Mälardalen University Licentiate Thesis
No.17

A PROCESS APPROACH FOR SENIOR MANAGEMENT INVOLVEMENT IN SOFTWARE PRODUCT DEVELOPMENT

Christina Wallin

2003

Department of Computer Science and Engineering
Mälardalen University

ISSN number: 1651-9256
ISBN number: 91-88834-17-4
Printed by Arkitektkopia, Västerås, Sweden
Distribution: Mälardalen University Press

ABSTRACT

To make business of software product development it is important that the right software products are developed the right way. Today there are a number of software development models that support project management to successfully execute software development projects, but they typically do not ensure that the resulting software products will be successful from a business perspective. To achieve this senior management involvement is needed to meet business objectives and deliver sustained and actual benefits to the customer and the organization.

This thesis presents one possibility to achieve effective senior management involvement in software product development projects by using stage-gate new product development models from traditional product development, and by suggesting a way to combine these models with contemporary software development models through pre-gate milestones.

For verification, experiences from a corporate wide software platform deployment initiative are collected and discussed. The initiative used a stage-gate new product development model for project selection and steering, and an incremental software development model for project management and execution.

ACKNOWLEDGMENTS

I want to thank my advisors Ivica Crnkovic, Fredrik Ekdahl and Stig Larsson for help and support during the work with the present thesis. I also want to thank ABB Corporate Research for giving me this opportunity to study and to collect experiences, and the Department of Computer Science and Engineering at Mälardalen University for providing good education and working environments.

Special thanks I want to give to Anders, for all support, and help, and discussions, and reviews, and pep-talk ;-)

Västerås 2003

LIST OF PUBLICATIONS

The following peer-reviewed papers and articles have been published at international conferences and journals.

Papers included in the thesis

The following papers and articles are included in the present thesis.

- Christina Wallin, Stig Larsson, Fredrik Ekdahl, Ivica Crnkovic, *Combining Models for Business Decisions and Software Development*, Proceedings of Euromicro Conference, September 2002

- Christina Wallin, Stig Larsson, Fredrik Ekdahl, *Integrating Business and Software Development Models*, IEEE Software November / December 2002

- Christina Wallin, Ivica Crnkovic, *Three Aspects of Successful Software Development Projects, "When are projects canceled, and why?",* Proceeding of Euromicro Conference September 2003

Papers not included in the thesis

The author has also co-authored the following paper;

- Rikard Land, Ivica Crnkovic, Christina Wallin, *Integration of Software Systems – Process Challenges*, Proceeding of Euromicro Conference September 2003

TABLE OF CONTENTS

LIST OF FIGURES

1 INTRODUCTION

"There are two ways to win at new products. One is to do projects right – building in the voice of the customer, doing the necessary up-front homework, using cross-functional teams, and so on. The other way is by doing the right projects – namely, astute project selection and portfolio management." [9]

In 1994 W. W. Gibbs wrote, in his article about Software's Chronic Crisis [13], that "… for every six new large-scale software systems that are put into operation, two others are cancelled" and "… three quarters of all large systems are 'operating failures' that either do not function as intended or are not used at all."

Although this problem was identified and the idea of Software Engineering was formed already 1968, the problem still existed in 1994. Some of the reasons were that "… the vast majority of computer code is still handcrafted … by artisans using techniques they neither measure nor are able to repeat consistently" and "… industry does not uniformly apply that which is well known best practice".

Others, e.g. Jones [18] and the Software Engineering Institute (SEI) [25], have through extensive software project evaluations identified that the fundamental problem is the organization's inability to manage software development projects, not the development methodologies or technologies used.

According to The Standish Group [29] CHAOS 2001 study [19], the situation is still similar today as it was ten or even 40 years ago. The Standish Group evaluation of 30 000 software application projects in large, medium and small cross-industry U.S. companies shows that only 28% were successful, i.e. they were completed on time and on budget, with all features and functions originally specified. 23% of the projects failed completely. Among the rest (the challenged projects) average cost overrun was 45%, time overrun was 63% and required features delivered was 67%. See: Figure 1 and Figure 2.

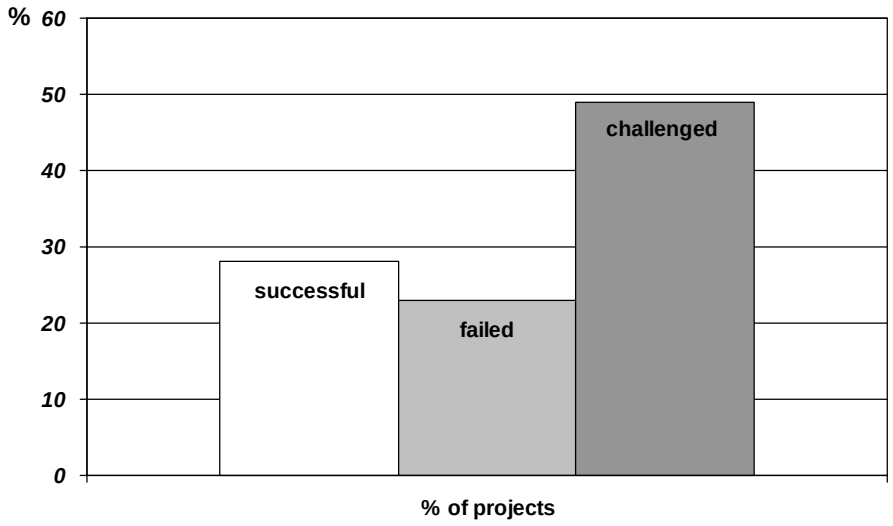

Figure 1: The resolution of 30 000 software application projects in large, medium and small cross-industry U.S. companies evaluated by The Standish Group in 2000 [19]

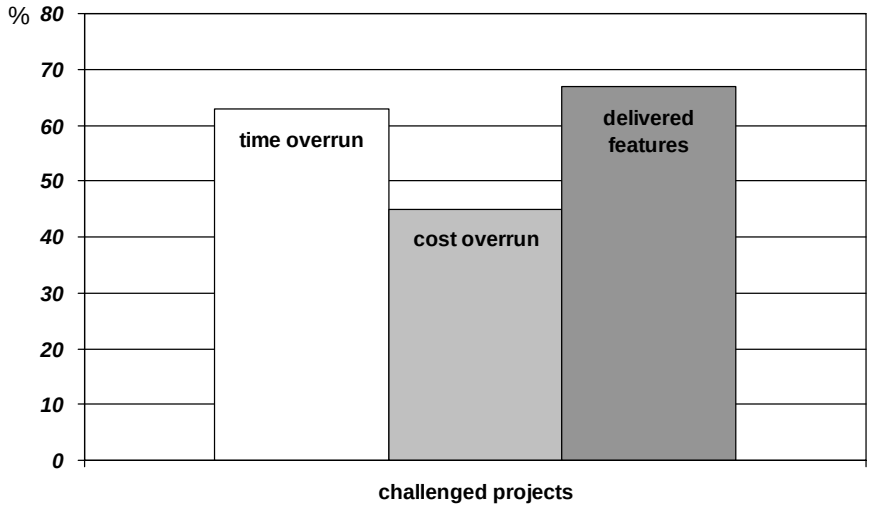

Figure 2: The average results of the challenged projects evaluated by The Standish Group in 2000 [19]

Two important success factors for software product development projects are: senior management involvement (executive support) [19][31] and customer (user, client) involvement [3][19][28][31]. The second, customer involvement, is typically addressed in some way or another in most software development models, while the involvement of senior management is not, at least not explicitly. But, it is important that senior management is involved in an effective way [7][28][30][31]. In addition to the traditional funding and staffing of projects, senior management should be supportive and assist project management in conflict resolution, risk management and change control, but not interfere in project work. Senior management should ensure that the project supports corporate goals, business objectives, benefits the customer and organization. In addition senior management should resolve political conflicts, approve or cancel projects and make business decisions. Senior management should not act as project managers, customer representatives or technical experts. Senior management should not change the goals and scope of the projects based on personal opinions instead of facts.

Today there are a number of available software development models, from the classical Waterfall model [27], to today's agile methodologies like Extreme Programming (XP) [3] and Dynamic Systems Development Method (DSDM) [28]. There are also a number of software process models like ISO/IEC[1] Lifecycle Processes [15][16] and CMU/SEI[2] Capability Maturity Models [8][25]. Software development models typically specify the software engineering processes needed to develop and maintain a software product or system. They support the management and engineering in software development projects and provide the infrastructure to execute software development projects successfully, i.e. to deliver products with the expected features, functions and qualities on the expected time at the expected cost. Software process models typically specify all software engineering processes needed in the entire lifecycle of a software product or system.

[1] ISO = the International Organization for Standardization, IEC = the International Electrotechnical Commission

[2] CMU = the Carnegie Mellon University., SEI = the Software Engineering Institute

But, software development models and software process models typically do not ensure, from a business perspective, that the 'right' software products are developed. They do not ensure that business objective are met, that benefits both for the customer and the development organization are delivered, and that the organization spends its resources on the 'right' development projects. To achieve this, senior management involvement and clear business objectives are needed in the development projects, as complements to customer involvement and experienced project management [7][18][19][31].

1.1 Research Questions and Hypothesizes

One main question for this research is how to involve senior management in software product development projects. One possible way to find an answer is to look at traditional[3] product development. Successful traditional product development organizations typically use a stage-gate new product development process run by senior management and with the focus on business objectives [9][14]. A stage-gate new product development process breaks the product development process into a pre-determined set of stages. The entrance to each stage is a gate that serves as a go/kill checkpoint. In the 1997 PDMA Report [14] from the Product Development & Management Association (PDMA), some success factors for new product development are identified. The most important is "a formal New Product Development (NPD) Process supported by Top Management". Almost 70% of the most successful new product development organizations in the survey used a stage-gate new product development process.

Formal stage-gate new product development models are typically used in traditional product development companies for synchronizing product development activities undertaken by different functional areas within an organization and for gathering and preparing information needed for making business related decisions in product development projects [1][9][12].

[3] A traditional product in this thesis means any goods or services product for the consumer and/or business market [14].

But, stage-gate new product development processes do not support the actual development of software in general. As they do not address the problems due to the relative immaturity of software engineering compared to other engineering disciplines (e.g. the relative absence of standardized procedures and solutions), they do not work against a software development culture that can jeopardize the final success of project outcomes by failing to deliver products with expected features, functions, qualities and benefits on time and budget.

In spite of this we claim that the usage of a stage-gate new product development process is an effective means to support software product development. This claim we express through a hypothesis.

Hypothesis 1 : The usage of a stage-gate new product development process is an effective way to achieve effective executive support in software product development as it already is in traditional product development.

The usage of a stage-gate new product development process will increase the success of software product development. Unpromising projects will be cancelled in favor of more promising projects, and the organization will be ready for the project outcomes when the actual development is finished.

Another research question is if it is possible to use a stage-gate new product development process in combination with well-known software development models, or does the usage of a stage-gate new product development process constrain the software development process? We express the answer to this question through the second hypothesis.

Hypothesis 2 : It is possible to combine stage-gate new product development models with software development models without violating the principles of any of them, and at the same time achieve both effective senior management involvement and needed degree of flexibility in the software development processes.

The new product development models and software development models should be separated,

and only connected through a well-defined interface. The main reason is that large companies typically want only one common new product development model (as senior management want to interact with all development projects in the same manner) but they need different software development models as different types of software products are developed.

1.2 Research Motivation

In the late 1990's and early 2000's ABB developed and deployed a stage-gate new product development model, described in the ABB Gate Model Business Decision Layer [1]. The model should be used in all product and technology development projects and the main purpose is to ensure that all products, and their responsible organizations, are well prepared for the product release when the product is released. The author of this thesis was actively involved in the development, training and the deployment of the ABB Gate Model and learned during that work some common problems when applying the model to software product development projects. The most characteristic are the following.

- **The development project uses the new product development model as a software development model** resulting in a document driven sequential (waterfall) development lifecycle without support for the actual software engineering work.

- **Senior Management uses the new product development model as a project monitoring model** resulting in control of document production instead of the evaluation of the business value of the project and its outcomes.

These problems indicate that there is a lack of knowledge in how senior management and software development projects should interact, maybe due to the relative immaturity of the software engineering discipline.

The goal of this research is to validate the two hypotheses and gain better knowledge of how senior management and software development projects can interact.

1.3 Research Methodology

This research focuses on the need for business focus through senior management involvement in software product development, combined with the need for freedom and flexibility in the choice and implementation of software development processes in the development projects.

For validation of the first hypothesis, experiences were collected from the ABB IIT[4] Deployment Initiative where the ABB Gate Model was introduced for steering a large number of different projects worldwide. The ABB IIT Deployment Initiative includes a repository with information stored from all projects included in the ABB IIT Deployment project portfolio. The information was synthesized and analyzed to provide data for conclusions about the results of using a stage-gate new product development process in software development projects.

In order to verify the second hypothesis, a method of combining formal stage-gate new product development models with any type of software development model (sequential, incremental or evolutionary) through pre-gate milestones was motivated, defined and exemplified. The method was formulated after a survey of selected stage-gate new product development models, software development models and software process models described in papers, articles, reports and books referenced in this thesis. All models were analyzed regarding their support for senior management involvement and customer involvement, two of the most important success factors for software development projects [19].

1.4 Contribution

Through this research, the possibility of using a stage-gate new product development process in software product development to effectively achieve senior management involvement is motivated and evaluated. The possibility to combine stage-gate new product development models with

[4] IIT = Industrial Information Technology

software development models is demonstrated. A repeatable method for combining the two model types by means of pre-gate milestones is also defined. This method and several examples of model combinations are thoroughly described and published in the following paper and article:

Combining Models for Business Decisions and Software Development

This paper describes how Cooper's Stage-Gate™ model and ABB Gate Model Business Decision Layer can be combined with the Unified Process software development model.
Published in *Proceedings of Euromicro Conference, September 2002*

Integrating Business and Software Development Models

This article describes how the ABB Gate Model Business Decision Layer can be combined with the Unified Process, Extreme Programming and Microsoft's Sync-and-Stabilize software development models
Published in *IEEE Software November/December 2002*

Experiences from the usage of a stage-gate new product development process in software product development were drawn from a major software platform deployment program, the ABB IIT Deployment Initiative.

The experiences indicates that development projects within the program, through the requirements on information from the stage-gate new product development process, were motivated to develop their business case early and thoroughly, and that unpromising projects typically were canceled before the actual software development had started and any major investments were done. But, the experiences also indicate that unsuccessful project execution (exceeding schedule and budget) typically is not a strong enough reason to cancel projects. These experiences are presented and published in the following paper:

Three Aspects of Successful Software Development Projects, "When are projects canceled, and why?"

This paper describes the experiences gained from using the ABB Gate Model Business

Decision Layer in the ABB IIT Deployment Initiative

Published in *Proceeding of Euromicro Conference September 2003*

The papers and the article are presented below with a short summary of their respective contribution and a presentation of the specific contribution of the author of this thesis. The papers and the article are reprinted in Appendix A- C.

1.4.1 Combining Models for Business Decisions and Software Development

"*Today there is a number of established software development lifecycle models (SDLMs) supporting software development. Correct implementation of these models helps develop software products the right way, but this does not ensure that the right products are developed. Successful product development companies often use business decision models (BDMs) to facilitate the selection of products and projects for investment, but these models do not necessarily facilitate actual development of the software. One of the current challenges in the software community is to combine BDMs and SDLMs, including mapping of business decision gates and major lifecycle milestones. This is needed to achieve synergies between the two model types and to support the development the right products the right way, as well as to gain control over company investments. This paper analyzes two BDMs, proposes mappings to an established SDLM, and describes experiences of using them in a large, multinational engineering company.*"

The author of this thesis contributes to this paper with the definition of the pre-gate milestones and the model combination examples.

1.4.2 Integrating Business and Software Development Models

"*By mapping business decision gates to major software development milestones, organizations can relate technical life-cycle models to business decision models. The authors mapped Unified Process, Synch-and-Stabilize, and Extreme Programming life-cycle examples to the ABB Gate Model for product development projects.*"

The author of this thesis contributes to this article with the definition of the pre-gate milestones and the model combination examples.

1.4.3 When are projects canceled, and why?

"Successful project execution, successful technical solutions or a promising business case, are they equally important selection criteria in a product development process? We have used experiences gained from a large multinational industrial company that is currently deploying a software product line strategy to try to answer that question. The product line's core assets include, among other things, a new software platform that is introduced to the company's software development organizations by means of a portfolio of targeted pilot projects. A business decision-making process is used to select and prioritize projects within the portfolio. This paper report findings from an analysis of a large number of projects and will indicate that the three criteria are not equally important."

The author of this thesis contributes to this paper with the knowledge about the ABB IIT Deployment Initiative and the collection, analysis and synthesis of project data and the conclusions drawn.

1.5 Summary of Remaining Chapters

Chapter 2 provides a state of the art survey of development models. It also discusses three different project stakeholder groups (customers, senior management and developers) and how different project types (contract, commercial and internal development) affect the staffing of these stakeholder groups. It provides a brief summary of software development models, software process models and stage-gate new product development models. The different model's lifecycles, phases and major milestones or gates are analyzed, as well as their support for senior management involvement and customer involvement in software product development projects.

Chapter 3 provides a survey of related work. It describes Barry Boehm's anchor point milestones, project portfolio management, project models like Ericsson's PROPS and the ABB Gate Model and 'on-site customers' in agile methodologies.

Chapter 4, and the three appendices reprinting published papers, present the contribution of this research and describe the details on how to combine software development models and stage-gate

new product development models. It discusses the difference between information and documentation and the rational behind, and purpose of, pre-gate milestones. Chapter 4 also describes experiences of the usage of a stage-gate new product development process in the ABB IIT Deployment Initiative. The conclusions that can be drawn and lessons learned from the experiences are discussed and it also raises a question about how important development project execution success is to the overall success of software product development.

Chapter 5 summarizes the conclusions made from this research and Chapter 6 suggests future research to further investigate the importance of senior management involvement in software product development projects.

Three appendices provide reprints of published papers and articles included in this thesis.

2 STATE OF THE ART – DEVELOPMENT MODELS

The survey of the state of the art covers stage-gate new product development models, software process models and software development models and the information is collected from books, articles and papers. The survey analyses the different model's lifecycles, major milestones and/or gates, and support for user involvement and senior management involvement. The survey analysis will provide the background needed to suggest how stage-gate new product development and software development models could be combined to achieve active and effective senior management involvement in software product development.

2.1 Categories of New Products

There are several categories of new products and most companies have a mixed portfolio of these products [9], see Figure 3:

New to the world (innovations). Theses products are the first of their kind and will create a new market. Only 10% of all new products belong to this category.

New to the company. These products are not new on the market but form a new product line in a particular company or organization. About 20% of all new products belong to this category.

Additions to product line. These products are new to a particular company or organization, and maybe also new to the market, but fits into an already existing product line. This category includes about 26% of all new products.

Revisions of existing products. These new products are improved and/or enhanced replacements of existing products. About 26% of all new products belong to this category.

Repositionings. Already existing products are used in new applications or are retargeted to new markets. This category includes about 7% of all new products.

Cost reductions. These products are not new to the market but are cost reduced although similar replacements of existing products. About 11% of all new products belong to this category.

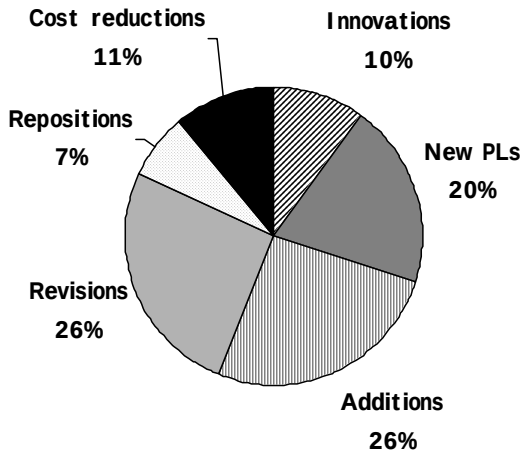

Figure 3: Categories of New Products [9]

2.2 Product and Development Lifecycles

"A process Lifecycle is defined as a sequence of Phases that achieve a specific goal." [24]

2.2.1 Product Lifecycle Phases

Every product (also software products) has a *product lifecycle* [16]. Although the lifecycles vary according to the nature, purpose, and use of the product there is an underlying, essential set of characteristic phases (or stages) that exists in the complete lifecycle of any product. The phases

represent the major lifecycle periods associated with a product and they relate to the state of the product. The phases provide organizations with a framework within which management has high-level visibility and control of the product. Figure 4 shows a frequently encountered example of lifecycle phases [16]: concept definition, product development, production, utilization and retirement.

During the *concept* phase stakeholders' needs are identified, concepts are explored and viable solutions are proposed. The *development* phase encompasses refinement of requirements, description of the solution and construction, verification and validation of the product. In the *production* phase the product is manufactured and certified for operation and in the *utilization* phase the product is operated (used) and supported and maintained. Finally, during the *retirement* phase the product is stored, archived or disposed.

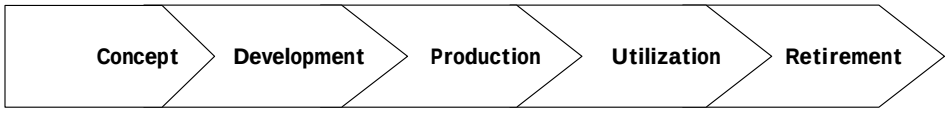

Figure 4: Generic Product Lifecycle [16]

Rajlich and Bennett [26] describe a slightly different view of the *software product lifecycle*. The concept phase is called initial development. The production phase is omitted since the 'production' of a software product typically is a minor activity in the end of the development phase. The utilization phase is actually a series of evolution and servicing cycles, and the retirement phase is divided into phase-out phase and closedown phase. See Figure 5.

