, and respectful environment is a precondition for any successful agile transformation. Argyris' (1993) insight into the role that management plays in the transformation confirms the previous finding regarding the alignment of different improvement measures, concerning the alignment of *Goals, Means,* and *Enablers* vectors.

3.10 Results

The experiences of using *Agile Methods* on a large scale that have been referred to suggest that besides having systematic adoption the organization must also have a holistic way of working and alignment of *Intention* in order to gain all the benefits. Using Scrum at the team level will help the respective teams but unless the whole organization's working practices are linked with each other in a meaningful manner, the organization will miss the overall goal, such as improved time-to-market. Sub-optimizations will not lead to improvements on a holistic level; we witnessed this painfully when development programs were delivering at a rate the surrounding organizations were not able to receive, integrate, test, and deliver.

A recent study proposed that the appreciation of *Agile Methods* seemed to increase once they had been adopted and applied in practice (Chow, Cao 2008), which indicated that while there was likely to be resistance among the agile adopters in the organization, time and experience of applying the methods had a positive effect on this resistance. The two studies carried out by the author in 2008 and 2009 confirm these findings. Further research into how the new technology adoption curve could be applied here is suggested. Change management in the context of agile deployment also provides further research opportunities.

3.10.1 Generality of the Presented Model of Agile Transformation

The *Model of Agile Transformation* (Fig. 19) is generic, and thus it can be applied to any organization aiming to engage in agile transformation. The *Agile Framework* presented in Fig. 20 can be used similarly in other organizations — or an alternative model could be built that was based on the *Agile Principles* and practices. One could, e.g., build a similar model for putting continuous integration into use. In that case the *Goals* could be, e.g., fast feedback, better dependency management, and quality. *Enablers* could be, e.g., fast build machines, test automation tools and team-building practices. *Means* could consist of *Information radiators*, automated quality gates, and a *Test Repository*.

3.10.2 Benefits of Agile Methods

Intention (or *Goals*) provides just one angle on the CAS framework. An *Agile Organization* could also be studied from the other seven angles of the CAS framework. For example, one very interesting research dimension would be the *Strategies and Actions* — what kind of development processes (game strategies) will lead to the best outcome.

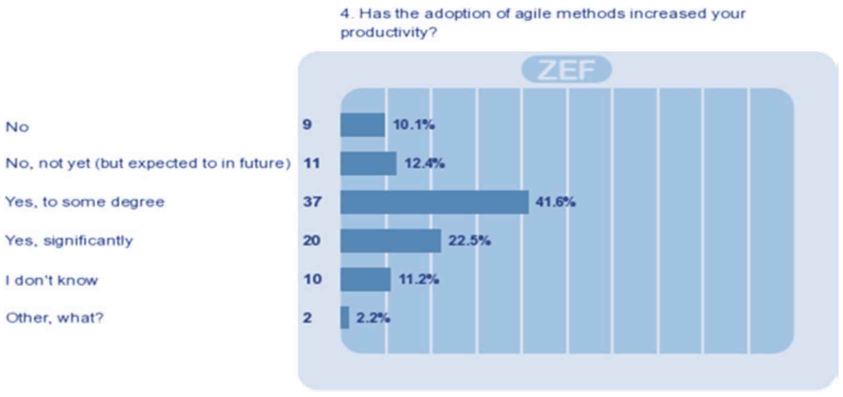


Fig. 24. Perceived Benefits of Agile Methods in terms of Increased Productivity. (Paper IV).

The benefits of *Agile Methods* provide an additional angle for viewing the agile transformation. Paper IV presents some statistics from this angle. Although the survey population was quite small (only 96 respondents), the results showed that people feel more productive in this new mode and the subjective feeling of quality has increased (Fig. 24 and Fig. 25). These findings were consistent with the positive anecdotal evidence gained from program managers and their superiors. (Paper IV)

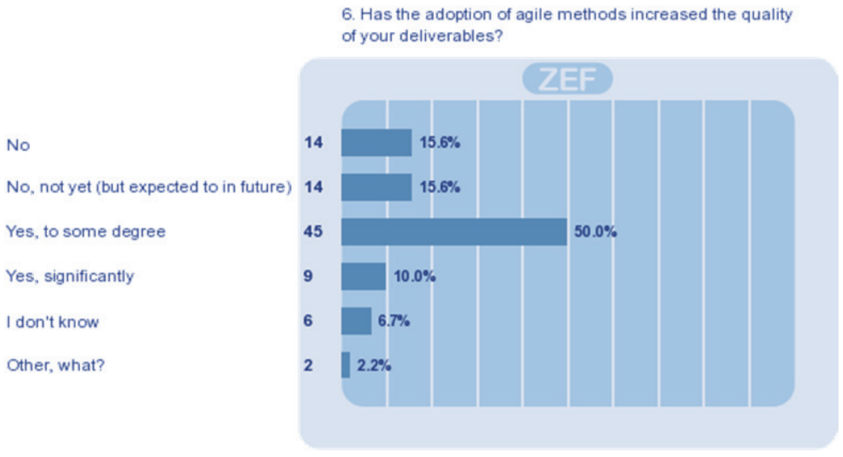


Fig. 25. Perceived Benefits of Agile Methods in terms of Increased Quality. (Paper IV).

4 Research Methods

Different research methods can be structured according to the taxonomy of Järvinen and Järvinen (2004). This taxonomy is presented in Fig. 28. This research applies *Action Research*, which belongs to *Innovation building and evaluating studies* in Fig. 28. The researcher has been working as a *Reflective Practitioner* in the organization under research.

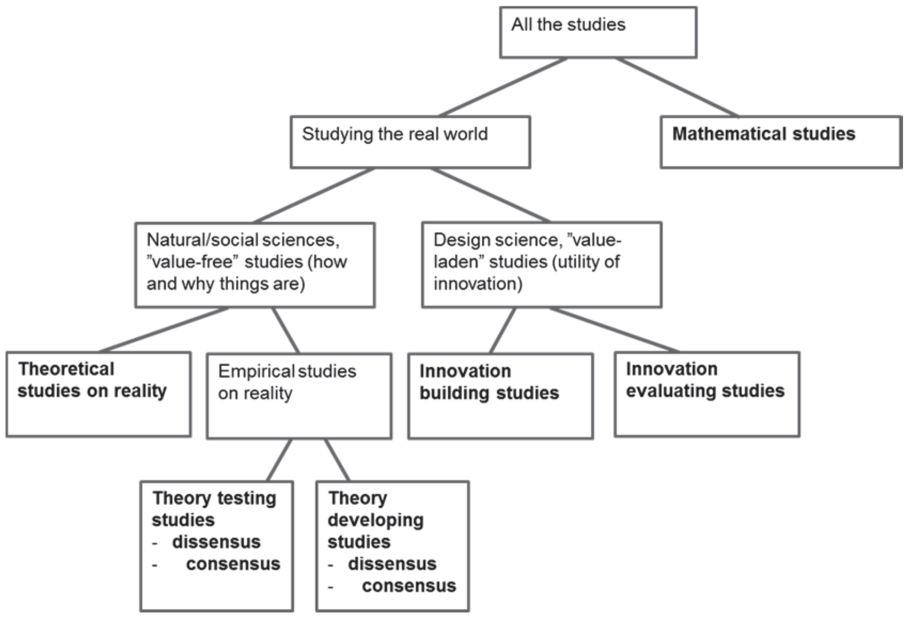

All the studies
Studying the real world
Mathematical studies
Natural/social sciences, "value-free" studies (how and why things are)
Design science, "value-laden" studies (utility of innovation)
Theoretical studies on reality
Empirical studies on reality
Innovation building studies
Innovation evaluating studies
Theory testing studies - dissensus - consensus
Theory developing studies - dissensus - consensus

Fig. 26. Taxonomy of Different Research Methods. (Järvinen and Järvinen 2004, Järvinen 2010)

Carr and Kemmis (1986) provide a classic definition of action research.

> *Action research is simply a form of self-reflective inquiry undertaken by participants in social situations in order to improve the rationality and justice of their own practices, their understanding of these practices, and the situations in which the practices are carried out.*

This definition is firmly tied into self-reflection, and the realm — frame of understanding — of the practitioner. As a way of working it is thus very close to *reflection-in-practice* (Smith M 2007).

In *Action Research* the researcher is also participating in the research context, acting as a *change agent* (Järvinen and Järvinen 2004). This means that *Action Research* also lacks the usual requirement of the neutrality of the researcher.

Action Research has been blamed for be insufficiently theory-oriented and too focused on finding solutions to practical problems. *Research-in-action* tries to bring these two elements (theory and practice) closer to each other. In *research-in-action* the researcher is not just someone making an intervention but is actually changing the organization under research. The researcher is helping the clients to change the world in ways that match the values and theories they possess. (Järvinen and Järvinen 2004)

Papers V and VI contain qualitative and quantitative analysis, which belong to *Mathematical studies* in Fig. 26. However, in this dissertation, these qualitative and quantitative studies can be considered as part of the *Action Research* that was conducted. *Action Research* can utilize any methodology, qualitative or quantitative, but when experimentation is involved in *Action Research,* it is most probably quasi-experimental in nature (Wiersma and Jurs 2005). *Quasi-experimental research* involves the use of subjects in experiments, rather than assigning subjects at random to experimental treatments. Thus the research conducted in this dissertation has a quasi-experimental nature.

4.1 Reflecting-in-action

Ordinary people and professional practitioners often think about what they are doing, sometimes while doing it. Knowing-in-action refers to know-how and putting that know-how into action, applying knowledge to instrumental decisions. Reflecting-in-action refers to the case when we are thinking about what we are doing. In such cases we compare the outcome to the expected outcome, and try to find logic and reason when surprises happen. Reflection in-action[35] is not a rare event. On the other hand, knowing-in-action may limit the scope and depth of reflection. (Schön 1991)

Whereas an expert thinks that "I am presumed to know, and must claim to do so, regardless of my own uncertainty", the reflective practitioner thinks that "I am presumed to know, but I am not the only one in the situation to have relevant and

[35] The different terms used here might cause confusion. The actual practice is called *reflecting-in-action,* where the researcher *reflects in-action* on the basis of a theory of *Action Research* called *reflection-in-action*.

important knowledge. My uncertainties may be a source of learning for me and for them." (Schön 1991) In this way reflection-in-action is close to organizational double-loop learning, i.e., the practitioner is always looking for the governing variables, i.e., why the actions cause certain consequences.

When someone reflects-in-action, he becomes a researcher in the practice context. He is not dependent on the categories of established theory and technique, but constructs a new theory of the unique case. His inquiry is not limited to deliberation about means and ends separately, but defines them interactively as he frames a problematic situation. He does not separate thinking from doing, ratiocinating his way to a decision which he must later convert to action. Because his experimenting is kind of action, implementation is built into his inquiry. (Schön 1991)

Reflective practitioners keep up a *conversation with the situation.* It starts with inquiries, which turn into frame experiments. When the reflective practitioner becomes aware of his frames, he also becomes aware of the possibility of alternative ways of framing the reality of his practice. (Schön 1991) Choosing the wrong frames, or too little knowledge-in-action, may be the reason why reflection-in-action fails. An alternative way to fail is to test the theories only within a limited scope, or taking in only biased information.

In this work, the researcher has been working in the industry for eighteen years, and of that she has been leading projects for ten years. She has been studying *Agile Methods* since 2004, and since 2007 she has been applying those methods in a very large organization. The biases of the feedback have been excluded with the use of surveys and quantitative methods.

4.2 Action Research

Avison *et al.* (1999) explain that qualitative methods such as *Action Research* have their value in explaining what is going on in organizations. *Action Research* is unique in the way it associates research and practice, combining change and reflection immediately in a problematic situation. It is an iterative process involving researchers and practitioners acting together on a particular cycle of activities, including problem diagnosis, action intervention, and reflective learning. Every iteration of the action research process adds to the theory, so it is more likely to be appropriate for a variety of situations.

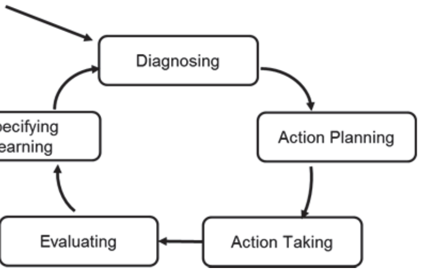


Fig. 27. Action Research Cycle. (Susman and Evered 1978).

Susman and Evered (1978) have described *Action Research* as a cyclical process consisting of five phases; see Fig. 27. Järvinen and Järvinen (2004) state that the cycle *1. Diagnosis ... 2. Planning ... 3. Deployment ... 4. Reflection ... 5. Learning ...* typically continues for many rounds.

An alternative to this cycle is presented in Fig. 28. This representation starts from *Analysis* and *Reflection*, raises a *Question*, seeks answers, and, through *Fieldwork*, comes back to analysis. *Action Research* cycles of this type are more typically used in *Education Research*. This model also resembles more accurately how this research work has been conducted. Papers I and II present the *Analysis and reflection,* Paper III *Raises a question,* Paper IV *Is seeking answers*, and Papers V and VI are *Doing fieldwork.*

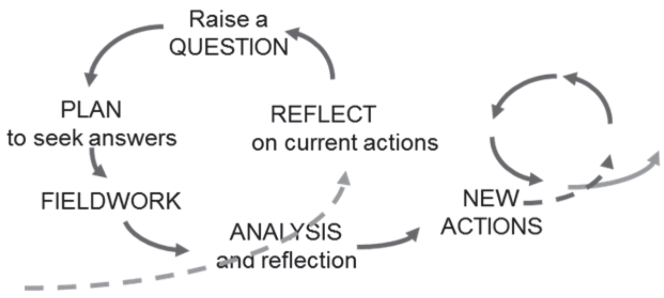


Fig. 28. Cycles of Action Research. (Victoria University of Wellington, 2011).

