The Journal of Systems and Software 83 (2010) 1387–1403

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

# Architecture analysis of enterprise systems modifiability – Models, analysis, and validation

Robert Lagerström *, Pontus Johnson, David Höök

*Industrial Information and Control Systems, The Royal Institute of Technology, Osquldas väg 12, 100 44 Stockholm, Sweden*

ARTICLE INFO

*Article history:*
Received 4 September 2009
Received in revised form 11 January 2010
Accepted 14 February 2010
Available online 4 March 2010

*Keywords:*
Enterprise architecture
Software change cost estimation
Software modifiability
Metamodel
Probabilistic relational models

ABSTRACT

Enterprise architecture (EA) models can be used in order to increase the general understanding of enterprise systems and to perform various kinds of analysis. This paper presents instantiated architectural models based on a metamodel for enterprise systems modifiability analysis, i.e. for assessing the cost of making changes to enterprise-wide systems. The instantiated architectural models detailed are based on 21 software change projects conducted at four large Nordic companies. Probabilistic relational models (PRMs) are used for formalizing the EA analysis approach. PRMs enable the combination of regular entity-relationship modeling aspects with means to perform enterprise architecture analysis under uncertainty. The modifiability metamodel employed in the analysis is validated with survey and workshop data (in total 110 experts were surveyed) and with the data collected in the 21 software change projects. Validation indicates that the modifiability metamodel contains the appropriate set of elements. It also indicates that the metamodel produces estimates within a 75% accuracy in 87% of the time and has a mean accuracy of 88% (when considering projects of 2000 man-hours or more).

## 1. Introduction

Managing software systems today is a complex business. In order to achieve effective and efficient management of the software system landscape it is essential to be able to assess the current status of system qualities such as availability, performance, security, and modifiability, as well as estimate their status in different future scenarios. Estimation of these qualities is however a great challenge that to a large extent can be addressed by introducing models as a means of abstraction. This is achieved with enterprise architecture modeling.

The purpose of this paper is to: (1) present instantiated architectural models based on the metamodel supporting analysis of systems modifiability, i.e. estimation of enterprise software system change cost. The created models are instantiations of 21 software change projects studied in multiple case studies conducted at four large Nordic companies, (2) discuss the validity of the modifiability metamodel elements by studying data collected in workshops and surveys (in total 110 experts were surveyed) and (3) discuss the estimation capabilities of the metamodel by studying the actual and estimated costs of the 21 software change projects.

The contribution of this paper is the instantiation and validation of the modifiability metamodel. (Lagerström et al., submitted for publication) thoroughly describes this metamodel and (Lagerström et al., 2009a) details the creation method which it is based on. Thus, the research work conducted is divided into three papers with different focus and contributions.

### 1.1. Enterprise architecture

In recent years, *Enterprise Architecture* (EA) has become an established discipline for business and software system management (Ross et al., 2006). EA describes the fundamental artifacts of business and IT as well as their interrelationships (Zachman, 1987; Lankhorst, 2005; Ross et al., 2006; Winter and Fischer, 2007; The Open Group, 2009). Architecture models constitute the core of the approach and serves the purpose of making the complexities of the real world understandable and manageable (Winter and Fischer, 2007). EA ideally aids the stakeholders of the enterprise to effectively plan, design, document, and communicate IT and business related issues, i.e. they provide decision support for the stakeholders (Kurpjuweit and Winter, 2007).

In relation to supporting decisions, a key underlying assumption of EA models is that they should provide some more aggregated knowledge than what was merely put into the model in the first place. Software application architecture, for instance, does not only keep track of the set of systems in an enterprise and their internal relationships, it also provides information about the dependencies between the systems. More broadly, the dependencies between the business and the software systems are covered in an EA. Conclusions can therefore for instance be drawn about the consequences in the enterprise given that one specific system is unavailable.

* Corresponding author. Tel.: +46 8 790 68 66; fax: +46 8 790 68 39.
*E-mail address:* robertl@ics.kth.se (R. Lagerström).

0164-1212/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2010.02.019

Enabling this type of analysis is extremely important for providing value of EA to its stakeholders. Unfortunately, EA frameworks rarely explicitly state neither what kinds of analysis that can be performed given a certain model nor the details on how the analysis should be performed (Johnson et al., 2007; Franke et al., 2009). Another permeating problem in EA modeling is the uncertainty that is related to the development of the model. For instance, if a model is the result of a very thorough and recent investigation or a quick read-through of old documents, the resulting quality of the model will differ. In turn this will impact the quality of the decision support the model offers for its various stakeholders. Are all the software systems in the model still in use, is the data flow still as depicted, and is the process structure illustrating what is really happening? This kind of uncertainty is not addressed by todays EA frameworks. Again the user of the models is simply left to her subjective best knowledge or gut feeling when estimating to what extent the EA model, and the analyses based on it, can and should be trusted.

This paper employs a formalized approach to enable analysis of EA models. The approach also copes with empirical uncertainties by not considering the information in EA models as fixed but rather as probabilities. The underlying fundamental formalism in the approach is called *Probabilistic Relational Models* (PRMs) (Friedman et al., 1999), which in turn employs statistical mathematics of Bayesian networks (Jensen, 2001; Neapolitan, 2003).

### 1.2. Enterprise system modifiability

As discussed in the previous subsection, enterprise architecture models can be used to analyze different system qualities and provide information for the decision maker regarding different scenarios. In this paper the focus will be on enterprise software system modifiability, i.e. the cost of making changes to enterprise-wide software systems. Cost is here defined as man-hours spent on a change project.

Business environments today need to progress and change rapidly to keep up with evolving markets. Most business processes are supported by software systems and as the business processes change, the systems has to be modified in order to continue supporting the processes. Modifications include extending, deleting, adapting, and restructuring the enterprise software systems (Bass et al., 1988). Modification efforts can range from adding a functional requirement in a single system to implementing a service oriented architecture for a complete enterprise.

An essential issue with today's software systems is that many of them are interconnected, thus a modification to one system may cause a ripple effect among other systems. It is also common that the systems have been developed and modified during many years. Making further changes to these systems might require a lot of effort from the organization, for example due to a large amount of previous modifications implemented ad hoc. Problems like these raise questions for IT decision makers such as: Is there enough documentation describing the system? Has the documentation been updated correctly after each modification? Is the source code easy to understand? Or, which systems are interconnected?

According to IEEE Standards Board (1990), maintenance and maintainability can be interchangeably used with the terms modification and modifiability. Maintenance is in IEEE Standards Board (1990) defined as the ease with which a software system can be modified, maintainability is defined as the process of modifying a software system or component. With this confirmed we can lean towards and enlight several studies indicating that modification work is the phase in a software system's lifecycle that consumes the greatest portion of resources: Harrison and Cook (1990) report that over 70% of the software budget is spent on maintenance, Pigoski (1997) refers to studies stating that the maintenance cost, relative to the total life cycle cost of a software system, has been increasing from 40% in the early 1970s up to 90% in the early 1990s, and Jarzabek (2007) states that "the cost of maintenance, rather than dropping, is on the increase".

The activities of modifying enterprise systems are typically executed in projects, and IT decision makers often find it difficult to estimate and plan their change projects. Thus, a large proportion of the projects aiming to modify a software system environment fail, i.e. the projects tend to take longer time and cost more than expected. Laird and Brennan (2006) declare that 23% of all software projects are cancelled before completion, whereas of those completed only 28% were delivered on time, and the average software project exceeded its budget by 45%. This can often occur due to lack of information about the systems being changed. According to Laird and Brennan (2006), software engineers must be able to understand their activities, as well as manage the risks, through estimation and measurement. Therefore, it would be useful for IT decision makers to gather more information in a structured manner and use this information to analyze how much effort a certain modification of an enterprise software system would require.

This paper will address these issues of software change by employing enterprise architecture modeling. Instantiated architectural models for estimation of software change project cost, i.e. for systems modifiability analysis, will be presented.

### 1.3. Enterprise architecture for modifiability analysis

As stated in Section 1.1, the exact procedure or algorithm for how to perform a certain analysis given an architecture model is very seldom provided by EA frameworks. Most frameworks do however recognize the need of providing special purpose models and provide different viewpoints intended for different stakeholders. Unfortunately, most viewpoints are designed from a model entity point of view, rather than a stakeholder concern point of view. Thus, assessing a quality such as the modifiability of a system is not something that is performed in a straight forward manner. The Department of Defense Architecture Framework (DoDAF) for instance provides products (i.e. viewpoints) such as *systems communications description*, *systems data exchange matrix*, and *operational activity model* (Department of Defense Architecture Framework Working Group, 2007). These are all viewpoints based on a delimitation of elements of a complete metamodel, and they are not explicitly connected to a certain stakeholder or purpose.

The Zachman framework connects model types describing different aspects (*Data*, *Function*, *Network*, *People*, *Time*, and *Motivation*) with very abstractly described stakeholders (*Strategists*, *Executive leaders*, *Architects*, *Engineers*, and *Technicians*) (Zachman, 1987, 2009), but does not provide any deeper insight how different models should be used. The Open Group Architecture Framework (TOGAF) is explicitly stating stakeholders and concerns for each viewpoint they are suggesting (The Open Group, 2009). However, neither the exact metamodel nor the mechanism for analyzing the stated concerns, are described.

In relation to modifiability, the most appropriate viewpoints provided by TOGAF would arguably be *the Software Engineering View*, *the Systems Engineering View*, *the Communications Engineering View*, and *the Enterprise Manageability View*. In the descriptions of these views one can find statements such as; "the use of standard and self-describing languages, e.g. XML, are good in order to achieve easy to maintain interface descriptions". However, the exact interpretation of such statements when it comes to architectural models or how it relates to the modifiability of a system as a whole, is left out. Moreover, these kinds of "micro theories" are only exemplary and do not claim to provide a composed theory for modifiability or similar concerns.

Other, more formalized analysis mechanisms for enterprise architecture models, may be found in e.g. (Lankhorst, 2005) for performance and availability, for software architecture languages in Allen (1997) where for instance dead-lock and interoperability analyses are provided, Architecture Analysis and Design Language (AADL) (Society of Automotive Engineers, 2009) provides availability, security, and timeliness analyses, and UMLsec (Jürjens, 2005) provides security analysis. None, however, offer architecture models for software system modifiability analysis.

Since there are no EA frameworks or metamodels focusing on modifiability analysis available the present paper aims at filling this gap in enterprise architecture.

### 1.4. Outline

The remainder of the paper is structured as follows: Section 2 presents the probabilistic relational models which serve as the underlying formalism for the enterprise architecture models presented. The subsequent section describes the method employed for designing the modifiability metamodel, i.e. metamodel for enterprise software system change cost estimation. Next, in Section 4 the modifiability metamodel is presented. The qualitative part of the metamodel is based on existing academic literature and the quantitative part of information gathered in expert surveys and workshops. Section 5 contains information regarding the modeling approach for instantiating the architectural models. A set of instantiated architectural models for software change project cost based on four multiple case studies are described in Section 6 in order to illustrate the applicability of the EA analysis approach. In Section 7 the metamodel for modifiability is validated by considering the correctness of the qualitative structure and the estimation capabilities of the quantitative structure. The qualitative structure is validated with survey and workshop data from 110 experts. Actual costs and estimated costs from 21 software change projects are compared in order to validate the quantitative structure. COCOMO, function points, and planning poker are other available estimation methods. Section 8 compares the estimation capabilities of the modifiability metamodel with the capabilities of these methods. Discussion and research outlook is presented in Section 9. Finally, Section 10 summarizes the paper with conclusions.