During the *initial development* phase the first functioning version of the product is developed from scratch to satisfy initial requirements. During the *evolution* phase the capability and functionality of the product is extended, modified or deleted iteratively. At certain intervals a new version of the product is released. In the *servicing* phase only minor defects in the product are repaired and during the *phase-out* phase the product is still used but not serviced any more. Finally during the *close-down* phase the product is withdrawn from the market.

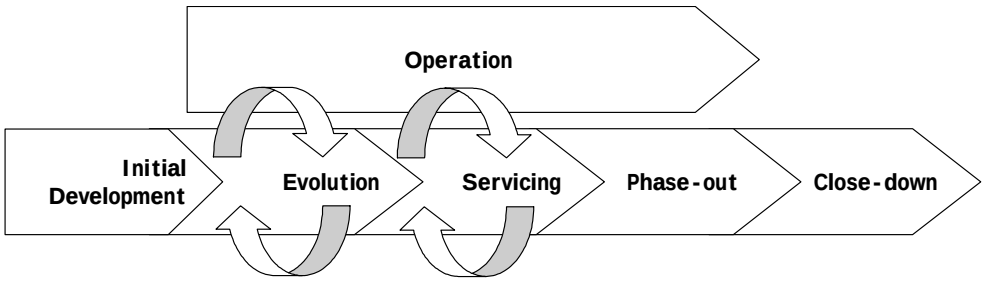

Figure 5: A simple software product lifecycle [26]

Extreme Programming [3] describes a software product lifecycle that consists of three different phases: initial development, maintenance and retirement. After the *initial development* phase the software system enters the *maintenance* phase where it is refined (evolved) in cycles and supported, until it finally enters the *retirement* stage when refinement is no longer needed or possible.

2.2.2 Development Lifecycle Phases

From a management perspective initial development and each evolution cycle are organized in a sequence of development phases concluded by major milestones as indicators of progress. Working through the phases produces a software product version and is called a *software development lifecycle* [20]. Typically an existing software product will evolve into its next version by repeating the same sequence of phases, although maybe with different emphasis.

Typically, a software development lifecycle consists of distinct phases, e.g. according to Rational Unified Process (RUP): inception, elaboration, construction and transition. See Figure 6. During the *inception* phase the vision, the scope and the business case for a product version are specified. During the *elaboration* phase development activities and required resources are planned, features are specified and the software architecture is designed. The *construction* phase encompasses the building of the product and the evolvement of the vision, the architecture and the plans until the product is ready for transfer to the users. During the *transition* phase the product is deployed to the users, trained, supported and maintained.

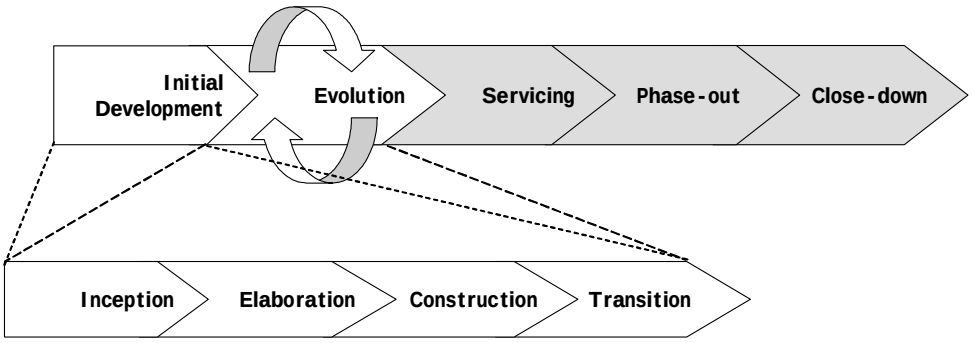

Figure 6: Software development lifecycle [20]

Extreme Programming [3] describes a similar software development lifecycle with four phases: exploration, planning, iterations-to-release and productionizing: During the *exploration* phase requirements are elicited and technologies architectures are explored. During the *planning* phase the release plan is developed. During the *iterations-to-release* phase the product built and during the *productionizing* phase the product is certified and tuned.

2.2.3 Development Lifecycle Types

Software development models implement development activities differently to satisfy different business and risk mitigation strategies. Performing activities concurrently or in different orders can lead to development lifecycle types with distinctly different characteristics. Sequential, incremental and evolutionary lifecycle types; or hybrids of these, are the most common.

In *sequential* lifecycle types the main development activities are performed in sequence. Milestones are used as assessment and transition points. Requirements are specified at the beginning and the software product is validated and released at the end of the development lifecycle. The Waterfall model [27] is a typical representative of the sequential lifecycle type.

In *incremental* lifecycle types, e.g. as in the Cleanroom model [21], the software product is built, validated by the customer and released piece by piece at several occasions during the development

lifecycle. As in sequential lifecycle types, requirements are specified at the beginning of the development lifecycle but then the work is organized into a sequence of verifiable and executable product increments, where each increment provides additional functionality. The content of each increment is defined in such a way that they accumulate into the complete final product.

In *evolutionary* lifecycle types, e.g. as in the original Spiral model [4], requirements (and estimates) are continuously refined during the development lifecycle. The software product grows via experimentation and elaboration through a series of prototypes.

Most contemporary software development models have hybrid lifecycle types, e.g. Microsoft Solution Framework (MSF) [23], Rational Unified Process (RUP) [17] and Extreme Programming (XP) [3], which combine the basic types. These lifecycles are sequential in the macro (phases and major milestones) and micro (analysis, design, implementation and verification within each iteration) perspective. At the same time incremental (internal or external) versioned releases for assessment, validation and usage are advocated. Evolutionary refinement of requirements and estimates are also recommended.

The selection and/or development of a software development model by an organization depend on several factors, including the business context, the nature and complexity of the software product, the stability of requirements, the technology opportunities, the need for different capabilities at different times, and the availability of funding and resources. [16].

A sequential lifecycle is useful when the requirements on the software product are clear, the technology known and the staff experienced e.g. in evolutions of mature products. A sequential lifecycle is easy to plan and manage, but missing or misunderstood requirements are typically discovered very late.

An incremental lifecycle is useful if early delivery of subsets of the final product is required. An advantage of an incremental lifecycle is that the customer can validate parts of the final product,

and mistakes can be corrected, early. A disadvantage of an incremental lifecycle, compared to a sequential lifecycle, is that more effort is needed for project management and product verification and validation.

An evolutionary lifecycle is useful if the requirements on the software product are vague or the technology is unknown or the staff is inexperienced, i.e. if the projects risks are high e.g. in the initial development of a new product. An advantage of an evolutionary lifecycle is that the final product evolves through a series of prototypes that can be thrown away if the solution is not satisfactory. A disadvantage of an evolutionary lifecycle is that it is difficult to plan and manage.

A hybrid lifecycle combines the advantages of the basic types and is useful in many cases. It is not uncommon in software development that planned time-to-market has to be kept and product parts have to be released during the development although requirements etc. will evolve (change) during the development.

2.3 Software Development Projects

The development of a new product version (i.e. a development cycle) is typically performed as a project. A project delivers one or more products to a customer or end user, and it has a definite beginning and a definite end and operates according to a plan [8]. Such a plan specifies the product(s) to be delivered or implemented, the resources and funds to be used, the work to be done, and a schedule for doing the work. A project can be composed of (sub)projects, and in some cases such a main project is called a program.

Development projects can be of different types and involve different stakeholders depending on the type of product developed.

2.3.1 Project Stakeholders

A project stakeholder is a group or an individual that is affected by, or accountable for the outcome of a project [8]. Stakeholders may include project management, developers, suppliers, customers (acquirers), end users, support and maintenance, senior management etc. A "relevant stakeholder" is a stakeholder that is identified for involvement in specified activities and is included in an appropriate project plan [8] i.e. they play a role in the project.

The view of the relevant stakeholders differs from model to model, and that may be a source of confusion. The Spiral model [5] treats all relevant stakeholders as one group, while the Waterfall model [27] and XP [3] have divided the relevant stakeholders into two groups, the 'development team' and the 'customers'. ISO/IEC 12207 [15] and ISO/IEC 15288 [16] also have two groups called 'supplier' and 'acquirer'. In these cases 'customers' or 'acquirer' typically means any relevant stakeholder outside the development team. Other models like SW-CMM [25], CMMI [8], RUP [17] and DSDM [28] defines several project roles, e.g. RUP defines five groups including 32 different roles that perform activities in a project.

To be able to analyze different models support for 'senior management involvement' and 'customer involvement' (i.e. the two most important success factors in software product development [19][31]) this thesis will divide the relevant stakeholders involved in software product development projects into three main groups based on their different concerns; Customer, Senior Management and Development Team. See Table 1.

The *customers* are the external stakeholders, either individuals or projects or organizations, who are responsible for accepting the product, who will ultimately pay for the project and its outcome and who are the ultimate recipients of the developed product and its artifacts [8][11][17]. The customers are external to the development project, but are not necessarily external to the development organization. Synonyms for customer are purchasers or acquirers.

Senior management represents the internal stakeholders whose primary focus is the long-term vitality of the organization, rather than short-term project and contractual concerns [8]. Senior management has authority to direct the allocation or reallocation of resources. Senior management includes the executive sponsor [30], or owner, of the project and other relevant managers of the organization such as marketing and sales, development, production, support and maintenance and training etc.

The *development team* represents the internal stakeholders responsible for project management and the development of the product.

Table 1: Project stakeholder concerns

Stakeholder group	Concerns
Customers	- Consistency with functional requirements and usage scenarios. - Quality attributes like performance, reliability, interoperability and others. - Acceptance tests. - Etc.
Senior Management	- Business case, feasibility and risks - Schedule, budget and staffing. - Architecture and product portfolio compatibility. - Consistency with policies and processes and practices. - Etc.
Development Team	- Requirements, usage scenarios and specifications. - Acceptance criteria. - Frameworks and components. - Architecture, design and technologies - Policies, processes and practices. - Etc.

2.3.2 Project Types

Three generic types of software product development projects can be identified; Contract, Commercial and Internal development projects [3][11][17]. The purpose of this section is not to

describe all possible project types, only to put some light on the different stakeholder groups; customer, senior management and development team.

Contract development is the case when a development team produces a software product on a given end customer specification, and for that end customer only [20]. The development team and senior management are in the same development organization while the customer is outside, although typically with close contact with the development organization. See Figure 7. A synonym for contract development is customer order development [11]. In some cases contract development seen from the viewpoint of the customer is called outsourcing [3].

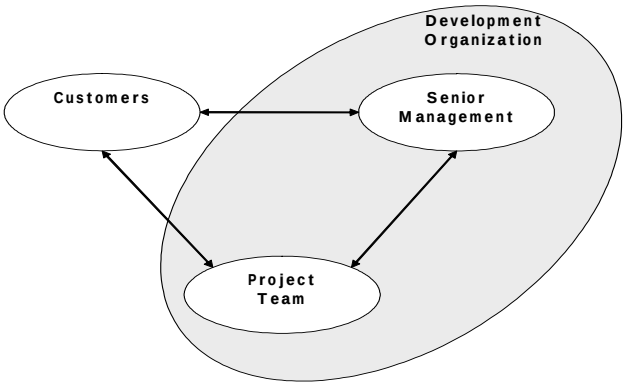

Figure 7: Contract development

Commercial development is the case when the development team produces a software product to be put on the market [20]. The customer is typically not available for direct contact with the development organization, but the customer is instead represented by someone inside the development organization, a product manager or account manager or similar. See Figure 8. The role of the customer representative is in this context to elicit market requirements and to be the receiver of the resulting product(s). (Of course a product/account manager has many other responsibilities too.)

Synonyms for commercial development are e.g. speculative development [20], shrink-wrap development [3], product-provisioning development [11] and mass-market development [23].

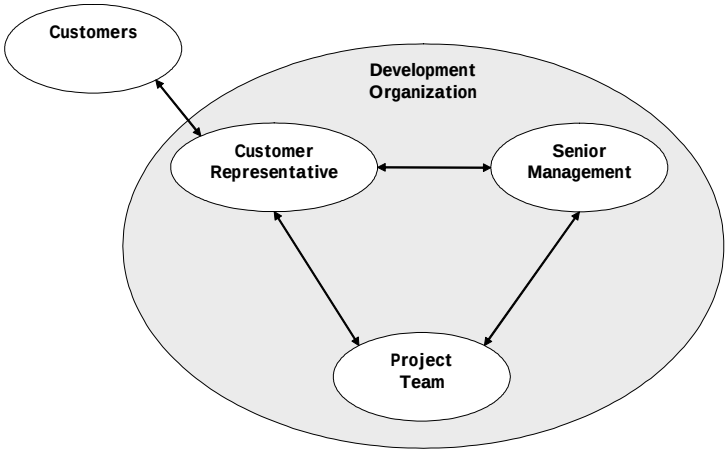

Figure 8: Commercial Development

Internal development is the case when the customers, senior management and the development team are in the same organization [11][20]. See Figure 9. In some cases the customer and senior management are actually the same individuals. A synonym for internal development is in-house development [3].

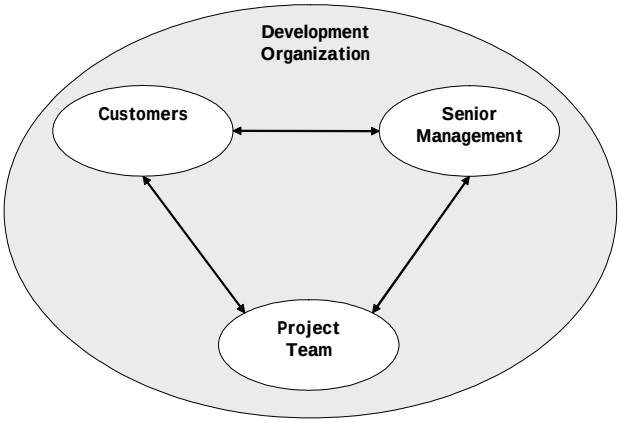

Figure 9: Internal development

In reality organizations can perform projects of any variant or combination of these generic types. It is for example not uncommon that contract projects are organized in the same way as

commercial projects are described here i.e. the development team communicates with the customer through a customer representative. It is also not uncommon that both commercial product components and customer specific product components are developed in one project. In product line organizations the customers of internal or commercial development projects may be contract development projects.

2.4 Software Development Models

Most software development models describe the work performed in the actual development of software. Individuals or small groups or tool vendors with long experience of software development are typically those that define these models based on what they think are good software development practices.

Software development models typically include a development lifecycle, including phases and milestones, describing in which order development processes and activities should be executed and when. Software development models typically also include description of different project stakeholders and their involvement in different project activities.

Many different software development models have been defined and described over the last 30 – 40 years, all with the purpose to support successful development project execution. In this section a few software development models are briefly described and analyzed regarding their suggested development lifecycle and their support for customer involvement and senior management involvement.

2.4.1 Brief History

1970 Royce described the first model, later named the Waterfall model, with the purpose to manage software development projects, "… arriving at an operational state, on-time, and within cost." [27]. The Waterfall model was highly influential and is still used, more or less, in many organizations.

In the 1980's new software development models were defined, addressing some of the difficulties with the Waterfall model e.g. early completion of documents and big-bang delivery. Two examples are the Cleanroom model [21] and the Spiral model [4], both advocating incremental and iterative development. The Cleanroom model is used by for example US Department of Defense (DoD) and US National Aeronautics and Space Administration (NASA). The Spiral model is also used by US DoD, but also by for example. Xerox and AT&T.

In the 1990's, following the object-orientation paradigm shift, the next evolution of software development models took place. The most known model of this generation is probably the Rational Unified Process (RUP) [17][20] although there are many more. RUP is commonly used in the development of financial applications for example banks and insurance.

Now in the 2000's, agile software development models [2] e.g. Extreme Programming (XP) [3] and Dynamic Systems Development Method (DSDM) [28] are the latest approaches to address the challenges with successful software development. Agile methodologies are widely used and for example the DSDM consortium has almost 200 members including banks, consultancy firms and universities.

2.4.2 Overview of the Models

The Waterfall model is the basis of all software development models [22] and it is still frequently used. In the original Waterfall model [27], a project progresses through an orderly sequence of software development activities from system requirements specification to operations. In this model activities and phases are the same. In the end of each activity the project holds a formal review to determine whether or not it is ready to advance to the next activity. The Waterfall model is document driven, which means that the main work products that are carried from activity to activity are documents. The original Waterfall model was intended for the development of large software systems. It recognizes the need for iterative work and it includes activities (phases) for e.g. architectural design and prototyping, features that are often forgotten when the model is criticized.

Cleanroom is a model intended for the development of high quality software with certified reliability [21]. The Cleanroom name is borrowed from hardware clean rooms and Cleanroom focuses on defect prevention instead of defect correction, and validation (certification) of reliability in intended usage environment. Cleanroom is incremental, the product is released and validated piece by piece, and the model recommends small and specialized teams for specification, development and validation.

The Spiral model is a risk-oriented software development model that breaks up a software development project into subprojects [4][6][22]. Each subproject addresses one or more risks until all major risks have been addressed. After all risks have all been addressed, the Spiral model terminates in the same way as the Waterfall model. The Spiral model is useful for high risk projects.

Rational Unified Process (RUP) is a software development model describing nine software engineering workflows: business modeling, requirements, analysis and design, implementation, test, deployment, configuration and change management, project management and environment [17][20]. All these workflows are executed more or less in parallel during a project. RUP is founded on six good practices for software development: develop iteratively, manage requirements, use component based architectures, visually model software, continuously verify quality and control changes. RUP is incremental and iterative, very extensive and described in its every detail. Each core workflow is decomposed in detailed workflows with corresponding activities, roles and artifacts. RUP is aimed for object oriented software development and is tightly coupled with the Unified Modeling Language (UML).

Extreme Programming (XP) [3] has its roots in the Smalltalk community. It is a refinement of practices from a numerous of projects during the early 90's. One of the cornerstones is its strong emphasis on testing. XP puts testing as the foundation of development, with every developer writing tests before they write the code. XP is a system of twelve good practices (planning game, small releases, metaphor, simple design, continuous test, re-factor, pair programming, collective ownership, continuous integration, 40-hour week, on-site customer and coding standards) for

development of software when the risk is high. XP requires small development teams (that follows the software product its entire product lifecycle) and automated testing to be effective and efficient.

The Dynamic Systems Development Method (DSDM) [28] started in Britain in 1994 as a consortium of companies who wanted to use Rapid Application Development (RAD) and iterative development. Starting with 17 founders the consortium now has over a thousand members and has grown outside Britain. Being developed by a consortium, it has a full time organization supporting it with manuals, training courses, accreditation programs, etc. It also carries a price label. DSDM has nine underlying principles (good practices), that are used as the foundation for the model: active user involvement, empowered teams, frequent delivery, fitness for business purpose, iterative and incremental development, reversible changes, high-level requirements, integrated testing and collaboration and cooperation.

2.4.3 Phases and Major Milestones

A *phase* is a work definition such that its precondition defines the phase entry criteria and its goal (often called a "milestone") defines the phase exit criteria. Phases are defined with the additional constraint of sequentiality; that is, their enactments are executed with a series of milestone dates spread over time and often assume minimal (or no) overlap of their activities in time [24].

A *milestone* is a significant event in a project, usually the completion of certain deliverables. Milestones can be event based or calendar based. If calendar based milestone dates have been agreed upon, they are often very difficult to change [8]. Milestones are formal checkpoints to measure progress and major milestones mark the transition from one phase to another. Minor milestones split large work efforts into workable and manageable pieces. [17][23].

The original Waterfall model [27] consists of eight phases; system requirements, software requirements, preliminary program design, analysis, program design, coding, testing and operations. The first (system requirements) and the last (operations) are outside the actual software development leaving six phases in the software development lifecycle. There are no milestones

defined in the model, but when implemented, the completion of each phase is typically marked with a major milestone [5].

The Cleanroom model [21] does not specify any specific phases or milestones, but three phases can be identified from the model: analysis & specification, increment planning and incremental builds. During analysis & specification phase requirements are analyzed and functions and usage are specified. During the incremental planning phase customer requirements and resources are allocated to a series of software increments and the development schedule is defined. During the incremental build phase the software is designed, built, verified and validated increment by increment.

In the original Spiral model [4] a phase is a cycle in the spiral and each cycle is completed by a review involving relevant stakeholders. The cycles (phases) are not named and the number of cycles is not defined. The end-of-cycle reviews should cover all work products developed during the previous cycle as well as the plans and resource requirements for next cycle. The goal with the reviews is to achieve mutual commitment between all relevant stakeholders to the next cycle. One major difficulty with the original spiral model was its lack of major milestones to serve as commitment and progress checkpoints for the entire development lifecycle [5]. This difficulty was addressed by the development of a set of anchor point milestones: Life Cycle Objectives (LCO), Life Cycle Architecture (LCA) and Initial Operational Capability (IOC). These anchor points, or major milestones can be described as stakeholder commitment points in the software development lifecycle: the LCO milestone is the stakeholders' commitment to support architecting; the LCA milestone is the stakeholders' commitment to support development and the IOC milestone is the stakeholders' commitment to support the release and operations.