According to Avison *et al.*, different types of action research can be categorized into four types:

1. *Action research* focusing on change and reflection;
2. *Action science* trying to resolve conflicts between espoused and applied theories;
3. *Participatory action research* emphasizing participant collaboration; and
4. *Action learning* for programmed instruction and experiential learning. (Avison, Lau, Myers, and Nielsen 1999)

This research belongs to *Action research focusing on change and reflection.* Avison *et al.* further note that "action researchers have large and complicated stories to tell", which is exactly true in the case of this research work as well. Not only is the subject of this research large and complex but also the subject organization is.

4.2.1 Canonical Action Research

Davison, Martinsons, and Kock (2004) present principles for *Canonical Action Research*. The word "canonical" is used to formalize the association with the iterative, rigorous, and collaborative process-oriented model developed by Susman & Evered. *Canonical Action Research* presents a set of principles and criteria for the practice and review of the research done. However, these principles should not be followed blindly and inflexibly, but rather used to facilitate the clear and systematic presentation of ideas and findings, helping researchers to justify their choices of action, their contributions to knowledge, and their conclusions. This section presents reflections on the way this research conforms to the five principles of *Canonical Action Research* and the associated criteria.

The First Canonical Action Research Principle of the Researcher — Client Agreement

1a. *Did both the researcher and the client agree that Canonical Action Research was the appropriate approach for the organizational situation?*

The researcher was working in the client organization, so there was no contract made in connection with the research in the traditional sense. Action research has been a widely applied research method in the client organization in similar cases; see, e.g., Kinnula (1999).

The researcher was in a visionary position, trying to figure out ways to apply *Agile Methods* in the client organization. All the research results were of immediate interest in the client organization, and the feedback that the researcher got from the client organization impacted directly on the research.

1b. *Was the focus of the research project specified clearly and explicitly?*

The research was started and focused by the researcher's insight that there is a need for this kind of understanding of agile on a large scale. It should be noted that this research was started in the researcher's own spare time. The researcher was even advised to seek an area closer to her daily job at the time of the beginning of the research. However, agile deployment and, later, agile coaching, gradually became the researcher's main job. After that the research was linked with the researcher's main job — but the ideas or topics for the research, such as the topic of the survey questionnaire, originated from the researcher, not from the client organization.

1c. *Did the client make an explicit commitment to the project?*

An explicit commitment was made in January 2007, when the researcher started to champion agile deployment at Nokia. The earlier research done was based on the researcher's experiences at Nokia Networks.[36]

The client organization had expressed interest in the research. There were many other practitioners interested in this topic, but rather working on the engineering and design side, not on the operative development side of the organization. Dialogue with these people has been very fruitful.

1d. *Were the roles and responsibilities of the researcher and client organization members specified explicitly?*

Yes. The roles and responsibilities were clearly specified, but also changed during the research period. As described, the researcher started the research in her own spare time. In 2007, she was made responsible for the transformation project. As the project grew, more people were deemed responsible for the actual deployment,

[36] Nokia Networks was the name of the unit at Nokia responsible for manufacturing telecommunications and data networking equipment. Since 2007 it has been a joint venture with Siemens, called Nokia Siemens Networks.

and the researcher took on the role of an advisor and evangelist. That means that she was advising people responsible for change projects, training people on agile, and creating visions and blueprints for how to use *Agile Methods* in the organization under research.

1e. *Were project objectives and evaluation measures specified explicitly?*

From the year 2007 onwards the researcher was engaged in several agile deployment change projects, which each had clear objectives and evaluation measures. Likewise, the research papers each had specific research questions that were answered in the respective papers.

1f. *Were the data collection and analysis methods specified explicitly?*

Yes. Especially in Papers V and VI, which used quantitative methods, the data collection methods were specified with great accuracy. The data collection and analysis for Paper IV presenting the scaled-up model was wider than what was actually presented in the paper. Papers I-III were based on the experiences at Nokia Networks and the data available there at the time, plus extensive literature research.

The Second Canonical Action Research Principle of the Cyclical Process Model (CPM)

2a. *Did the project follow the CPM or justify any deviation from it?*

The research followed iterations. Each research paper was an iteration of its own, contributing more to the knowledge of the researcher and the client organization. As a whole, all the research papers form a bigger cycle, contributing to the theories associated with agile methods and approaches on a large scale.

2b. *Did the researcher conduct an independent diagnosis of the organizational situation?*

Several such diagnostic exercises were conducted in the research organization (after 2007) but not by the researcher. However, the researcher had access to all these data. More than the diagnosis, the research was affected by the direct feedback that the researcher got from the client organization all the time. Surveys proved to be useful tools in order to separate not just the loudest but also the most relevant messages from the feedback.

2c. *Were the planned actions based explicitly on the results of the diagnosis?*

The actions were planned on the basis of the theory. Before the actions were started the theory was verified through an analysis of the current situation (but not by the researcher). However, the research results impacted on the succeeding deployment rounds in the subject organization.

2d. *Were the planned actions implemented and evaluated?*

Yes. All the actions ended up with detailed and accurate implementation plans. The researcher was also partially responsible for the implementation. Data related to the implementation phase are confidential for the company and thus cannot be referred to or presented here.

2e. *Did the researcher reflect on the outcomes of the intervention?*

Yes. Much of this reflection is explained in the research papers. Some of these data, however, are internal to the subject organization and confidential for the company.

2f. *Was this reflection followed by an explicit decision on whether or not to proceed through an additional process cycle?*

Yes. Especially the quantitative studies performed in 2008 made a significant impact on the subject organization.

Deploying agile also happened in cycles in the subject organization. The deployment itself was also incremental and iterative — and agile. However, the subject organization was not always making steady progress towards agility — sometimes steps were also taken backwards toward a more traditional way of carrying out projects. But the next rounds after these backlashes again moved towards more agile operations.

2g. *Were both the exit of the researcher and the conclusion of the project due to either the project objectives being met or some other clearly articulated justification?*

The researcher has not exited from the research organization. The individual change projects have been completed, but the change is still continuing. The cycles of deployment have had several successes — but, as described, there is no limit to how agile an enterprise can become.

The Third Canonical Action Research Principle of Theory

3a. *Were the project activities guided by a theory or set of theories?*

Yes, by a set of theories that were all related to *Agile Methods*. The researcher was in a central role to guide which of these theories should be applied in practice and how. Many other practitioners also presented their opinions. The most important theories applied were *Agile Principles* in general (Cockburn 2001), Scrum (Schwaber 2004 and 2007), and Continuous Integration (Humble and Farley 2011). Theories and practices presented by Leffingwell (2007), Larman and Vodde (2009, 2010), Lean Thinking (Poppendieck 2003 and 2010), Product Development Flow (Reinertsen 2009), Agile Portfolio Management (Vähäniitty and Rautiainen 2008), and Agile Estimation and Planning (Cohn 2005) were also applied.

3b. *Was the domain of investigation, and the specific problem setting, relevant and significant to the interests of the researcher's community of peers as well as the client?*

Yes. There has been a lot of interest in the research both internally in the organization and externally.

3c. *Was a theoretically based model used to derive the causes of observed problem?*

Yes. Each deployment round started with a vision presenting the current problem and the possible solutions — most of which were created by the researcher — and why these solutions would work.

3d. *Did the planned intervention follow from this theoretically based model?*

Yes. When I applied for a position to be in charge of agile deployment in December 2006, my understanding of how to apply agile on a large scale was thoroughly tested. The theory was then further discussed by the Operational Development team and the subject organization. Later on these discussions continued between the researcher and the people responsible for the deployment projects.

3e. *Was the guiding theory, or any other theory, used to evaluate the outcomes of the intervention?*

The outcomes were compared against the deployment plan. The results were discussed and compared against the theory. Any deviations from the expected results were thoroughly analyzed.

The Fourth Canonical Action Research Principle of Change through Action

4a. *Were both the researcher and client motivated to improve the situation?*

Yes. At the beginning there were some individuals who were not so motivated but the top management was motivated. After some time the situation turned around; the developers were the most motivated, and the management was not always fully behind the change. However, around 2009 everybody agreed that the situation needed an improvement. Not everybody agreed about the ways in which this improvement should be made, though.

4b. *Were the problem and its hypothesized cause(s) specified as a result of the diagnosis?*

Yes, these were discussed several times. In 2007 the problem and the solutions were written as an elevator pitch. Later on, various other visionary documents and change plans were created.

4c. *Were the planned actions designed to address the hypothetical cause(s)?*

Yes. The change plans and actions planned are, however, company confidential, and cannot be described here.

4d. *Did the client approve the planned actions before they were implemented?*

These discussions took a long time. The discussion at the beginning took half a year. Eventually there was agreement about what to do. This happened with every implementation cycle.

4e. *Was the organization situation assessed comprehensively both before and after the intervention?*

Yes. The assessment before could have been more thorough. In March 2007 it was not obvious to everyone that a change was needed. The success of the deployment was also assessed after the iteration at hand was completed. The researcher was collecting stories about success and also other feedback all the time together with her colleagues. The situation was reflected on on a weekly basis.

4f. *Were the timing and nature of the actions taken clearly and completely documented?*

No. The change programs were all planned and documented but not as part of this research. However, as each of the research papers forms an iteration of its own, the research described in these papers was documented thoroughly.

The Fifth Canonical Action Research Principle of Learning through Reflection

5a. *Did the researcher provide progress reports to the client and organizational members?*

The researcher reported her work as she usually does. The change projects had their own reporting. All the research results were presented and discussed in the client organization.

5b. *Did both the researcher and the client reflect upon the outcomes of the project?*

Yes. The discussions on research results took place prior to the writing of the scientific papers.

5c. *Were the research activities and outcomes reported clearly and completely?*

The research results were reported clearly and completely only internally. There are a lot of survey data and results that were not shown publicly when the scientific reports were being written. But the data that were selected to be used in the research reports were complete and clear.

5d. *Were the results considered in terms of implications for further action in this situation?*

The research results were discussed and their implications for the succeeding implementation rounds were agreed accordingly.

5e. *Were the results considered in terms of implications for action to be taken in related research domains?*

The results were considered in terms of the future of their implications for further research actions by the researcher only. Each of the research papers contains a section that considers "future research" items. In the client organization the results were discussed only from the point of view of what implications or enablers of the agile deployment might still be unknown.

5f. *Were the results considered in terms of implications for the research community (general knowledge, informing/re-informing theory)?*

There has been a continuous dialogue between the researcher and the research community. This dissertation collects and presents the final contributions to the theory-building.

5g. *Were the results considered in terms of the general applicability of Canonical Action Research?*

This section considers the general applicability of *Canonical Action Research* to the results of the research. As the research was performed in iterations and increments, the cyclic process models are applicable in this research.

Davison *et al.* (2004) further state that learning is the most critical activity in *Action Research*: both the research community and the clients should learn. If this principle is taken as a measure of successful research then this research has been truly successful as significant learning has taken place here. It is also suggested by Davison *et al.* (2004) that the learning happening in a research project is marked down as soon as “lessons learned” happen throughout the project to ensure that all the lessons have been documented both clearly and comprehensively. The incremental learning is clearly visible in the different research papers (see Chapter 1).

4.3 Concept Analysis

Concept Analysis may be needed when the available concepts are unclear, outmoded, or unhelpful. The purpose of Concept Analysis is to distinguish between the defining attributes of a concept and its irrelevant structure or to determine its internal structure by breaking it down into simpler elements.

Concept Analysis is used to:

1. clarify overused vague concepts;
2. promote mutual understanding among colleagues, and
3. provide a precise operational definition that by its very nature has *Construct Validity* and accurately reflects its theoretical base (Walker and Avant 1995)

Nupponen (2003) suggests that not just the ones developing terminology — to whom Concept Analysis was originally developed — but also the subject specialists could benefit from Concept Analysis. She states that special experts in a specific subject area, such as software architects, could also benefit from Concept Analysis because their work is largely done using language and context-specific special concepts.

In this dissertation Walker and Avant's (1995) method for Concept Analysis is used. Walker and Avant's method for Concept Analysis consists of the following steps:

1. select a concept;
2. determine the aims or purpose of analysis;
3. identify all the uses of the concept;
4. determine the defining attributes;
5. construct a model case;
6. construct additional cases;
7. identify antecedents and consequences;
8. define empirical referents.

In this dissertation the concepts that have been analyzed are *Agile Software Development*, agility, and agile-in-large. The purpose of the analysis of *Agile Software Development* and agility was to define these concepts in such a way that it could be used as the basis for discussion about how this concept could be used on a large scale, i.e., in large software development organizations. The purpose of the analysis of the agile-in-large concept was to clarify the meaning of this concept.

The literature was researched in order to identify all the definitions of *Agile Software Development* and agility, and the emphasis in each of the definitions was studied. Likewise, all aspects of agile-in-large were studied. For both concepts, a model case was constructed, as well as a borderline case and a contrary case. The antecedents — events that must occur prior to the occurrence of the concept — or the consequences — events that occur as a result of the occurrence of the concept — were discussed later in this dissertation, when the question of using these methods was discussed. The empirical referent was the subject organization under research.

5 Contributions of Papers

The research in Papers I-III, entitled "*How to Steer an Embedded Software Project: Tactics for Selecting the Software Process Model*", "*How to Steer an Embedded Software Project: Tactics for Selecting Agile Software Process Models*", and "*Combining Agile Software Projects and Large-Scale Organizational Agility*", is innovation-building. The data collection in these papers is based on contemporary literature research that is validated by empiricism.

The research in Paper IV *"Implementing Program Model with Agile Principles in a Large Software Development Organization*", is theory testing that is backed up with qualitative data (frequencies). The data collection is based on collected deployment metrics and a web-based survey.