## 2. Probabilistic relational models

As stated in the introduction *Probabilistic Relational Models* (PRMs) serve as the underlying formalism for the enterprise architecture models described in this paper. Previously proposed formalisms such as the *Extended influence diagrams* are presented in Johnson et al. (2007) and Lagerström (2007).

A PRM (Friedman et al., 1999) specifies a template for a probability distribution over an architecture model. The template describes the metamodel for the architecture model, and the probabilistic dependencies between attributes of the architecture objects. A metamodel acts as a pattern for the instantiation of models. In other words, a metamodel is a description language used when creating models (Lankhorst, 2005; Johnson and Ekstedt, 2007; Kurpjuweit and Winter, 2007; The Open Group, 2009). A PRM, together with an instantiated architecture model of specific objects and relations, defines a probability distribution over the attributes of the objects. The probability distribution can be used to infer the values of unknown attributes, given evidence of the values of a set of known attributes.

An architecture *metamodel* $\mathscr{M}$ describes a set of *classes*, $\mathscr{X} = X_1, \ldots, X_n$. Each class is associated with a set of *descriptive attributes* and a set of *reference slots*. The set of descriptive attributes of a class $X$ is denoted $\mathscr{A}(X)$. Attribute $A$ of class $X$ is denoted $X \cdot A$ and its domain of *values* is denoted $V(X \cdot A)$. For example, a class *DeveloperTeam* might have the descriptive attribute *Expertise*, with domain $\{High, medium, low\}$. The set of reference slots of a class $X$ is denoted $\mathscr{R}(X)$. We use $X \cdot \rho$ to denote the reference slot $\rho$ of class $X$. Each reference slot $\rho$ is typed with the *domain type* $Dom[\rho] = X$ and the *range type* $Range[\rho] = Y$, where $X; Y \in \mathscr{X}$. A slot $\rho$ denotes a relation from $X$ to $Y$ in a similar way as Entity-Relationship diagrams. For example, we might have a class *DeveloperTeam* with the reference slot *IsAResourceOf* whose range is the class *ChangeOrganization*.

An architecture *instantiation* $\mathscr{I}$ (or an architecture model) specifies the set of objects of each class, the values for the attributes, and the references of the objects. For example, Fig. 6 presents an instantiation of the change project metamodel of Fig. 4. It specifies a particular set of changes, systems, documents, etc., along with their attribute values and references. For future use, we also define a *relational skeleton* $\sigma_r$ as a partial instantiation which specifies the set of objects in all classes as well as all the reference slot values, but not the attribute values.

A probabilistic relational model $\Pi$ specifies a probability distribution over all instantiations $\mathscr{I}$ of the metamodel $\mathscr{M}$. This probability distribution is specified similar to a Bayesian network (Jensen, 2001), which consists of a qualitative dependency structure and associated quantitative parameters.

The *qualitative* dependency structure is defined by associating with each attribute $X \cdot A$ a set of parents $Pa(X \cdot A)$. Each parent of $X \cdot A$ is defined as $X \cdot \tau \cdot B$ where $B \in \mathscr{A}(X \cdot \tau)$ and $\tau$ is either empty, a single slot $\rho$ or a sequence of slots $\rho_1, \ldots, \rho_k$ such that for all $i, Range[\rho_i] = Dom[\rho_{i+1}]$. For example, the attribute *Cost* of class *ChangeProject* may have *ArchChangeActivity.System.Understandability* as parent, thus indicating that the cost of a prospective software modification project depends on the understandability of the systems changed in the architecture. Note that $X \cdot \tau \cdot B$ may reference a set of attributes rather than a single one. In these cases, we let $A$ depend probabilistically on some aggregate property over those attributes, such as the logical operations *AND*, *OR*, and *NOR*. In this paper we use the arithmetic operations *SUM* and *MEAN* as *aggregate* functions. For instance, if there are several systems changed in a modification project, we might aggregate each systems' understandability into a mean understandability of the whole architecture.

Considering the *quantitative* part of PRMs, given a set of parents for an attribute, we can define a local probability model by associating a *conditional probability distribution* (CPD) with the attribute, $P(X \cdot A | Pa(X \cdot A))$.

We can now define a PRM $\Pi$ for a metamodel $\mathscr{M}$ as follows. For each class $X \in \mathscr{X}$ and each descriptive attribute $A \in \mathscr{A}(X)$, we have a set of *parents* $Pa(X \cdot A)$, and a CPD that represents $P_{\Pi}(X \cdot A | Pa(X \cdot A))$.

Given a relational skeleton $\sigma_r$ (i.e. a metamodel instantiated to all but the attribute values), a PRM $\Pi$ specifies a probability distribution over a set of instantiations $\mathscr{I}$ consistent with $\sigma_r$:

$$P(\mathscr{I} | \sigma_r, \Pi) = \prod_{x \in \sigma_r(X)} \prod_{A \in \mathscr{A}(x)} P(x \cdot A | Pa(x \cdot A))$$

where $\sigma_r(X)$ are the objects of each class as specified by the relational skeleton $\sigma_r$.

A PRM thus constitutes the formal machinery for calculating the probabilities of various architecture instantiations. This allows us to infer the probability that a certain attribute assumes a specific value, given some (possibly incomplete) evidence of the rest of the architecture instantiation. In addition to express and infer uncertainty about attribute values as specified above, PRMs also provide support for specifying uncertainty about the structure of the instantiations. A more detailed description of this topic is, however, beyond the scope of the paper.

## 3. Creating the modifiability metamodel

Creating a good enterprise architecture metamodel is not trivial. Obviously, it is important that the metamodel is tailored for the management tasks it should support, i.e. what kind of analysis the metamodel will be subjected to. For instance, if one seeks to employ an enterprise architecture model for evaluating IT/business alignment, the information required from the model differs radically from the case when the model is used to evaluate the modifiability of an enterprise software system. This section aims at presenting the method employed when the modifiability metamodel was designed. The method focuses on creating a special kind of metamodel called probabilistic relational model (PRM). As described in Section 2, PRMs contain attributes which are causally related. These are used for analysis and estimation with probabilities, while regular entity-relationship models and UML-class diagrams do not. For the interested reader a more detailed description of the method is presented in Lagerström et al. (2009a).

### *3.1. Method for creating the qualitative part of the metamodel*

This subsection presents the method for creating the qualitative part of the modifiability metamodel, i.e. the classes, reference slots, attributes and their parents.

The method employed for creating metamodels for decision support is of an iterative nature and focuses on finding a set of appropriate *a priori* measures for a chosen goal, i.e. finding measures with high correlation with and causal influence on the selected goal. This can be done in several ways, for instance by studying research literature, conducting experiments, doing case studies, or using expert opinions.

The first step in the method is to select the goal that the metamodel under design is supposed to support. In this paper the goal considered is modifiability, i.e. change cost. Then, variables affecting the goal, and variables affecting previously identified variables are identified and causally related to each other. This iterative process continues until all paths of variables, and causal relations between them, have been broken down into variables that are directly controllable by the decision maker, see part 1 and 2 of Fig. 1. The process of finding variables affecting modifiability is supported by knowledge elicitation guidelines and control steps. These have been described in Johnson et al. (2007), and Lagerström et al. (2007, 2009a). The result of the iterative process is a goal break-down. The goal break-down is based on scientific knowledge, exhibiting variables that are controllable by the decision maker and causally linked to a well-defined goal, e.g. as in part 3 of Fig. 1. The next step is to translate these into metamodel classes, attributes, and reference slots. The attributes of the metamodel classes correspond to the variables found in the goal break-down part of the method. The goal break-down example presented in part 3 of Fig. 1 corresponds to the metamodel example presented in part 4 of the same figure.

By using the guidelines for metamodel design, the qualitative part of a modifiability analysis metamodel was finally obtained. This qualitative part is based on existing academic literature. Sources were chosen based on how well they fit the research topic (modifiability, maintainability or change cost analysis) and that they are written in an academic way (either peer-reviewed papers or well-cited books). Figs. 2 and 3 present two views of the modifiability metamodel. A view contains classes, reference slots, attributes, causal dependencies, and aggregating functions. The metamodel view illustrated in Fig. 4 contains the main classes and attributes, as well as the causal structure of the metamodel. However, this view neither presents the aggregation functions nor the multiplicities.

### *3.2. Method for creating the quantitative part of the metamodel*

This subsection presents the second part of the metamodel creation method, defining the quantitative part of the metamodel. That is, to define the conditional probability distributions (CPDs) related to each attribute.

The metamodel creation method employs a knowledge elicitation approach previously published in Lagerström et al. (2009a,b). This approach takes expert opinions into consideration when defining the conditional probability distributions without the need of introducing the experts to the concepts of conditional probabilities and PRMs. The algorithms used for defining the CPDs in the metamodel are based on the effect one attribute $x$ has on a related attribute $y$. In the modifiability case the effect was found by using a questionnaire among experts, during both workshops and in an online survey. The effect is measured on an ordinal scale with three states *High effect*, *Low effect*, and *No effect*.

The resulting CPDs will look like the following example between five attributes and their joint effect on *Component change activities cost*, cf. Table 1.

As described in Section 2 there are some attributes that are related as aggregates rather than by causality. In these cases, the probabilistic dependency is on some aggregate property over those