Rational Unified Process (RUP) [17] defines a development lifecycle divided into four sequential phases; inception, elaboration, construction and transition. During the *inception* phase the end product vision and business case should be specified and the project scope is defined. During the *elaboration* phase planning of the necessary activities and required resources should be done as well as specification of features and architectural design. During the *construction* phase the product should

be built evolving vision, architecture and plans until the product is ready for delivery. During the *transition* phase the product should be finalized and deployed. Each phase is concluded with a major milestone; Lifecycle Objective (LCO), Lifecycle Architecture (LCA), Initial Operational Capability (IOC) and Product Release (PR). The first three major milestones are adopted from Boehm's anchor point milestones.

The XP software development lifecycle consists of four different phases: exploration, planning, iterations-to-release and productionizing [3]. During the *exploration* phase the initial set of user stories is elicited, the technologies are explored and different architecture alternatives are evaluated. During the *planning* phase user stories are prioritized and the release plan is developed. During the *iterations-to-release* phase the product built iteratively and incrementally until the product is ready to go into production. During the *productionizing* phase the product is certified and tuned to go into production. XP does not define any (major) milestones, but the completion of each phase can be marked with one.

DSDM is a software development model with five development phases: feasibility study, business study, functional modeling, design and build and implementation [28]. The *feasibility study* phase considers the feasibility of the project in business and technical terms as well as the suitability of the project for an evolutionary development approach. The *business study* phase defines the high level functionality and the major business entities affected. The *functional modeling* phase is used to construct and demonstrate the required functionality using a working prototype. The *design and build* phase is used to refine the functional prototype, particularly to meet non-functional requirements. The *implementation* phase includes the handover to users followed by a review of the project's success. DSDM does not define any (major) milestones either but as in XP the completion of each phase can be marked with one.

2.4.4 Support for Customer Involvement

The Waterfall model [27] does not explicitly define the customer role but uses the term as opposite to the contractor role and indicates three occasions in the development cycle where the 'customer'

should be involved in a formal way for insight, judgment and commitment; the preliminary software review (PSR), the critical software review (CSR) and the final software acceptance review (FSAR). The first occasion, the preliminary software review, is after the preliminary design phase (i.e. after architectural design and pilot solution development). The second occasion, the critical software review, is actually several design review occasions during the detailed design phase. The third occasion, the final software acceptance review, is the acceptance test before final delivery.

In the Cleanroom model [21] the 'customer' is a member of the project team. The term 'customer' may here mean external or internal sponsor, end user, or any other party that is appropriate for defining requirements and evaluating the evolving system. The 'customer' is involved for requirements definition and for assessment and validation of the software product after each increment.

In the Spiral model [5] the 'user' role is concerned with customer issues like consistency with requirements, usage scenarios, and run-time quality attributes, and is involved in the post-cycle (end-of-phase) reviews and at the anchor point milestones reviews together with all other relevant stakeholders.

In Rational Unified Process [17] the 'customer' is involved in several review occasions: iteration plan reviews, iteration evaluation criteria reviews, iteration acceptance reviews, lifecycle milestone reviews and project acceptance review.

In the agile software development models like Extreme Programming (XP) [3] and Dynamic Systems Development Method (DSDM) [28] it is assumed that the 'customer' is a member of the project team and takes active part in the development together with the development team. They are continuously involved in requirements and test specifications, project planning and in performing acceptance tests. In contract development projects or internal development projects the 'customer' can be selected from the customer organization, while in commercial development a customer representative typically comes from the development organization, e.g. a product manager. In XP the 'customer' represents all relevant stakeholders outside the development team

while in DSDM the ‘customer’ is represented by two specific roles, the Ambassador User and Advisory User.

2.4.5 Support for Senior Management Involvement

The Waterfall model does not describe any specific support for ‘senior management involvement’, and neither does the Cleanroom model. But the usage of the term ‘customer’ in both the Waterfall model and Cleanroom model can be interpreted as including also senior management although senior management (business) concerns are typically not directly addressed at the review occasions.

The Spiral model uses the term ‘customer’ to describe the role that is concerned with typical business issues in the project like schedule, budget, feasibility, risks, progress and product portfolio compatibility. The ‘customer’ is involved in the post-cycle (end-of-phase) reviews and at the anchor point milestones reviews together with all other relevant stakeholders.

In Rational Unified Process (RUP) senior management is involved only at a couple of upfront review occasions: project approval review and project planning review. At these occasions issues like: customer benefits, internal business benefits, technical benefits, return on investment (ROI) and the availability of resources are addressed.

In addition the RUP model also defines a project management role defined called Project Reviewer. The Project Reviewer is responsible for evaluating project planning and assessment artifacts at major review points in the project's lifecycle. These review events mark points at which the project may be cancelled if planning is inadequate or if progress is unacceptably poor. The Project Reviewer role requires business, technical, and software project management experience, and decision-making ability at the senior management level. The Project Reviewer is a project team member and one of the tasks for that role is to summon senior management, quality assurance or customer representatives for major review meetings.

In the Extreme Programming model 'senior management involvement' is not addressed specifically, senior management is treated as any 'customer'. A senior management representative (e.g. the project sponsor) has to be available for the development team 'on-site'.

In the Dynamic Systems Development Method model senior management is represented by the executive sponsor role. The executive sponsor commits resources (funds and staff), provides business knowledge, monitors the business case and is the ultimate decision-maker in the business area. But, the executive sponsor is only involved when the project runs into problems that cannot be resolved by the project itself.

2.5 Software Process Models

Software process models describe what software or system engineering processes are needed in a mature and capable development organization. These models are typically defined by some independent organization (e.g. ISO/IEC[5]) or institute (e.g. CMU/SEI[6]) and can be regarded as process requirements specifications, or roadmaps, that can guide organizations in how they can improve their capability to develop software. Software process models only describe what processes and activities are needed, not how or when they should be executed. Software process models can typically be used in capability assessments or evaluations of organizations. It is not uncommon that an organization, especially in the US, has to prove its conformance to a specific software process model to get a contract with a customer.

[5] ISO = the International Organization for Standardization, IEC = the International Electrotechnical Commission

[6] CMU = the Carnegie Mellon University., SEI = the Software Engineering Institute

2.5.1 Overview of the Models

ISO/IEC 12207 Software Lifecycle Processes [15] is an international standard that groups seventeen software engineering processes in three different categories: primary processes, supporting processes and organization processes. Primary processes are: acquisition, supply, development, operation and maintenance. Organization processes are: management, infrastructure, training and improvement. Supporting processes are: documentation, configuration management, quality assurance, verification, validation, review, audit, and problem resolution. Each software engineering process is divided into a set of activities and each activity is further divided into a set of tasks. Each task is formulated, as a process requirement including one of the words 'shall', 'will' or 'may'. This ISO standard describes the software engineering processes, but not any details on how to implement or execute the different activities and tasks included. It is typically used as a requirement specification for software processes definition and improvement activities as it describes what processes are needed in software engineering and how these processes should be structured and interconnected.

ISO/IEC 15288 System Lifecycle Processes [16] is a successor of ISO/IEC 12207 and an international standard describing twenty-five system-engineering processes grouped together in four different process categories: Enterprise Processes, Agreement Processes, Project Processes and Technical Processes. See Figure 10.

Each process is described with a purpose, expected outcomes and performed activities. The structure of this standard is most of all like a checklist where performed activities and expected outcomes can be ticked off. ISO/IEC 15288 applies to the entire lifecycle of a system or product (not only software and not only development), during six product lifecycle stages: conception, development, production, utilization, support and retirement.

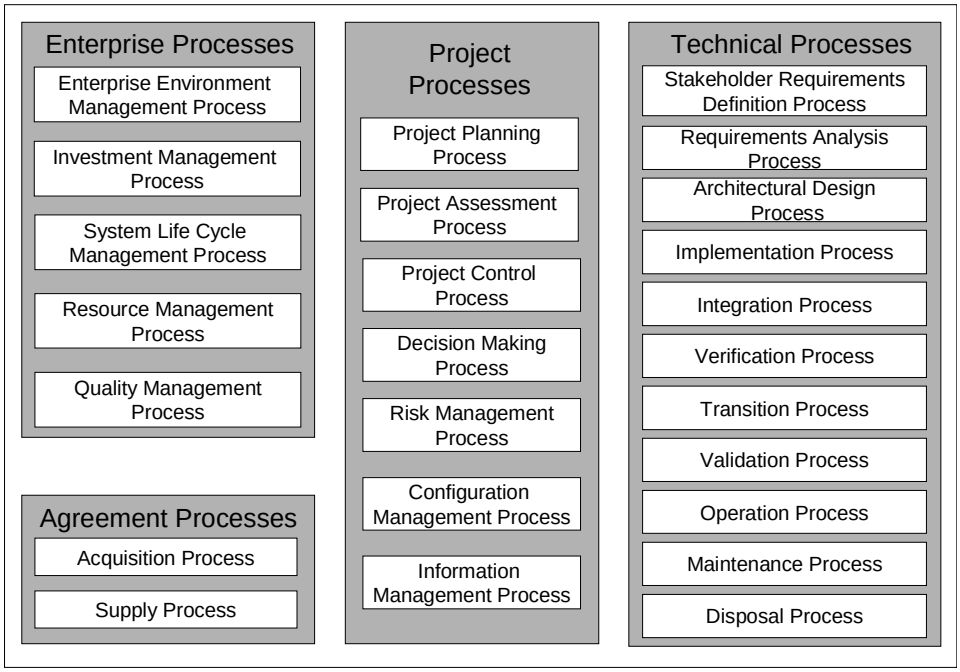

Figure 10: ISO/IEC 15288 processes and process categories [16]

The CMU/SEI Software Capability Maturity Model (SW-CMM) [25] defines eighteen software engineering processes and groups them into five different maturity levels: Initial, Repeatable, Defined, Managed and Optimizing. The SW-CMM grouping indicates which processes provide the foundation for other processes and can be used for assessing the maturity of software developing organizations as well as a guide for software process improvement. Each software engineering process is described with one or more goals and a set of common features grouped together in five different feature categories; commitment, ability, activities, measurements and verification. Each common feature is described by a set of key (good) practices.

The CMU/SEI Capability Maturity Model Integration[SM] (CMMI) [8] is a successor of SW-CMM and describes twenty-four system engineering processes. There are two representations of CMMI; one is the staged representation in maturity levels as in SW-CMM, and the other is a continuous representation in four different process categories similar to ISO/IEC 15288. In the continuous

representation each process in graded in six different capability grades and if the two representations of CMMI are combined, a minimum set of good software engineering practices can be identified. The structure of CMMI is similar to that of SW-CMM where the system engineering processes are described with goals and common features.

2.5.2 Support for Customer Involvement

In the ISO/IEC 12207 model and in the ISO/IEC 15288 model the customer is called the acquirer. Both models have a defined process, the acquisition process, described with activities, tasks and expected outcomes. The acquisition process describes how the acquirer should be involved in the development of a software product or system.

Neither the SW-CMM model nor the CMMI model defines any specific 'customer involvement' process but the customer is identified in both models as a relevant stakeholder and the customer should be involved in (formal) reviews of relevant work products and also in quality (QA) reviews of the development project. These models define the customer as the individual or organization that is responsible for accepting the product and authorizing payment to the developing organization [8][25].

2.5.3 Support for Senior Management Involvement

In ISO/IEC 12207 'senior management involvement' is not addressed.

ISO/IEC 15288 indicates the appearance of business decision gates between each product lifecycle stage and there is also a set of Enterprise Processes (enterprise environment management, investment management, system lifecycle processes management, resource management and quality management) with the purpose "…to support projects and ensure the satisfaction of organizational objectives …" Specifically the purpose of the investment management process is to "initiate and

sustain sufficient and suitable projects in order to meet the objectives of the organization" and the purpose of the resource management process is to "provide resources to projects".

SW-CMM and CMMI defines senior management as "a management role at a high enough level in an organization that the primary focus is the long-term vitality of the organization, rather than short-term project and contractual concerns and pressures." Typically a senior manager would have responsibility for multiple projects.

SW-CMM recognizes the senior management should review development projects resources, schedules, risks and technical solutions on a regular basis. SW-CMM also recognizes that reviews of the software project should be performed periodically to determine the actions needed to bring the software project's performance and results in line with the current and future needs of the business, but the role responsible for these reviews is not defined.

In a similar way CMMI specifies that senior management is responsible for establishing and communicating guiding principles, direction, and expectations for the organization and senior management should review activities, status, and results periodically and event driven. Relevant stakeholders (including senior management) should review project commitments, plan, status and risks at planned project milestones. The business view is an integrated part of CMMI and equally important as customer needs. Business objectives should influence both how different engineering processes are implemented and how development projects are executed.

2.6 New Product Development Models

New product development models differ from software process models and software development models as they address product development in general and from a business perspective. Not only software or system development issues are covered, but also things as market analysis, competitor and intellectual property evaluations and organizational preparations for manufacturing, maintenance and support etc.

Another difference is that new product development models typically include gates (business decision points) where it is decided if a project is allowed to continue with or without redirections, or where the project is canceled. Business decisions should be based on the results from market and competitor evaluation, technology feasibility, business strategy, intellectual property rights, product quality and status and available resources within the organization. Many organizations today use a well-defined new product development process when developing new products (not only software) or new product versions. Such a new product development process generally consists of a number of different development stages separated by business decision gates. The number of stages and gates may differ from organization to organization.

Typical stage-gate new product development models are; Cooper's Stage-Gate™ Model [9], Ericsson's PROPS[7] Project Steering Model [12] and the ABB Gate Model Business Decision Layer [1].

2.6.1 Overview of the Models

Cooper's Stage-Gate™ model [9] decomposes a product development lifecycle into stages and gates. Each stage consists of a set of parallel activities and processes performed by different actors within an organization. Each activity at each stage is designed to gather all information needed as input to the upcoming business decision gate and to reduce risks associated with the creation or evolution of a product.

Ericsson's PROPS project steering model [12] is based on experiences gained from use within Ericsson companies all over the world. It describes the steering aspects of development project work with (toll)gates representing business decision points.

[7] The PROPS model is named after the project that developed it; the PROjektet för Projekt Styrning (PROject for Project Steering).

The ABB Gate Model business decision layer [1] defines eight gates where major business decisions are made. The model serves as a framework for the various activities, e.g. software development, hardware development, competitor monitoring, intellectual property management, training and marketing etc. included in product development.

2.6.2 Stages and Gates

In Cooper's Stage-Gate™ model the first stage of the model is the *discovery* stage. It begins with an idea for a new product or product version. During the *scooping* stage, the main objectives are to assess market and technology and identify the key product requirements. At the *business case* stage information needed to decide if it is feasible to develop the product is gathered. The *development* stage mainly deals with the development of the product according to the product and project definitions. At stage four, *testing and validation*, the product is finally verified and validated, and the final stage, the *launch* stage, includes activities for marketing and sales and for production or operation.

The gates between each product development stage are the decision points. The procedures at each gate are similar; the results from the activities performed in the stage preceding the gate, together with a decision criteria checklist are used as input to the business decision. The output from the gate is a go/no-go/hold/redo decision accompanied by relevant plans for the next stage.

Cooper's Stage-Gate™ model distinguishes five gates: Gate 1, the *idea screen* is the first occasion where resources are committed to the product development. Gate 2, the *second screen* decision point, is essentially a repeat of the previous gate although a bit more rigorous and based on the information gathered during the scooping stage. Gate 3, the *go to development* is the decision point and the last chance to stop the development before large investments are made. A go decision at this point is both a financial and resource commitment and an agreement on the product and project definition established during the build-business-case stage. Gate 4, the *go to testing* decision point is based on a post-development assessment to make sure that the product and project are still

attractive to the market and to the organization. Finally Gate 5, the *go to launch* decision point, is the last point at which the project can be stopped and the product canceled.

In PROPS the product development lifecycle is divided into four development stages: the pre-study, feasibility study, execution and conclusion. The *pre-study* stage is a preparatory stage before the project has been formally started. The purpose is to verify or not whether a business idea is technically and commercially feasible. The *feasibility study* is the stage during which the project is outlined. The purpose of this stage is to provide a solid foundation for successful project execution and completion. The *execution* stage is the stage during which the project is executed and the outcome is handed over to the customer and the receiving organization that will manage the project outcome in the future. During the *conclusion* stage experiences made in the project are documented and lessons learned transferred to the development organization before the project is formally closed.

PROPS defines six (toll)gates, TG0 – TG5. At each gate, the project sponsor makes a decision on how to continue the project. The purpose of TG0 is to ensure that the following pre-study is based on an idea for a business opportunity that is assessed to be aligned with the organization's business objectives. The purpose of TG1 is to ensure that the pre-study is aligned with the organization's business objectives and that the benefits for the customer are considered. The purpose of TG2 is to ensure that the following project execution is based on a business case aligned with the organization's business objectives, and that the benefits for the customer are considered. The purpose of TG3 is to minimize the technical and commercial risks before major investments have been made in the project. The purpose of TG4 is to ensure that the quality and scope of the project outcome is confirmed and the purpose of TG5 is to ensure that the project conclusion is based on the customer's acceptance of the project outcome.

The ABB Gate Model does not explicitly define any stages. In product development projects it is implicitly assumed that the project management model used, defines the needed development lifecycle phases. In this way ABB Gate Model is not used as an independent process but it is closely related to the development processes selected by the project.

The ABB Gate Model defines eight gates. Each gate is a decision point where those who are responsible for the outcome of the project evaluate the achieved results from a business point of view and determine whether to continue the project or not. A decision to continue may include alterations to the project such as changed scope or plan. Gate 0 (G0) Start Project, is an agreement to initiate the feasibility evaluation. The focus between G0 and G1 should be on analysis of the requirements. Gate 1 (G1) Start Planning, is an agreement on the scope of the project. The requirements agreed here will control the planning made between G1 and G2. Gate 2 (G2) Start Execution, marks the agreement on requirements, concept and project plans. The focus from G2 to G3 should be on specification of functions and architecture. Gate 3 (G3) Confirm Execution is a confirmation that target dates can be met and that the project executes according to the project description and plan. After G3, the focus is on implementation. Gate 4 (G4) Start Introduction, is an agreement that project outcomes can be released for acceptance testing. The focus from G4 to G5 should be on validation, on preparation for the market introduction and on production preparations. Gate 5 (G5) Release Product, is an agreement to hand-over of the project outcomes to the line organization. G5 also indicates that the project activities should be finished and focus in the period up to G6 is on finalizing any remaining issues. Gate 6 (G6) Close Project, is an agreement that the project can be brought to an end. Gate 7 (G7) Retrospective Investigation of Project is a follow-up of the project to check if the results are acceptable, and to feed experiences back to the organization.

2.6.3 Support for Customer Involvement

Cooper talks frequently in "Winning at New Products" [9] about the importance of customer without defining exactly who the customer is. He says "The definitions of unique and superior and benefit are constructed from the customer's perspective – so they must be based on an in-depth understanding of customer needs, wants, problems, likes, and dislikes." Cooper also says that "Seeking customer input and feedback is a vital and ongoing activity throughout development, both to ensure that the product is right and also to speed development toward a correctly defined target." He recommends constant and iterative validations tests under actual usage conditions of the product as it develops. In contract development projects and internal development projects this is typically not a big problem, but to solve this in commercial development projects Cooper

suggests involvement of potential users and dealers at demonstrations and try-out events, or setting up a "user's panel" – an ongoing group of potential customers that act as a team of advisors, or establish customer partnerships.

PROPS recognize that 'customer involvement' is important, but does not further describe how this will be achieved. PROPS only indicate that "customer representatives can appear in a number of different roles" [11].

The ABB Gate Model does neither define who the customer is nor how 'customer involvement' should be supported, but describes 'customer benefits' complementary to 'ABB (organizational) benefits'.

2.6.4 Support for Senior Management Involvement

Cooper defines a number of responsibilities for senior management. One is the commitment to make available necessary resources to product development, including funds, people and time. Another responsibility is the close involvement in product development projects including approval of plans and budgets and go/kill business decisions at the gates. Senior management should also be held accountable for product development results in general. This is true about development results in their own development projects in particular.

In PROPS the project sponsor (a senior manager) is the owner of the project steering process and there is a separate guideline describing the project steering process in detail, including responsibilities, activities and business decision criteria for each 'tollgate'.

In the ABB Gate Model senior management is represented by the Gate Owner (the project sponsor) and the Gate Meeting Participants (other relevant senior managers). Similar to PROPS project sponsor, the Gate Owner is the owner of the business decision layer (the project steering

process) and the model is described in a guideline and complemented with checklists describing the business decision process in detail, including project assessments and business decision criteria.

2.7 Conclusion

All software development models evaluated in this thesis support 'customer involvement' in the development projects. The Spiral model [4][5], Rational Unified Process [17], Cleanroom [21] and the Waterfall model [27] recommend frequent and formal reviews of project outcomes. Extreme Programming [3] and the Dynamic Systems Development Method [28] recommends an 'on-site customer' continuously working together with the development team. Although some development organizations and/or projects do implement their own development processes without support for active involvement from either the real customer or a customer representative, the models by themselves recognize the importance of and support 'customer involvement'.

The importance of 'senior management involvement' is not equally recognized in software development models. Extreme Programming [3], the Spiral model [4][5], Cleanroom [21] and the Waterfall model [27] regard senior management as any 'customer' leaving it up to the development organization and/or project to understand that senior management should provide and validate business and organizational requirements. Rational Unified Process [17] involves senior management only up front and the Dynamic Systems Development Method [28] involve senior management only when a project runs into serious problems.

The software process models ISO/IEC 12207 [15], ICO/IEC 15288 [16], CMU/SEI SW-CMM [25] and CMU/SEI CMMI [8] recognize the importance of 'customer involvement' and all except ISO/IEC 12207 recognizes the importance of 'senior management involvement'. These models recognize process needs for both aspects but they do not describe how or when the involvement should take place; only that customer involvement' and 'senior management involvement' should be implemented. It is up to the organizations or projects that implement the processes to decide on what 'periodically' or 'event driven' or 'on a regular basis' stands for.