Paper V, *"Agile methods replacing rapidly the traditional methods at Nokia: A survey of opinions on agile transformation*"*,* uses 1) quantitative studies (non-parametric and parametric tests): frequencies, mean, std. deviation, ANOVA (analysis of variance), clustering and 2) qualitative studies (categorization of opinions) as the research methods. The data collection was done with a web-based survey.

Paper VI, *"Agile Transformation Study at Nokia – One Year After",* also uses quantitative studies (non-parametric tests): percentages of frequencies, Kruskall-Wallis H, and a Chi-Square test as the research methods. The data collection was done with a web-based survey.

5.1 Paper I: How to Steer an Embedded Software Project: Tactics for Selecting the Software Process Model

This paper discusses how the right selection of a process model can help tackle problems in modern large new product development projects, which are typically characterized by many uncertainties and frequent changes. Often the embedded software development projects working on such products face many problems compared to traditional, placid project environments.

The paper compares many process models against each other, from plan-based waterfall and incremental models to evolutionary models such as spiral and RUP (Rational Unified Process) and up to agile models (FDD, ASD, XP). For comparison, a no-process model (i.e., ad hoc) is also considered.

From the point of view of this thesis the paper adds to the understanding of what the most typical problems in traditional development are and how agile models can help to foresee and possibly solve these problems. The comparison matrix gives a good understanding of how *Agile Methods* can help reveal some problems that traditional programs typically suffer from earlier (if followed up and implemented properly). From the selection matrix it can be seen, for example, that XP's practice of having a customer on-site when defining the requirements should (at least in theory) help against the problem of having an "unattractive software release (wrong, obsolete, or missing features)". However, there are also problems where none of the processes give really good answers — such as, e.g., "geographically dispersed teams" or "unsuitable or low-quality tools".

5.2 Paper II: How to Steer an Embedded Software Project: Tactics for Selecting Agile Software Process Models

This paper focuses (similarly to Paper I) on the right selection of the process model. The difference between these two papers is that Paper II focuses more on the *Agile Aspect* (as its title states), and compares different agile process models against each other (whereas Paper I compares agile models to traditional ones).

The idea for writing this paper was based on the results of Paper I: what if the problem space in agile projects is different from the problems traditionally seen, i.e., if there is a set of new problems that agile processes introduce, and how can we possibly solve these problems?

The process models compared in the comparison matrix presented in Paper II are: RUP (Rational Unified Process), FDD (Feature-Driven Development), ASD (Adaptive Software Development), Scrum, and XP (Extreme Programming). For comparison, at this time an ad hoc approach (no process at all) was also presented in the comparison matrix. Unfortunately, at the time of writing, not much had been written about the problems in agile projects, so not many adjustments were made to the problem list. Thus we wanted to shed more light on the conditions in which each of these methods was born and where these methods are at their best and the basic thinking behind each of these methods. This was the primary reason for presenting a "software process characteristic matrix" in Appendix B of Paper II.

5.3 Paper III: Combining Agile Software Projects and Large-Scale Organizational Agility

This paper proposes a theoretical framework for understanding the multidimensional nature of agility in large product development environments when putting *Agile Methods* into use, and highlights certain practical considerations of such usage. It does not specifically look at any *Agile Method* as such, but tries to understand the phenomena of agility more holistically, putting them into the framework of a large development organization. The foundations of the paper lie both in the researcher's experience of such environments and in an extensive literature study of the related fields of *Agile Manufacturing* and *Agile Organizations*. First we listed all the characteristics we found in the literature of such organizations. To our surprise, many of these characteristics were not aligned at all, but could be seen as pulling in different directions. Because of this we concluded that there need to be *forces* that are countering each other — forces originating both in agility and in other sources. Then we categorized the descriptions of agility so that similar descriptions (forces that had a similar nature or forces impacting on the same level of the organization) were put together. The final step was trying to come up with a descriptive name for each of these categories.[37]

The paper contributes to understanding by presenting an *Agile Organization* as a field of different forces that have different impacts. Some of these forces are dependent on each other and some counter or are misaligned with each other. We named these forces *Goals*, *Enablers*, and *Means*. Kettunen (2010) enlarged the framework later by adding *Impediments* and *Needs* to this frame. At the time of writing the paper we had already discussed the Impediments but did not see those as tangible objects but rather as a result of countering forces. Later on in our discussions we concluded that, e.g., an outdated tool could be seen as such a force of its own, an Impediment. *Needs* originated from the understanding that there must also be forces external to the organization that impact on the organization. *Needs* are forces that the organization has selected as a response to the external business forces. These are different from *Goals* as the latter originate from the organization's own ambitions. *Enablers* are elements that the organization needs to reach the *Goals*; they can be thought of as prerequisites. *Means* are ways to

[37] This was done by the author. The moment at which she was trying to keep all these data in her head and come up with a descriptive term was really difficult — there were several hundred items. The author has never felt so close to insanity either before or after this moment.

reach the *Goals*, answering the question “How can the goal be reached”. For example, an organization’s goal could be faster ROI, which is driven by a more competitive business environment. The *Means* to reach this could be both creating smaller increments and investing in continuous deployment. The underlying *Enablers* are the tools for performing continuous integration and automatic testing, as well as the new capabilities that the personnel need.

As a practical measure the paper presents the agility evaluation matrix. The matrix could be used, e.g., as a basis for discussion of the current status and how to enhance the organization’s agility further.

5.4 Paper IV: Implementing the Program Model with Agile Principles in a Large Software Development Organization

This paper presents the model scaling agility up to the program level that was in use in the subject organization in 2008. The actual model described in the paper consists of nested control loops, a system for scaling up backlogs and *Product Owners* by forming a hierarchy of *Product Owners* and following Scrum principles on each of the loop levels as described in Paper IV: Agile Policy.

Because of her broad theoretical background in scaling *Agile Methods*, the researcher was able to present a blueprint for changing all software development into an agile mode as early as in April 2007, but it took half a year before this model was even approved by the operational development team, and a full year before the blueprint was actually approved. The original blueprint is presented in Appendix D.

The model that was implemented was only a subset of this blueprint. Interestingly, the fact that agility was not implemented holistically from the beginning was later on identified by many as one source of problems. The *Agile Framework* (if understood as a force field) could have been used to forecast the other source of problems: misaligned improvement initiatives that consume the organization’s energy and cause random behavior.

The contribution of the paper is to show that a scaled-up model actually works. The scaled-up model was piloted in a project with five nested levels of control loops and over 500 people. It also proved that satisfactory results can be gained at a project level.

It took the subject organization more than two years to start to follow all the practices mentioned in the *Agile Policy*. A lot of this change focused on how projects were managed and followed up — in fact, full change on the project level was not possible before the traditional management on the next level had been replaced by people whose views were more compatible with agile projects.

5.5 Paper V: Agile Methods Rapidly replacing the Traditional Methods at Nokia: A Survey of Opinions on Agile Transformation

This paper presents results from an organization-wide survey of the usage of *Agile Methods* in the subject organization. In the study it was found out that the subject organization's opinions on *Agile Methods* were mainly positive. It was also found out that the positive opinions are related to the practitioners' length of practical experience with *Agile Methods*, and that this relation was statistically significant for opinions on transparency and collaboration. We could also see that long experience of traditional methods had a negative impact on the respondents' opinions on *Agile Methods*, and that this relation was statistically significant for the opinions on effectiveness, increased collaboration, work being more organized/planned, and agile enabling errors to be detected earlier. A cluster analysis of the opinions of the group that had long experience of both traditional and *Agile Methods* revealed that the long experience of *Agile Methods* prevailed over the long experience of traditional methods: that is, there were more agile advocates in the group that had long experience of both methods. Examination of the open-ended answers about the problems and benefits of the usage of *Agile Methods* by *agile advocates* and *opponents* revealed that these groups were looking at agility from different perspectives: *agile advocates* (*the positive group*) had more comments and a better understanding of *Agile Methods* than the *opponents* (*the negative group*).

Before the study there were mixed loud opinions on *Agile Methods*. What the majority really thought and felt was a surprise. The reported correlations in opinions and differences in opinion between the more experienced and less experienced respondents were also a surprise.

5.6 Paper VI: Agile Transformation Study at Nokia – One Year After

This paper presents results from two surveys in the subject organization, studying if the positive opinions towards *Agile Methods* are of a permanent nature, or if the opinions have changed after one more year of the usage of *Agile Methods*. The results revealed that most respondents were satisfied with the agile way of working and would like to continue in agile mode. They also thought that using *Agile Methods* was important for the future. In two consecutive studies it was seen that the opinions of the people who had actual experience of *Agile Methods* had stayed the same and that the general opinion towards agility had remained extremely positive.

The Kruskall-Wallis H test revealed statistically significant results with the actual experience of *Agile Methods* and the attitudes people had towards *Agile Methods* for both years' data. This led to a recommendation that people should try these methods in practice. The Chi-Square test of Independence told us that none of the variables that were tested (importance, satisfaction, would like to go back) were independent of the length of the practitioner's agile experience. The opinions of different groups that had different agile experience also seemed to have come closer to each other during the year between the two consecutive surveys.

The paper gave us an indication that opinions on the *Agile Methods* had changed more positively for good. It also gave us a hint that the reasons for the negative opinions should be studied in detail — there could possibly be some problems in some areas of agile deployment that could be corrected. It also confirmed the assumption made in the previous paper that agile adoption should be a systematic *deployment* rather than a more free will-based *adoption.*

6 Discussion

Overall, a lot of new research has been conducted in this dissertation. As of today, no-one else has studied the views and convictions in an organization going through an agile transition on such a large scale as was done by the researcher in Papers V and VI. Vodde (2004) and Abrahamsson (2007a and 2007b) have shown similar studies from a large development organization but these were done on a smaller scale and did not include any further analysis with quantitative studies. Korhonen (2010) used quantitative studies when surveying agile and error management practices but within a smaller entity.

The *Agile Framework* presented in Paper III is a new contribution. Later Vidgen and Wang (2009) presented the Agile Organizing Framework, which was discussed in Section 2.9.1.

The Concept Analysis performed in Chapters 2 and 3 is a new approach to defining what agile is on a large scale. This dissertation also introduces the concept of an *Agile Enterprise*. The term is not new, neither for the agile community[38] nor for the research into software development, but is used, e.g., by Pulkkinen and Hirvonen (2005).

In Chapter 3 a suggestion was made that *Agile Methods* should be deployed on all levels of the organization. Schwaber (2007) presents a way to organize the work of an enterprise in order to use Scrum, but the book does not discuss thoroughly how the organizational structure should support agility, i.e., the problem of how to create an *Agile Organization*. Leffingwell (2011) recognizes the three layers (team, portfolio, and program) and also lists some problems caused by a legacy mindset. But the viewpoint presented in this dissertation is somewhat unique, trying to present a more holistic model.

A model of Agile Transformation is also presented in Chapter 3. This model is a new and unique approach to how to implement an *Agile Enterprise* in practice.

6.1 Key Findings

The first key finding in this dissertation is that the definitions of *Agile Methods* vary a lot. Thus in this dissertation the concepts of agility, *Agile Software Development*, and agile-in-large scale have been analyzed. Model cases, borderline cases, and contrary cases have been represented for *Agile Software*

[38] Google gave 218,000 hits for “Agile Enterprise” + software on 8 May 2012.

Development and agile-in-large. A new concept of an *Agile Enterprise* has been introduced to mark the use of *Agile Methods* in large software development organizations.

Secondly, the use of *Agile Methods* on different levels of an organization was discussed. Example cases of successes and failures on different levels of usage were presented. Instead of thinking about the optimal level on which *Agile Methods* should be used in an organization, the conclusion is that the whole enterprise needs to be transformed.

Thirdly, we presented a practical model for transforming an organization to put these *Agile Methods* into use. An application of this model to a large enterprise was also shown. Lastly, it was shown via survey results how the opinions on agile in an enterprise may differ between people and develop over time.

6.2 Implications for Research

There have been claims that scaling *Agile Methods* are the last thing that should be done (Kähkönen 2004). In contrast to this, our research defines the concept of an *Agile Enterprise* and shows a model of how agile transformation can be done. Such definitions and models have largely been missing so far. Additionally, when most organizations talk about "large-scale", they mean "dozens" of teams rather than hundreds (Sutherland 2001).

The measured impact in Paper V (the 2008 survey of opinions) regarding the actual agile deployment was that most respondents agreed on all accounts with the benefits generally claimed for *Agile Methods*. (Paper V) These benefits include a higher level of satisfaction, a feeling of effectiveness, increased quality and transparency, increased autonomy and happiness, and the earlier detection of defects. It can be assumed that these benefits were the reason why in both the 2008 and 2009 surveys the majority of the respondents would not have liked to return to the old way of working (e.g., in the survey in the year 2008 60% of the respondents said they would like to stay in agile mode). The conclusion from Paper V and Paper VI is that *Agile Methods* are here to stay.

Using quantitative methods to measure the benefits claimed for *Agile Methods* is a somewhat new approach when conducting software process model research. The research is also more focused on the benefits of *Agile Methods* than the perception of these methods.

This dissertation has also contributed to the understanding of how *Agile Methods* should be used on different organizational levels. There are a few books on how to manage agile projects, but the idea of transforming all the levels of the organization is — on the basis of what we know — a new contribution.
Few research papers have based their research on the understanding of the complexity of *Agile Enterprises* as *Complex Adaptive Systems*. This is a new research area where a lot of new insights can be gained.

6.3 Implications for Practice

There has been interest in the industry in *Agile Methods*, but for large software development organizations models of how to deploy large-scale agile have been missing. This work gives one view on how such large-scale agility could be achieved.

As teams start to follow *Agile Principles* and practices on a large scale (teams are *Enablers* for agility), new organization-wide mechanisms, i.e., *Means*, are needed to run and explain an *Agile Enterprise*. Fig. 19 presents one example of such an *Agile Framework* upon which an *Agile Enterprise* can base its way-of-working model (e.g., such as the one presented in Appendix E).
Even though the way of working needs to be commonly agreed and understood (this corresponds to the alignment intention in a *Complex Adaptive System Framework*), it may not be enough just to replace the traditional processes with agile processes — the processes of an *Agile Enterprise* will also have to change.