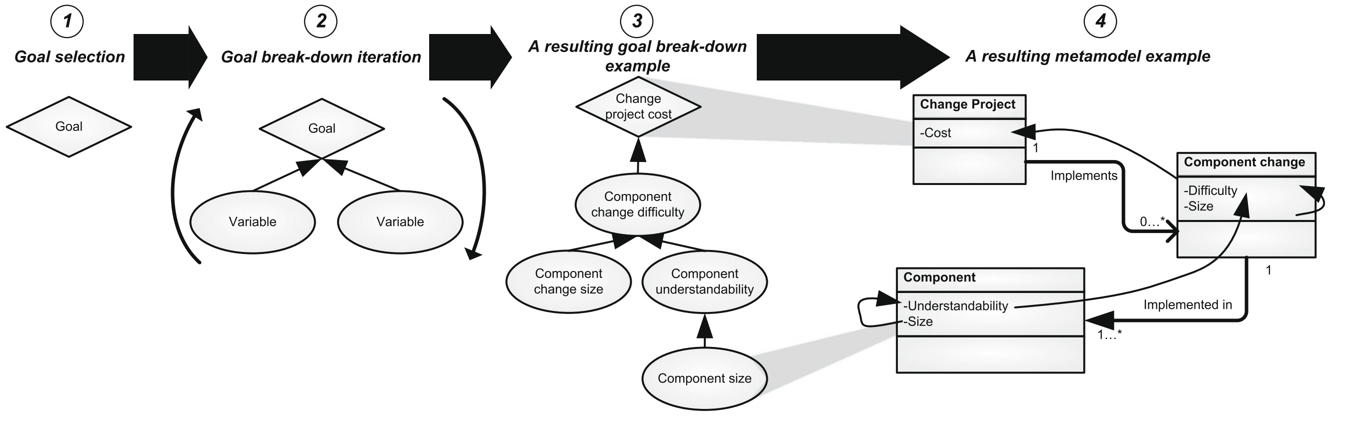

**Fig. 1.** The iterative goal break-down process, a resulting goal break-down example, and the corresponding metamodel.

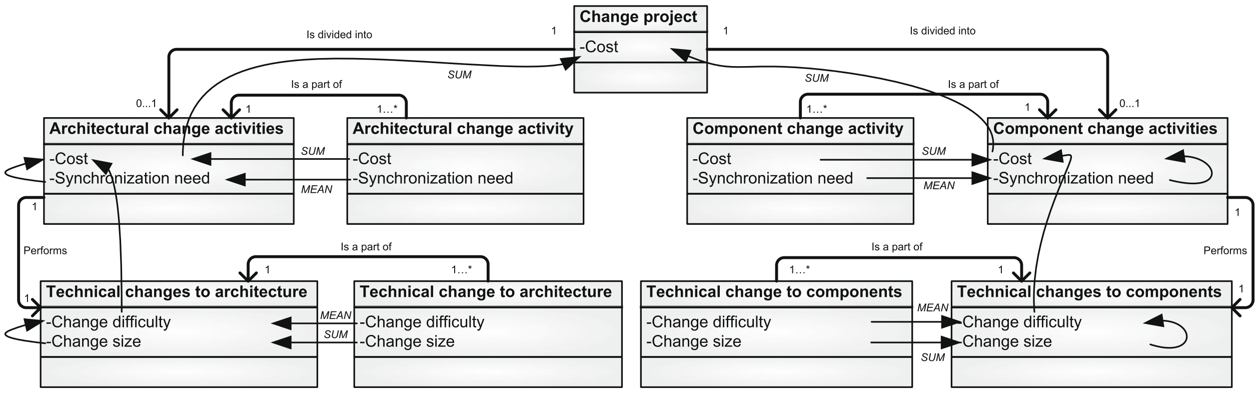

**Fig. 2.** The project and activity view of the modifiability PRM.

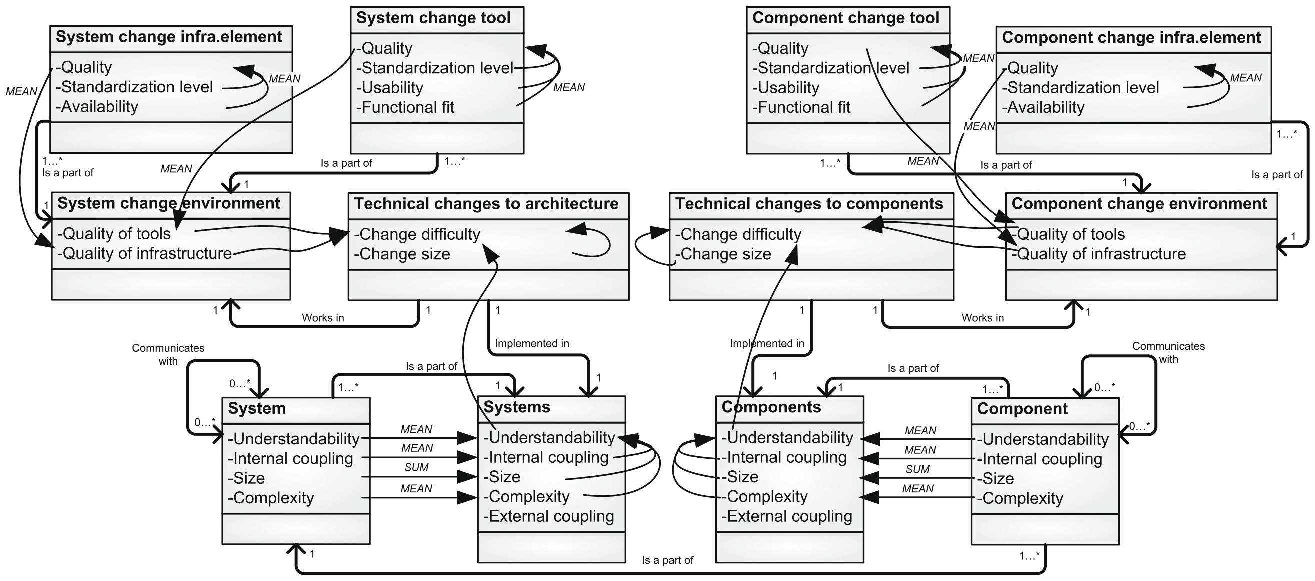

**Fig. 3.** The system view of the modifiability PRM.

attributes, such as the arithmetic operations *SUM* and *MEAN*. In the metamodel, cf. Figs. 3 and 4, there are several classes having underlying classes that serve as *Is-a-part-of* classes. E.g. in the system view, cf. Fig. 3, the system change environment consists of a number of tools. Here, the quality of each tool aggregates by the *MEAN* operation to the quality of the change environment. Since the aggregate functions are not based on causality but rather on aggregates of attributes, these CPDs are not defined based on data and can not be found with experiments. These aggregate functions are definitions decided by the metamodeler and are in the modifiability case intuitively set as *SUM* or *MEAN*.

## 4. The modifiability metamodel

This section presents an enterprise architecture metamodel for modifiability analysis. Outlined are the classes of the metamodel, its reference slots and attributes. The interested reader is referred to Lagerström et al. (submitted for publication) for a complete description of the metamodel. The main elements of the metamodel textually described in this section are depicted in Fig. 4.

The suggested metamodel for modifiability analysis focuses on software systems and their surrounding environment involved in or affected by modifications implemented in a change project. The main element of the metamodel is the *Change project* class. Since modifiability in this paper is defined as cost of change the natural attribute of the change project class is *Cost*.

Change projects are divided into *Architectural change activities* and *Component change activities* (Boehm, 1981; Bass et al., 1988; Grubb and Takang, 2003). Architectural change activities represent modifications on an architectural level, while a component change activity concerns modifications of a single component of an application. The two activity types both have *Cost* as their main attribute. The sum of these costs constitute the total cost of a change project. Moreover, these two change activities are attributed with *Synchronization need* as a measure of the need for alignment between activities involved in the change project.

The two sorts of change activities are supported by a *Change management process* possessing the attribute *Maturity*, emphasizing the importance of having a rigid and proper support function for the change project being conducted (Boehm, 1981; Oman et al., 1992; Pigoski, 1997; Smith, 1999; Grubb and Takang, 2003; Kan, 2003; The IT Governance Institute, 2007; April and Abran, 2008; Simonsson, 2008).

Component change activities and architectural change activities perform technical changes to software components and system architecture respectively. These changes are illustrated in the metamodel by the classes *Technical changes to components* and

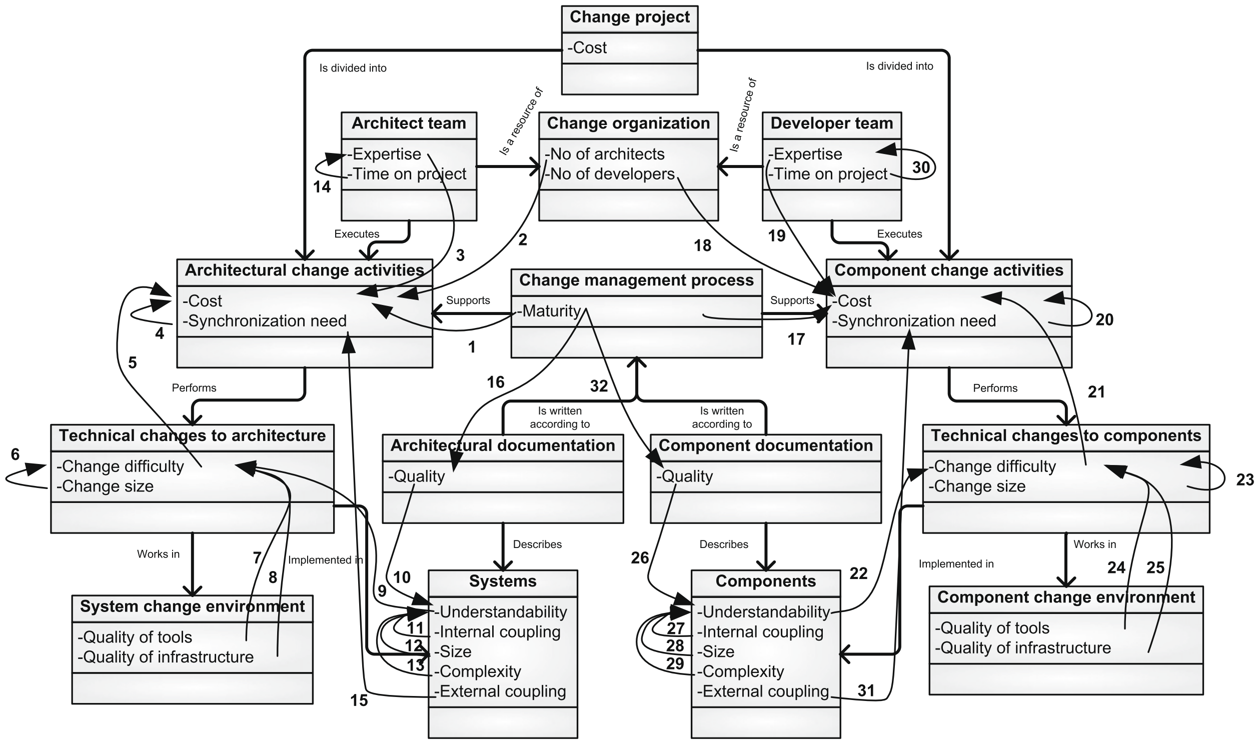

**Fig. 4.** The modifiability PRM containing classes and attributes, including numbering of the attributes causal dependencies.

**Table 1**
The resulting conditional probability distribution for the attribute *Cost* of the class *Component change activities* affected by the *Change management process maturity*, *Number of developers in the change organization*, *Developer team expertise*, *Change difficulty of the technical changes to components* and *Synchronization need of the component change activities*.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Change mgmt. process maturity (17) | | Not mature | | | | | | … | Very m. |
| Change org. no of developers (18) | | Many | | | | | … | … | Few |
| Developer team expertise (19) | | Low | | | | … | … | … | High |
| Tech. changes to comp. change diff. (21) | | Difficult | | | … | … | … | … | Easy |
| Comp. change act. synch. need (20) | | High | Med. | Low | … | … | … | … | Low |
| Component change activities cost | High | 0.88 | 0.73 | … | … | … | … | … | 0.06 |
| | Medium | 0.06 | 0.21 | … | … | … | … | … | 0.06 |
| | Low | 0.06 | 0.06 | … | … | … | … | … | 0.88 |

*Technical changes to architecture*. The two classes are attributed with *Change difficulty* and *Change size* (Chan et al., 1996; Pigoski, 1997; International Organization for Standardization, 2001; Grubb and Takang, 2003; April and Abran, 2008).

Fig. 2 presents a view illustrating the change project class, the classes for change activities and technical changes with their respective attributes, reference slots, causal structures, and aggregating functions.

Change activities are executed either by architects, belonging to an *Architect team*, or by developers belonging to a *Developer team*. Attributes needed for cost estimation related to these classes are *Time on project* and *Expertise* (Boehm, 1981; Oman et al., 1992; Chan et al., 1996; Fenton and Pfleger, 1997; Pigoski, 1997; Smith, 1999; Grubb and Takang, 2003; April and Abran, 2008).

The documentation supporting the change activities is written according to the change management process. The set of documentation is modeled as either *Architectural documentation* or *Component documentation* depending on its content. The most important aspect of the documentation is its quality, hence *Quality* is the main attribute for both types of documentation (Oman et al., 1992; Pigoski, 1997; Smith, 1999; Aggarwal et al., 2002; Grubb and Takang, 2003; April and Abran, 2008).

The architect team and developer team are modeled as resources of the *Change organization* in the metamodel. The relevant attributes related to cost estimation for change organizations are the total number of architects and developers involved in the project. Thus *No of architects* and *No of developers* serve as the two attributes of the change organization (Oman et al., 1992; Chan et al., 1996; Fenton and Pfleger, 1997; Pigoski, 1997; Grubb and Takang, 2003; Putnam and Myers, 2003; April and Abran, 2008).