The new product development models described in Cooper's Stage-Gate™ model [9], Ericsson's PROPS model [12] and the ABB Gate Model [1] specifies in detail processes for 'senior management involvement' in development projects. The need for 'customer involvement' is recognized but not supported.

3 RELATED WORK

The suggested method to combine new product development models with software development models was formulated by the author of this thesis after investigating several models for new product development and software development.

One contribution to the method comes from the identified need for so called “anchor point milestones” in the software development lifecycle [5][6][17] as originally described by Professor Barry Boehm 1996.

Another contribution comes from the Ericsson PROPS model [11] and the ABB Gate Model [1], both describing three-layered product development process architectures separating project steering processes from project management processes and product development processes.

A third contribution comes from the typical need for senior management to manage a portfolio of projects within their organization. To be able to perform this task senior management needs relevant information about all their projects, and one way for them to achieve this is to apply a new product development process as a project steering process for each project [10][11].

Last, but not the least, one contribution comes from contemporary agile software development models where the need for business focus in software development projects is addressed by requiring that the ‘business people’ (i.e. relevant stakeholders) work together with the development team on site and on a daily basis [2][3][28].

3.1 Anchor Point Milestones

After some years of usage of the original Spiral model [4] its lack of intermediate milestones to serve as commitment progress checkpoints was identified [5][6]. This was addressed by the

definition of a set of anchor point milestones; Life Cycle Objectives (LCO), Life Cycle Architecture (LCA), and Initial Operational Capability (IOC). These anchor point milestones are described as stakeholder commitment points in the software development lifecycle:

- Life Cycle Objectives is the stakeholders' commitment to support architecting
- Life Cycle Architecture is the stakeholders' commitment to support full development lifecycle
- Initial Operational Capability is the stakeholders' commitment to support operations.

The key question to answer at the two first anchor point milestones, Life Cycle Objectives and Life Cycle Architecture, is if the software product developed using the specified architecture and processes will support the operational concept, realize the prototyping results, satisfy the requirements, and finish within the budgets and schedules in the plan? The key question at the last anchor point milestone, Initial Operational Capability, is if the software, the receiving site and the users, and the support and maintenance organizations are prepared for the operation of the software product?

Rational Unified Process (RUP) has adopted, and slightly modified, the anchor point milestones as its major milestones [17]. The Life Cycle Objectives milestone is the stakeholders' commitment to support architecting and the Life Cycle Architecture milestone is the stakeholders' commitment to support full development lifecycle as in the original definition. The original Initial Operational Capability anchor point milestone is split into two major milestones, Initial Operational Capability and Product Release (PR). Rational Unified Process' Initial Operational Capability milestone is the stakeholders' commitment to support deployment and acceptance (beta) test and the Product Release milestone is the stakeholders' commitment to support the release of the product.

3.2 Ericsson PROPS and the ABB Gate Model

Both Ericsson PROPS [11] and the ABB Gate Model [1] actually consist of three models each. In PROPS they are called, the Project Steering Model, the Project Management Model and the Project Work Model. In the ABB Gate Model the corresponding models are called; the Business Decision Layer, the Project Management Layer and the Execution Layer. See Figure 11.

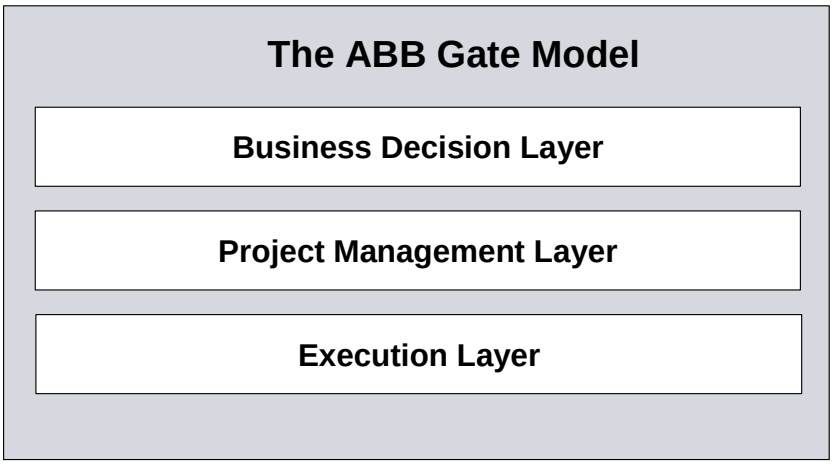

Figure 11: The ABB Gate Model different layers

The purpose of both PROPS *Project Steering Model* and ABB Gate Model *Business Decision Layer* is to ensure that long term strategic business perspectives are always present in the project, and that the project and its outcomes are aligned with the organization's business direction.

In PROPS the *Project Management Model* describes all project management activities and artifacts for all development phases in a project. The ABB Gate Model *Project Management Layer* only describes activities and artifacts that have to be added to, or modified in, an organizations existing project management process to provide the information required by the Business Decision Layer.

An interesting characteristic of both PROPS and the ABB Gate Model is that neither the *Project Work Model* nor the *Execution Layer* is defined. In order to obtain a complete project model in a

specific project, a work/execution model has to be selected and/or defined and linked to the other two models. This process architecture supports well the method to combine processes suggested in the next main section in this thesis.

It is difficult to find any published evaluations of the usage of Ericsson PROPS or the ABB Gate Model. In general the usage of a stag-gate new product development process for project steering improves the success rate for the developed products [14]. Experiences from the ABB IIT Deployment Initiative show that the usage of a stag-gate new product development process for project steering (the ABB Gate Model Business Decision Layer) prevents the execution of projects with an unpromising business case [34]. It is equally difficult to find any published reports on how different software development models have been linked with the respective model. One typical opinion experienced by the author of this thesis is however that these models enforce sequential (waterfall) software development models.

3.3 Project Portfolio Management

Project portfolio management is a business process run by senior management, where an organization's list of projects is constantly revised and updated [10][11]. In the project portfolio management process new and ongoing projects are evaluated, prioritized and directed according to the business strategy and goals, and project resources are allocated and reallocated. A project portfolio management process could encompass a number of ongoing project steering processes including periodic reviews of individual projects, typically by means of a stage-gate new product development process [14].

Using a stage-gate new product development process for steering projects can facilitate the selection of the most strategic and valuable projects in the portfolio. Un-strategic or mediocre projects can be killed based on business facts and resources can be focused on the 'right' projects.

3.4 'On-Site Customers'

In contemporary agile software development models the need for business focus in software development projects is addressed by requiring that 'customers' and developers must work together daily throughout the project [2]. Taking into account that 'customer' in agile methodologies represents any relevant stakeholder (including senior management) outside the development team [3]; an 'on-site customer' implies that the project sponsor (or some other responsible senior manager) has to be available full time on-site for the development team. This may be achievable in small organizations with few and small projects, but in large organizations (such as Ericsson or ABB) with large development projects distributed over several countries and time zones, this solution will soon become problematic.

3.5 Conclusion

Neither of the investigated models explicitly describes how stage-gate new product development processes (used for project steering) can be effectively combined with software development processes, as we have done. The anchor point milestones provide defined occasions where relevant stakeholders should commit themselves for supporting the next phase of a development project, but the actual business decision process is not further described. Project models like Ericsson PROPS and the ABB Gate Model provides frameworks for combining the models, but does not describe how. The solution with continuously available relevant stakeholders suggested in agile methodologies is not generally applicable in the same way as the solution suggested in the next section.

4 CONTRIBUTION

The first research hypothesis was that the usage of a formal stage-gate new product development process is one possibility to achieve effective senior management involvement in software product development projects. Experiences collected from the ABB IIT Deployment Initiative [34] indicate that this hypothesis can be confirmed (see Appendix C). The vast majority of the projects within the initiative are steered by senior management by means of the stage-gate new product development process defined in the ABB Gate Model Business Decision Layer. Unpromising projects are canceled and they are typically canceled early, before any major investments have been made. Promising projects are allowed to continue with or without redirections. The decision to continue or cancel a project is based on an evaluation of the project's business case, the product's fit within the product portfolio and the project's execution status.

The second hypothesis was that it is possible to combine a formal stage-gate new product development model with most software development models, without violating the principles of any of them, achieving both effective senior management involvement and the needed degree of freedom and flexibility in the software development process. This hypothesis can be confirmed theoretically by the suggested method and examples on how to combine the models by means of pre-gate milestones [32][33] (see Appendix A and Appendix B). The method requires no modification of either model, it is only a question of project planning and executing the project according to the plan.

The principles behind the pre-gate milestones and the model combinations as well as the ABB IIT Deployment evaluation are described in the following sections.

4.1 Pre-gate Milestones

It is important to understand the difference between information and documentation. Relevant information is what is needed to make business decisions at the gates, not necessarily written

documents. For example the existence of a project plan document does not ensure that the project is realistic and feasible and that needed resources are available. In the same way the existence of detailed design models does not ensure that all technical risks are solved. Business decision makers should analyze relevant information and make gate decisions based on facts, not only sign off on a checklist that activities have been performed and that a document exists. However, the baseline of a document can be an important milestone in the development project, concluding that required work is done.

The gates in the new product development process should not be used as milestones in the software development, as the gates have different purposes than the milestones. However, to be able to perform an evaluation of the product and project before a gate, the required information has to be available. This typically means that some key milestones in the software development plan have to be passed. The purpose of the pre-gate milestones defined as a result of this research [32][33] is to ensure that the relevant information is available in time to allow for a project and product evaluation before a business decision is made at a gate. See Figure 12.

Note that pre-gate milestones should not only be specified in the software development plan, but also in the marketing plan, competitor monitoring plan, intellectual property management plan, training plan, service plan, quality assurance plan, and hardware development plan and so on. Consequently, all pre-gate milestones, in all plans, ought to be passed before the evaluation of the product and project takes place.

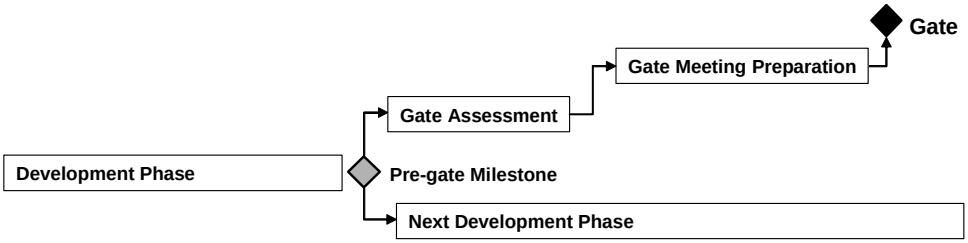

Figure 12: Pre-gate Milestone

4.2 Model Combinations

It is also important to understand the difference between combining processes and merging them. Merging all needed software product development processes into one monolithic process model is common but often results in a model that is difficult to use. Most software development models are monolithic. For example, in his book Extreme Programming Explained [3] Kent Beck stresses that caution should be taken to the fact that the practices of Extreme Programming complement and depend on each other. He explicitly states that the "practices support each other [and] the weakness of one is covered by the strengths of another".

If a monolithic process model is specific enough to provide guidance on project and working methods and tools, it will probably fit only a few development projects and requires tailoring for each project. Rational Unified Process has for example a discipline called 'environment' [17] built into the model with the purpose to tailor the Rational Unified Process itself to fit a specific project.

On the other hand, if a monolithic process model is generic enough to fit most projects and different methodologies and technologies within an organization it will probably not provide enough support to a specific project.

If a development process instead is regarded as a composition of different process components that communicates through defined interfaces, a much more flexible development model can be achieved. A project could select among available optional process components included in the model, combine them with mandatory process components in the model and create a project specific development process that satisfies that projects needs.

It was a whish in the beginning of this research that the ability to choose software development model should be independent of the stage-gate new product development model used. The suggested solution is that the software development process should provide (some of the) needed information, and that the new product development process should evaluate that information.

Remember that the purpose of the pre-gate milestones [32][33] is to ensure that the relevant information is available in time to allow for a project and product evaluation before a business decision is made.

Using the concept of pre-gate milestones, mapping new product development gates and software development major milestones becomes quite straightforward. Looking at the purpose of each gate and the fulfillment criteria for each major milestone it is fairly easy to judge which major milestone can be used as pre-gate milestone to which gate. If there are fewer major milestones than gates additional pre-gate milestones have to be identified among the minor milestones to ensure that relevant information is available in time fore the gate. In other cases gates may be combined and passed on one occasion.

Comparing Boehm's anchor point milestones with Cooper's Stage-Gate™ gates indicates that the Life Cycle Objective milestone can be used as pre-gate milestone to Gate 2, the Life Cycle Architecture milestone can be used as pre-gate milestone to Gate 3 and the Initial Operation Capability milestone can be used as pre-gate milestone to Gate 5. There is no major milestone needed before Gate 1, as Gate 1 is where the decision to start the project is made. It is only before Gate 4 a major milestone is missing. See Table 2.

Other examples of model combinations can be found in the paper Combining Models for Business Decisions and Software Development [32] see Appendix A and the article Integrating Business and Software Development Models [33] see Appendix B.

Table 2: Boehm's anchor point milestones compared with Cooper's Stage-Gate™ model gates.

Anchor Point Milestone	Criteria	Gate	Purpose
Not available	Not needed, project is not started	GATE 1 Idea screen	Commitment to product idea and to start preliminary investigation.
LCO Lifecycle objective	- Top-level objectives and scope defined. - Operational concept defined. - Top-level functions and architecture defined. - Lifecycle model and stakeholders identified	GATE 2 Second screen	Commitment to preliminary investigation and to start construction of the business case.
LCA Lifecycle architecture	- Project plan specified - Functions, interfaces and quality attributes specified. - Architecture specified. - Release criteria specified	GATE 3 Go to development	Commitment to the business case and to start development.
Not available	Not available	GATE 4 Go to testing	Commitment to development outcomes and to go to beta-test (validation), trail production and trail sells.
IOC Initial operation capability	- Product ready - Site ready - Users, operators, and maintainers ready	GATE 5 Go to launch	Commitment to validation and to start market launch and full production or operations

4.3 Evaluation of the ABB IIT Deployment Initiative

The ABB IIT Deployment Initiative is an ongoing corporate wide product line deployment initiative including central funding and expert support. One goal with the initiative is to deploy the new software platform and other core assets within the company's different software product development organizations, i.e. to deploy the foundation for the company wide software product line. One major task in the initiative is to demonstrate the value of the new software platform

across the company's businesses areas through a number of targeted pilot solutions. The pilot solutions can either be a product prototype aimed for many customers or an individual solution developed for a single customer. A thorough description of the experiences gained from the usage of a stage-gate new product development process in the ABB IIT Deployment Initiative can be found in paper "Three Aspects of Successful Software Development Projects" [34], see Appendix C.

The author of this thesis analyzed the portfolio of pilot projects included in the ABB IIT Deployment Initiative to evaluate when (at which gate) projects are canceled by senior management, and for what reason (business, technology or project execution). One expected result achieved due to the senior management involvement in the pilot projects is that unpromising pilot projects (i.e. projects with an unpromising business case) are canceled, and they are typically canceled early before any major investments have been made. Another expected, but not achieved result, is that unsuccessful project execution should lead to project cancellation, or at least redirection. Not a single pilot project in the portfolio is canceled due to schedule overrun, although almost all active pilot projects are (very) late.

5 CONCLUSION

To facilitate that the right software products (from a business perspective) are developed the right way, software development models and new product development models could be used in combination. The question is how to combine them in a good way. Software products or systems can span over embedded systems with high demands on real-time response and security, to large distributed business systems handling huge amounts of data, to all kinds of information technology systems with extreme demands on time-to-market. Different software products build on different technologies and have to satisfy different types of functional and quality requirements. Different software products have to satisfy different standards and regulations and will be handled by different types of organizations and will have different development lifecycles. Even if a company introduces one common new product development model in its organizations, it has to be possible to combine it with several different software development models [7][22].

The research results presented in this thesis include a straightforward method with several examples on how to combine stage-gate new product development processes with software develop processes by means of pre-gate milestones [32][33]. The research results also include experiences that indicate that the usage of a stage-gate new product development process is an effective way to achieve senior management involvement and business focus in software development projects [34].

5.1 Limitations

Neither of the two hypotheses presented in this thesis are validated through controlled case studies and the ABB IIT Deployment pilot projects evaluated in this research can not be regarded as representative for software product development in general, as they all develop a first version of the respective product.

There is no data indicating if the usage of a new product development processes increases or decreases the workload on project management. Some project managers complain that the

collection of information needed at each gate increases their work load, while others find that the information requirements enables them to focus on the right things during each development phase e.g. to develop the business case early in the project.

To plan a project and then execute the project according to the plan requires a certain level of maturity within an organization [8][25]. This level of maturity is probably not generally reached among the projects evaluated in this research. For example, in some of the projects evaluated proper usage of the pre-gate milestones failed. Senior management did not require that information was available in time before a gate and project management did not provide the information on time either. In some cases, the missing information led to that the gate meeting (where a continue-or-cancel decision should be made) turned into a project reporting meeting, the gate was only used as a milestone and the actual gate decisions were postponed to some vague point in time (sometimes several months) after the corresponding gate meeting.

5.2 Generally Valid Results

Using a stage-gate new product development process to achieve effective senior management involvement, and a business perspective, into software product development ought to be generally valid. The number of stages and gates and the evaluation criteria for each gate should although be defined for each organization and type of project.

Combining stage-gate new product development models and software development models by means of pre-gate milestones also ought to be generally valid. By using the reasoning behind the pre-gate milestones any stage-gate new product development model can be combined with any software development model. The reasoning behind the pre-gate milestones (providing relevant information timely) can probably also be used to combine several other models.

The validation of the hypotheses was done within one company and one project portfolio. There is a possibility that different results can occur in another company or under other circumstances. We

have demonstrated that at least in one case the hypotheses are verified. By using reasoning we have also shown that it is expected that the hypotheses are valid in a more general context.

5.3 Lessons Learned

Based on the literature survey and experiences from the ABB IIT Deployment Initiative we have learned the following lessons:

- **All evaluated software development models support active customer involvement** in software product development in some way or another. In practice this fact is often neglected or forgotten, the question why remains.

- **The implementation of a stage-gate new product development process within a software development organization is typically well received** both by senior management and by project management, at least within ABB. The process makes both parties focus on important business issues in each development phase and facilitates the communication. Projects are evaluated from a business perspective on a regular basis and unpromising projects are canceled early releasing resources for other tasks.

- **Using any software development model in an immature organization is difficult**, even with the support of a new product development process. The need to plan and follow the plan is not understood. Pre-gate milestones are not respected, i.e. relevant information is not provided timely. Schedule overrun is a rule, not an exception and the model is blamed.

5.4 The importance of project execution success

A new question that has emerged during the work with this research is if development project execution success (i.e. to deliver required features and functions on time and budget) really is important for successful software product development? Common sense says project execution success has to be important. Overrun in development budget, unplanned maintenance and support costs due to unexpected low quality, or decreased market due to limited features and functions etc., ought to have impact on the products business case and return on investment. But the experiences gained during this research do not confirm this.

6 FUTURE WORK

The findings in this research have spawned some new research ideas:

- **Industrial Validation**: The combination of new product development models with software development models by means of pre-gate milestones as presented in this thesis are for the most part based on theoretical reasoning. An important consideration in empirical research is validation, the ability of the research results to apply to the world outside the research situation. The results outlined in this thesis are straightforward and can be meaningfully applied to most software product development organizations. Empirical validation of the results in an industrial setting would be beneficial. Interesting questions to answer could for example be:

  - What extra activities (effort) are needed for the preparation of the required information before each gate?

  - How well do the major milestones criteria in the different software development models fit with the information requirements at the corresponding gates?

- **Project Execution Success and Software Product Success**: One common purpose with all software development models evaluated in this thesis is that they will support the delivery of software products with required features, function and quality on **time** and **budget**. This research can however not verify that product delivery on time is important. As long as the business case is promising and the product fits in the organization's product portfolio and strategy, time and budget seems to be of (much) less importance. It would be interesting to study this phenomenon further.

- **Extending the new product development process for Project Portfolio Management**: This research indicates that using a new product development process to achieve senior

management involvement (and steering) in single software product development projects is effective. However large organizations typically run several (software) product development projects at the same time, i.e. a portfolio of projects. Research into defining how a new product development process could be used also to support project portfolio management would complement the model.

- **Process Components and Interfaces:** One of the motivations for this research was the whish to be able to combine a new product development process with a software development process without merging them into one (monolithic) development process. This process combination approach indicates the possibility to identify or define process components and process component interfaces in a similar way as software components and software component interfaces. Further research in this area can include topics like component based process definition, process architectures and process qualities. One question to answer could be:

  - Can process components be defined so they are exchangeable? For example can code reviews in a formal development process be exchanged with pair programming from Extreme Programming?

7 REFERENCES

[1] ABB, *ABB Gate Model for Product Development 1.2*, 9AAD104000, ABB, 2003

[2] Beck Kent, Et al, *Principles behind the Agile Manifesto*, http://agilemanifesto.org/, Mars 2003

[3] Beck, Kent. *Extreme Programming Explained*, ISBN 0-201-61641-6, Addison-Wesley, 2000

[4] Boehm, Barry. *A Spiral Model of Software Development and Enhancement*, IEEE Computer, May 1988, Page(s) 61-72.

[5] Boehm, Barry. *Anchoring the Software Process*, IEEE Software, July 1996, Page(s) 73-82.