Getting *Means*, *Enablers,* and *Goals* aligned is a long-term task. Alignment here means that the CAS *Actors* will have alignment between the *intentions.* One way to reach this kind of unity in a *Complex System* like the one shown in Fig. 20 would be to nominate a person at the top to lead such activities with enough support from the organization. Without this kind of leadership the middle management might not be unified enough to get the necessary changes deployed. Papers V and VI stated that agility is here to stay, so over time, agility will become something we do (like object-oriented programming) rather than a hype that we change to something else. But keeping an organization agile is not something that can be taken for granted: the Borland case (Maples 2009) shows that organizations that have had a very positive history with *Agile Methods* can revert to non-agile methods.

A typical way to tackle these kinds of problems is to nominate a leader and support organization for this kind of activity. A similar case exists with quality.

We have established practices to set up a quality organization with a VP of quality in order to provide the necessary support to the surrounding organization. Thus this would justify a VP of Agility with an organization and a mandate to support long-term operational improvement. The responsibility of such a VP would be the alignment of all management in such a way that the agile viewpoint would never be absent from discussions. Putting *Enablers* into place requires a lot of hard work and money. Such a VP would secure sufficient funding and priority for implementing the *Enablers*. The role should be combined with the respective operational development role — as traditional process improvement and the related roles would no longer be needed in an *Agile Enterprise*. A failure to have such a role would mean that every time an organization change is made all the agile supportive roles, such as agile coaches (in the Agile VP's direct or virtual organization), will be discarded until they are discovered again as a result of practical need. The Agile VP's agenda would also include keeping agility on the agenda and in active communication, and to see that no contradictory activities get started. An *Agile Organization* would constantly reinvent its agility and restructure itself — as *Business Agility* and *Organizational Agility* state. Communicating such changes should be systematic and planned, so that the intention gets aligned.

Putting the *Enablers* in place requires systematic and planned investments, according to the experience gained in the organization studied here. Once the *Enablers* of an *Agile Enterprise* are in place, the operations and actions can be very responsive. However, this responsiveness is only one force impacting on the success of the organization. According to Moore (2011), *Execution power* is the weakest of all powers; *Category power*, *Company power*, *Market power* and *Company power* are all superior to it. According to Laukkanen (2012) and Moore (2011), an organization should have different modes of operational execution depending on the novelty of the product, i.e., depending on whether the organization is operating in new, growing, or mature markets. In the context of this dissertation this would mean different *Agile Aspects* being emphasized in all these different cases. It should be noted that Moore (2011) states that execution that is too slow can tie up all the resources, so that an organization cannot free any resources for radical or disruptive innovations. Using *Agile Methods* on a large scale is a good solution to that problem.

Traditionally, a large-scale software development effort has followed a reductionist model: work has been split into requirements that have then been allocated to different parts of the organization to implement and then integrated as one product. There has been a major change management process to guard all this from changes. Now iterations and increments with continuous changes are replacing the former lengthy projects. This may even impact on how the work is split into parts if new cross-functional teams replace traditional layered architecture-based component teams.

An *Agile Framework* that is based on Scrum and Extreme Programming will optimize for development speed, as presented in Fig. 20. There are also other alternatives and other *Goals* that could be pursued with organization-wide agility. Hugos (2009) suggests that an *Agile Organization* should be optimized for responsiveness. This aspect is called *Business Agility*. Each organization must understand its business environment and define what aspects of an *Agile Enterprise* are the most important in the specific case.[39]

Section 2.8.4 presents other possible aspects of an *Agile Enterprise*, namely *Strategic Agility*, *Organizational Agility*, *People Agility*, *Tools Agility*, *Organizational culture*, and the *Agility of the product that is built*. It is unclear if all these aspects could be stretched in one organization.

An organization that has all these aspects stretched would have at least the following qualifications:

- the organization values multi-skilled people and intellectual assets;
- it consists of many virtual organizations, reconfigured dynamically, striving to combine the advantages of small entrepreneurial companies and large-scale production economics. Parts of the products would be created with collaborators and subcontractors, leveraging the core competencies and resources of each;
- it releases new product features and varieties frequently and also new breakthrough products following the prospective market changes;
- its products are based on flexible designs and are open and customizable;
- it has open, flexible process frameworks that product development teams use to improve themselves according to their current needs;
- it has tools that support a flexible way of working.

Such agility could be reached if:

1. The product development organization makes a conscious strategic choice of its targeted business operational model (e.g., product leadership). Different companies need different types of agility, depending on the market

[39] E.g. quality requirements may differ a lot, depending on what type of software is produced.

environment and product types. The actions should be based on this understanding of the best business processes.

2. Large organizations developing complex products and a wide range of product variants have to be supported both for individual release projects and longer-term product evolution and portfolio management. Change creation and proactive capabilities have to be considered. It can be argued that many current *Agile Methods* with a tendency to focus on customer requests for the next sprint or next iteration only have a more reactive than proactive nature. In some cases, several internal iterations would be needed in order to create a new visionary product consisting of a multitude of bigger features. What we need is an *Agile Method* that would also cover the product vision or business process iteration and improvement (e.g., business model, revenue generation).
3. An organization appoints a leader for this kind of large-scale agility, whose duty is to develop the model of the way of working and capabilities and probe the organization for improvement. This is going one step further than James M. Morgan and Jeffrey K. Liker (2006), who state in the *Toyota Production System* that an organization under change needs a change agent, but that the change agent does not need to be an expert on lean product development. *Agile Organizations* do not only need a change agent, but the whole organization, like any other *Complex Adaptive System*, needs to understand the shared *Goals,* i.e., the *Intention* of the organization. The lack of such an *Intention* may be the reason why many lean transformations have become only superficial, leading only to partial results.

Antti Vasara, Senior Vice-President of Nokia Mobile Phones Product Development, states the following: “Formerly, we could not get *Enterprise Agility* to work as there was so much misalignment between the management of what needs to be done. Getting *Enterprise Agility* to work was possible after the management alignment in my organization.”

Agile Methods provide some improvement opportunities for small teams. But on a large scale the supposed risks and rewards are larger by orders of magnitude. In practice this was witnessed in 2008 and 2009, when the continuous integration efforts were focused on the project level. Projects achieved their goals, but the whole organization did not gain any benefit, as there was too much big-bang integration happening between different projects. Later, when organization-wide continuous integration activities were deployed successfully, improvements in cycle time that were even as high as 1000% were measured in the subject organization. This resulted in significantly faster error correction and development cycle times.

6.4 Validity of the Work

The concerns about the validity of the research can be divided into concerns about its *External Validity*, *Internal Validity*, and *Construct Validity.*

6.4.1 External Validity

External Validity defines how widely the results can be generalized to other populations, situations, and conditions (Wiersma and Jurs 2005).

This dissertation presents an *Agile Framework* for using *Agile Methods* in large software development organizations. This *Agile Framework* is based on extensive theoretical studies, and has been validated by being used in practice, as presented in Chapter 3. Several loops of learning have happened through several separate deployment actions, as described in Chapter 4.

Davison, Martinsons, and Kock (2004) suggest interviewing multiple participants (i.e., triangulation of data) and using different approaches (e.g., qualitative and quantitative methods) to address concerns about validity. Tuomi and Sarajärvi (2002) list five different ways in which data can be triangulated: 1) the usage of multiple data sources; 2) the usage of multiple researchers; 3) the usage of multiple theoretical angles; 4) the usage of multiple methods, and 5) the usage of multiple analysis methods.

Three of the research papers included in this dissertation present validation through different quantitative surveys. One research paper also presents the results of a qualitative analysis of data retrieved by two open questions in the survey concerning the perceived benefits and problems associated with using *Agile Methods*. This can be understood as contributing to triangulation by analysis, i.e., analysis is performed by using different methods (Tuomi and Sarajärvi 2002). In the above-mentioned surveys, several roles in the organization were present. This contributes to triangulation by multiple data sources (Wiersma and Jurs 2005). The data were represented internally for comments, and compared to similar surveys conducted at Nokia Siemens Networks with similar results (Vodde 2004, Abrahamsson 2007a, Abrahamsson 2007b). This contributes to triangulation by researcher (different researchers get similar results independently from different organizations) (Tuomi and Sarajärvi, 2002). This would indicate that the results are likely to be generalizable to other large organizations.

6.4.2 Internal Validity

Internal Validity measures to which extent the results can be interpreted accurately. *Internal Validity* deals with adequate and appropriate procedures so that results, including cause-and-effect conclusions, can be interpreted with confidence. Were the research procedures conducted appropriately? Were there possibilities for introducing bias? Are the procedures described accurately so the reader can understand what was done? Was the analysis appropriate? (Wiersma and Jurs 2005) Here, the question of *Internal Validity* can be divided into three sub-questions. Was the number of respondents large enough? Were the data collected accurately? Were the statistical tests used the correct ones?

The reported benefits of *Agile Methods* were based only on a survey of a limited population only, as already explained in Section 6.3, in which the limitations of the survey were discussed. However, the surveys in the years 2008 and 2009 were sent to large organizations. In the 2008 survey we had more than 1000 respondents from 7 different countries in Europe, North America, and Asia. 90% of the respondents represented Research and Development (R&D), while the others represented mainly marketing, design, or other support organizations. The total response rate was 33% of the population of the study. The 2009 survey was conducted with the same organization. The response rate in the 2009 survey was much lower — in spite of our best efforts we only got 576 responses, which is significantly lower than the year before. The reason for the lower interest shown towards the survey in 2009 remains unclear. (Paper V, Paper VI)

Both the 2008 and 2009 surveys were questionnaire-based and computer-assisted. Both were also web-based. Nardi (2003) proposes using web-based surveys to eliminate human error caused by manual data handling and to make a higher response rate possible while also increasing speed and reducing cost.

Specific attention was paid to the statistical tests used both in Paper V and Paper VI. Both parametric (ANOVA) and non-parametric (Kruskal-Wallis H, Chi-Square test for Independence) statistical methods were used. ANOVA (Analysis of Variance) tests whether the means of different groups can be fitted to a probability distribution (F-ratio). As one-way ANOVA assumes homogeneity of variance, Levene's test was used for homogeneity. A post-hoc analysis was done with Tukey HSD. (Field 2009)

The Kruskal-Wallis H test is a non-parametric test that does not assume normality. It tests the equality of population medians among the groups. Instead of using variance, it replaces the actual values with ranks — otherwise it is identical to the one-way ANOVA (analysis of variance) parametric test. The Kruskal-Wallis H test is like the Mann-Whitney U test, but it can be used with multiple values (Mann-Whitney U is limited to nominal variables with only two values). In addition to the Kruskal-Wallis test, the χ^2 test of independence between respondents' background in agile way of working and attitudes was performed. In the Chi-Square test of Independence, the frequency of one nominal variable is compared with different values of the second nominal variable. The Chi-Square test of Independence is used when we have two nominal variables. Like many other non-parametric tests, the Chi-Square test of Independence does not assume the normality of the data, and can also take multiple variable values.

Miles and Huberman (1994) proposed linking qualitative and quantitative data to provide richer detail and deeper understanding for the research. Taking these issues into consideration, the analysis includes both quantitative and qualitative data, the latter collected through free-form text entry fields in the survey.

In the qualitative analysis first the data were checked against the categories, and these were cross-checked and unclear cases were discussed to conclude whether comments had been interpreted in the right way and the true meaning of the comments captured in the process. Few of these cases led to the category being changed. Finally, we checked that different interpretations of open comments would not impact on the final results, i.e., that a change in the interpretation of a comment that had multiple interpretations would not change the result or the ordering of the categories. When we concluded that different interpretations could not affect the order of the categories or the conclusions, we concluded that both the process and the results were correct and the probability of any errors small enough. (Paper V)

The large number of responses allowed not only descriptive data analysis including frequencies and cross-tabulation but also more sophisticated quantitative data analysis. The clustering used in this study provides an analysis method that seeks to build up a number of clusters. It is a widely used method in biology, medicine, and market research, especially with segmenting (Rapkin and Luke 1993). Cluster techniques are used to create taxonomies, identify patterns of associations, and distinguish sub-groups in samples. *Cluster Analysis* emphasizes diversity rather than a central tendency. In *Cluster Analysis* the distance between data points is measured, and the data points closer to each other are marked as belonging to the same cluster. *Cluster Analysis* is especially suitable for studying differences in a sample. Even though general linkages exist, *Cluster Analysis* can be used to demonstrate patterns that do not conform to the majority in the sample. (Rapkin and Luke 1993)

In the 2008 study, *Cluster Analysis* was used to discover small differences between the groups, primarily consisting of two different attitude shifts. First there was analysis of the (negative) impact of the length of experience of traditional methods on opinions towards agility and, secondly, there was analysis of the (positive) impact of actual experience of agility on the attitudes and opinions detected. Then we drilled into the differences by analyzing different respondent groups using the attitude question "would you go back?". Finally, a *Cluster Analysis* of answers to the same attitude question, "would you go back?" was used to determine the opinions of those people who belong to both of these groups, i.e., those who have long experience of both non-agile and *Agile Methods*.

As a conclusion, it can be stated that the results were adequately analyzed and interpreted.

6.4.3 Construct Validity

Construct Validity is concerned about obtaining the right measures for the concept being studied. It refers to confounding or misunderstanding the variables in the research. Understanding the research results depends on understanding how the constructs were defined in the particular study. When constructs are used in research they should be defined in a way that is consistent with the prevailing thinking and research in the domain. (Wiersma and Jurs 2005) Here, the concerns of *Construct Validity* can be understood in two ways: were the questions and terms in the surveys that were conducted unambiguous and did the researcher then use correct variables in the research analysis? The clarity of the terminology was tackled by using standard agile terms in the survey. We also tested the questions by doing a separate pilot survey with a team of roughly one hundred respondents before the renewed survey was distributed more widely. The researchers' extensive experience in the field under study, as well as the fact that the results were also reviewed in the subject organization, guarantees that the variables were not misunderstood.