The classes technical changes to architecture and components each work in their respective environment, defined in the metamodel by the classes *System change environment* and *Component change environment*. These two classes both have the attributes

*Quality of tools* and *Quality of infrastructure* (Boehm, 1981; Oman et al., 1992; Fenton and Pfleger, 1997; Pigoski, 1997; Smith, 1999; Grubb and Takang, 2003; April and Abran, 2008).

The modifications in a change project are either implemented in *Systems* or *Components*. These classes are both attributed with *Understandability*, *Internal coupling*, *Size*, *Complexity*, and *External coupling* (Boehm, 1981; Oman et al., 1992; Chan et al., 1996; Fenton and Pfleger, 1997; Granja-Alvarez and Barranco-Garcia, 1997; Pigoski, 1997; Zuse, 1997; Bass et al., 1988; Smith, 1999; Aggarwal et al., 2002; Grubb and Takang, 2003; Kan, 2003; Matinlassi and Niemel, 2003; Putnam and Myers, 2003; Laird and Brennan, 2006; April and Abran, 2008). In Fig. 3 a view illustrating systems, components, and the change environment classes is presented.

The metamodel depicted in Fig. 4 presents the main classes, their reference slots, and the causal dependency structure of their attributes. Thus, this figure neither includes reference slot multiplicity nor does it include aggregating functions. However, it does contain additional information regarding the causal dependencies. These dependencies are numbered in order to relate them to the CPDs explained in Section 3.2 and the data collection presented in Section 7.2. Two views of the metamodel, cf. Fig. 2 and 3, have been used to illustrate the aggregating classes and attributes. There are other views of the metamodel, these are presented in Lagerström et al. (submitted for publication). Together the views constitute the whole metamodel. It is however to large to fit into one single figure and still be readable, thus the focus on different views.

For the interested reader Lagerström (2007) presents an early version of the modifiability metamodel.

## 5. Modeling

This section presents the approach of instantiating the modifiability metamodel, as well as the possible benefits gained when it is employed in change projects.

### 5.1. Metamodel instantiation

The purpose of the modifiability metamodel, described in Section 4, is for it to be instantiated in software change projects in order to estimate the project cost (in man-hours) and to highlight risks, thus supporting decision making and planning.

The process of instantiating the metamodel basically contains two steps: data collection and modeling. Based on the metamodel elements the modeler gets input to what data she should collect for her models. The modifiability metamodel, cf. Fig. 4, contains information regarding what classes to instantiate into objects, how these relate to each other, and what attributes they need for change cost analysis.

In project M studied at a large Nordic transportation company (described in Section 6.4) data was collected by interviewing and surveying people involved in the project, and by studying project documentation. Additional data was also collected with the development tools used at the company, for instance the size and component internal coupling measures. Based on the data collection an architecture model was instantiated for the project, the main view is illustrated in Fig. 10. The instantiated model view presented illustrates project M on a high level, i.e. excluding the underlying classes and their attributes.

According to the metamodel the instantiated model should for instance contain systems and how these relate to each other. In project M there were seven systems involved in the change: ticket delivery, youth travel, webshop, sales, content management, seat booking, and journey planner. Each system has, according to the metamodel, the attributes understandability, internal coupling, size, and complexity. The collection of systems also has the attribute external coupling (calculated based on the information in the instantiated model). Fig. 9 illustrates the external coupling view of the instantiated architecture model for project M.

The attributes of the metamodel are all defined with the scales described in Lagerström et al. (submitted for publication). Excerpts of four attribute scales are presented in Table 2. Using a fine grained scale provides clear definitions and allows for statistical analysis. These scales are used when data is collected for the instantiation of the metamodel. However, since the approach is formalized with probabilistic relational models, the input for the instantiated architectural model needs to be on a discrete scale with a finite number of states. Also, the method used for defining the CPDs of the metamodel restricts the attribute scales to be discrete with three defined states, cf. Section 3.2. Therefore, the data collected is transformed according to the rules described in Table 2.

### 5.2. Probabilities estimation vs. cost in man-hours

The PRM based metamodel presented handles all attribute values as probabilities, including the goal attribute *Cost*. This means that change cost has probability values being either *High*, *Medium*

**Table 3**
Cost intervals for categorization of change project size.

| Segment/cost | High | Medium | Low |
|---|---|---|---|
| Large | 40,000 | 6000 | 1000 |
| Medium | 12,000 | 6000 | 1000 |
| Small | 5000 | 2500 | 1000 |

**Table 2**
Attribute definitions, data collection scales, transformation rules, and model calculation scales for the class system.

| Class | Attribute | Definition | Data collection scale | Transformation rule | Model calculation scale |
|---|---|---|---|---|---|
| System | Understandability | Time, in percentage compared to total change activity time, spent on trying to understand the system. | {0,. . .,100} | 41-100 % = Difficult<br>21-40 % = Normal<br>0-20 % = Easy | {Difficult, Normal, Easy} |
| | Internal coupling | Number of actual relations between the components in this system divided with the number of possible relations. | {0,. . .,100} | 41-100 % = Tight<br>16-40 % = Normal<br>0-15 % = Loose | {Tight, Normal, Loose} |
| | Size | Total number of components in the system. | {0, 1, 2....∞} | 500 - ∞ = Large<br>40 - 499 = Medium<br>0 - 39 = Small | {Large, Medium, Small} |
| | Complexity | Subjectively evaluated. "How complex is the system?" | {Complex, Medium, Not complex} | N/A | {Complex, Medium, Not complex} |

or *Low*. To enable estimations in man-hours these probabilities need to be translated into man-hours. In this paper the probability transformation is approached by the introduction of segments, cf. Table 3. These segments take into consideration whether the projects are considered being of either *Large*, *Medium* or *Small* size.

To be able to utilize the estimation capabilities of the metamodel for software change project cost analysis within the accuracy ranges in Table 6 (further described in Section 7.3), a subjective prediction of the size of the project needs to be done. This segmentation of project size is preferably performed by project managers having experience of the project culture in the company where the architectural models are being applied.

Here a simple categorization aiming at determining if, dependent on e.g. company culture, the largest projects usually turns out to demand around 40,000, 12,000 or 5000 man-hours. Some companies always perform their work in small projects that in the worst cases means around 5000 man-hours. Other companies work in large projects costing approx. 40,000 man-hours. Thus, when using the modifiability metamodel the modeler (e.g. project manager) must make a decision whether the project he or she is modeling is a *large*, *medium*, or *small* project. For a *large* project this means that the probabilities of the project being *High*, *Medium*, or *Low* in the metamodel are mapped to the man-hour levels 40,000, 6000 and 1000, respectively. This has so far only been done a posteriori and further research is being conducted to classify these more objectively and generally. Once a project has been fitted into a segment, then it is possible to make a cost estimation in man-hours with a reasonable accuracy. Table 3 depicts the transformation from probabilities of *High*, *Medium*, and *Low* cost to number of man-hours for *Large*, *Medium*, and *Small* projects.

The cost in man-hours, *C*, is calculated as an expectation value $E(C)$.

$$E(C) = \sum_{i=L}^{H} U_i P(C = i),$$

where $U_i$ is the utility value of the attribute *Cost* given that the *Cost* equals *i*.

In Fig. 10 we can see that the attribute *Cost (prob.)* is *High* with a 26% probability, *Medium* with a 33% probability, and *Low* with a 41% probability. Since project M is considered to belong to segment *Medium*, Table 3 gives us the utility values $U_{High} = 12,000, U_{Medium} = 6000$, and $U_{Low} = 1000$.

Thus the expectation value for the cost in project M is

$$E(C) = 12000 * 0.26 + 6000 * 0.33 + 1000 * 0.41 = 5510.$$

### 5.3. Modeling benefits

A decision maker, typically project managers working with software change, will be provided with four sorts of valuable information when utilizing the modifiability metamodel. Firstly, the expected costs of a set of change projects can be estimated providing a cost ranking of the projects in the company portfolio. This means that a more rational decision making is enabled.

Secondly, it is possible for a decision maker to test different scenarios regarding the objects and attributes in the instantiated architectural model in order to try to lower the cost of a specific project and thereby providing increased decision support for software project portfolio prioritization. This could for example be realized by involving other developers having more suitable experience within the chosen design specific language in the project.

Thirdly, when a project has been chosen from the portfolio to be initiated the architectural model will be able to aid the project manager in the planning phase of the project. The manager can, for instance, get suggestion on architectural and development team size, as well as elaborate how the team expertise will affect the outcome.

Fourthly, the instantiated architectural model can be used for conducting risk analysis. The model is able to expose parts of the project with a high risk of increasing its cost. Hence, action plans can be set up accordingly to mitigate the occurrence of these risks. This enables the resources chosen for the project to be optimized for the project specific change activities. Hence, the risk of the project exceeding its given budget and timeframes are somewhat more controllable.

## 6. Architectural models and analysis

Several case studies have been conducted based on the modifiability metamodel and the enterprise architecture analysis approach presented. In all, four multiple case studies have been conducted and within these 21 software change projects have been studied. In the first case two projects within a Nordic consultancy firm were studied (projects A and B). The second case considered eight projects conducted within a large Nordic manufacturing company (projects C to J). Case three contained two projects at a leading Nordic software and hardware vendor (projects K and L). In the fourth case nine change projects where studied at a Nordic transportation company (projects M to U).

This section focuses on presenting the companies involved in the four multiple case studies. Four projects are presented in text and with instantiated architectural models, one from each case study. The actual costs and the estimated values of all 21 software change projects are presented in Section 7.

### 6.1. Case study 1 – Nordic consultancy firm

This consultancy firm, established in 1993, is one of the leading Nordic consulting firms in the field of product data management (PDM) and related software technology areas such as product lifecycle management (PLM), product engineering systems, product database applications, and document management.

Two projects (called project A and B) were studied at the consultancy firm. Both projects were large software change projects with the aim of implementing a number of end-user requirements in several systems. The projects were implemented at a large Nordic IT-provider and the development was done by a subcontractor. *Project A* is further described in the next paragraph and Fig. 5 presents an instantiated architectural model view focusing on the project and change activity classes.

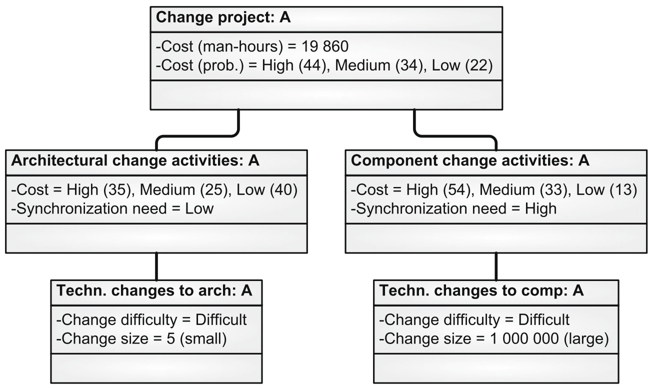
Change project: A
-Cost (man-hours) = 19 860
-Cost (prob.) = High (44), Medium (34), Low (22)
Architectural change activities: A
-Cost = High (35), Medium (25), Low (40)
-Synchronization need = Low
Component change activities: A
-Cost = High (54), Medium (33), Low (13)
-Synchronization need = High
Techn. changes to arch: A
-Change difficulty = Difficult
-Change size = 5 (small)
Techn. changes to comp: A
-Change difficulty = Difficult
-Change size = 1 000 000 (large)

**Fig. 5.** An excerpt of the instantiated PRM for project A presenting the classes: *change project*, *change activities*, and *technical changes*.