[6] Boehm, Barry. *Spiral Development: Experience, Principles, and Refinements*, CMU/SEI-2000-SR-008, SEI, July 2000.

[7] Bonner, Ruekert, Walker. *Upper management control of new product development projects and project performance*. The Journal of Product Innovation Management 19, PDMA, 2002, Page(s) 233-245

[8] CMMI Product Team, *Capability Maturity Model ® Integration (CMMI ^{SM})*, Version 1.1, CMU/SEI-2002-TR-011, SEI, Mars 2002

[9] Cooper Robert G., *Winning at new Products, Third Edition*, ISBN 0-7382-0463-3, Perseus Publishing, 2001

[10] Cooper, Edgett, Kleinschmidt. *Portfolio Management for New Product Development: Results of an Industry Practices Study*, R&D Management, vol. 31, no. 4, October 2001, pp. 361-380

[11] Ericsson, *Introduction to PROPS*, ISBN 91-89438-11-6, Ericsson Business Consulting, 2000

[12] Ericsson, *PROPS Manual for Project Sponsors*, ISBN 91-89438-15-9, Ericsson Business Consulting, 2000

[13] Gibbs W. Wayt, *Software's Chronic Crisis*, Scientific American, September 1994

[14] Griffin, Abbie, *Drivers of NPD Success: The 1997 PDMA Report*, Product Development & Management Association, 1997

[15] IEEE/EIA 12207.0-1996, *Industry Implementation of International Standard ISO/IEC 12207:1995*, Standard for Information Technology - Software life cycle processes, 1998

[16] ISO/IEC 15288, *System Engineering - System Life Cycle Processes*, First Edition, ISO/IEC, 2002

[17] Jacobson, Booch and Rumbaugh, *The Unified Software Development Process*, ISBN 0-201-57169-2, Addison-Wesley, 1999

[18] Jones, Capers. *Patterns of Software Systems Failure and Success*. ISBN 1-850-32804-8. International Thomson Computer Press. 1996

[19] Johnson, Boucher, Connors, Robinson. *Collaborating on Project Success*, Software Magazine, February/March 2001

[20] Kruchten, Philippe. *A Rational Development Process*, Crosstalk, July 1996

[21] Linger, Tramell. *Cleanroom Software Engineering Reference Model*, Version 1.0, CMU/SEI-96-TR-022, SEI, November 1996

[22] McConnell, Steve, *Rapid Development*, ISBN 0-201-54664-7, Addison-Wesley, 1997

[23] Microsoft. *Microsoft Solution Framework Overview White Paper*, Microsoft Corporation, December 10, 1999

[24] OMG, *Software Process Engineering Metamodel Specification*: Version 1.0, OMG, November 2002

[25] Paulk, Curtis, Chrissis, Weber. *Capability Maturity Model ^{SM} for Software*, Version 1.1, CMU/SEI-93-TR-024, SEI, 1993

[26] Rajlich, Bennett. *A Staged Model for the Software Life Cycle*. IEEE Computer, July 2000

[27] Royce, Winston W., *Managing the Development of Large Software Systems*, IEEE WSCON, August 1970

[28] Stapleton, Jennifer. *DSDM – Dynamic Systems Development Method*, ISBN 0-201-17889-3, Pearson Education, Harlow 1997

[29] The Standish Group International Inc., http://www.standishgroup.com/, October 2003

[30] The Standish Group International Inc., *CHAOS: A Recipe for Success*, 1999

[31] Thomsett Rob, *Project Pathology: A Study of Project Failures*, American Programmer, July 1995

[32] Wallin, Larsson, Ekdahl, Crnkovic, *Combining Models for Business Decisions and Software Development*, Euromicro 2002

[33] Wallin, Larsson, Ekdahl, *Integrating Business and Software Development Models*, IEEE Software November/December 2002, IEEE Computer Society

[34] Wallin, Crnkovic, *Three Aspects of Successful Software Development Projects, "When are projects canceled, and why?"*, Euromicro Conference 2003, IEEE Computer Society

GLOSSARY

Customer. A *Customer* is the party (individual, project, or organization) responsible for accepting the product or for authorizing payment. The customer is external to the project, but not necessarily external to the organization. The customer may be a higher level project. Customers are a subset of stakeholders [8].

Lifecycle. A process *Lifecycle* is defined as a sequence of phases that achieve a specific goal. It defines the behavior of a complete process to be enacted in a given project or program [24].

Phase. A *Phase* is a specialization of work definition such that its precondition defines the phase entry criteria and its goal, called "Milestone" in this case, defines the phase exit criteria. Phases are defined with the additional constraint of sequentiality; that is, their enactments are executed with a series of milestone dates spread over time and often assume minimal (or no) overlap of their activities in time [24].

Product. A *Product* is any tangible output or service that is a result of a project and that is intended for delivery to a customer or end user. [8]

Project. A *Project* is a managed set of interrelated resources that delivers one or more products to a customer or end user. This set of resources has a definite beginning and end and typically operates according to a plan. Such a plan is frequently documented and specifies the product to be delivered or implemented, the resources and funds used, the work to be done, and a schedule for doing the work. A project can be composed of projects [8].

Project Sponsor. The *Project Sponsor* is the Senior Manager who is commercially and financially responsible for the project and its outcome. The Project Sponsor is the primary risk taker for the project and makes the business decisions, based on an assessment of the project's alignment with the organizations business direction [11]. Synonyms for Project Sponsor include **Executive Sponsor**.

Project Team Member. A *Project Team Member* is an individual working in a project, either within the project management function or within the project execution function.

Stakeholder. A *Stakeholder* is a group or individual that is affected by or in someway accountable for the outcome of an undertaking. Stakeholders may include project members, suppliers, customers, end users, and others [8].

Relevant Stakeholder The term *Relevant Stakeholder* is used to designate a stakeholder that is identified for involvement in specified activities and is included in an appropriate plan [8].

Senior Manager. The term *Senior Manager* refers to a management role at a high enough level in an organization that the primary focus of the person filling the role is the long-term vitality of the organization, rather than short-term project and contractual concerns and pressures. A senior manager has authority to direct the allocation or reallocation of resources.

A senior manager can be any manager who satisfies this description, including the head of the organization. Synonyms for Senior Manager include **Executive** and **Top-level** Manager [8].

APPENDIX A: COMBINING MODELS FOR BUSINESS DECISIONS AND SOFTWARE DEVELOPMENT

Christina Wallin, Stig Larsson, Fredrik Ekdahl, & Ivica Crnkovic

Proceedings of Euromicro Conference, September 2002

Abstract:

Today there is a number of established software development lifecycle models (SDLMs) supporting software development. Correct implementation of these models helps develop software products the right way, but this does not ensure that the right products are developed. Successful product development companies often use business decision models (BDMs) to facilitate the selection of products and projects for investment, but these models do not necessarily facilitate actual development of the software. One of the current challenges in the software community is to combine BDMs and SDLMs, including mapping of business decision gates and major lifecycle milestones. This is needed to achieve synergies between the two model types and to support the development the right products the right way, as well as to gain control over company investments. This paper analyzes two BDMs, proposes mappings to an established SDLM, and describes experiences of using them in a large, multinational engineering company.

1. Introduction

Development of software products is typically done in projects, using different SDLMs. Used properly; these enable the projects to deliver products with expected quality and functionality, on time and on budget. This is however not enough for a commercial success.

is at least equally important that the right product development projects are selected and that sound business decisions govern the projects until the product is launched. Progression from one development lifecycle phase to another should be based on a deliberate business decision, verifying that the development project and the product are still feasible to the development organization and

the market. Organizations successful in developing new products typically use some kind of BDM to achieve this [1].

Using both a BDM and a SDLM can however introduce problems in the interaction between business decisions and software development. As there are different models (both for business decisions and software development) it might not be obvious how these models interface and how they should be synchronized. Another problem is that some software developing organizations, introducing a BDM, experience that it forces them to use a particular type of SDLM, which might not be perceived as appropriate from the development point of view.

Consequently, mappings of SDLMs and BDMs are needed for clarity and to fully realize the synergies between these two model types.

This paper aims to demonstrate the concept of mapping BDMs and SDLMs. The outline of the paper is as follows: Section 2 describes two BDMs, the Stage-Gate™ Model [1] and the ABB Gate Model for Product Development [3]. Section 3 gives a short overview of software development lifecycle phases and major milestones as defined in the Unified Process [2]. Section 4 discusses the mapping of business decision- and software development lifecycle models by introducing pre-gate milestones. Finally the last section is a summary of experiences, and concludes with plans for future work.

2. Business Decision Models

A business decision should be based on the results from evaluating the market and competitors, the technology feasibility, the business strategy, intellectual property rights, product quality and status, and available resources within the organization. Today, many organizations use a well-defined model for preparing and making business decisions in product development projects. Such BDMs generally consist of a number of different development stages separated by business decision gates. Cooper's Stage-Gate™ model [1][4] is a good example of a BDM.

2.1 Cooper's Stage-Gate™ Model

Cooper's Stage-Gate™ model, shown in Figure 1, divides the development project lifecycle into six stages, separated by five gates. Each stage consists of a set of parallel activities and processes performed by different actors within an organization. Each activity in each stage is designed to gather information needed as input to the upcoming business decision gate and to reduce risks associated with the creation or evolution of a product.

The first stage of Cooper's model is the *discovery* stage. It begins with an idea for a new product or product version. During the *scoping* stage, the main objectives are to assess market and technology and identify the key product requirements. During the *business case* stage, information needed to decide if it is feasible to develop the product is gathered. The *development* stage mainly deals with the development of the product according to the product and project definitions. In stage four, *testing and validation*, the product is finally verified and validated, and the final stage, the *launch* stage, includes activities for marketing and sales, and for production or operation.

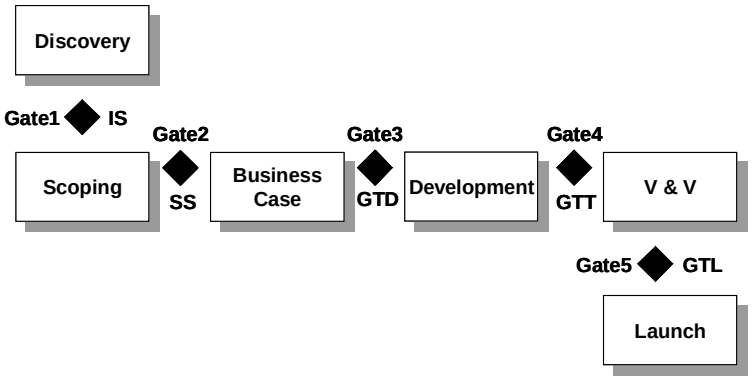

Figure 1: Cooper's Stage-Gate™ Model

The gates between each product development stage represent distinct business decision points. The procedures at each gate are similar; the results from the activities performed in the stage preceding the gate, together with a decision criteria checklist are used as input to the business decision. The

output from the gate is a go/no-go/hold/redo decision accompanied by relevant plans for the next stage.

Cooper's Stage-Gate™ model uses five gates: Gate 1, *idea screen* (IS), is the first occasion where resources are committed to the project. Gate 2, *second screen* (SS), is essentially a repetition of Gate 1, although more rigorous and based on the information gathered during the scoping stage. Gate 3, *go to development* (GTD), represents the last chance to stop the project before significant investments are made. A go decision at this point represents both a financial and resource commitment and an agreement on the product and project definition established during the build-business-case stage. Gate 4, *go to testing* (GTT), is based on a post-development assessment to make sure that the product and project are still attractive to the market and to the organization. Finally Gate 5, *go to launch* (GTL), is the last point at which the project can be stopped and the product cancelled.

Cooper's Stage-Gate™ Model is widely accepted in development and manufacturing industry. Many companies use this or slightly modified versions of the model, and experience reports from the companies are positive [5][6][7]. One of the advantages of a BDM is the permanent awareness of the business goals and the presence of clear alternatives for the decisions. Another advantage is that the commitment of funding for a project is low at the start and increases as the project progresses when the stakeholders become more confident that the project will ultimately be successful. A difficulty with using BDMs is the significant requirements on the information needed for the business decisions already early in the project. If the development lifecycle model used is not well defined and accurately synchronized with the BDM, there is a high risk that the information used for the decision is not well prepared. Another consequence of poor synchronization of the models is the risk of spending unnecessary effort and time to produce information that is neither needed nor a natural result of the activities conducted so far in the project.

2.2 The ABB Gate Model

The ABB Gate Model for Product Development [3] defines eight gates where major business decisions are made. The model serves as a framework for the various activities, e.g. software

development, hardware development, competitor management, intellectual property management, training and marketing, etc. included in a product development project.

There are several reasons why ABB developed its own model. The primary reason is that the model has grown from the company's own experience and to some extent from shared experience with other companies (such as the PROPS [8] model from Ericsson). This approach gives a good tuning of the model to specific company needs, but requires extra effort for development and maintenance, and increases the risk of ending up with a model that is not in line with general trends or applicable de-facto standards.

The ABB Gate Model does not explicitly define any stages. It is implicitly assumed that the selected SDLM includes the phases or stages needed. The results from the development activities provide information used as input to the gates. This way the ABB Gate Model is not used as an independent and self-sustained process but it is closely related to the development processes. When using a BDM as an independent process, there is an apparent risk for a mismatch with the development process. Experiences from ABB show that such a mismatch (in strategy, in goals, in procedures and techniques) can lead to misunderstandings between the developers and management, to decisions based on faulty information, to decisions made on the wrong level, or even to decisions not being made at all. On the other hand, good synchronization between the models increases both decision and development quality. For this reasons it is crucial that the BDM is distinguished from, but tightly coupled with the SDLM. One way to achieve this is to utilize gates as the decision points, and refrain from specifying lifecycle stages as activities separate from the decision process.

2.2.1 Gate Model Roles

There are two specific roles defined in the ABB Gate Model, the Gate Owner and the Gate Assessor.

The Gate Owner is the person, or group of persons, that have the responsibility and authority to decide if a product should be developed or not, and if a development project should run or not. The Gate Owner is also responsible for the funding of the development project and for the availability of required resources. Generally this is the product or customer responsible.

The task of the Gate Assessor is, on behalf of the Gate Owner, to evaluate the product and the project before a gate, to produce a gate assessment report, to suggest a gate decision, and to present the assessment result at a gate meeting. The Gate Owner appoints the Gate Assessor and it is recommended that the Gate Assessor is external to the development project, to be able to be as objective as possible. This is a demanding role, as the Gate Assessor needs to be both experienced and competent, trusted by the Gate Owner and respected by the organization.

2.2.2 Gate Procedure

The gate procedure in itself is fairly simple and consists of only two activities, the gate assessment and the gate meeting. Input to the gate assessment is documents prepared by the project, interviews with project stakeholders and a gate assessment checklist. The assessment is done over an extended period, typically a calendar week, and involves both the project manager and the gate assessor. The output from the assessment is a report (a slide presentation) addressing the checklist items for the current gate. The assessment report together with other relevant material is made available to the gate meeting participants prior to the gate meeting to give them time to prepare. At the gate meeting, a go/no-go/hold/redo decision is made for the project. After the gate meeting the decision and any identified actions are communicated to all stakeholders.

2.2.3 Gates

The first six gates (Gate 0 to Gate 5) are decision points used to determine whether or not the development project should continue. Gate 6 and 7 are used to close the development project and to capture experience.

Input to Gate 0, *Start Project* (SP), is a feasibility study report or a project proposal including analysis of the market, competitors, intellectual properties, product strategy, risks, needed resources and required technology.

At Gate 1, *Start Project Planning* (SPP), the development project scope should be defined in terms of functions, features and quality as well as business constraints such as for example time to market, in such detail that it can be used for planning.

At Gate 2, *Start Execution* (SE), the project should be planned in terms of specified requirements, effort, time and cost estimates, procedures for quality assurance, risk management, configuration management and so on.

At Gate 3, *Confirm Execution* (CE), all major risks should be addresses and all technical solutions proposed.

At Gate 4, *Product Introduction* (PI), all functions and features should be implemented and the product should be ready for Beta, or acceptance- test and marketing.

At Gate 5, *Product Release* (PR), the product should be ready for release to the market or customer.

At Gate 6, *Close Project* (CP), the development project should be closed and product hand-over to manufacturing and/or service and maintenance should be confirmed.

At Gate 7, *Retrospective Investigation of Project* (RIP), an evaluation of the project and product should be done to evaluate its business success.

3. Software Development Lifecycle Models

The software development lifecycle can be divided into a number of phases, generally 3-5, indicating the main focus of the development work at that time; investigating the scope, constructing the software or deploying the results. There are almost as many names for these phases as there are software-developing organizations, but in the context of this paper the lifecycle phases defined in the Unified Process (UP) [2] will be used. According to UP, each software development lifecycle consists of four phases: inception, elaboration, construction and transition. Each phase can then be further subdivided into steps (iterations), see Figure 2.

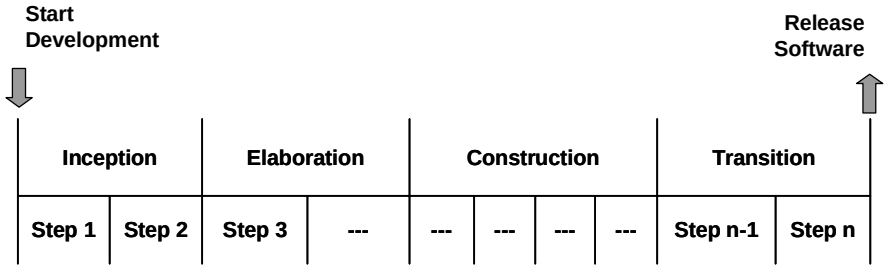

Figure 2: Unified Process Development Lifecycle Phases

The progression from one phase to the next is made when passing a major milestone. A milestone is defined as a scheduled event that marks the completion of one or more important tasks and it is used to measure achievements and development progress. At a milestone, a predefined set of deliverables should have reached a predefined state to enable a review.

In his article "Anchoring the Software Process" [9] Barry Boehm describes the three critical milestones essential for successful management of system development:

Life Cycle Objectives (LCO) - stakeholders' agreement on the system's top-level objectives such as: system boundaries, operational concept, system requirements, system and software architecture, and development lifecycle plan and feasibility rationale. The primary goal of the inception phase,

which is concluded with the LCO milestone, is to set the technical scope of the software, outline the architecture, identify critical risks, and build a proof-of-concept prototype.

Life Cycle Architecture (LCA) - stakeholders' agreement on the system's elaborated objectives, especially the system and software architecture, and complemented with a risk assessment and risk management plan. The goal of the elaboration phase, concluded with the LCA milestone, is a stable software architecture, identified significant risks, specified quality requirements, most functional requirements captured, and planned schedule, cost and resources.

Initial Operational Capability (IOC) - the system is prepared for operation and support, the deployment site is prepared and users, operators and maintainers are prepared and trained. The general objective of the construction phase, concluded with the IOC milestone, is a software capable of initial operation, that is, it is ready for Beta testing.

The UP has adopted these three critical milestones as major software development milestones, modified the interpretation of them slightly and added a Product Release milestone to end up with four major software development milestones, see Figure 3. By the end of the transition phase, concluded with the PR milestone, the software should be tested, corrected and ready for a formal release, including all documentation and all required preparation of the manufacturing or operation environment.

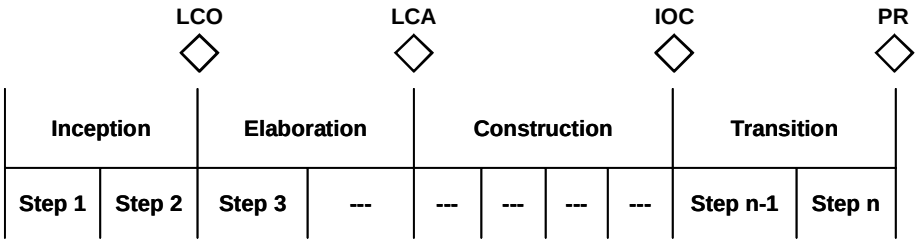

Figure 3: Major Milestones in the Unified Process

4. Model Mapping

To synchronize business decisions with development activities, in particular software development, a set of points for synchronization is needed. As the gates in the BDMs and major milestones in the SDLMs are clearly distinguishable, it is natural to use them for this purpose. It is desirable that the selection of SDLM is independent of the BDM used.

4.1 Pre-Gate Milestones

Gates should not be milestones in the software development plan as they belong to different processes. However, to be able to perform the assessment of the product and project before a gate, the required information has to be available; i.e. typically some key milestones have been passed. These milestones can be designated as Pre-Gate Milestones, see Figure 4. Note that pre-gate milestones are not specified only in the software development plan, but also in the marketing plan, competitor monitoring plan,, intellectual property management plan, training plan, service plan, quality assurance plan, hardware development plan and so on. Consequently, all pre-gate milestones, in all plans, ought to be passed before the assessment of the product and project takes place.

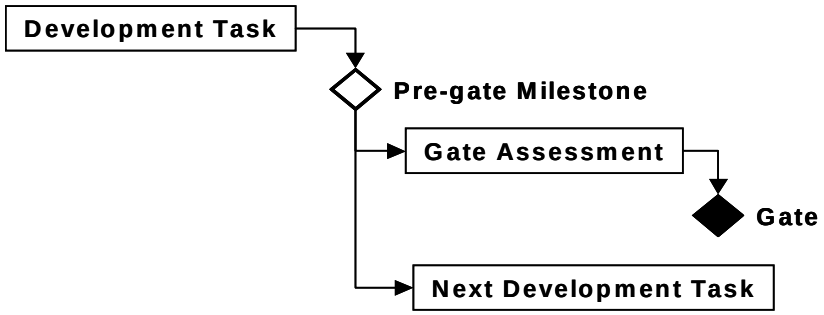

Figure 4: Pre-Gate Milestone

All product development tasks continue during the gate assessment but if a decision to stop the project is made at the gate, all planned tasks are cancelled.