A necessary characteristic of validity is *Reliability*. A study cannot be valid while lacking reliability. *Reliability* means the consistency of the research and the extent to which the studies can be replicated. *Reliability* can further be divided into *Internal Reliability*, which refers to the extent that data collection, analysis, and interpretations are consistent, given the same conditions, and to *External Reliability*, which deals with the issue of whether or not independent researchers could replicate studies in the same or similar settings. (Wiersma and Jurs 2005)

The questions of reliability refer to the issue of whether other researchers would get similar results with a similar research setting. An example of such concern could be, e.g., the response rate: were the respondents to these surveys biased in some way? This could be the case, for example, if only those respondents who were more satisfied with *Agile Methods* answered the survey. But some smaller studies show similar results (Melnik and Mauer 2006, Tessem and Mauer 2007), so the *Reliability* concerns are small.

7 Conclusions

This dissertation started by analyzing different definitions of *Agile Software Development* and agility. After careful analysis, new definitions were presented that could be used as the basis for scaling *Agile Software Development*.

Second, we explained that processes provide one view on the deployment of large-scale agility. Different levels of an organization each need to be transformed. The transformation was discussed at a project (or team) level, at product development level, and at the enterprise level. It was concluded that the *Agile Aspects* provide one way to see the enterprise-level agility. These aspects were then used in the Concept Analysis and the *Agile Enterprise* concept was defined.

Thirdly, a model for agile transformation was presented. It is important to reach an alignment in an organization regarding the *Goals*, *Means,* and *Enablers*. As organizations are *Complex Adaptive Systems,* the *Intention* as defined by the CAS framework can be measured in order to show how unified this understanding is. Failure to reach a unified *Intention* in the organization causes misalignment in operations that are actually seen as impediments.

7.1 Answering the Research Question: How can Agile Methods be put into Use in Large-Scale Software Development Organizations?

This research question is a question about transformation. Changing from traditional development to *Agile Software Development* is like changing from football to rugby: the rules of the game are different. That is because traditional systems are based on reductionism and approach a system as the sum of its parts, while *Agile Software Development* and agility are based on complexity. Friction and tension can easily arise between parts of the organization that work on the basis of different assumptions if only a part of the organization goes through the transformation. To judge from experience in the organization under research, transformation of the whole organization would lead to best results.

The understanding of *Complex Adaptive Systems* can be used when defining how to apply *Agile Methods*. The framework for *Complex Adaptive Systems* tells us that in order to work well, any such system must have agreement on: 1) information and connections; 2) intention (*Goals*); 3) communication among the agents; 4) interaction; 5) payoffs; 6) strategies and actions; 7) cognition, and 8) the focus and heterogeneity of the model. (Miller and Page 2007)

Agile Methods can be used by forming different models (following the strategy that defines what we are trying to achieve with agility) based on *Goals*, *Means,* and *Enablers* and probing those in practice. The *Cynefin Framework* (Snowden and Boone 2007) suggests probing before decision making as a leadership mechanism with *Complex Systems* since there are many possible choices — and one can only know after probing which alternative leads to the best results.

The *Goals, Means,* and *Enablers* model for agile transformation can be used to initiate change projects that put *Enablers* in place. Once the organization starts to get feedback, the model can be adjusted on the basis of this feedback.

The communication of *Goals*, *Means,* and *Enablers* is also one way to reach a unity of *Intentions* across the organization. One possible high-level model was presented in Fig. 20. The *Enablers* in this model were picked from Scrum and Extreme Programming practices, which lead to the optimization of development speed. Another type of framework could be developed to optimize for other *Goals*.

7.1.1 On what Level(s) of the Organization should Agile Methods be used in order to maximize the benefits to the Enterprise?

This research question is about the levels on which Agile Methods should be used within the organization. Using *Agile Methods* on the team level only gives benefits on the team level only, and may lead to friction in interfaces between that team (or teams) and the rest of the organization. When *Agile Methods* are used only at some level, the conflicting models (the traditional one that is based on reductionism, seeing the whole as the sum of its parts, and the agile holistic model) cause tension and misalignment in the organization.

If the enterprise would like to maximize the benefits on the enterprise level, it should use an enterprise-level model for its agile adoption. Changing the whole organization and transforming all the levels of it that are interlinked or connected to each other could help the organization to reach the goals it was pursuing with its *Agile Enterprise Model*. This requires the new way of working to be communicated across the whole organization.

Many *Agile Methods* (like Scrum or Extreme Programming) indicate how to apply these methods on the team level only. One way to use these methods on the project level is presented in Paper IV and Appendix E. One possible model for how to use these methods on the enterprise level is presented in Appendix D.

7.1.2 How can these Methods be applied in Large-Scale Software Development Organizations?

This research question is about how to build a model of an *Agile Enterprise*. Using *Agile Methods* on a large scale requires organization-wide learning. The researcher presented a model of an *Agile Enterprise* as early as in 2007 (see Appendix D) but that was not widely accepted in the subject organization, as the organization was not ready for it.

Learning Organizations explain the underlying assumptions, expanding the analytical frame to explicitly identify and challenge the underlying assumptions (Argyris 1993). Transforming an organization from a traditional one into an *Agile Enterprise* requires this type of learning. The *Goals*, *Means,* and *Enablers* presented in Fig. 19 provide a mechanism for communicating and presenting these underlying assumptions.

The model for agile transformation that is presented in Fig. 20 and which is based on *Enablers* retrieved from Scrum and Extreme Programming practices is optimized for the speed of development. There are also other *Agile Aspects* that could be pursued with organization-wide agility.

The *Agile Aspects* (i.e., what is gained by being agile) are context-sensitive, relative, and improved in relation to a specific organization level. The extent of the improvement depends on the organizational level the agile transformation can penetrate: the higher the level transformed, the bigger the improvement.[40] An *Agile Enterprise* would invent new and better ways to be agile continuously, based on the feedback it receives.

Organization-wide agility could be described as force vectors on each organization level. Getting permanent improvements happening amongst such a set of vectors would require the alignment of *Goals, Means,* and *Enablers.* This would require strong coherence and unity of improvement measures, and thus most probably would best be led from the top.

Failure to get the *Goals*, *Means*, and *Enablers* aligned would lead to problems — as well as to the misalignment of agile actions on various

[40] An example: resource fluidity is an *Aspect* of strategic agility. Resource fluidity can happen at the team level (team members changing tasks with each other), across teams (team members or work is exchanged across the teams), or on the organizational level (all team members can change jobs as needed, or all work can be reallocated or teams can be reallocated). An organization needs to understand the impact, causes, and consequences of the different options. The solution selected must be based both on the organization's capabilities and the market needs (how much flexibility is actually needed).

organizational levels. These kinds of misalignments would manifest themselves in practice as water-scrumming or similar problems. Such problems need to be detected so that corrective steps can be taken. A survey can be proposed as one tried and tested tool for this kind of organizational development. A survey can make the organization raise its awareness of its own actual state, and the future development efforts can be based on that knowledge.

When adopting *Agile Methods*, it should be remembered that a traditional reductionistic approach to processes and process development no longer holds but one should understand agile processes from a more holistic viewpoint. When people complain that they are "failing Scrum" they typically list a set of practices that a specific team or specific teams do not follow, e.g., like the Nokia Test (Santana 2009). Even the Nokia Test is not a perfect measure, as Scrum teams may bend and twist the rules if they learn new and better ways of being agile from a retrospective viewpoint. Because of this learning aspect, sometimes people demanding strict obedience to such rules or such tests are seen by others as "Agile Zealots" (Berard 2003). The problem is not if the team fails to obey some rule, but if the team fails to obey the spirit or "Geist" of the self-emergence of these agile models. This provides an opportunity for using statistical tests for organizational development. Improvement itself may also be far more complex than a tick-in-the-box checklist of practices to be followed up and involve people rather than processes.

7.2 Limitations of the Work

The research done in this dissertation is *Action Research* of a quasi-experimental nature (see Chapter 4). Wiersma and Jurs (2005) state that when considering problems of validity with quasi-experimental research, limitations should be clearly identified, the equivalence of groups should be discussed, and possible representativeness and generalizability should be argued on a logical basis.

The biggest limitation on the generalization of the results is that this research was done in one company only. That should not affect the theory-building part, but here the theory-testing part could be biased by the organizational culture and the specific interpretation of agility. A question for *External Validity* remains, i.e., how broadly these results can be generalized. But the results indicating that agile transformation needs to be organization-wide and the model of agile transformation — illustrated in Fig. 20 — are generic and not bound to any organization as such.

The scope of this work is limited to one organization only. This impacts on, e.g., the figures represented in this dissertation: the possible arrows in Fig. 20 going outside the subject organization or external forces impacting on the subject organization are mostly missing. Having a wider ecosystem type of view on an organization might impact on the results.

In this work, the usage of *Agile Methods* on a large scale is mainly reflected against the CAS framework *Intention* (*Goals*) by studying the opinions of the users. The *Strategies and actions* in the CAS framework were reflected to some extent through the perceived benefits of *Agile Methods*. This gives us a view on how well the current model has been adopted into use (and how well the *Goals*, *Enablers,* and *Means* have been put into use), but focusing on other points in the CAS framework might prove to be useful too.

7.3 Future Work

The future research possibilities are numerous. First, the ideas presented in this dissertation could be applied in different organizational contexts and settings and expanded further. Second, the research done in this dissertation is not comprehensive, but limited by, e.g., the viewpoints on the CAS framework that were selected, which leaves room for studying *Agile Methods* from other viewpoints. Third, there are many other viewpoints that are excluded, including, e.g., change management. The new understanding introduced in this work could lead to a transformation of the current understanding of large-scale development organizations, which typically manifests itself in the culture of the organization.

7.3.1 Expanding the Ideas Presented in this Work

This research gives some insights into the organizational development towards agility, but is far from being comprehensive. A model for agile transformation has been presented. The model is a thinking tool that can be applied in several different ways.

The *Aspects* of an *Agile Enterprise* need further clarification and testing. We should better understand how the different *Aspects* could be reached at the enterprise level, so that maximum benefits could be achieved.

7.3.2 Expanding from the Selected Viewpoints

The study of the relations between the attitudes and opinions of practitioners and their agile experience, i.e., the study of *shared intention* was just one out of many possible viewpoints on agile development, and just one angle on an *Agile Enterprise* as a *Complex Adaptive System*. Most urgently, a model needs to be developed of the relationship between how beneficial a certain method is perceived to be and if that is related to the actual usage of the methods. There is also still room for studying the different practices and their impact on performance and quality, although quite many such studies already exist (Rico, Sayani, and Sone 2009).

7.3.3 Future Research on other Excluded Viewpoints

Further research on the measurable impacts of the usage of *Agile Methods* is needed in the industry. This could range from the deployment of a holistic agile model to the application of individual *Agile Practices* such as continuous integration or test-driven development in large organizations. Through these studies we would be able to assess the benefits that are claimed, such as improved quality, time-to-market, and developers' morale on the scale of an *Agile Enterprise*.

This dissertation studies the usage of *Agile Methods* in a large organization, but not from a change management viewpoint. There are psychological and motivational viewpoints that were not discussed. This leaves room for many interesting research areas and topics in the future. There are also numerous opportunities to develop the leadership aspect of *Agile Enterprises*.

Organizations deploy *Agile Methods* because of business benefits. How much organic growth, as presented by Holling (2001), can be generated with these methods is an interesting topic. Do *Agile Methods* have their limits? How resilient can an *Agile Enterprise* be? How are agility and resilience linked with an organization's business model? What is the relationship between disruptive innovations that combine innovation and technology in new ways and thus typically provide this kind of organic growth and *Agile Methods*? Do *Agile Methods* provide better soil for growing disruptive innovations than traditional methods?

7.3.4 Researching the Transformation of Organization Culture

In organizations based on mechanistic models and reductionism one of the assumptions is that there is some superior intelligence that knows the best way to organize and run the work. In models based on complexity this view is changed. The system is developed by heterogeneous agents that may or may not be intelligent. The discovery of *group intelligence* — that a group can statistically give better answers than even the most educated professionals (Fisher 2009) — can change how organizations and processes have been developed so far. So far group intelligence has been applied, e.g., for developing concepts for new television series and new software games, but applying group intelligence to organizational development provides many interesting opportunities. This could change how we think about processes and process development.

As a summary it can be concluded that the opportunities for further research are numerous; we are only at the beginning of understanding agility.

References

Aaen I (2003) Software Process Improvement: Blueprints Versus Recipes. IEEE Software 20(5): 86–93.

Abbas N, Gravell A & Wills G (2008) Historical Roots of Agile Methods: Where did "Agile Thinking" come from? Agile Processes in Software Engineering and Extreme Programming. Lecture Notes in Business Information Processing 9(4): 94–103. DOI: 10.1007/978-3-540-68255-4_10.

Abrahamsson P (2007a) Speeding up Embedded Software Development: Application of Agile Processes in Complex System Development Projects. Innovation Technology for European Advancement (ITEA) Agile Innovation Report. URI: http://www.itea2.org/innovation_reports. Cited May 2012. URI: http://www.itea2.org/project/result/download/result/5583. Cited May 2012.

Abrahamsson P (2007b) Agile Software Development of Embedded Systems. ITEA2 Symposium, Berlin, Germany, 18–19 October. URI: http://www.agile-itea.org/public/papers/ITEA-Symposium_oct-07.pdf. Cited May 2012.

Abrahamsson P, Oza N & Siponen M (2010) Agile Software Development Methods: A Comparative Review. Agile Software Development: Current Research and Future Directions. Springer.