Project A ended up costing approximately 20,000 man-hours, where 6800 man-hours were used for the architectural changes and 13,200 man-hours for the component changes. The largest costs were the development cost and test cost for the component changes. The smallest cost was the learning cost for the architectural changes. Comparing the project outcome with the estimation provided by the instantiated model, cf. Fig. 5, we can see that the accuracy of the estimated value is as high as 99%. We can also see in the model that the technical changes of the components are both difficult and large, thus it was no surprise that the largest costs were the development and test costs for the component changes.

### 6.2. Case study 2 – Nordic manufacturing company

The second case study was conducted at a large Nordic manufacturing company. The company is active in some 100 countries and has over 30,000 employees. More than 2000 people work with development and research, mainly within the Nordic region. The production facilities are spread out in Europe and Latin America.

In this multiple case study eight projects (labeled project C to J) were studied. We will focus on presenting some information regarding *Project F*. Project F was initiated since the company predicted increased sales and thus an increased amount of employed products. To still be able to manage the products a new function in the product management portal was needed. In addition to this there was a need for a more secure, redundant and scalable server environment to be set up along the development of the new software application. The project had several objectives concerning improvement of the product management business, namely; to reduce the amount of hours spent on administration related to product management, to improve the support for the distributors of the products, to improve the quality of the already present services, to obtain a scalable, redundant and secure communication infrastructure and finally, to future proof the communication and server environment in terms of more long lasting functionality.

The main risks found in project F, cf. Fig. 6, were: the expertise of the developer team was considered to be low since the people involved only had 0–1 years of experience with change work and with the programming language used. Also, the architect team, which has high expertise, only spend 0–20% of their time on this particular project. The change management process is measured to be only at maturity level 1 (scale: level 0–5). Furthermore, the external coupling of the systems involved were considered to be high.

Project F ended up costing approximately 20,000 man-hours and the modifiability metamodel estimated the cost to 15,230 man-hours. Thus, the estimation accuracy for this project is 76%.

### 6.3. Case study 3 – Nordic software and hardware vendor

The third case study was conducted at a large Nordic software and hardware vendor. The company is a global leader in its domain. The company has over 100,000 employees worldwide and they have customers and local distributors in over 100 countries. The case study was carried out at one of the development departments at the company. The department's work mainly focus on management and further development of an information and control system. This system is sold as a standard product, however each implementation is modified in order to match the unique requirements of the customer.

The projects studied in this case were two projects (named project K and L) in a large program consisting of numerous projects. The projects, although part of a program, were treated by both the customer and the vendor as separate projects. The main program's purpose was to implement a large upgrade of a National Control Center in a large non-European country. The upgrade consisted of a large amount of additional functionality that was supposed to be added to a standard product delivered by the vendor. In the project presented here, from now on called *Project L*, the main task was to add a new component to the standard product. The main function of this new component is to provide a tool for

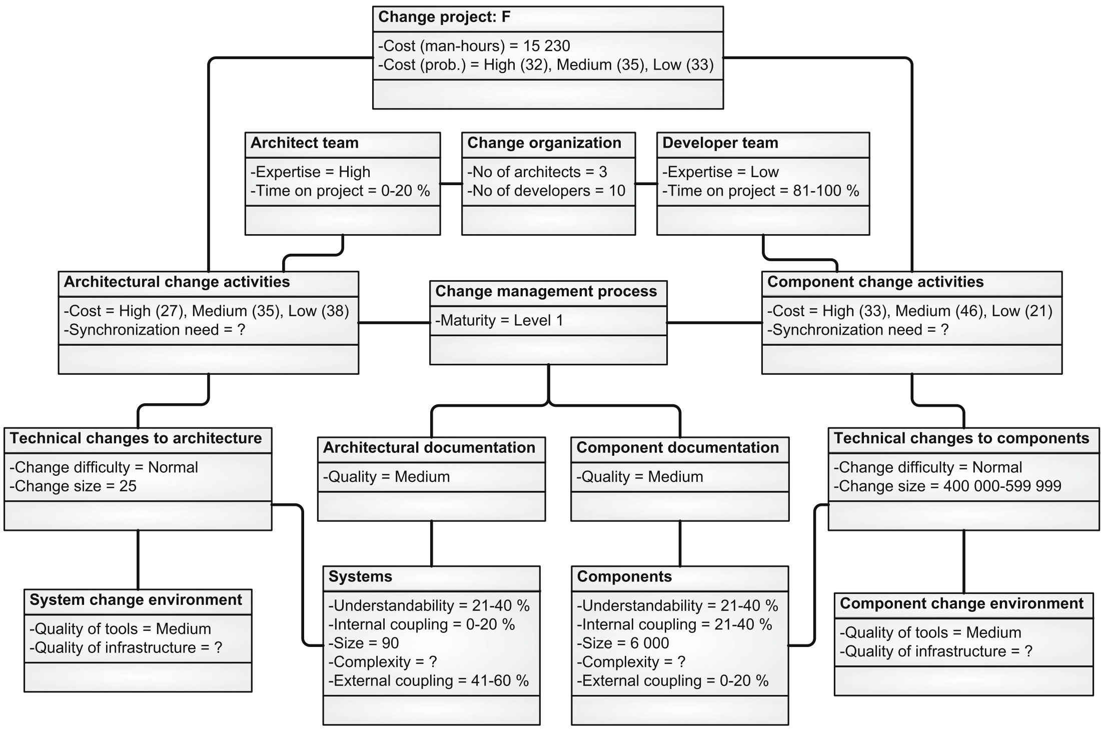
Change project: F
-Cost (man-hours) = 15 230
-Cost (prob.) = High (32), Medium (35), Low (33)
Architect team
-Expertise = High
-Time on project = 0-20 %
Change organization
-No of architects = 3
-No of developers = 10
Developer team
-Expertise = Low
-Time on project = 81-100 %
Architectural change activities
-Cost = High (27), Medium (35), Low (38)
-Synchronization need = ?
Change management process
-Maturity = Level 1
Component change activities
-Cost = High (33), Medium (46), Low (21)
-Synchronization need = ?
Technical changes to architecture
-Change difficulty = Normal
-Change size = 25
Architectural documentation
-Quality = Medium
Component documentation
-Quality = Medium
Technical changes to components
-Change difficulty = Normal
-Change size = 400 000-599 999
System change environment
-Quality of tools = Medium
-Quality of infrastructure = ?
Systems
-Understandability = 21-40 %
-Internal coupling = 0-20 %
-Size = 90
-Complexity = ?
-External coupling = 41-60 %
Components
-Understandability = 21-40 %
-Internal coupling = 21-40 %
-Size = 6 000
-Complexity = ?
-External coupling = 0-20 %
Component change environment
-Quality of tools = Medium
-Quality of infrastructure = ?

**Fig. 6.** The main view of the instantiated PRM for project F of the multiple case study at a Nordic manufacturing company.

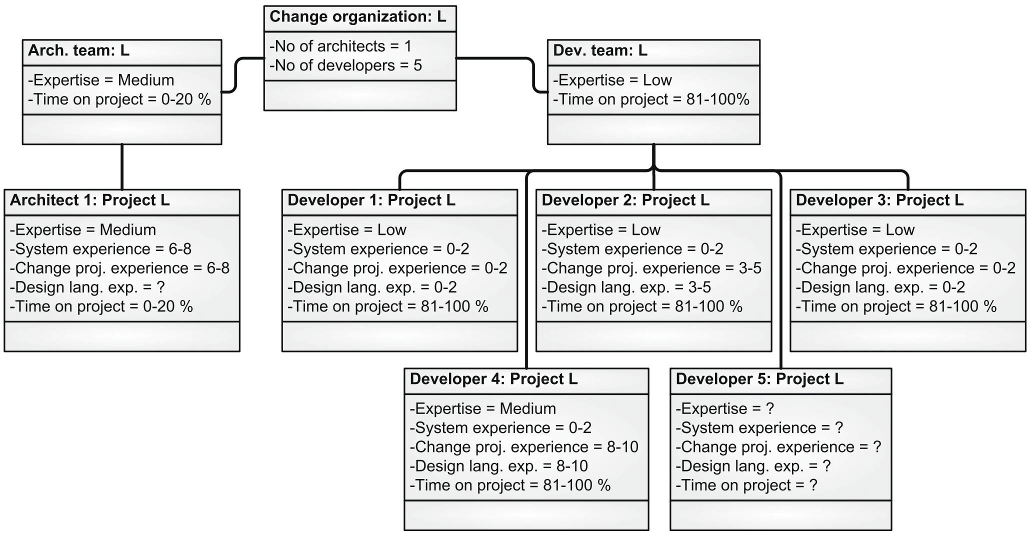

**Fig. 7.** An organizational view of the instantiated PRM for project L.

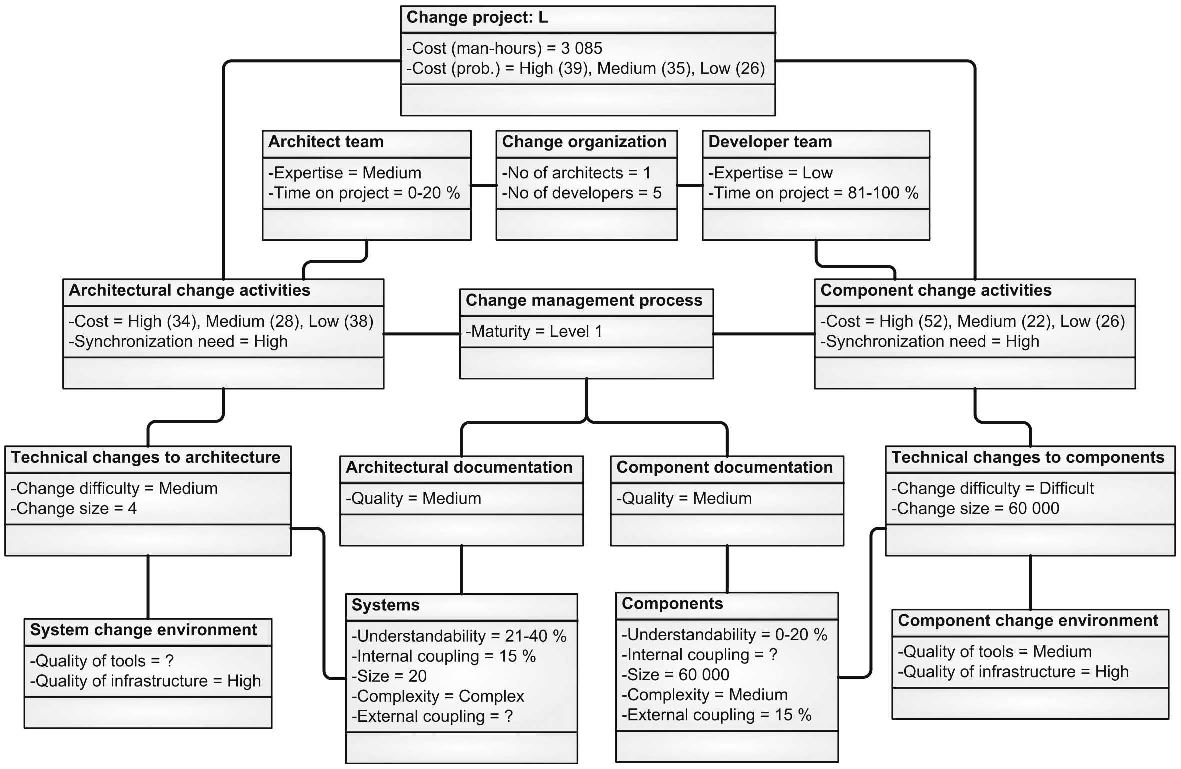
Change project: L
-Cost (man-hours) = 3 085
-Cost (prob.) = High (39), Medium (35), Low (26)
Architect team
-Expertise = Medium
-Time on project = 0-20 %
Change organization
-No of architects = 1
-No of developers = 5
Developer team
-Expertise = Low
-Time on project = 81-100 %
Architectural change activities
-Cost = High (34), Medium (28), Low (38)
-Synchronization need = High
Change management process
-Maturity = Level 1
Component change activities
-Cost = High (52), Medium (22), Low (26)
-Synchronization need = High
Technical changes to architecture
-Change difficulty = Medium
-Change size = 4
Architectural documentation
-Quality = Medium
Component documentation
-Quality = Medium
Technical changes to components
-Change difficulty = Difficult
-Change size = 60 000
System change environment
-Quality of tools = ?
-Quality of infrastructure = High
Systems
-Understandability = 21-40 %
-Internal coupling = 15 %
-Size = 20
-Complexity = Complex
-External coupling = ?
Components
-Understandability = 0-20 %
-Internal coupling = ?
-Size = 60 000
-Complexity = Medium
-External coupling = 15 %
Component change environment
-Quality of tools = Medium
-Quality of infrastructure = High

**Fig. 8.** The main view of the instantiated PRM for project L of the multiple case study at a Nordic software and hardware vendor.

operators to evaluate and analyze optional transactions with other companies. The new component also required integration with several other already existing components of the system.