4.2 Mapping Gates and Milestones

Using the concept of pre-gate milestones, mapping Cooper's Stage-Gate™ model product development gates and the UP software development major milestones becomes straightforward, see Figure 5. A go decision at Gate 1, idea screen (IS), is a prerequisite for the software development, as well as all other activities. Each major milestone in the UP can then be used as a pre-gate milestone to the corresponding business decision gate. An evaluation of the technical feasibility, an analysis of the market, an evaluation of the development and manufacturing/ operation capability, an estimation of development time and cost, and an investigation of any legal and regulatory constraints, provide input to Gate 2, second screen (SS). The scope and technical feasibility of the software should be defined in the UP inception phase.

A detailed technical appraisal, detailed market investigations and market research studies, as well as competitive analyses, investigations of needed internal investments, and detailed business and financial analyses provide input to the go-to-development (GTD) decision at Gate 3. A stable software architecture and planned schedule, staff and cost for the software development is the result of the UP elaboration phase.

Concurrent with the technical construction, market analysis and customer feedback activities are undertaken. Regulatory, legal and patent issues are resolved and test plans, market launch plans, production or operation plans are developed. When all this is done the project is ready to pass Gate 4, go-to-test (GTT). By the end of the UP construction phase, the software is ready for Beta testing.

During Beta testing, the market should be evaluated to determine expected market share and revenues, and the business and financial analyses should be revised. All this information has to be regarded at Gate 5, go-to-launch (GTL), as this is the last point at which the product can be stopped before a major commitment to production, service, maintenance, training and so on is made. Output from the UP transition phase, which is a software ready for formal release.

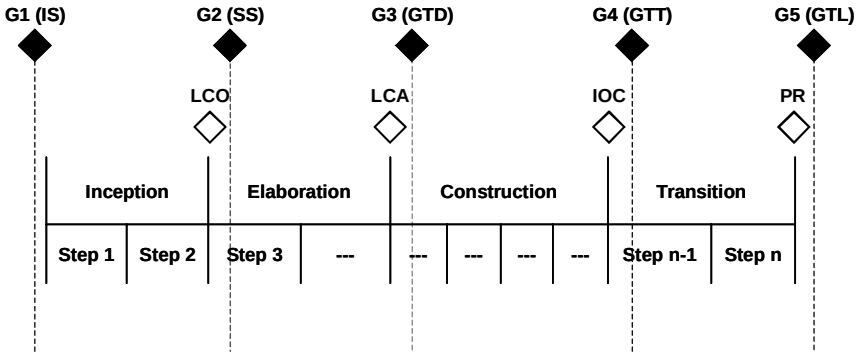

Figure 5: Mapping between Cooper's Stage-Gate™ model gates and Unified Process major milestones

This sample mapping illustrates that there is a good match between the information needed at the gates, and what is required to pass the milestones.

4.3 ABB Gate Model Mapping

The mappings of the ABB Gate Model gates and the UP major milestones much resembles that of Cooper's Stage-Gate™ model and UP, see Figure 6.

As in the Stage-Gate™ model mapping, a go decision at the ABB Gate Model Gate 0, start project (SP), is a prerequisite for starting the software development subproject, as well as the other subprojects. The goal of ABB Gate Model Gate 1, start project planning (SPP), is to agree on the project scope and the scope and technical feasibility of the software should be defined in the UP inception phase. The goal of ABB Gate Model Gate 2, start execution (SE), is to agree on the project plan and the goal of the UP elaboration phase is a stable software architecture and planned schedule, staff and cost for the software development. The goal of ABB Gate Model Gate 4, start introduction (SI), is to agree on the product readiness for piloting and market introduction, and the goal of the UP construction phase is to provide software ready for Beta testing. Finally the goal of ABB Gate Model Gate 5, release product (RP), is to agree on the product readiness for release and the goal of the UP transition phase is software ready for a formal release.

It is only before ABB Gate Model Gate 3, confirm execution (CE), that a major milestone in the UP, usable as a pre-gate milestone, is missing. Instead, a minor milestone indicating the finalization of an iteration [2] should be selected and used as a pre-gate milestone in the software development subproject.

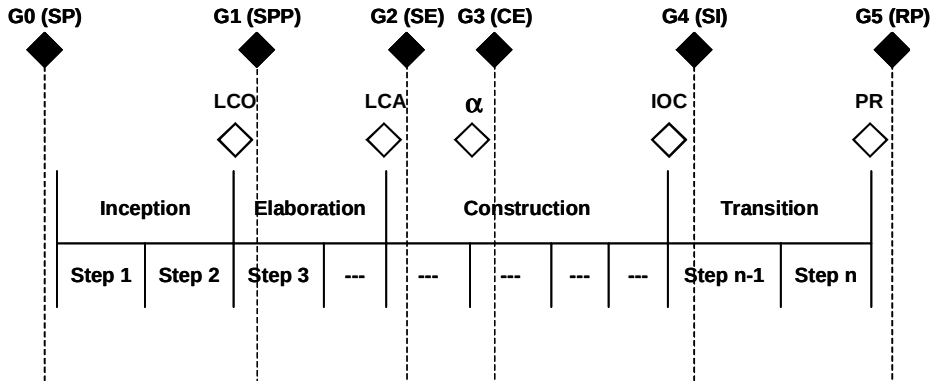

Figure 6: Mapping of ABB Gate Model gates and Unified Process major milestones

The main purpose of Gate 3 is to agree on the proposed technical solution, which must be taken into account when selecting an appropriate minor milestone. Experience indicates that it is good practice to decompose the UP construction phase into sub-phases and to mark the completion of the first sub-phase with a software release milestone that is also used as a pre-gate milestone to Gate 3. At this point the software has been integrated for the first time and the technical solution can be assessed. Consequently, it is an appropriate time for a gate decision.

When working with mappings in ABB, the experience is that in order to allow sufficient time for assessments and gate meeting preparations, the pre-gate milestones need to be passed at least two weeks before the corresponding gate.

5. Conclusion and future work

By combining a business decision model with a software development lifecycle model, but at the same time recognizing the need for two separate models, several advantages are achieved. In this paper we have illustrated how an organization can have the possibility to select a software development lifecycle model independently of the chosen business decision model. This also makes it easier for the organization to adopt future software development lifecycle models. Through a proper combination of the two model types, organization will also avoid trying to meet the need for a business decision model with the means of a software development lifecycle model, or vice versa.

The combination of models is theoretically simple, but may lead to misconceptions, e.g. that a business decision model forces the use of a waterfall like software development lifecycle model. This is typically a result of a misunderstanding of the gate concept. Instead of evaluating the business aspects of the software project and product at appropriate points in time, the gate assessments are used to "tick off" that deliverables are completed and the gates are regarded only as additional milestones.

The mapping between the ABB Gate Model and local development lifecycle models has shown positive results so far, with some variations. Organizations using UP have adopted the ABB Gate Model smoothly without serious problems. In some cases the organizations have experienced problems gathering all the information needed for the decisions at the early gates, especially at gate 2, which actually is an indication of the organization's maturity level.

The first organizations implementing the mapping between gates and major milestones have experienced a higher degree of management understanding as well as increased speed in the development. The projects have reduced the non-value added project tasks, and the clearly identified business decision model has made it possible to introduce iterative development in the software development subprojects, since the requirement from management on project status visibility is satisfied by the gate model.

Future efforts will be focused on mapping the ABB Gate Model and a wider set of software development lifecycle models, such as different variants of incremental or evolutionary models. To achieve this, a generic method for mapping different business decision models to software development lifecycle models will be developed. At the same time, the long-term effects of the deployment and use of the ABB Gate Model together with different software development lifecycle models will be analyzed. Also, experiences from the combination of business decision models and development models for other product development activities, such as intellectual property development or development of marketing and sales material, are needed to get the whole picture of developing the right software products in the right way.

6. References

[1] Cooper Robert G., Winning at new Products, Third Edition, ISBN 0-7382-0463-3, Perseus Publishing, 2001

[2] Jacobson Ivar, Grady Booch and James Rumbaugh. The Unified Software Development Process, ISBN 0-201-57169-2, Addison-Wesley, 1999, pp. 8-13, 102-104 and 410.411.

[3] GP-PMI 9AAD102113, ABB Gate Model for Product Development 1.1, ABB

[4] Cooper Robert G., Doing it right, Winning at new Products, Ivey Business Journal July/August, 2000

[5] Cormican, K. and O'Sullivan, D., Product Manager: A Decision Support Tool for Design Engineers". Proceedings of the European Network of Excellence on Advanced Methodologies and Tools for Manufacturing Systems International Scientific-Technical Workshop, 1999, Ufa, Russia.

[6] Johansson, J.; Nilsson, L. Product planning at an Electrolux subsidiary Engineering and Technology Management, 1998. Pioneering New Technologies: Management Issues and Challenges in the Third Millennium, IEMC '98 Proceedings

[7] National Renewable Energy Laboratory, Stage Gate Management in the Biofuels Program. (2001). 41 pp.; NICH Report No. MP-510-31541.

[8] Ericsson Project Management Institute, project-management model PROPS, http://www.ericsson.com/epmi/index.shtml

[9] Boehm Barry. Anchoring the Software Process, IEEE Software, July 1996, pp. 73-82.

APPENDIX B: INTEGRATING BUSINESS AND SOFTWARE DEVELOPMENT MODELS

Christina Wallin, Stig Larsson & Fredrik Ekdahl

IEEE Software, November/December 2002

By mapping business decision gates to major software development milestones, organizations can relate technical life-cycle models to business decision models. The authors mapped Unified Process, Synch-and-Stabilize, and Extreme Programming life-cycle examples to the ABB Gate Model for product development projects.

Today, software product development cannot generally be regarded as successful. Only about one of four software development projects are completed on time and on budget, with all the features and functions originally specified [1]. Running a software project is a complex task in itself; making the resulting product a commercial success is even harder.

Software development life-cycle models and business decision models contribute to the control of product development in different ways. However, both kinds of models have limitations. SDLMs do not ensure that resources are used in the right projects, that the market is available, or that the organization is ready for a release. Similarly, business decision models do not support software development, so development might take place with uncontrolled changes and inadequate time for verification and validation.

Thus, successful software product development requires that the project use both a business decision model and an SDLM. This requires careful definition of the interfaces, or mapping, between the two model types, as well as to any other model related to software product development. The ABB Gate Model, presented here, supports decision makers with business-relevant project and product information, increases mutual understanding and improves visibility

between decision makers and developers during product development, educates decision makers in software engineering problems and solutions, and educates developers in business issues.

How business issues hurt software development

Many business-related problems face software product development. First, stakeholders typically scrutinize their software development projects from a business perspective only at startup, if at all; they do not revisit the business case over the course of the project. Often, they do not identify market, technology, or schedule problems until the project has gone astray.

Second, because new technology drives software product development, stakeholders typically examine a project's business aspects less carefully than the technical solutions. This is of course a serious mistake, especially when a project is targeting a market that is new to the organization and when knowledge about this market is limited. Unfortunately, limited knowledge often leads to even less activity in trying to understand the business aspects.

Third, decision makers who don't understand the basics of software engineering change the target continuously without looking at resulting costs and delays. This is probably a result of the common view that developers can easily adapt software to last-minute requirements. However, decisions to change or add new functionality often overlook the tasks that go along with code changes – for instance, changed architecture and design documentation, changed user documentation, regression testing, redesign of test cases for verification and validation, and changes to training, marketing and support material, and so on.

Finally, project managers can feel squeezed in the middle. Typically, decision makers want facts as soon as possible, but ask for finalized documents. For example, a manager might want to know if the selected technical solution is feasible, but instead asks if the detailed design document is ready. On the other side, developers might not think they can provide enough information when the decision makers want it. They often think that business decision models imply a waterfall-like development life cycle, so those who want to use modern development practices might resist using

any such model. Also, modern practices such as the Unified Process and Extreme Programming require iterative and incremental development, which leads to late finalization of documents.

Business decision models

Delivering a product with expected quality and functionality, on time, and on budget is seldom enough to achieve commercial success. It is at least equally important to choose the right product development projects and to have a mechanism for closing down projects that no longer show sufficient potential.

Good business decisions are based on facts elicited through careful evaluation of key elements of the business situation – for example, market, competitors, technical feasibility, strategy, intellectual property, product quality, and resource availability. To facilitate the collection of relevant facts in time to make business decisions, many organizations use a well-defined process.

Several well-known business decision models exist, of which Cooper's Stage-Gate Process Model [2] is one example (see below for more information). Typically, they comprise a number of different development stages separated by decision points, often referred to as decision gates. The gates represent distinct decision points at which stakeholders decide the project's future.

Cooper's Stage-Gate Process Model

Cooper's Stage-Gate Process Model, shown in Figure A, breaks the development project life cycle into six stages and five gates. Each stage consists of a set of parallel activities, of which software development is only one, performed by different functions within an organization. Each activity in each stage is designed to gather information needed as input to the upcoming business decision gate and to reduce risks associated with the development project.

The stage before the actual development project starts, the Discovery stage, begins with an idea for a new product or product version. Generally, a product manager collects the information needed as input to the first business decision gate.

Gate 1, the Idea Screen decision point, follows the Discovery stage and is the first occasion where decision-makers commit resources to the product development project. The product manager presents the idea to the stakeholders from development, marketing and sales, service and maintenance, manufacturing, training, and so on, who together decide whether to start a development project based on the idea.

During the first product development stage, the Scoping stage, the main objective is to assess the market and technology to identify key product requirements.

Gate 2, the Second Screen decision point, essentially repeats the previous gate, although with more rigorous

requirements and based on the information gathered during the Scoping stage.

The second development stage, Building the Business Case, includes a detailed investigation that clearly defines the product, market, organization, development project, competitors, intellectual properties, and so on in preparation for deciding whether developing the product is feasible.

Gate 3, the Go to Development decision point, is the gate prior to the Development stage and the last chance to stop the project before the organization makes significant investments. A go decision at this point represents both a financial and resource commitment to the project as well as an agreement on the product and project definition established during the Building the Business Case stage.

The third stage, Development, mainly deals with the product's physical development according to the product and project definitions. The deliverable from this stage should be a product ready for beta testing.

Gate 4, the Go to Testing decision point, is based on a post development assessment to ensure that the product and project are still attractive to the market and to the organization. A go decision at this point is an agreement on the verification and validation plans and also on marketing and operation plans.

In stage four, Testing and Validation, the product is verified and validated in-house or at friendly customers' sites.

Finally Gate 5, the Go to Launch decision point, is the last point at which the project can be killed and the product cancelled. A go decision here is an approval of the marketing and operation plans and the startup of full production or operation.

The final stage, Launch, includes, for example, activities for marketing and sales and for production or operation.

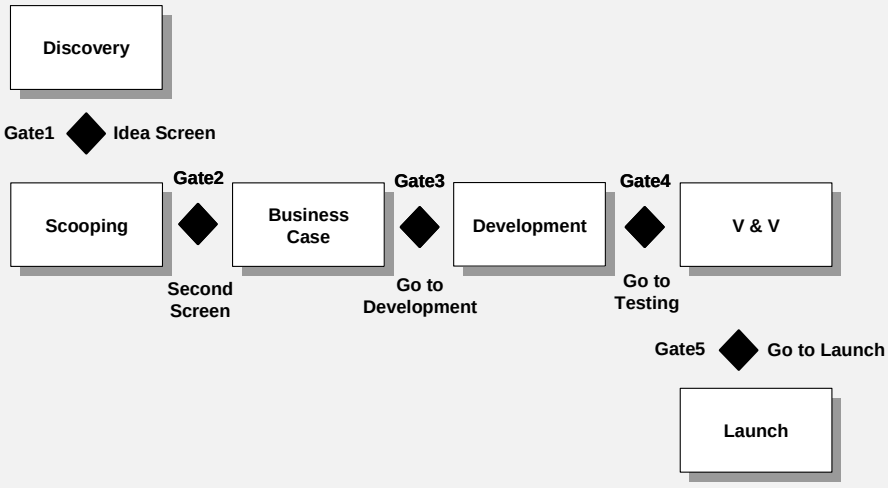

Figure A. Cooper's Stage-Gate Process Model.

Software development life-cycle models

Several SDLMs support software development projects. When correctly implemented, they help projects deliver products with expected quality and functionality, on time and within budget.

Most SDLMs divide the development life cycle into several phases, generally three to five. However, there are almost as many names for these phases as there are SDLMs (see Figure 1). Phase names typically indicate the main activity performed in that phase and do not distinguish the concerns of project management and software development. This article uses the life-cycle phases defined in Microsoft's Synch-and-Stabilize Life Cycle [3], the Unified Software Development Process [4], and Extreme Programming [5] as examples. These models are commonly known, and their life-cycle phase names cannot be confused with software development activities such as analysis, design, implementation, verification, and validation, as described in the traditional waterfall model. Moreover, these three approaches' phase names indicate the product's maturity rather than the development activities performed.

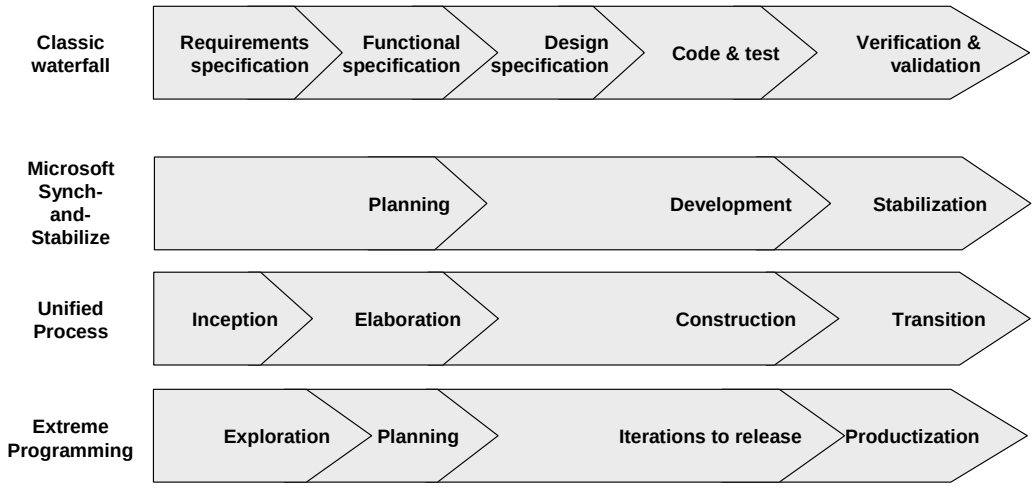

Figure 1: Phase names in four software development life-cycle models: waterfall, Synch-and-Stabilize, Unified Process, and Extreme Programming.

In most SDLMs, passing a major milestone marks the transition from one development phase to the next (see Figure 2). Of the three models just listed, only XP does not mention milestones. The Unified Process uses the three anchor-point milestones that Barry Boehm defined [6] (Life-Cycle Objectives, Life-Cycle Architecture, and Initial Operational Capability) to mark each phase's conclusion and the stakeholders' commitment to move ahead. The UP also adds a Product Release

milestone that concludes the Transition phase. Synch-and-Stabilize identifies three major milestones, each concluding a phase [3].

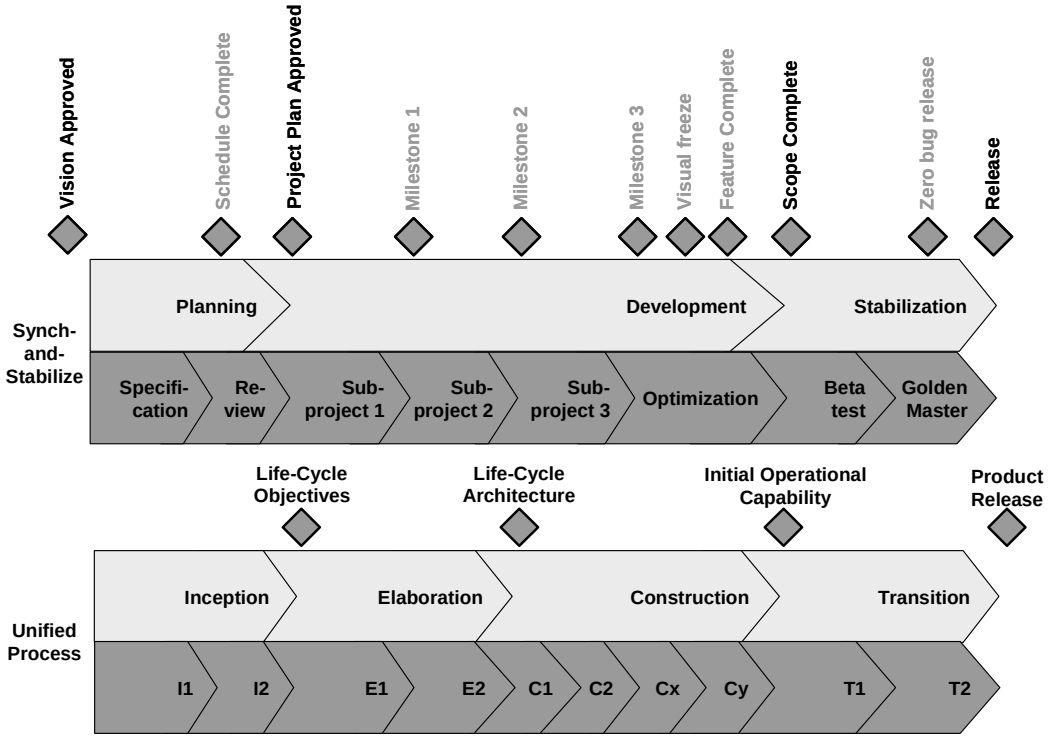

Figure 2: The Synch-and-Stabilize and Unified Process milestones.