Agile Manifesto (2001) URI: http://www.agilemanifesto.org. Cited July 2011 and May 2012.

Ambler SW (2007) Disciplined Agile Software Development: Definition. URI: http://www.agilemodeling.com/essays/agileSoftwareDevelopment.htm. Cited Nov 2007 and May 2012.

Ambler S (2010a) Reworking the Agile Manifesto. URI: https://www.ibm.com/developerworks/mydeveloperworks/blogs/ambler/entry/reworking_the_agile_manifesto14?lang=en. Cited July 2011 and May 2012.

Ambler S (2010b) Agile Data Method. URI: http://www.agiledata.org/. Cited July 2011 and May 2012.

Ambler S (2011) Examining the Agile Manifesto. URI: http://www.ambysoft.com/essays/agileManifesto.html. Cited July 2011 and May 2012.

Anderson DJ (2003) Agile Management for Software Engineering: Applying the Theory of Constraints for Business Results. Prentice Hall.

Anderson DJ (2010) Kanban: Successful Evolutionary Change for Your Technology Business. Blue Hole Press.

Appelo J (2010) Complexity vs. Lean, the Big Showdown. Proceedings of Lean Enterprise Software and Systems. Springer.

Appelo J (2011) Management 3.0. Leading Agile Developers, Developing Agile Leaders. Addison-Wesley.

Argyris C (1993) Knowledge for Action. A Guide to Overcoming Barriers to Organizational Change. Jossey-Bass.

Atkinson SR & Moffat J (2005) The Agile Organization: From Informal Networks to Complex Effects and Agility. Information Age Transformation Series.

Avison D, Lau F, Myers M & Nielsen PA (1999) Action Research. Communications of the ACM 42(1): 94–97.

Barnett L (2003) Best Practices for Agile Development. Forrester Research, August 27, 2003.

Barnett L (2004) Adopting Agile Development Processes: Improve Time-To Benefits For Software Projects. Forrester Research, March 25, 2004.

Bassett PG (1997) Framing Software Reuse: Lessons from the Real World. Yourdon Press Computing Series.

Beck K (2004) Extreme Programming Explained. 2nd ed. Addison-Wesley.

Beck K (2003) Test-Driven Development by Example. Addison-Wesley.

Benefield G (2008) Rolling out Agile in a Large Enterprise. Proceedings of the 41st Hawaii International Conference on System Sciences.

Berard E (2003) Misconceptions of the Agile Zealots. The Object Agency, L.L.C. URI: http://svspin.org/events/2003_08b.pdf. Cited May 2012.

Bogsnes B (2007) Implementing Beyond Budgeting: Unlocking the Performance Potential. Wiley.

Blackburn J, Scudder G & Wassenhove LN (1996) Improving Speed and Productivity of Software Development: A Survey of European Software Developers. IEEE Transactions on Software Engineering 12: 875–885.

Boehm B & Turner R (2004) Balancing Agility and Discipline – A Guide for the Perplexed. Boston MA, Addison-Wesley/Pearson Education.

Carr W & Kemmis S (1986) Becoming Critical. Education, Knowledge and Action Research. Routledge.

Chow T & Cao D (2008) A Survey Study of Critical Success Factors in Agile Software Projects. Journal of Systems and Software 81: 961–971.

Cockburn A (1998) Surviving Object Oriented Projects. Addison-Wesley Professional.

Cockburn A (2001) Agile Software Development. Addison-Wesley.

Cockburn A & Highsmith J (2001) Agile Software Development, the People Factor. Computer 11: 131–133. DOI 10.1109/2.963450.

Cockburn A (2005) Two Case Studies Motivating Efficiency as "Spendable" Quantity. Proceedings of the International Conference on Agility, 2005.

Cockburn A (2006) Agile Software Development: the Cooperative Game. 2nd ed. Addison-Wesley.

Cohn M (2007) Agile Estimation and Planning. Prentice Hall.

Cohn M (2009) Succeeding with Agile: Software Development using Scrum. Addison-Wesley.

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): 37–44.

Curtis B, *et al.* (1988) A Field Study of the Software Design Process for Large Systems. CACM 31(11): 1268–1287.

Davison R, Martinsons M & Kock N (2004) Principles for Canonical Action Research. Info Systems J 14: 65–86.

Dooley K (1997) A Complex Adaptive Systems Model of Organization Change. Nonlinear Dynamics, Psychology, and Life Sciences, Vol. 1, No. 1. Human Sciences Press, Inc.

Doz Y & Kosonen M (2008) Fast Strategy. How Strategic Agility will help You Stay ahead of the Game. Wharton School Publishing.

Eckstein J (2004) Agile Software Development in the Large: Diving Into the Deep. Dorset House Publishing.

Evans ND (2002) Business Agility. Strategies for Gaining Competitive Advantage through Mobile Business Solutions. Prentice Hall.

Feller J & Fitzgerald B (2000) "A Framework Analysis of the Open Source Software Development Paradigm", in Proceedings of the Twenty-First International Conference on Information Systems (Brisbane, Queensland, Australia). International Conference on Information Systems. Association for Information Systems, Atlanta, GA: 58–69.

Field A (2009) Discovering Statistics using SPSS. third ed. Sage Publications.

Fisher L (2009) The Perfect Swarm: The Science of Complexity in Everyday Life. Basic Books.

Fowler M & Highsmith J (2001) The Agile Manifesto. Software Development August.

Fowler M (2004) Is Design Dead? http://www.martinfowler.com/articles/designDead.html. Cited July 2011 and May 2012.

Fowler M (2009) Feature Branch. URI: http://martinfowler.com/bliki/FeatureBranch.html. Cited May 2012.

Fuggetta A (2000) Software Process: A Roadmap. Proc. Future of Software Engineering (ICSE): 27–34.

Garvin D (1987) Competing on the Eight Dimensions of Quality. Harvard Business Review 65(6): 101–109.

Glazer H (2001) "Dispelling the Process Myth: Having a Process Does Not Mean Sacrificing Agility or Creativity." CrossTalk 14(11): 27–30.

Gnatz M, *et al.* (2003) The Living Software Development Process. SQP 5(3): 4–16.

Goldman S, Naegel R & Preiss K (1994) Agile Competitors and Virtual Organizations: Strategies for Enriching the Customer. Wiley.

Gould P (1997) What is Agility? IEE Manufacturing Engineer 76(1): 28–31.

Grant T (2010) Tech Vendors Supporting Agile Must Be Adaptive. For Technology Product Management & Marketing Professionals. Forrester Research.

Grenning J (2001) Launching XP at a Process-Intensive Company, IEEE Software, November/December: 3–9.

Haeckel SH (1999) Adaptive Enterprise: Creating and leading Sense-and-Respond Organizations. Harvard Business Press.

Hamel G (2007) The Future of Management. Harvard Business School Press.

Highsmith J (1999) Adaptive Software Development: A Collaborative Approach to Managing Complex Systems. Dorset House Publishing Company.

DOI, Declaration of Interdependence (2005) URI: http://www.pmdoi.org/. Cited July 2011 and May 2012.

Highsmith J (2002) Agile Software Development Ecosystems. Addison-Wesley.

Highsmith J (2005) Agile for the Enterprise: From Agile Teams to Agile Organizations. Executive Report 6(1). http://www.cutter.com/project/fulltext/reports/2005/01/index.html. Cited May 2012.

Holling C (2001) Understanding the Complexity of Economic, Ecological, and Social Systems. Springer Ecosystems 4(5): 390–405. DOI: 10.1007/s10021-001-0101-5.

Hugos MH (2007) The Greatest Innovation since the Assembly Line. URI: http://www.bpminstitute.org/articles/article/article/the-greatest-innovation-since-the-assembly-line.html. Cited Jan 2012 and May 2012.

Hugos M. H (2009) Business Agility. Sustainable Prosperity in a Relentlessly Competitive World. Microsoft Executive Leadership Series. John Wiley & Sons.

Humble J & Farley D (2011) Continuous Delivery. Reliable Software Releases through Build, Test, and Development Automation. Addison-Wesley.

Industrial XP. URI: http://industrialxp.org/. Cited August 2005 and May 2012.

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

IEEE (2011) URI: http://www.ieee.org/about/index.html. Cited July 2011 and May 2012.

Iivari J & Iivari N (2010) Organizational Culture and the Deployment of Agile Methods: the Competing Values Model View. Agile Software Development, Current Research and Future Directions. Springer.

Ikonen M, Kettunen P, Oza N & Abrahamsson P (2010) Exploring the Sources of Waste in Kanban Software Development Projects. Software Engineering and Advanced Applications (SEAA), 2010 36th EUROMICRO Conference. DOI: 10.1109/SEAA.2010.40.

Johnson NF (2007) Two's Company, Three is Complexity: A Simple Guide to the Science of All Sciences. Oneworld.

Järvinen P & Järvinen A (2004) Tutkimustyön metodeista. Opinpajan kirja. Tampere.

Järvinen (2010) Pertti Järvinen’s Web Page. URI: http://www.cs.uta.fi/~pj/PJWeb.htm. Cited July 2011 and May 2012.

Katayama H & Bennett D (1999) Agility, Adaptability and Leanness: A Comparison of Concepts and a Study of Practice. Int J Production Economics 60–61: 43–51.

Keenan F (2004) Agile Process Tailoring and Problem Analysis (APTLY), Proceedings of the 26th International Conference on Software Engineering: 45–47.

Kettunen P & Laanti M (2005) How to Steer an Embedded Software Project: Tactics for Selecting the Software Process Model. Information & Software Technology 47(9): 587–608.

Kettunen 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).

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

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

Kettunen P (2009) Agile Software Development in Large-Scale New Product Development Organization: Team-Level Perspective. Helsinki University of Technology, Doctoral Dissertation. TKK Dissertations 186.

Kettunen P (2010) A Tentative Framework for Lean Software Enterprise Research and Development. Proceedings of Lean Enterprise Software and Systems Conference, Springer.

Kinnula A (1999) Software Process Engineering in Multisite Environment. An Architectural Design of a Software Process Engineering System. Academic Dissertation, Department of Information Processing Science and Infotech, Oulu University. A333.

Kim WC & Mauborgne R (2005) Blue Ocean Strategy: How to Create Uncontested Market Space and Make Competition Irrelevant. Harvard Business Press.

Kniberg H & Skarin M (2010) Kanban and Scrum – Making the Most of Both. Lulu.com.

Kontio J (2005) Why Agile Now? The Agility Needed for Business Success at 3rd Agile Software Development Seminar (ASDS 2005), Oulu, Finland, Sept 29.

Korhonen K (2010) Exploring Defect Data, Quality and Engagement during Agile Transformation at a Large Multisite Organization. Proceedings of XP 2010 conference. Springer.

Korhonen K (2012) Supporting Agile Transformation with Defect Management in Large Distributed Software Development Organization. Dissertation, Tampere University of Technology, Publication 1032.

Kähkönen T & Abrahamsson P (2004) Achieving CMMI Level 2 with Enhanced Extreme Programming Approach, Proceedings of the Fifth International Conference of Product Focused Software Process Improvement (PROFES): 378–392.

Kähkönen T (2004) Agile Methods for Large Organizations – Building Communities of Practice. In Proc Agile Development Conf. (ADC): 2–10.

Laanti M (2008) Implementing Program Model with Agile Principles in a Large Software Development Organization. COMPSAC '08 Proceedings of the 2008 32nd Annual IEEE International Computer Software and Applications Conference. DOI: 10.1109/COMPSAC.2008.116.

Laanti M (2010) Agile Transformation Study at Nokia – One Year After. Lean Enterprise Software and Systems. Lecture Notes in Business Information Processing 65, Part 1: 3–19. DOI: 10.1007/978-3-642-16416-3_2.

Laanti M, Salo O & Abrahamsson P (2010) Agile Methods Rapidly Replacing the Traditional Methods at Nokia: A Survey of Opinions on Agile Transformation. Information and Software Technology. URI: http://dx.doi.org/10.1016/j.infsof.2010.11.010.

Larman C (2003) Agile & Iterative Development. A Manager’s Guide. Addison-Wesley Professional.

Larman C & Vodde B (2009) Scaling Lean & Agile Development: Thinking and Organizational Tools for Large-Scale Scrum. Addison Wesley.

Larman C & Vodde B (2010) Practices for Scaling Lean & Agile Development. Large, Multisite, and Offshore Product Development with Large-Scale Scrum. Addison-Wesley.

Laukkanen S (2012) Making sense of ambidexterity. Doctoral dissertation, Hanken Economics and Society.

Lazko W & Sounders D (1995) Four Days with Dr. Deming. Addison-Wesley.

Leffingwell D (2007) Scaling Software Agility. Best Practices for Large Enterprises. The Agile Software Development Series.

Leffingwell D (2011) Agile Software Requirements. Lean Requirements Practices for Teams, Programs, and the Enterprise. Addison-Wesley.

Lehrer J (2009) How We Decide. Mariner.

Lehtonen I (2009) Communication Challenges in Agile Global Software Development. University of Helsinki. URI: http://www.mendeley.com/research/communication-challenges-agile-global-software-development-1/. Cited May 2012.

Lindvall M, *et al.* (2004) Agile Software Development in Large Organizations. IEEE Computer 37(12): 26–34.

Liker J (2004) The Toyota Way: 14 Management Principles from the World's Greatest Manufacturer. McGraw-Hill.

Lycett M, Macredie R, Patel C & Paul R (2003) Migrating Agile Methods to Standardized Development Practice. Computer 36: 79–85.

Lyytinen K & Rose G M (2006) Information System Development Agility as Organizational Learning. European Journal of Information Systems 15: 183–199.

Manzoni L & Price R (2003) Identifying Extensions Required by RUP (Rational Unified Process) to Comply with CMM (Capability Maturity Model) Levels 2 and 3. IEEE Transactions on Software Engineering 29(2): 181–192.