As can be seen in the instantiated organizational view of project L, cf. Fig. 7, there were six persons involved in project L. Five developers and one architect. Since many of the developers only had 0–2 years of experience for several of the experience attributes, the average expertise level of the developer team was assessed to be *Low*.

Project L was estimated to 800 man-hours by the project manager at the company. His estimate was based on his experience without any tool support. As can be seen in Fig. 8, the instantiated model of project M provided an estimate of 3085 man-hours. The actual cost in man-hours was approximately 3200. In this case the expert estimate had an accuracy of 25%, while the metamodel provided an estimate with a 96% accuracy.

## 6.4. Case study 4 – Nordic transportation service company

The fourth case study was conducted at a large Nordic transportation company. The company provides transportation services mostly within one of the Nordic countries, but also has some ser-

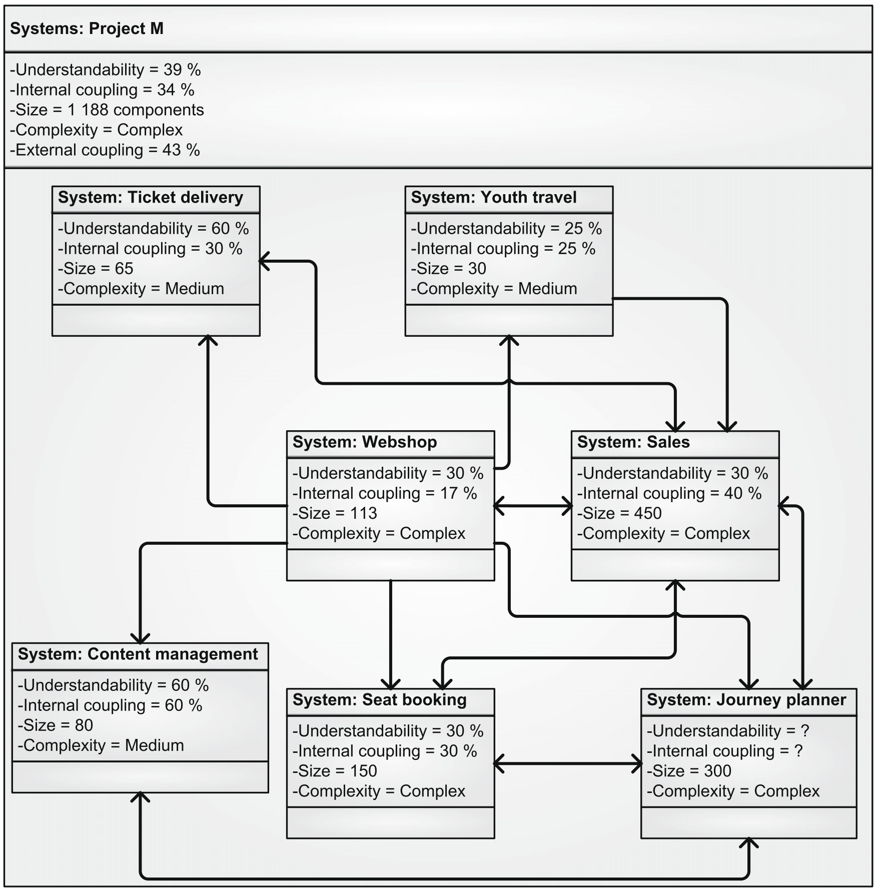

**Fig. 9.** A systems external coupling view of the instantiated PRM for project M.

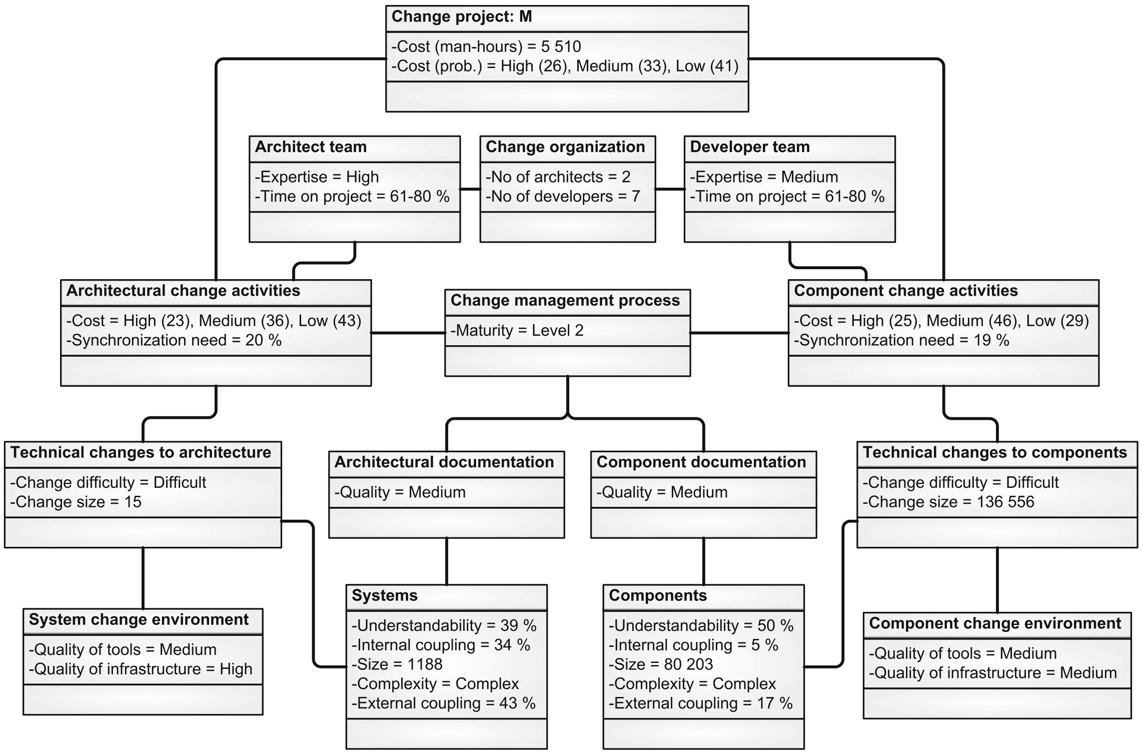
Change project: M
-Cost (man-hours) = 5 510
-Cost (prob.) = High (26), Medium (33), Low (41)
Architect team
-Expertise = High
-Time on project = 61-80 %
Change organization
-No of architects = 2
-No of developers = 7
Developer team
-Expertise = Medium
-Time on project = 61-80 %
Architectural change activities
-Cost = High (23), Medium (36), Low (43)
-Synchronization need = 20 %
Change management process
-Maturity = Level 2
Component change activities
-Cost = High (25), Medium (46), Low (29)
-Synchronization need = 19 %
Technical changes to architecture
-Change difficulty = Difficult
-Change size = 15
Architectural documentation
-Quality = Medium
Component documentation
-Quality = Medium
Technical changes to components
-Change difficulty = Difficult
-Change size = 136 556
System change environment
-Quality of tools = Medium
-Quality of infrastructure = High
Systems
-Understandability = 39 %
-Internal coupling = 34 %
-Size = 1188
-Complexity = Complex
-External coupling = 43 %
Components
-Understandability = 50 %
-Internal coupling = 5 %
-Size = 80 203
-Complexity = Complex
-External coupling = 17 %
Component change environment
-Quality of tools = Medium
-Quality of infrastructure = Medium

**Fig. 10.** The main view of the instantiated PRM containing data for project M of the multiple case study at a Nordic transportation company.

vices in neighboring countries. They carry hundreds of millions of passengers every year, mainly with train services, and employ several thousand people.

In total, nine software change projects (project M to U) were studied at the transportation company. We here focus on presenting one project, from now on called *Project M*. Project M was a project focusing on modifying the company's public ticket webshop into an almost completely new version of it. The focus was on adding and changing functionality in the webshop software, making it possible for the passengers to buy tickets online easier, faster, more secure and more reliable. One especially important part of the project was to increase the usability with focus on the user interface for finding journeys, choosing ticket type, view price information, and pdf-ticket printing. The project included developing a new web flow and creating new integration solutions towards the sales, journey planner, and ticket delivery systems.

The webshop system is integrated with a sales system. Also, the webshop system and the sales system are integrated with a seat booking system. Furthermore, the sales system and the seat booking system are integrated with a journey planning system. In all, there are seven systems, coupled together, involved in the change project, i.e. seven systems in need of modifications. In Fig. 9 an external coupling view of the instantiated architecture model presents the systems in project M and how these communicate.

The instantiated model presented in Fig. 10 presents project M on a high level, i.e. excluding the aggregating classes and their attributes. As can be seen in this figure the estimated cost was 5510 man-hours and the actual cost was 5810 man-hours. The metamodel provided an estimation with a 95% accuracy in this case.

## 7. Validation

The elements and structure of the modifiability metamodel as well as its estimation capabilities need to be evaluated and validated.

While the use of academic papers reflecting research serves as a good foundation for the metamodel creation, it is not completely trustworthy. There are several reasons for this. First, the scientific literature is not complete, so when creating a metamodel, it might be necessary to fill in some blanks with hypotheses unsupported by the literature. Second, the scientific literature is not always coherent, so when creating a metamodel it might be unavoidable to make more or less controversial choices. Third, the modeler might be biased and thus involuntarily introduce distortions. Expert validation of the metamodel attributes serves as a good function in minimizing these uncertainties.

Since the attribute PRMs of the metamodel are defined based on expert knowledge mapped to a three graded scale, there is a risk that the estimations are less accurate than wanted. Data based on 21 software change projects is used for testing the estimation capabilities of the metamodel.

The three most important questions are: (1) Are there attributes missing in the metamodel that should be added? (2) Are there superfluous attributes in the metamodel that could be removed? (3) Does the metamodel provide good estimation capabilities?

Together, the first and second question determines whether the metamodel contains the appropriate elements, and will be addressed in Sections 7.1 and 7.2 respectively.