Both UP and S&S also use minor milestones; in the UP, each iteration ends with a minor milestone, whereas S&S uses a number of predefined minor milestones concluding various sub-projects.

Mapping business decision models and SDLMs

A milestone is a scheduled event that marks the completion of one or more important tasks. The project manager uses milestones to measure and show achievements and development progress. At a milestone, a predefined set of deliverables should have reached a predefined state to enable a review. A gate, on the other hand, is a go-or-no-go decision point in the product development

cycle, where all relevant business facts are brought together [2]. At each gate, the decision maker uses the results from the preceding stage's activities together with a decision criteria checklist as input to the business decision.

Developers should not treat gates as software development milestones (see Figure 3), but they must pass some key milestones to be able to supply the decision maker with the required information in time before the gate. These important milestones could be called *pre-gate milestones*; they reflect the mapping between the business decision model and the SDLM. Of course, pre-gate milestones are not only in the software development plan but also, for example, in plans for marketing and competitor management, business, intellectual property management, training, customer service, quality assurance, hardware development, and so on. The project should pass all pre-gate milestones in all plans before the corresponding business decision at the gate.

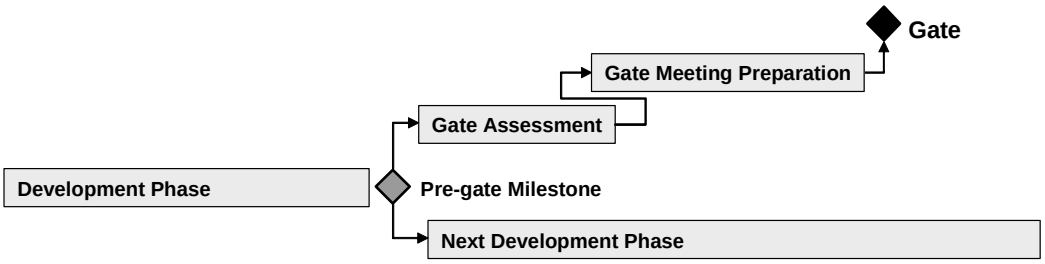

Figure 3: A pre-gate milestone's relation to a gate.

Mapping a business decision model's gates to an SDLM's major milestones is straightforward (see the examples in Figure 4). A go decision at the first gate is a prerequisite to start software development as well as all the other activities. At this point, we can start the project if we decide that the intended product is a strategic fit, attractive to the market, and technically feasible. We can then use major milestones in the software development life cycle as pre-gate milestones corresponding to the business decision gates. If the gates outnumber the major milestones, we must select suitable minor milestones as pre-gate milestones.

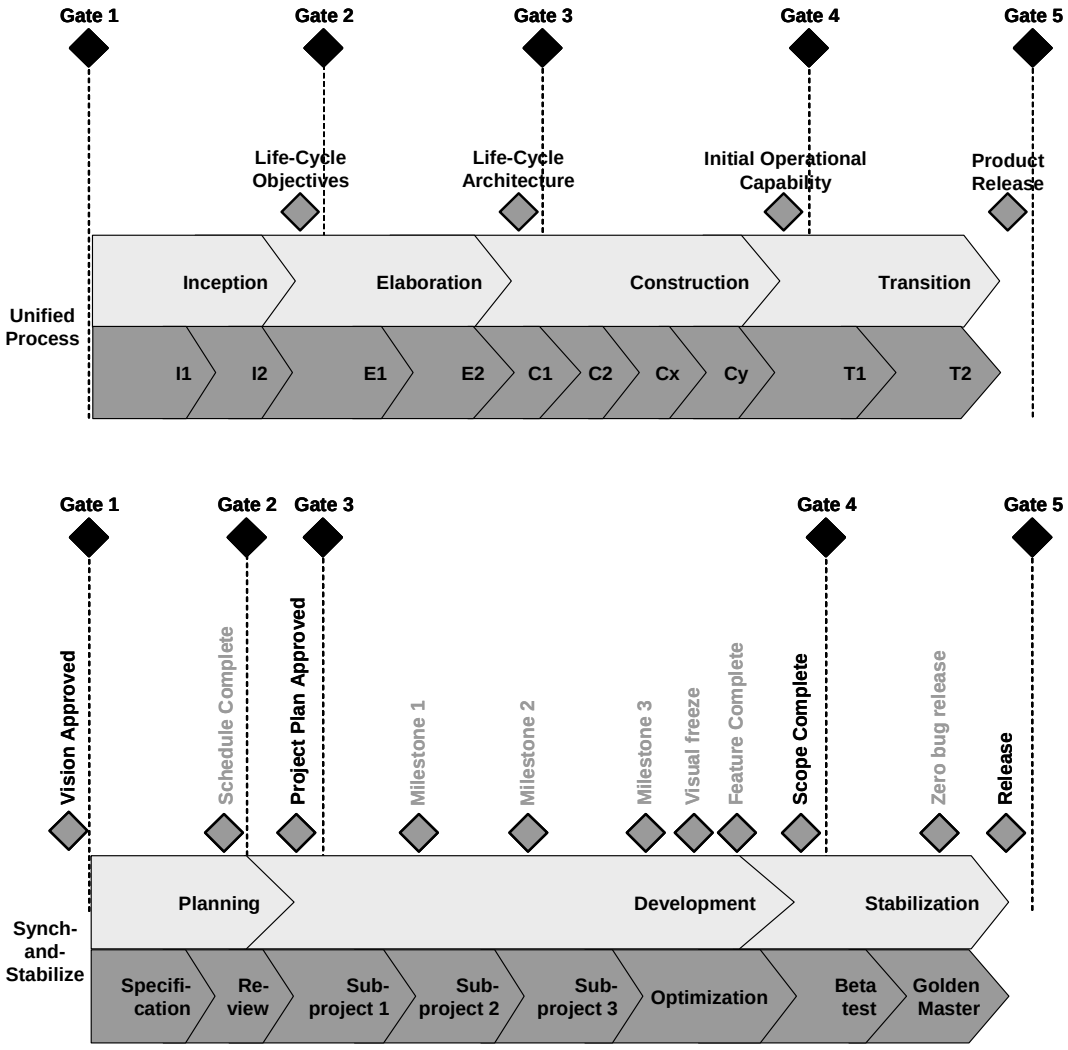

Figure 4: Comparing Cooper's Stage-Gate Model to the Unified Process and the Synch-and-Stabilize model.

Mapping SDLMs and the ABB Gate Model

To raise the quality of its product development business decisions, ABB developed the ABB Gate Model [7], a project control model reminiscent of Cooper's Stage-Gate. The ABB Gate Model consists of eight gates: gates 0 through 5 are true decision gates where the project can actually be canceled; gates 6 and 7 are used for follow-up and for a retrospective investigation of project experiences.

Mapping the UP major milestones and the ABB Gate Model gates is almost as straightforward as mapping to Cooper's Stage-Gate. It is only before ABB's Gate 3, Confirm Execution, that the UP is missing a pre-gate major milestone. Here, project management can choose a minor milestone indicating the finalization of an iteration or sub-phase [4] as a pre-gate milestone. Table 1 summarizes the requirements for the ABB Gate Model gates and for the UP's major milestones.

Table 1: Mapping the ABB Gate Model gates and the major Unified Process milestones.

ABB gate	Gate's purpose	UP's corresponding major milestone	Milestone's content
G0	Agree to start project	–	Project start
G1	Agree on project scope	Life-Cycle Objectives	Software's scope set
G2	Agree on requirements and project plan	Life-Cycle Architecture	Stable architecture and planned software development schedule, staff, and cost
G3	Confirm consensus regarding proposed technical solution	-	Minor milestone should be selected
G4	Agree on the product's readiness for piloting and market introduction	Initial Operational Capability	Software ready for beta testing
G5	Agree on release	Product Release	Software's formal release

Mapping the ABB Gate Model to XP resembles mapping to the UP but adds one complication. Because the time for planning in an XP project should be short, separating Gate 1 and Gate 2 is unnecessary (see Figure 5). (The recommended time for the planning phase in XP projects is about one week.) The proposed solution is to combine Gate 1 and Gate 2 and use the end of the planning phase as the point in time for a combined Gate1/Gate2.

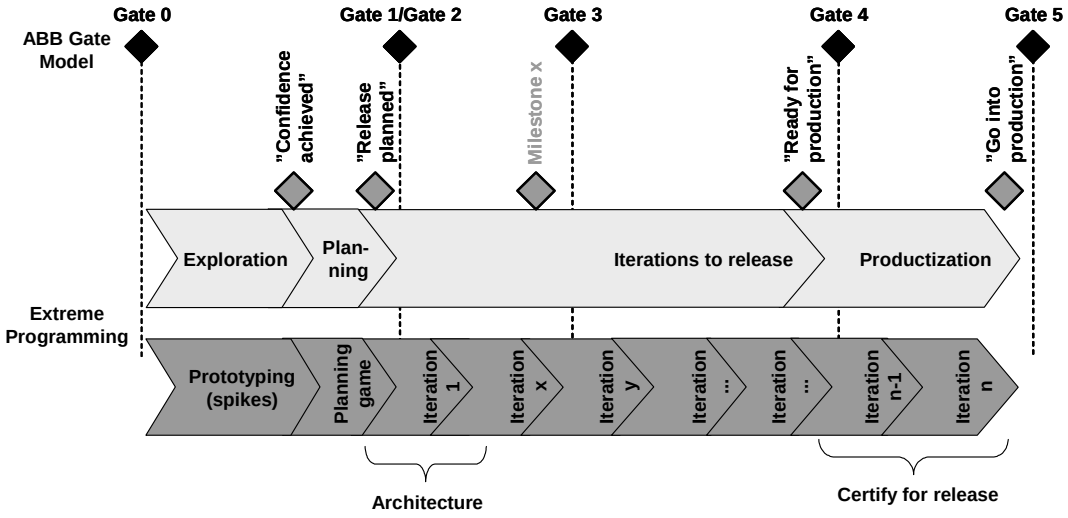

Figure 5: Mapping ABB Gate Model gates and Extreme Programming milestones.

When ABB first introduced a common decision model for product development, one of the developers' most common concerns was that adapting to the ABB Gate Model seemed to force the projects to use the waterfall development model. To clarify this issue, ABB made available to its developers all the mappings this article describes.

So far, the results are promising. Decision makers, project managers, and software engineers have reacted well to these mappings. Initial results show enhanced communication between the developers and the decision makers, increased focus on business aspects, and increased understanding of the differences between the models.

Current work focuses on making the mappings more widely known and used throughout ABB. By making these mappings available and broadly understood, ABB expects easier adaptation to future SDLMs, with new approaches to software development.

Acknowledgments

We recently presented a more detailed and theoretical version of this article at the 28th Euromicro Conference 2002. It is available in the proceedings, published by the IEEE Computer Society.

References

[1] J. Johnson et al., "Collaborating on Project Success," Software Magazine, Feb./Mar. 2001, www.softwaremag.com/archive/2001feb/CollaborativeMgt.html.

[2] R.G. Cooper, "Winning at New Products, 3rd ed"., Perseus Publishing, Cambridge, Mass., 2001.

[3] M.A. Cusumano et al., "Microsoft Secrets", Simon & Schuster, New York, 1998.

[4] I. Jacobson et al., "The Unified Software Development Process", Addison-Wesley, Boston, 1999.

[5] K. Beck, "Extreme Programming Explained", Addison-Wesley, Boston, 2000.

[6] B. Boehm, "Anchoring the Software Process," IEEE Software, vol. 13, no. 4, July 1996, pp. 73–82.

[7] ABB Gate Model for Product Development 1.1, tech. report 9AAD102113, ABB/GP-PMI, Västerås, Sweden, 2001.

APPENDIX C: THREE ASPECTS OF SUCCESSFUL SOFTWARE DEVELOPMENT PROJECTS

"WHEN ARE PROJECTS CANCELED, AND WHY?"

Christina Wallin & Ivica Crnkovic

Proceedings of Euromicro Conference, September 2003

Abstract:

Successful project execution, successful technical solutions or a promising business case, are they equally important selection criteria in a product development process? We have used experiences gained from a large multinational industrial company that is currently deploying a software product line strategy to try to answer that question. The product line's core assets include, among other things, a new software platform that is introduced to the company's software development organizations by means of a portfolio of targeted pilot projects. A business decision-making process is used to select and prioritize projects within the portfolio. This paper report findings from an analysis of a large number of projects and will indicate that the three criteria are not equally important.

1. Introduction

It has been suggested that a successful software development project is a project that is "completed on time and on budget, with all features and functions as originally specified" [10][7][8].

This definition of success is maybe enough for one-at-a-time custom software solution development, but for organizations aiming for some kind of software product line approach [9] another criterion of success has to be added. A successful software solution in a software product line could be defined as a solution that share "a common, managed set of features that satisfy the

specific needs of a particular market or mission and that are developed from a common set of core assets in a prescribed way." [3].

But, running a successful software development project with a successful software solution is in many cases still not enough. In commercial software product development the result, the software product, also has to be a success from a business perspective, i.e. it has to meet expected "financial criteria, return-on-investment (ROI), and market share" [4].

Business decision-making processes such as the Stage-Gate™ process [4], PROPS [6] and the Gate Model [11] recommend that criteria for all three above mentioned aspects of success (project execution, technical solution and business benefits) are evaluated for individual software product development projects periodically. Also agile software development methodologies address these aspects, although the evaluation is done on a more continuous basis. "Business people and developers must work together daily throughout the project." [1] But are all three aspects of success equally important, or even needed, for the software product development to be regarded as successful? The question is if successful software product development can be pictured as a stool with three equally important legs as in Figure 1?

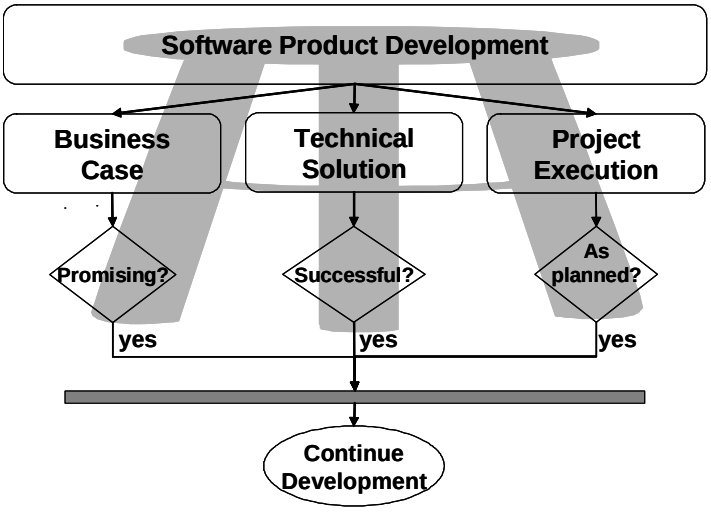

Figure 1: "The Software Product Development Stool"

If it is obvious that a resulting product will fail to meet expected business success criteria, is this a reason to cancel the project even if the technical solution and the project execution are successful? Lower than expected return-on-investment (ROI) and/or market shares may give lower profit and may require more than planned investments in e.g. marketing and sales.

Or, if it is obvious that e.g. the product line core assets cannot be used as expected for the solution, is this a reason to cancel the project when the business case is promising and project execution is successful? The usage of different technologies and technical solutions for different products within an organization may require more resources for development, support and maintenance than if a product line approach is used.

Finally, if it is obvious that a project will fail to deliver what is expected as planned, is this a reason for canceling the project when the business case is promising and the technical solution is successful? Typically resources like personnel and funds are limited. If a project is underestimated for some reason either the scope has to be decreased with the risk to disappoint the customer, resources have to be taken from other projects with the risk to delay them, or the delivery time has to be delayed. The later may have negative effects from a time-to-market perspective.

In many organizations canceling a project means a failure, or even a fiasco, as the resources and time spent will not result in achievement of expected results. Typically a primary goal of any development project is to successfully deliver expected results. This however, does not necessarily mean that the overall business goal will be achieved. By failing to cancel unpromising projects, living on the hope that problems can be overcome, risks increase for the failure of the overall business goal: (i) Resources are not available for allocation to more promising projects. (ii) Increased costs will make the initial business case obsolete. (iii) New projects are difficult to start until it is clear that ongoing projects will actually finish. This may discourage organizations to start new, experimental and risky projects, and in this way limit the creative forces in the organization.

This paper is based on experiences from a large multinational company that is currently moving from its traditional customer specific software solution strategy, to a software product line

approach. For that reason, major effort has been put into the development of core assets needed, including a new software platform. Their current step is to deploy the core assets among the software development organizations within the company, by means of a large number of targeted pilot solutions. The pilot development projects are managed in a project portfolio [5] with very precise criteria for success for business case, technical solution and project execution. The portfolio management uses a business decision-making process to make regular continue/cancel decisions for each individual pilot project. A repository of proposed, active, finished and canceled pilot projects is updated and revised continuously. Proposals for new pilots are evaluated and selected. Active projects are monitored, evaluated, supported or canceled. The repository contains information about each pilot such as current status, planned and actual dates and rational for business decisions and project documentation.

Our purpose of this paper is to present an investigation on how feasible is to cancel (or redirect) a project during its execution. Further, what could be the main criteria for canceling a project, from theoretical point of view, and even more important, what are the criteria from the experience.

The outline of the paper is as follows: Section 2 presents briefly the business decision-making process. Section 3 presents the product line deployment initiative, which provides the experiences. Section 4 presents the results of the research and section 5 analyses the findings and describes some conclusions and future work.

2. Business Decision-Making Process

The purpose of a business decision-making process such as Cooper's Stage-Gate Model™ [4], Ericsson's PROPS [6] and ABB's Gate Model [11], is to provide organizations with a procedure for better management of the organization's product development projects. See Figure 2.

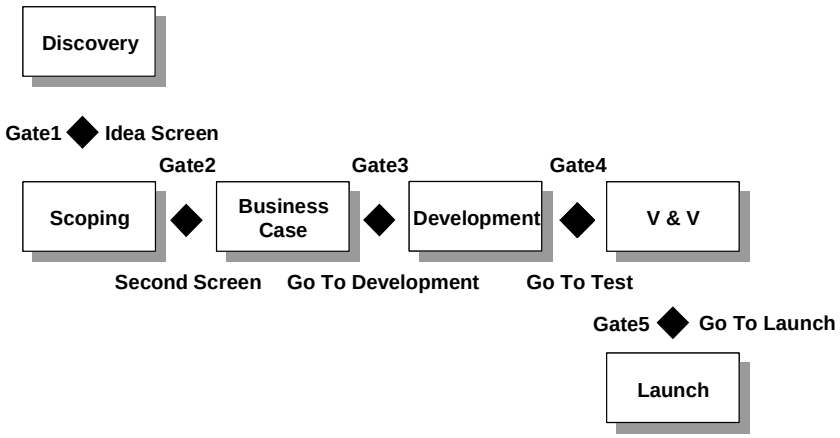

Figure 2: Cooper's Stage-Gate™ Business Decision-Making Model

If followed, such a procedure improves the possibility that projects are driven by business objectives. It provides defined management checkpoints, gates, where continue/cancel decisions regarding a project are made, based on correct and relevant information. A decision is based on an evaluation of the project to determine if it makes economic and strategic sense to proceed with the work. The procedures at each gate are similar; the results from the activities performed in the stage preceding the gate, together with a decision criteria checklist are used as input to the business decision. A decision to continue may of course include alterations to the project such as changed scope or plan. Typically at each gate project status, technical solutions and business issues are evaluated.

3. The Product Line Deployment Initiative

The company's ongoing product line deployment initiative is a corporate wide initiative including central funding and expert support. One goal with the initiative is to deploy the new software platform and other core assets within the company's different software product development organizations, i.e. to deploy the foundation for the company wide software product line. One major task in the initiative is to demonstrate the value of the new software platform across the company's businesses areas through a number of targeted pilot solutions. The pilot solutions can either be a

product prototype aimed for many customers or an individual solution developed for a single customer.

The initiative is performed by means of a managed project portfolio including a large number of pilot development projects, which are supposed to finish with a pilot product demonstrating the proposed solution, and plans for the transformation of the pilot to a commercial product.

One person is responsible for the overall management of the project portfolio. For each pilot project the portfolio manager cooperates with the actual research program manager (responsible for funding) and internal business partner representative (receiver of the result) in the business decision-making process.

3.1 The Platform Deployment Lifecycle

Each pilot solution is run through a platform deployment lifecycle of four stages; proposal, feasibility, pilot development project and productization (see Figure 3).