Maples C (2009) Enterprise Agile Transformation: The Two-Year Wall. Proceedings of the 2009 Agile Conference . URI: http://doi.ieeecomputersociety.org/10.1109/AGILE.2009.62.

McMahon P (2005) Extending Agile Methods: A Distributed Project and Organizational Improvement Perspective. CrossTalk 18(5): 16–19.

Melnik G & Mauer F (2006) Comparative Analysis of Job Satisfaction in Agile and Nonagile Software Development Teams, Lecture Notes in Computer Science 4044: 32–42. DOI:10.1007/11774129_4.

Miles M & Huberman A (1994) Qualitative Data Analysis; An Expanded Sourcebook. Second ed. SAGE Publications.

Miller J & Page S (2007) Complex Adaptive Systems: An Introduction to Computational Models of Social Life. Princeton Paperbacks.

Moore GA (2011) Escape velocity. Free your company's future from the pull of the past. HarperBusiness.

Morgan J & Liker J (2006) The Toyota Product Development System. New York, Productivity Press.

Nardi P (2003) Doing Survey Research: a Guide to Quantitative Methods, Pearson Education.

Nerur S, Mahapatra R & Mangalaraj G (2005) Challenges of Migrating to Agile Methodologies. Communications of the ACM 48(5): 73–78.

Nightingale D (2002) Development of a Lean Enterprise Transformation Maturity Model. Information, Knowledge, Systems Management 3(1): 15–30.

Norton D (2008) Five Reasons Organizations Fail to Adopt Agile Methods. Gartner 9.

Nupponen A (2003) Käsiteanalyysi asiantuntijan työvälineenä. In Koskela M & Pilke N (eds) Kieli ja asiantuntijuus. AFinLA-vuosikirja. Jyväskylä, Suomen Soveltavan Kielitieteen Yhdistys: 13–24.

O'Brian R (1998) An Overview of the Methodological Approach of Action Research. Faculty of Information Studies, University of Toronto. URI: http://www.web.net/~robrien/papers/arfinal.html. Cited May 2012.

Ohno T (1988) Toyota Production System. Beyond Large-Scale Production. Productivity Press.

Ojanperä T & Prasad R (1998) Wideband CDMA for Third Generation Mobile Communications. Artech House Publishers.

Osterwalder A & Pigneur Y (2010) Business Model Generation: A Handbook for Visionaries, Game Changers, and Challengers. Wiley.

Ould M (1999) Managing Software Quality and Business Risk. Chichester UK, John Wiley & Sons.

Oza N & Abrahamsson P (2009) Building Blocks of Agile Innovation. Booksurge.

Pulkkinen M & Hirvonen AP (2005) EA Planning, Development and Management Process for Agile Enterprise Development. Hawaii International Conference on System Sciences. Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05) - Track 8: 223c.

Poppendieck M & Poppendieck T (2003) Lean Software Development: An Agile Toolkit. Addison Wesley.

Poppendieck M & Poppendieck T (2010) Leading Lean Software Development: Results are Not the Point. Addison Wesley.

Preiss K (2005) Agility – the Origins, the Vision and the Reality. Proceedings of the International Conference on Agility.

Rand C & Eckfeldt B (2004) Aligning Strategic Planning with Agile Development: Extending Agile Thinking to Business Improvement. In Proc Agile Development Conf. (ADC): 78–82.

Rajlich V (2006) Changing the paradigm of software engineering. ACM Communications 49(8). DOI: 10.1145/1145287.1145289.

Rapkin B & Luke D (1993) Cluster Analysis in Community Research: Epistemology and Practice, American Journal of Community Psychology 21: 247–277.

Reinertsen D (2009) The Principles of Product Development Flow. Second Generation Lean Product Development. Celeritas Publishing.

Rico D, Sayani H & Sone S (2009) The Business Value of Agile Methods. Maximizing ROI with Just-In-Time Processes and Documentation. J Ross Publishing.

Rogers E (2003) Diffusion of Innovations. 5th ed. Free Press.

Salo O (2007) Enabling Software Process Improvement in Agile Software Development Teams and Organisations. Doctoral Thesis, University of Oulu. VTT Publications. URI: http://www.vtt.fi/inf/pdf/publications/2006/P618.pdf. Cited May 2012.

Salo O & Abrahamsson P (2008) Agile Methods in European Embedded Software Development Organizations: a Survey on the Actual Use and Usefulness of Extreme Programming and Scrum. IET Software 2: 58–64.

Santana C, Caetano D, Alexandre S & Rocha V (2009) Measuring the Effectiveness of Nokia Test in Very Small Teams. Proceedings of Software Engineering and Applications (SEA).

Schuh P (2004) Integrating Agile Development in the Real World. Charles River Media.

Schwaber K & Beedle M (2002) Agile Software Development with Scrum. Prentice-Hall.

Schwaber K (2004) Agile Project Management with Scrum. Microsoft Press.

Schwaber K (2007) The Enterprise and Scrum. Microsoft Press.

Schwaber K, Laganza G & D'Silva D (2007) The Truth about Agile Processes: Frank Answers to Frequently Asked Questions. Forrester Report.

Schön D (1991) The Reflective Practitioner: How Professionals Think in Action. Ashgate Publishing Limited.

Senge P (2006) The Fifth Discipline. The Art & Practice of The Learning Organization. Random House Business Books.

Sharifi H & Zhang Z (1999) A Methodology for Achieving Agility in Manufacturing Organizations: An Introduction. Int J Production Economics 62: 7–22.

Smith M (2007) Action research. The Encyclopedia of Informal Education. URI: http://www.infed.org/research/b-actres.htm. Cited May 2012.

Smith P (2007) Flexible Product Development: Building Agility for Changing Markets. Jossey-Bass.

Snowden D & Boone M (2007) A Leader’s Framework for Decision Making. Harvard Business Review.

Snowden D (2012) The new 3 C’s. Complexity Based Approaches to Project Management. Keynote Presentation at Scan Agile 2012 conference. URI: http://vimeo.com/39200403. Cited May 2012.

Sommerville I (1996) Software Process Models. ACM Computing Surveys 28(1): 269–271.

Stevens (2011) What is next for the Agile Manifesto. URI: http://www.dennisstevens.com/2011/02/13/whats-next-for-the-agile-manifesto/. Cited August 2011 and May 2012.

Stoneleigh (2010) Fractal Adaptive Cycles in Natural and Human Systems. URI: http://theautomaticearth.blogspot.com/2010/01/january-1-2010-fractal-adaptive-cycles.html. Cited May 2011 and May 2012.

Subramaniam V & Hunt A (2005) Practices of an Agile Developer – Working in the Real World. The Pragmatic Bookshelf.

Susman G & Evered R (1978) An Assessment of the Scientific Merits of Action Research. Administrative Sciences Quarterly 23: 582–603.

Sutherland J (2001) Agile Can Scale. IT Journal 14(12). Cutter. http://codespeak.net/svn/pypy/tag/funding-final/agile_can_scale.pdf. Cited May 2012.

Sutherland J (2004) Nativity Scene: How Scrum Was Born. URI: http://jeffsutherland.com/2004_12_01_archive.html. Cited May 2012.

Sutherland J (2005) Future of Scrum: Parallel Pipelining of Sprints in Complex Projects. Agile Development Conference Proceedings. URI: http://doi.ieeecomputersociety.org/10.1109/ADC.2005.28

Sutherland J & Schwaber K (2007) The Scrum Papers: Nuts, Bolts and Origins of an Agile Process. URI: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.108.814.

Svensson H & Höst M (2005) Introducing an Agile Process in a Software Maintenance and Evolution Organization. In Proceedings of 9th European Conference of Maintenance and Reengineering.

Takeuchi H & Nonaka I (1986) The New Product Development Game. Harvard Business Review, January 1.

Tessem B & Mauer F (2007) Job Satisfaction and Motivation in a Large Agile Team. Lecture Notes in Computer Science 4536: 54–61. DOI: 10.1007/978-3-5.40-73101-6_8.

Tuomi J & Sarajärvi A (2002) Laadullinen tutkimus ja sisällönanalyysi. Tammi.

Turner R & Boehm B (2003) People Factors in Software Management: Lessons From Comparing Agile and Plan-Driven Methods. CrossTalk 16(12): 4–8.

Version One (2011a) State of Agile Development Survey 2011. Whitepaper. Author unknown. URI: http://www.versionone.com/pdf/2011_State_of_Agile_Development_Survey_Results.pdf.

Version One (2011b) Agile Development Methodologies. Accessed on May 2011 and May 2012. http://www.versionone.com/Agile101/Methodologies.asp

Victoria University of Wellington, New Zealand (2001) URI: http://www.victoria.ac.nz/lals/lwp/master-gfx/lwp-gfx/op2_diagram.gif. Retrieved August 2011. Cited May 2012.

Vidgen R & Wang X (2009) Coevolving Systems and the Organization of Agile Software Development. Information Systems Research 20(3): 355–376.

Vilkki K (2010) When Agile is not Enough. Lean Enterprise Software and Systems. Proceedings of First International Conference, LESS2010. Springer.

Vodde B (2004) Nokia Networks and Agile Development. Euromicro 2006. URI: http://www.odd-e.com/material/2006/nokia_agile.pdf. Cited May 2012.

Vähäniitty J & Rautiainen K (2008) Towards a Conceptual Framework and Tool Support for Linking Long-term Product and Business Planning with Agile Software Development. Proceedings of the 1st International Workshop on Software Development Governance. DOI: 10.1145/1370720.1370730.

Walker L & Avant K (1995) Strategies for Theory Construction in Nursing, 3rd ed. Appleton Lange, London.

Walker L & Avant K (1995) Concept analysis. In Walker L & Avant K (eds) Strategies for Theory Construction in Nursing. 3rd ed: 37–54. See also URI: http://xms.tmu.edu.tw/xms/read_attach.php?id=1608. Cited May 2012.

Wallace W (1971) The Logic of Science in Sociology. Transaction Publishers.

Waters M & Bevan J (2005) Journey to Lean. IEE Engineering Management 15(4): 10–13.

Wells D (2009) Extreme Programming. A Gentle Introduction. URI: http://www.extremeprogramming.org/. Cited July 2011 and May 2012.

West D & Grant T (2010) Agile Development: Mainstream Adoption Has Changed Agility. Forrester Research.

West D & Hammond J (2010) The Forrester Wave: Agile Development Management Tools. Forrester Research.

West D (2011) Water-Scrum-Fall Is The Reality Of Agile For Most Organizations Today. Manage the Water-Scrum and Scrum-Fall Boundaries To Increase Agility. Forrester Research.

Wheelwright S & Clark K (1992) Revolutionizing Product Development: Quantum Leaps in Speed, Efficiency, and Quality. The Free Press.

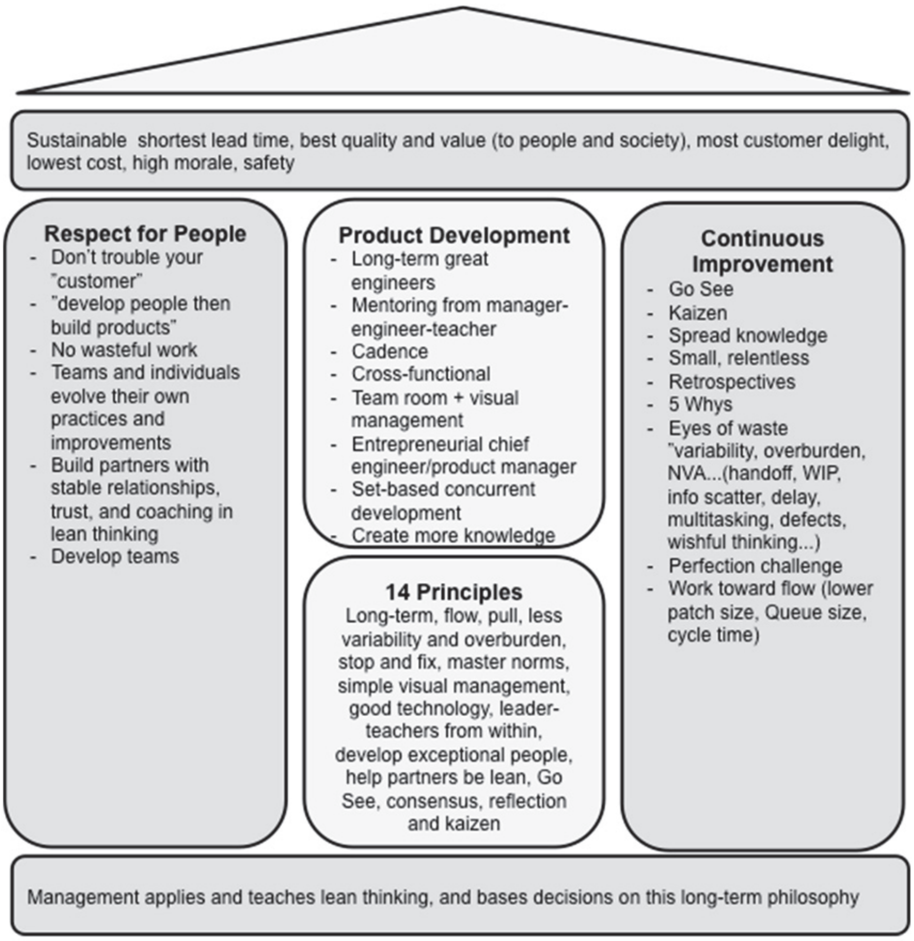
Wiersma W & Jurs S (2005) Research Methods in Education. An Introduction. Eighth Edition. Pearson Education.

Womack J & Jones D (1996) Lean Thinking. Banish Waste and Create Wealth in Your Corporation. Simon & Schuster.

Yourdon E (2002) Managing High-Intensity Internet Projects. Prentice Hall.