The third question concerns the whole metamodel, both the qualitative and quantitative structure i.e. both the classes, reference slots, attributes, and causal dependencies, as well as the conditional probability distributions and aggregate functions. The estimation capability is validated by studying 21 change projects and by comparing the estimated cost with the actual cost outcome of the projects. This is addressed in Section 7.3.

**Table 4**
The workshop and survey results for the question "which attributes are missing in the metamodel?".

| Class | Attribute | No of answers |
|---|---|---|
| Management | Degree of support | 3 |
| Testing process | Test coverage | 2 |
| System | Level of quality goals restrictions | 2 |
| | Platform independence | 2 |
| Requirements | Number of authors | 3 |
| Specification | Number of changes during project | 2 |
| Architecture goals | Prioritized and communicated | 3 |
| Business organization | Stability | 2 |
| Change organization | Geographic distance (culture and language diff.) | 3 |
| | Understandability of business objects | 2 |
| | Use of a common information model | 3 |
| Change activities | Time restrictions (deadline) | 2 |

### 7.1. Are there attributes missing in metamodel?

The first question, "are there attributes missing in the metamodel?", was posed to a number of respondents, both academics in an online survey and industrial people during workshops. Two workshops with industrial respondents, with 8 and 39 respondents, respectively, were carried out. One online survey was sent out to academics in the field of software modifiability, this survey had 50 respondents. Thus, the total number of respondents was 97.

In the workshops and surveys the respondents were also provided with questions regarding their qualification as experts. Fourteen persons were excluded from the two workshops and 10 persons were excluded from the survey. Either these respondents had too little experience (less than three years) or they themselves said that they did not feel certain at all about their answers. This resulted in 33 workshop experts, where 17 provided suggestions on attributes that could be missing in the metamodel. Of the 40 survey experts 13 provided suggestions on missing attributes. Thus, in total the number of experts with suggestions on missing attributes was 30.

The only attributes that two or more experts were missing are presented in Table 4. The attributes with the most votes in our survey and workshops had three experts out of the total 73 missing them, which is only 4.1%. If we consider the possibility that the 43 persons not missing any attributes at all skipped the open-ended questions due to time restrictions and instead restrict attention to the group that suggested new attributes we have three votes out of a total 30 (10%). Since there is no strong agreement among the 30 experts on which attributes are missing, the indication is that the metamodel contains an appropriate set of attributes. Thus, the answer to question one is: no, the modifiability metamodel does not seem to be missing any attributes. Nevertheless, the attributes suggested are interesting and should be explored in future case studies and surveys.

### 7.2. Does the metamodel contain attributes that can be removed?

The second question, "are there superfluous attributes in the metamodel that could be removed?", was addressed by analyzing the strength of the causal dependencies between the metamodel attributes, as elicited from the experts. That is, if experts find that there is a strong causal connection between two attributes, then these attributes should be present in the metamodel. Conversely, if experts find that there is no causal connection between two attri-

**Table 5**
Survey and workshop data regarding the strength of influence between causally related attributes in the metamodel. The data is also used for defining the CPDs as explained in Section 3.2.

| Relation | High effect | Low effect | No effect | I don't know | No of answers | No eff. percentage (%) |
|---|---|---|---|---|---|---|
| 1 | 25 | 19 | 2 | 6 | 52 | 4.5 |
| 2 | 15 | 26 | 7 | 4 | 52 | 17.1 |
| 3 | 44 | 7 | 0 | 1 | 52 | 0.0 |
| 4 | 22 | 24 | 2 | 4 | 52 | 4.3 |
| 5 | 44 | 7 | 1 | 0 | 52 | 2.0 |
| 6 | 28 | 19 | 0 | 5 | 52 | 0.0 |
| 7 | 29 | 17 | 1 | 5 | 52 | 2.2 |
| 8 | 27 | 18 | 2 | 5 | 52 | 4.4 |
| 9 | 43 | 8 | i | 0 | 52 | 2.0 |
| 10 | 37 | 13 | i | 1 | 52 | 2.0 |
| 11 | 33 | 14 | 3 | 2 | 52 | 6.4 |
| 12 | 41 | 9 | 1 | i | 52 | 2.0 |
| 13 | 44 | 6 | 0 | 2 | 52 | 0.0 |
| 14 | 28 | 20 | 1 | 3 | 52 | 2.1 |
| 15 | 36 | 14 | 1 | i | 52 | 2.0 |
| 16 | 27 | 18 | 3 | 4 | 52 | 6.7 |
| 17 | 23 | 13 | 2 | 4 | 42 | 5.6 |
| 18 | 14 | 21 | 1 | 6 | 42 | 2.9 |
| 19 | 38 | 4 | 0 | 0 | 42 | 0.0 |
| 20 | 18 | 8 | 2 | 14 | 42 | 7.7 |
| 21 | 35 | 5 | 0 | 2 | 42 | 0.0 |
| 22 | 36 | 5 | 1 | 0 | 42 | 2.4 |
| 23 | 10 | 25 | 4 | 3 | 42 | 11.4 |
| 24 | 23 | 16 | 1 | 2 | 42 | 2.6 |
| 25 | 23 | 15 | 1 | 3 | 42 | 2.6 |
| 26 | 29 | 12 | 1 | 0 | 42 | 2.4 |
| 27 | 30 | 11 | 1 | 0 | 42 | 2.4 |
| 28 | 27 | 12 | 1 | 2 | 42 | 2.6 |
| 29 | 39 | 2 | 0 | i | 42 | 0.0 |
| 30 | 29 | 10 | 1 | 2 | 42 | 2.6 |
| 31 | 31 | 8 | 0 | 3 | 42 | 0.0 |
| 32 | 18 | 15 | 3 | 6 | 42 | 9.1 |

butes, then the parent attribute could be removed from the metamodel. Table 5 summarizes the causal dependency strengths found.

There were in total 99 respondents answering the questionnaire. Thirteen were industrial people providing answers at two workshops and 86 were academics providing their answers via two online surveys.

The question asked to the respondents both in the workshops and surveys was: "How large is the effect presented in the following statements?" The respondents were provided with statements each corresponding to a causal dependency in the metamodel. See Fig. 4 for all corresponding causal relationships. The statements were all arranged as the following examples; "Change management process maturity affecting component documentation", which corresponds to the relationship labeled as number 32 in the metamodel. "Developer team expertise affecting the component change activities cost", corresponding to the causal dependency labeled as number 19. The answer alternatives were given to the experts on the following scale; *High effect*, *Low effect*, *No effect*, or *I don't know*.

*High effect* between two attributes means that in most cases when you change the value of the parent attribute the child attribute changes as well. That is, the attributes are causally dependent. *Low effect* between two attributes means that in some cases when you change the value of the parent attribute the child attribute changes as well. I.e. there is a low causal dependency between the attributes. *No effect* between two attributes means that in no (or very few) cases when you change the value of the parent attribute the child attribute changes as well. That is, there is no causal dependency between the attributes. The *I don't know* answer means that either the respondents did not understand the concepts in the question or the respondents understood the concepts but did not have the experience to estimate the actual effect.

In the workshops and surveys the persons were also provided with questions regarding their qualification as experts. In the workshops, two persons were excluded due to lack of expertise; both persons stated that they had not enough experience in the field. Fourteen persons were excluded from the surveys, either they had too little experience (less than three years), they themselves said that they did not feel certain at all about their answers, or the answer *I don't know* was given to more than 50% of the questions asked.

Since the attributes in general have either *high effect* or *high/low effect* in relation to their parents, whereas very few had *no effect*, the attributes in the metamodel all seem useful for modifiability analysis. As can be seen in Table 5, no causal dependency has more than 17.1% answers on *No effect*. We interpret these low percentages to indicate that there are no attributes in the metamodel with no effect on its causally related attributes and by that the cost of making changes. Thus the answer to question two is: no, the modifiability metamodel does not seem to have any superfluous attributes that could be removed. However, some causal dependencies in the metamodel do have some answers on *Low effect*. These dependencies will be further validated in future change projects and questionnaires. Possibly one or two attributes can be removed or replaced, but this requires more research.

### *7.3. Does the metamodel provide good estimation capabilities?*

The third question, "does the metamodel provide good estimation capabilities?", was addressed by conducting four different multiple case studies. In total 21 software change projects were modeled. Section 6 presents the four companies and four selected projects, one from each case study.

Since all studied projects are completed, data concerning their actual costs is obtainable. The estimated cost for each of the 21

**Table 6**
Studied projects with the actual costs, estimated costs, and accuracy of all conducted estimations.

| Project size segment | Project | Actual cost (man-hours) | Estimated cost (man-hours) | Accuracy |
|---|---|---|---|---|
| Large | A | 20,000 | 19,860 | 0.99 |
| | F | 20,000 | 15,230 | 0.76 |
| | B | 14,000 | 14,940 | 0.93 |
| | D | 9100 | 9580 | 0.95 |
| Medium | J | 6266 | 5920 | 0.94 |
| | H | 6228 | 5840 | 0.94 |
| | M | 5810 | 5510 | 0.95 |
| | P | 4458 | 6380 | 0.57 |
| | U | 3595 | 4110 | 0.86 |
| Small | L | 3200 | 3085 | 0.96 |
| | K | 3200 | 2195 | 0.69 |
| | E | 3000 | 2480 | 0.83 |
| | I | 2440 | 2170 | 0.89 |
| | C | 2300 | 2240 | 0.97 |
| | R | 2054 | 1915 | 0.93 |
| Small (1200 man-hours or less) | G | 1200 | 2095 | 0.25 |
| | T | 1082 | 1680 | 0.45 |
| | O | 952 | 2040 | <0 |
| | N | 894 | 2295 | <0 |
| | Q | 454 | 2005 | <0 |
| | S | 262 | 1805 | <0 |

projects is listed and compared to the actual cost of each project, cf. Table 6. The last column indicates the accuracy of the estimation, i.e. how close to the real value the estimation is. The accuracy is calculated using the magnitude of the relative error (*MRE*), as defined by Conte et al. (1985). Suppose *E* is the estimate of a value and *A* is the actual value, then the magnitude of the relative error for the estimate is

$$MRE = \frac{|A - E|}{A}.$$

Accuracy is calculated as $1 - MRE$. This can be seen as the primary measure for the quality of the estimation the modifiability metamodel is capable of. E.g. for project M we can see that the actual cost turned out to be 5810 man-hours. The model estimated the cost to be 5510 man-hours. Thus, the accuracy of the estimation is calculated to be 95%.

Studying Table 6 we can see that the smallest projects, ranging from 262 to 1200 man-hours, are the most difficult ones to estimate with the modifiability metamodel. This outcome, i.e. that the estimation capabilities of the metamodel would not work that well for the smallest software projects, was expected. For instance, projects with few persons and components involved do not need documentation and process support to the same extent as large projects do. For the smallest projects the metamodel would probably contain a different set of classes and attributes focusing on more detailed development issues. Also, the estimation need for really small projects might be superfluous since the estimation work might consume too much time in relation to the business value it provides. Thus, one issue with the metamodel estimation concerns the project size: at what size does the metamodel no longer provide an acceptable estimation accuracy? The data from the 21 studied change projects provides an indication that the metamodel should not be employed in change projects under 1200 man-hours. There are other more suitable methods and models for the smallest projects, e.g. planning poker as discussed in section 8.3.