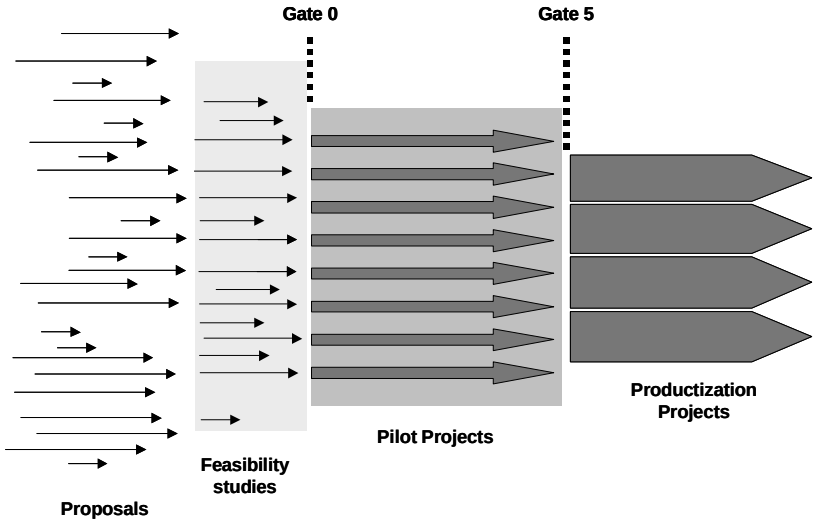

Figure 3: The Platform Deployment Lifecycle

3.1.1 The Proposal Stage

The Proposal stage starts with a collection of ideas for end customer solutions that can be enabled through the new software platform. For all ideas a solution proposal is developed. All proposals are evaluated according to:

- The possibility of commercial success – market need, market maturity, expected return, and access to a designated customer.

- The platform leverage – support of the solution architecture for the specific customer group.

- Project execution feasibility – are technology skills and resources available.

3.1.2 The Feasibility Study Stage

Promising proposals will be chosen for feasibility studies to compile the information needed for the responsible internal business partner management to make a decision to start a pilot development project. The feasibility study results in a report describing in greater detail the business case, market requirements, customer value, proposed solution architecture, competitor evaluation, and resource situation of this proposal.

3.1.3 The Pilot Development Stage

The Pilot Development stage is where the approved pilot development projects are run based on the feasibility study results. A business decision-making process is mandatory for all pilot projects. To improve project efficiency and to minimize the risk of failure due to inexperience, process and technology coaching is offered. An independent assessor, before each gate, makes formal project evaluation based on generic and project specific criteria for business success, solution success and project success. The evaluation is then used as input to the corresponding gate meeting which results in a continue or cancel decision for the actual pilot project.

In this case the business decision-making process has six gates (see Figure 4);

Gate 0 – Agree on feasibility study, start pilot project.

Gate 1 – Agree on pilot scope, start planning.

Gate 2 – Agree on project plan, start execution.

Gate 3 – Agree on technical solution, continue execution.

Gate 4 – Agree on final solution, start pilot installation and test at end-customer site.

Gate 5 – Agree on pilot release, handover solution to business partner.

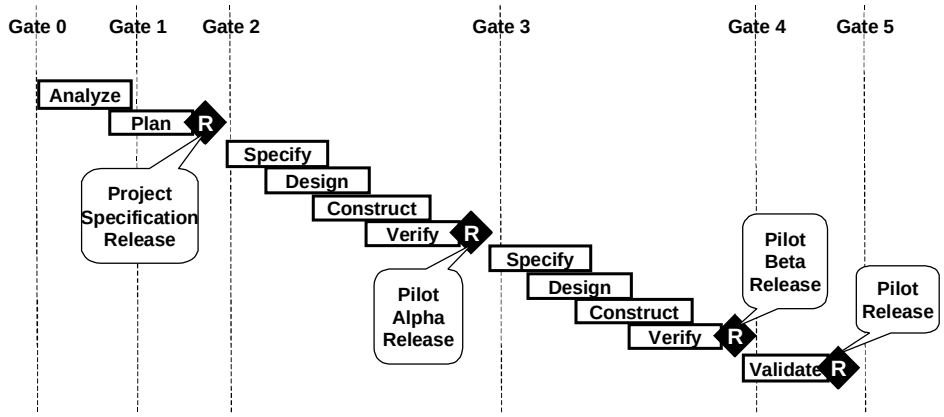

Figure 4: The Pilot Project Lifecycle

3.1.4 The Productization Stage

Based on the resulting pilot solution, the Productization stage can be entered. The pilot solution will be further developed by the responsible business partner to a commercial product complete with product documentation, marketing material, price, training, support etc.

3.2 Pilot Development Requirements

A set of specific business requirements is put on all pilot projects from the beginning;

- Business case: There has to be at least one external or internal end-customer committed to install and test the pilot solution, and there has to be an internal business partner committed to take over the pilot solution and develop it further to a commercial product.

- Technical solution: The new software platform has to be used for the solution.

- Project execution: The time-to-delivery (TTD) should not exceed 6 months and the staffing should consist of 4 – 8 persons.

The last requirement fits well with the recommendation in "CHAOS three pillars for project success" [10] that says "The smaller the team and shorter the duration of the project, the greater the likelihood of success." A project of this size should have about 50% chance of project execution success. This size of a project is also recommended by the agile community e.g. Extreme Programming [2].

4. Investigation

The aim of the investigation is to recognize a behavioral pattern of the pilot projects and the related business decisions. In particular of interest for us is an analysis of the cancel decisions. In which state the projects have been canceled, and what was the reason of their cancellation.

4.1 Information Sources

The main source of information about the pilot project portfolio and each pilot project is a database. In the database is project documentation such as the feasibility study report and project plan for each pilot project stored together with planned and actual dates for gates, gate assessment reports, gate decisions and minutes of the rational for the decisions. The database also contains bimonthly portfolio reports and results of a yearly questionnaire. Unfortunately information about effort spent and other costs are not stored. In addition to information in the database, executive summaries of the pilot projects are placed on the company's intranet.

4.2 Evaluation Database

Information from the source database has been collected and translated into an evaluation database. All ongoing feasibility studies and all finished, active and canceled pilot projects are recorded giving a total number of 82 projects.

Actual dates for gate passages and dates estimated in the feasibility study, planned in the project plan, and targeted at preceding gates are collected and recorded for each project. Time resolution is given in calendar weeks. The model distinguishes three types of time: estimated time that is specified at Gate 0, planned time, specified at Gate 2 (start of the development cycle), and actual time that is measured at Gate 5. Estimated, planned and actual Time To Delivery (TTD), i.e. the time between Gate 0 and Gate 5 (see Figure 5), and estimated, planned and actual Development Cycle Time (DCT), i.e. the time between Gate 2 and Gate 5 (see Figure 6) are calculated and recorded for each project. Delta times between estimated and actual (De) and between planned and actual (Dp) are calculated and recorded for each project as well.

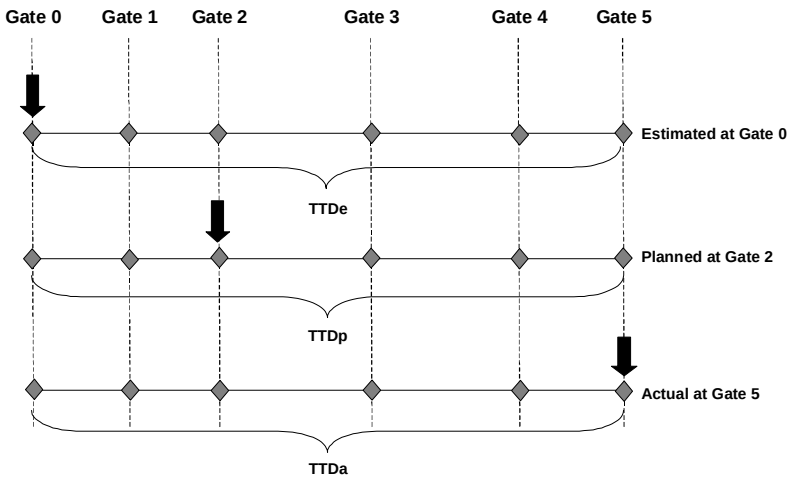

Figure 5: Estimated, planned and actual TTD

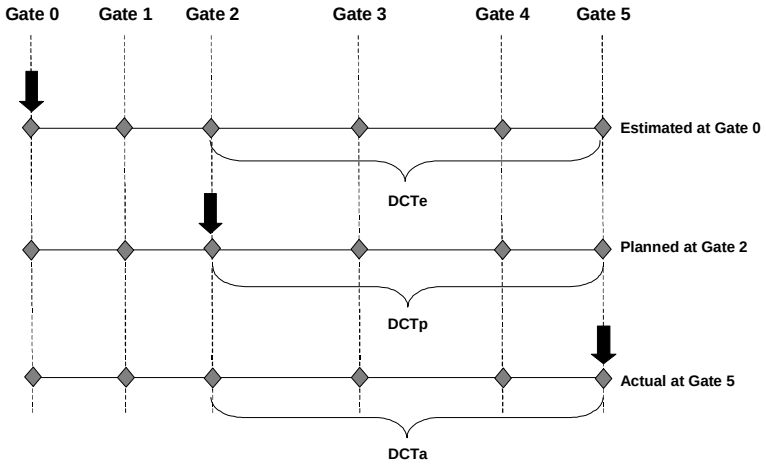

Figure 6: Estimated, planned and actual DCT

Nine projects are excluded from the portfolio evaluation due to incomplete source information leaving 73 projects for evaluation. Of these 73 projects are 8 (11%) finished, 29 (40%) ongoing and 28 (38%) canceled pilots. The rest 8 (11%) are ongoing feasibility studies (see Figure 7).

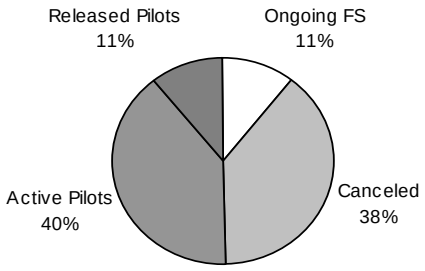

Figure 7: Distribution of pilot projects in the portfolio

5. Analysis

5.1 When are projects canceled, and why?

The main reason for canceling a project, 57% of the cases, is the lack of a promising business case. Either no end customer willing to test the pilot, or no internal business partner willing to take over the pilot for further development, is found. The second main reason, 25%, is insufficient feasibility study. There is not enough information to give the project a go to start. The applicability of the platform is the issue in 14% cases and project execution in 4% (see Table 1).

Table 1: Reasons and gates for canceling projects

	Gate 0	Gate 1	Gate 2	Gate 3	Gate 4	Gate 5	Total
Insufficient feasibility study	7	0	0	0	0	0	**7**
No committed end customer	5	4	0	1	0	0	**10**
No committed business partner	4	1	1	0	0	0	**6**
Platform not applicable	3	0	1	0	0	0	**4**
TTD exceeded	0	0	0	0	0	0	**0**
No available resources	0	1	0	0	0	0	**1**
Total	**19**	**6**	**2**	**1**	**0**	**0**	**28**

Projects are typically canceled early, the majority (68%) already at Gate 0 i.e. they are not even started, 21% at Gate 1, 7% at Gate 2 and 4% at Gate 3. No project that has passed Gate 3, i.e. an agreement on the technical solution is reached, is canceled. See Figure 8.

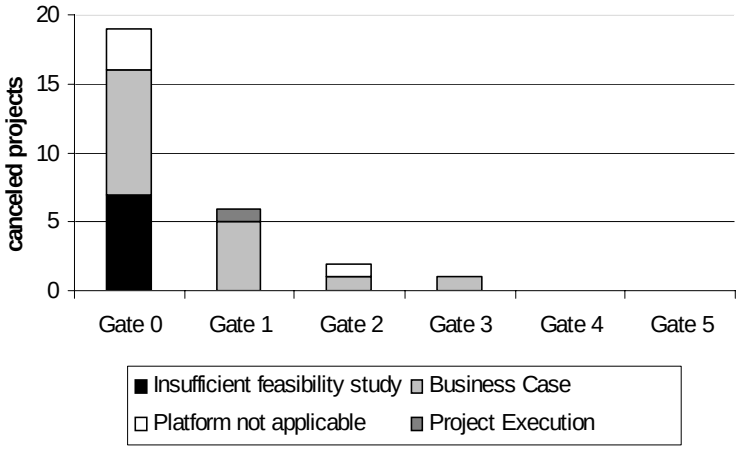

Figure 8: Projects canceled at each gate

Of the active and finished projects 73% already at Gate 0 have an estimated TTD (TTDe) that exceeds the stipulated TTD of max six months (26 weeks). The same amount, 73%, although not exactly the same projects have at Gate 2 a planned TTD (TTDp) that exceeds 6 months. Among the finished projects 88% have an actual TTD (TTDa) that exceeds 6 months. Of the projects still active, 64% should have passed Gate 5 (should have been finished) by now (week 310) according to their own plans.

The average TTD deviation against plan (TTDa – TTDp) is 37% (see Table 2 and Figure 9). The accuracy of the planning of TTD improves from Gate 0 to Gate 2. The average TTD deviation against estimation (TTDa – TTDe) is 83%. It is although interesting that the same improvement of the accuracy of DCT cannot be seen, the deviation from plan and the deviation from estimation is almost the same (see Table 2 and Figure 10).

Table 2: Estimated, planned and actual project times

	TTDe	**TTDp**	**TTDa**	**DCTe**	**DCTp**	**DCTa**	
Average (weeks)	33	40	54	23	22	32	**weeks**
Median (weeks)	32	38	55	23	21	32	**weeks**
Average deviation (actual-planned)			12			12	**weeks**
Deviation (%)			37			65	**%**
Average deviation (actual-estimated)			20			11	**weeks**
Deviation (%)			83			62	**%**

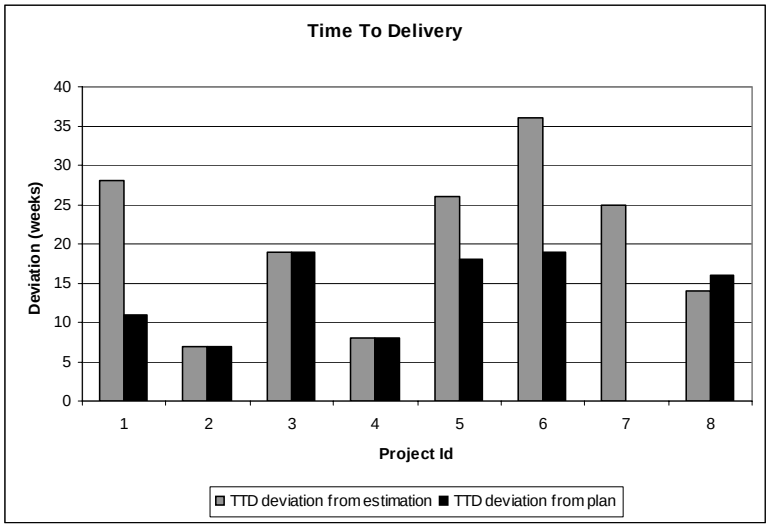

Figure 9: TTD deviation per project

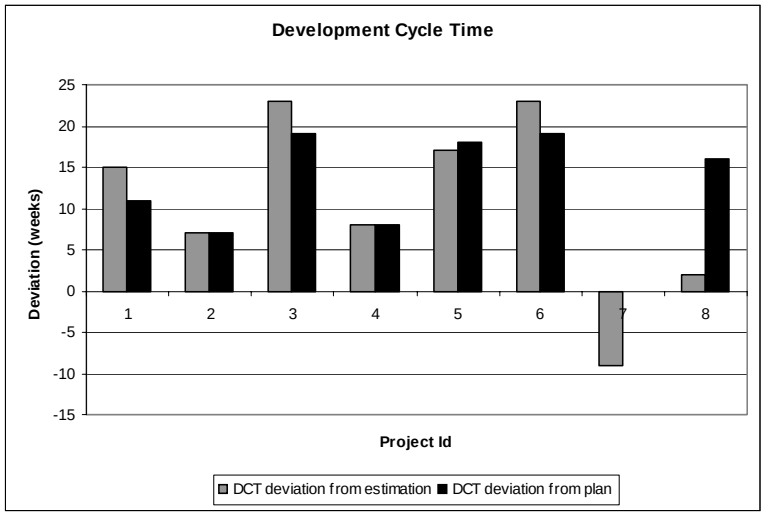

Figure 10: DCT deviation per project

5.3 Conclusions

Based on the findings in this study, successful development project execution is not considered as important as a promising business case and a successful technical solution when selecting and

prioritizing among projects in a project portfolio, although the extra cost for delayed projects can be as much as about half the costs saved by canceling projects without a promising business case as discussed below.

5.3.1 Business case

The specific requirements on the business case, to demonstrate market attractiveness, is that there is at least one end customer committed to install and test the pilot solution and one internal business partner committed to take over the project results. This requirement is validated thoroughly at each gate and if not fulfilled the project is canceled despite the possibility of a successful technical solution and successful project execution. Typically it is clear already in the feasibility study this requirement cannot be fulfilled, but in some cases the project gets an agreement to start at Gate 0 anyway and gets respite until Gate 1 to clarify the business case.

5.3.2 Technical solution

The specific technical solution requirement for each pilot project is that the new software platform should be used for the solution. If this requirement is not fulfilled, i.e. the software platform is not judged to be applicable for the suggested solution; the project is canceled from the portfolio. But, if the business case still is promising the project can be run anyway outside the portfolio, with other resources for funding and staffing, which is the case in one out of the four projects in this category.

5.3.3 Project execution

The specific requirement on project execution is that the result should be ready for delivery to the business partner, in less than 6 months (26 weeks) with a staffing of 4 – 8 persons. The TTD part of this requirement is not fulfilled at all in the active and finished projects. The estimated TTD exceeds 6 months in 73% of the cases and the average estimated TTD is 33 weeks (~8 months), but no project is canceled at Gate 0 for this reason. Also the planned TTD exceeds 6 months in 73% of the cases and the average planned TTD is 40 weeks (~10 months), but no project is canceled at Gate 2 either for this reason. Actual TTD exceeds 6 months in 88% of the finished

projects and the average actual TTD is 54 weeks (~13,5 months), more than twice the required. The staffing part of the requirement is not evaluated as only planned staffing, not actual, is reported in the source database.

Almost all finished pilots are delayed according to their own plans. Only one of the finished projects passed Gate 5 on plan. The average TTD deviation from plan (actual TTD – planned TTD) is 12 weeks. This gives an extra unplanned cost of approximately 12 person years assuming an average staffing of 6 persons in the 8 projects.

Among the active projects, 14 (64%) should have passed Gate 5 at the current point in time (week 310), but has not. Assuming they have the same average TTD deviation from plan and staffing as the already finished ones, at least another 23 person years extra cost is expected.

In this study neither unfulfilled project execution requirements (i.e. TTD less then 6 months) nor project delays according to their own plans are reasons to cancel projects, as long as the business case and technical solution are promising, i.e. project execution success is not considered equally important as the business case and the technical solution.

5.3.4 Business decision-making process

Using the business decision-making process, 25% of the pilot projects could be canceled already at the first screening at Gate 0, mainly due to the lack of a promising business case or an inadequate feasibility study. The saved cost for not running these projects could be calculated to approximately 57 person years (assuming 6 persons in 6 months for 19 projects). Adding the other 9 projects canceled at later gates gives an additional cost saving of approximately 20 person years more.

About 50% of the canceled projects can be re-opened if the business situation changes, i.e. the suggested technical solution and project execution are judged as feasible. (So far this has not happened.) Without a business decision-making process at least these projects would typically been

run much further and in the worst case the lack of a receiver of the project results would not have been recognized until the project was finished.

Another possible advantage of a business decision-making process, if used to manage a portfolio of projects as in this case, is the possibility to prioritize among the active projects and refocus resources to the most promising, important or time critical ones. No evidence that the business decision-making process is used in this way can be identified in this study. One explanation to this can be that the portfolio manager is not directly responsible for the funding of the projects.

5.4 Future work

The platform deployment initiative is still ongoing and new projects enter, and pass, through the platform deployment lifecycle. As more and more projects finish and pass Gate 5, deviation data will become more accurate and the study can be extended to investigate also why projects are delayed and when. Another interesting topic to investigate is if planning accuracy will improve over time. Are projects started late in the initiative better planned than the fist ones, the ones that are finished now? A third interesting topic is if the reasons for canceling projects will change over time. Will e.g. resource optimization be a reason for canceling projects eventually?

Finally it would be of interest to know if our findings are valid only for this particular project portfolio, or if the same conclusion can be derived for other companies as well. In our case, the projects analyzed have been performed in several different countries in Europe and US. As the company is a multinational organization, traditionally decentralized, with different local cultures, allowing large differences in software development practices, we believe that the results found are not specific for this initiative in particular, and definitely not specific for a particular company or country.

6. References

[1] Beck Kent, Et al, Principles behind the Agile Manifesto, http://agilemanifesto.org/, Mars 2003

[2] K. Beck, Extreme Programming Explained, Addison-Wesley, 2000.

[3] Clements Paul, Northorp Linda, Software Product Lines: Practices and Patterns, Addison-Wesley, 2001

[4] Cooper Robert G., Winning at new Products, Third Edition, ISBN 0-7382-0463-3, Perseus Publishing, 2001

[5] Cooper R., Edgett S., Kleinschmidt E. Portfolio management for new product development: results of an industry practices study, R&D Management, October 2001, Vol. 31 No. 4, pp. 361-380

[6] Ericsson, PROPS Manual for Project Sponsors, ISBN 91-89438-15-9, Ericsson Business Consulting, 2000

[7] Johnson Jim, Et. Al., Collaborating on Project Success, Software Magazine, February/March 2001

[8] Lawrence Brian, Hall Payson, The Problem of Project Management, Cutter IT journal Vol. 12 No 5, May 1999

[9] Software Engineering Institute, The Product Line Practice (PLP) Initiative, http://www.sei.cmu.edu/plp/plp_init.html, March 2003

[10] The Standish Group International Inc., CHAOS: A Recipe for Success, 1999

[11] Wallin, Larsson, Ekdahl, Crnkovic, Combining Models for Business Decisions and Software Development, Euromicro 2002