Zach (2011) Using Cynefin During RallyOn. URI: http://www.rallydev.com/coachingblog/?tag=cynefin. Cited July 2011 and May 2012.

Appendix A: House of Lean by Larman and Vodde


Sustainable shortest lead time, best quality and value (to people and society), most customer delight, lowest cost, high morale, safety

Respect for People
- Don't trouble your "customer"
- "develop people then build products"
- No wasteful work
- Teams and individuals evolve their own practices and improvements
- Build partners with stable relationships, trust, and coaching in lean thinking
- Develop teams

Product Development
- Long-term great engineers
- Mentoring from manager-engineer-teacher
- Cadence
- Cross-functional
- Team room + visual management
- Entrepreneurial chief engineer/product manager
- Set-based concurrent development
- Create more knowledge

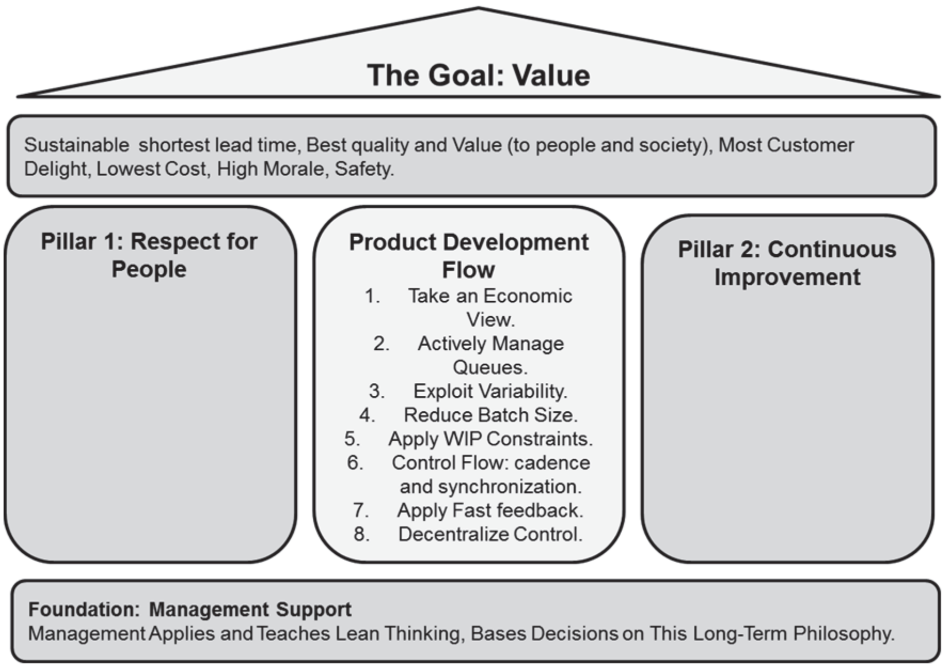
14 Principles
Long-term, flow, pull, less variability and overburden, stop and fix, master norms, simple visual management, good technology, leader-teachers from within, develop exceptional people, help partners be lean, Go See, consensus, reflection and kaizen

Continuous Improvement
- Go See
- Kaizen
- Spread knowledge
- Small, relentless
- Retrospectives
- 5 Whys
- Eyes of waste "variability, overburden, NVA...(handoff, WIP, info scatter, delay, multitasking, defects, wishful thinking...)
- Perfection challenge
- Work toward flow (lower patch size, Queue size, cycle time)

Management applies and teaches lean thinking, and bases decisions on this long-term philosophy

House of Lean by Larman and Vodde as presented by Larman and Vodde (2009).

Appendix B: House of Lean by Leffingwell



The Goal: Value
Sustainable shortest lead time, Best quality and Value (to people and society), Most Customer Delight, Lowest Cost, High Morale, Safety.
Pillar 1: Respect for People
Product Development Flow
1. Take an Economic View.
2. Actively Manage Queues.
3. Exploit Variability.
4. Reduce Batch Size.
5. Apply WIP Constraints.
6. Control Flow: cadence and synchronization.
7. Apply Fast feedback.
8. Decentralize Control.
Pillar 2: Continuous Improvement
Foundation: Management Support
Management Applies and Teaches Lean Thinking, Bases Decisions on This Long-Term Philosophy.

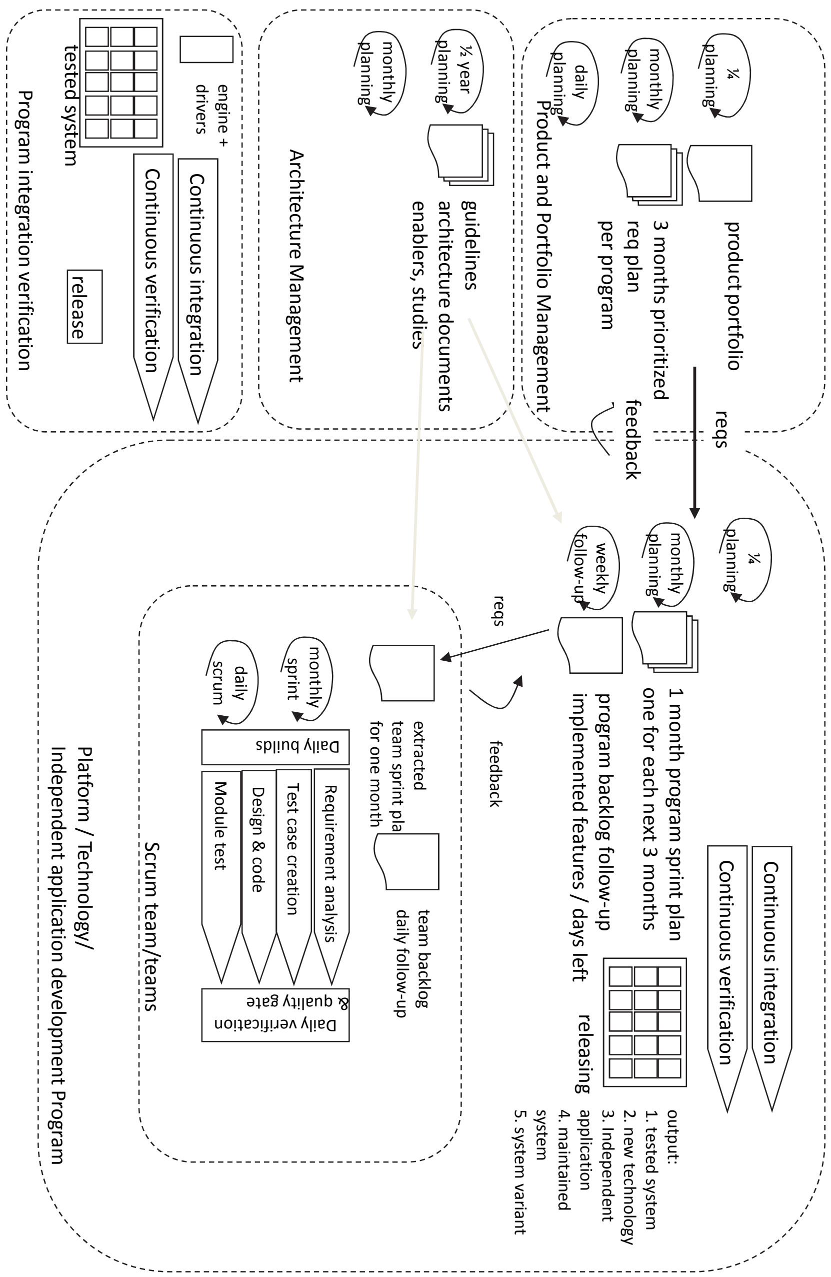
House of Lean as presented by Dean Leffingwell. (Leffingwell 2011).

Appendix C: Definitions of Agile Software Development

	Customer satisfaction / driven	Continuous delivery	Value	Early/ Frequent deliveries	Adaptability / Flexibility	Competitiveness	Customer satisfaction /benefit	Collaboration	Motivated individuals	Good environment	Support	Trust	Efficiency	Communication	Measure progress via deliverables	Sustainability	People	Focus on technical excellence	Good design as enabler of agility	Simplicity	Optimize work	Self-organization	Built-in improvement of efficiency and	Steerable / in-control	Rule-based	Profitability	Speed	Responsiveness
Agile Principles	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x					
Cockburn 2001													x	x			x							x	x			
Highsmith 2002					x																			x		x		
Anderson 2003																											x	
Larman 2003					x																						x	x
Schuh 2004			x														x										x	
Lyytinen 2006		x																										x
Suibramaniam 2005	x							x														x					x	
IEEE 2007																												x
Wikipedia 2007																												
DOI 2005			x		x			x		x							x											
Wikipedia 2010																						x						
Poppendieck 2010																												
Larman 2009																												
Larman 2010																												
Appelo 2011																												
Leffingwell 2011																												
Cohn 2009																								x			x	

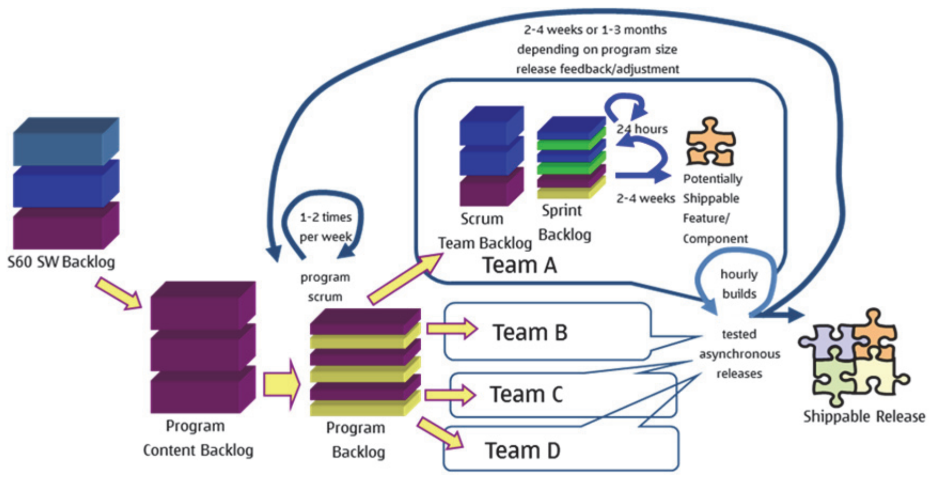
	Empowerment	Change	Feedback	Innovations / innovatitiveness	Iterative / non-sequential	Incremental	Less process-driven	Cost-conscious / low-cost	Conceptual framework / system	Maximize ROI	Delivery accuracy / predictability	Proactivity	Anticipation	Reliability	Situationality	Cross-functional	Evolutionary	Technical practices	Effectiveness	Systems improvement	Context-specific	Business benefits	Non-traditional	High-quality	Meets user needs	Process	Productivity	Visibility
Agile Principles																												
Cockburn 2001																												
Highsmith 2002																												
Anderson 2003																												
Larman 2003																												
Schuh 2004	x	x	x																									
Lyytinen 2006				x																								
Suibramaniam 2005					x	x	x	x																				
IEEE 2007					x																							
Wikipedia 2007					x				x																			
DOI 2005				x						x	x	x	x	x	x													
Wikipedia 2010					x	x										x	x											
Poppendieck 2010									x									x	x									
Larman 2009		x																										
Larman 2010																				x								
Appelo 2011																					x							
Leffingwell 2011																		x			x	x						
Cohn 2009					x			x			x												x	x	x	x	x	x

Appendix D: Blueprint for Agile Organization



Original drawn in 2007.

Appendix E: “Agile Program Management”



Original drawn in 2008

“There are about three kinds of scientists — the consolidator, the technical expert, and the artist. Consolidators accumulate and solidify advances and are deeply skeptical of ill-formed and initial, hesitant steps. That can have a great value at stages in a scientific cycle when rigorous efforts to establish the strength and value of an idea are central. Technical experts assess the methods of investigation. Both assume they are searching for the certainty of understanding.
In contrast, I love the initial hesitant steps of the "artist scientist" and like to see clusters of them. That is the kind of thing needed at the beginning of a cycle of scientific enquiry or even just before that. Such nascent, partially stumbling ideas, are the largely hidden source for the engine that eventually generates change in science. I love the nascent ideas, the sudden explosion of a new idea, the connections of the new idea with others. I love the development and testing of the idea till it gets to the point it is convincing, or is rejected. That needs persistence to the level of stubbornness and I eagerly invest in that persistence.” Holling, 2009

This thesis was created in my spare time; no paid work time was used for it.

Man's mind,
once stretched by a new idea,
never regains its original dimensions.
-- Oliver Wendell Holmes, Jr.

Publications

I Kettunen P & Laanti M (2005) How to Steer an Embedded Software Project: Tactics for Selecting the Software Process Model. Information & Software Technology 47(9): 587–608.

II Kettunen 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.

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

IV Laanti M (2008) Implementing Program Model with Agile Principles in a Large Software Development Organization. COMPSAC '08 Proceedings of the 2008 32nd Annual IEEE International Computer Software and Applications Conference. DOI: 10.1109/COMPSAC.2008.116.

V Laanti M, Salo O & Abrahamsson P (2010) Agile Methods Rapidly Replacing the Traditional Methods at Nokia: A Survey of Opinions on Agile Transformation. Information and Software Technology. DOI: 10.1016/j.infsof.2010.11.010.

VI Laanti M (2010) Agile Transformation Study at Nokia – One Year After. Lean Enterprise Software and Systems. Lecture Notes in Business Information Processing 65(1): 3–19. DOI: 10.1007/978-3-642-16416-3_2.

Original publications are not included in the electronic version of the dissertation.

ISBN 978-952-62-0033-0 (Paperback)
ISBN 978-952-62-0034-7 (PDF)
ISSN 0355-3191 (Print)
ISSN 1796-220X (Online)

UNIVERSITY of OULU
OULUN YLIOPISTO