A question raised regarding the project size is: how can one know which projects that are large enough for the modifiability metamodel in advance? This was analyzed and discussed during the fourth case study. In the nine projects studied at the transportation company (projects M–U) there was a clear distinction between the projects with acceptable accuracies and the ones being poorly estimated by the metamodel. Two variables available at project start that differed between these projects were found, these are *Number of systems* involved in the change and *System change distribution*. As we can see in Table 7 projects M, R and U are successfully estimated, while O, N, Q and S are poorly estimated. Projects P and T are borderline cases. In the successfully estimated projects there were between five and seven systems involved in the change and the change distribution was *high/medium* between the involved systems. In the less successful cases the number of systems involved were two–four and the change distribution was *low*. Thus, in projects with many systems being changed and where the changes are spread out between these systems the modifiability metamodel seems to produce more accurate estimates than in the case when there are few systems involved and when the changes are more focused to few systems. As a consequence of this, the answer to the posed question is: projects aiming to modify five or more systems and with a high change distribution between these systems seem to be the ones benefiting from employing the proposed modifiability metamodel.

According to Conte et al. an acceptable level for mean accuracy is something higher than or equal to 75%. Calculating the mean accuracy for the 15 projects above 2000 man-hours results in a 88% accuracy. Conte et al. use this mean accuracy notion to define a measure of prediction quality (*PRED*). In a set of *n* projects, let *k* be the number of projects where the mean accuracy is higher than or equal to *q*. Then

$$PRED(q) = k/n.$$

According to Conte et al. an estimation technique is acceptable if $PRED(0.75) = 0.75$. This means that in 75% of the time the estimated values falls within 75% of their actual values. In our case $PRED(0.75) = 0.87$, thus the metamodel is acceptable as an estimation technique. However, if we include the six projects of 1200 man-hours or less then $PRED(0.75) = 0.62$.

Thus the answer to question three is: yes, the modifiability metamodel provides good estimation capabilities (at least for soft-

**Table 7**
Studied projects in the fourth case study conducted at a large Nordic transportation company.

| Project | Actual cost (man-hours) | Estimated cost (man-hours) | Accuracy | Number of systems involved | System change distribution |
|---|---|---|---|---|---|
| M | 5810 | 5510 | 0.95 | 7 | High |
| R | 2054 | 1915 | 0.93 | 5 | High |
| U | 3595 | 4110 | 0.86 | 7 | Middle |
| P | 4458 | 6380 | 0.57 | 7 | High |
| T | 1082 | 1680 | 0.45 | 3 | Low |
| O | 952 | 2040 | <0 | 3 | Low |
| N | 894 | 2295 | <0 | 4 | Low |
| Q | 454 | 2005 | <0 | 2 | Low |
| S | 262 | 1805 | <0 | 3 | Low |

**Table 8**
Comparing the estimation accuracy of the modifiability metamodel, COCOMO II, and function points.

| Model/method | | | 75% accuracy, PRED(0.75)= (%) |
|---|---|---|---|
| COCOMO II | 1997 | Before stratification | 49 |
| | | After stratification | 55 |
| | 2000 | Before stratification | 68 |
| | | After stratification | 76 |
| Function points | | Model A | 64 |
| | | Model B | 68 |
| The modifiability metamodel | | All projects | 62 |
| | | All above 2000 man-hours | 87 |

ware change projects over 2000 man-hours). However, since the smallest projects (the ones of 1200 man-hours and less) seems to be more difficult for the modifiability metamodel to estimate, this will be addressed separately in future research. Another topic for further research related to the metamodel's estimation capability is the size segmentation discussed in Section 5.2. Also, future research will include studying more change projects from start to end to provide another level of validation.

## 8. Comparison with other models and methods

Since enterprise architecture is a discipline on the rise there are no alternatives within this modeling field that can be used for comparison. There are however other disciplines that have addressed the cost estimation problem. This section presents three of such alternative models and methods for cost estimation. These are briefly presented and their estimation capabilities are compared with the ones of the modifiability metamodel. Section 8.1 presents COCOMO II, Section 8.2 presents function points, and Section 8.3 presents planning poker.

### 8.1. COCOMO II

COCOMO, COnstructive COst MOdel, was in its first version released in the early 1980's. It became one of the most frequently used and most appreciated software cost estimation models of that time. Since then, development and modifications of COCOMO has been performed several times to keep the model up to date with the continuously evolving software development trends. The latest version of COCOMO, called COCOMO II, had its estimation capabilities calibrated in the year 2000 with the help of information from 161 project data points and eight experts. This latest calibrated version of COCOMO II uses the probabilistic Bayesian approach for turning a priori obtainable data into estimates of costs related to an a posteriori state of a software development or modification project (Chulani et al., 1999; Boehm et al., 2000).

Table 8 contains a comparison between COCOMO II and the modifiability metamodel. The COCOMO II.2000 calibration has an accuracy of 75% of the actual cost 68% of the time before stratification and 76% of the time after stratification. This is a major improvement compared to the performance of the COCOMO II.1997 version which had an accuracy of 75% of the actual cost 49% of the time before stratification and 55% of the time after stratification (Boehm et al., 2000). The COCOMO stratification is done by calibrating the model's multiplicative coefficient to each of their major sources of project data (Chulani et al., 1999). (Chulani et al., 1999) recommend organizations using COCOMO II.2000 to calibrate it using company specific data to increase the model accuracy. The modifiability metamodel on the other hand has an accuracy of 75% of the actual cost in 62% of the time before and in 87% of the time after its removal of the projects of 1200 man-hours or less.

**Table 9**
Comparing the mean estimation accuracy of the modifiability metamodel and planning poker.

| Model/method | Mean accuracy (%) |
|---|---|
| The modifiability metamodel, all above 2000 man-hours | 88 |
| Planning poker | 82 |

Based on this comparison the estimation capabilities of the modifiability metamodel (based on all projects) are similar to the capabilities of COCOMO II before their stratification. It is slightly better than the 1997 version and somewhat less accurate than the 2000 version. However, when focusing on the 15 projects of 2000 man-hours and above the modifiability metamodel seems to provide a 75% accuracy for more projects than COCOMO II. Before any conclusions can be drawn based on this there are still some issues that need to be addressed. Firstly, the stratification and the project exclusion are not the same, thus maybe not really comparable. Secondly, the COCOMO results are tested with 161 projects while the modifiability metamodel results are tested with 21 (15) projects. Hence, the COCOMO II results are probably more stable. Although there are some issues left to be solved our results show great promise for future research and use of the enterprise architecture approach and the modifiability metamodel.

### 8.2. Function points

The function points analysis is a method of quantifying the size and complexity of a software system in terms of the functions that the system delivers to the user. Counting function points is done by considering the linear combination of five basic software components (inputs, outputs, master files, interfaces, and inquiries), each at one of three levels: low, average or high. This count is called unadjusted function points (UFP). The final number of a software system's function points is arrived by multiplying the UFP by an adjustment factor that is determined by considering 14 aspects of processing complexity (Matson et al., 1994).

Matson et al. (1994) did a major study of function point's ability to estimate software cost in 1994. They analyzed function point data from 104 projects obtained from a major corporation. In the study two function point models were tested, one logarithmic transformation model (Model A) and one multiple regression model (Model B). Model A had an accuracy of 75% in 64% of the cases and model B in 68% of the cases, cf. Table 8. While the modifiability metamodel has a 75% accuracy in 62% of the cases before removing the projects of 1200 man-hours or less and 87% after. Thus, both function point models perform slightly better than the modifiability metamodel before project removal. However, when the smallest projects are removed the metamodel seems to perform better.

### 8.3. Planning poker

Planning poker is a so called expert estimation method especially useful in agile software projects. The idea is to divide a project into smaller functional parts, which are described as short stories. For each described story the team members in the project are supposed to write down an estimate on a card, without any discussion among the team members. The estimates for each story are then discussed (if different) and new estimates are written down. This iterative process continues until the team has written similar estimations for all stories.

Moløkken-Østvold (2008) presents a study of using planning poker for combining expert estimates in software projects. In their study of one software project containing 55 stories they found that the mean estimation accuracy was 82%, cf. Table 9. While, the mean estimation accuracy of the modifiability metamodel was 88% (when studying the 15 project above 2000 man-hours). As we can see they provide rather similar estimation accuracies.

The project studied by Moløkken-Østvold (2008) ended up in approximately 430 man-hours and had the mean accuracy of 82%. Thus, planning poker seems to be a good complement to the modifiability metamodel for the projects of 1200 man-hours and less. It is however unclear how appropriate planning poker is for large projects.

## 9. Discussion and research outlook

During the research work, several issues concerning the validity have been addressed. Multiple sources in both the development phase and the validation phase were used. Key stakeholders have been addressed for reviews of drafts. A chain of evidence was established. Theory was developed and used. There are however some threats to the validity. These threats mainly concern the 21 change projects studied in the validation phase when focusing on the estimation capabilities. Since there was no possibility of studying projects from start to end most data gathered in these case studies was collected after the projects were finished. Thus providing a final cost of the projects for comparison, but also inferring an uncertainty of the a priori value of the data. Another issue being a threat to the validity concerns the scales used when collecting and analyzing data, especially the transformation between these scales. The cost interval segmentation used when estimating the costs is also a validity issue since this so far only has been done after the projects were finished.

The quantitative part of the metamodel was created by surveying modifiability experts both from industry and academia, in total 138 people were surveyed. Of these, 110 were classified as modifiability experts as explained in Section 7. The industry participants were surveyed during three separate workshops; one at a company working with enterprise software system change, one at the Software Metrics network of the Swedish software and computer association and one at the second annual Enterprise Architecture Symposium at the Royal Institute of Technology. The academic participants were chosen based on their publications at the 11th and 12th European Conference on Software Maintenance and Reengineering. Although the participants have been chosen based on their expertise in the field of enterprise system modifiability and that both industry practitioners and academics are present in the survey, it is difficult to be absolutely sure that the sample is representative. We do however believe that the metamodel foundation with academic literature, the survey data and the case studies provide results sound enough. This can nevertheless be further addressed in future research.

Reliability was achieved by documenting the research work during all phases. All case studies have their own reports describing data collection and analysis. The aim has been to operationalize as many steps as possible, e.g. conducting surveys with predefined choices and using methods with well-defined guidelines and rules, thereby allowing other investigators to repeat the work.

To the industrial practitioner, the present paper assists the enterprise modeling effort when the concern is focused on modifiability and enterprise software change. If the metamodel is employed in the beginning of change projects the modeler will get cost estimations and the possibility to highlight risks early.

To the EA tool industry, the present paper hints to potential new features or products. A tool incorporating the provided metamodel would give a qualitative and quantitative support to the user's modeling effort. The current market for EA tools does not explicitly consider different analyses, especially not with focus on modifiability and change cost.

To the scientific community, the presented metamodel combines the approach of enterprise architecture modeling for analysis with the approaches of modifiability analysis and change cost estimation. These communities can benefit from this work as well as continue contributing in the combined area by: (1) extending/improving the metamodel, (2) conducting more case studies and (3) surveying more experts.

As discussed in Section 7 there are some metamodel elements that after further research perhaps could be reconsidered, i.e. removed or changed. Also, in the surveys and case studies some new elements have been suggested to be included in the metamodel (indicating possible blind spots in existing academic literature). Before any such additions can be made some additional research is needed. Since the current version of the metamodel mainly focuses on change organization, change environment, documentation and software system related issues, special focus could be on the business related elements of enterprise architecture. Many business issues are now implicitly included in the change difficulty attribute of the technical changes class. New classes to add and test might be; management, business organization, requirement specification and business process.

## 10. Conclusions

Enterprise architecture models can be used to increase the general understanding of enterprise systems and specifically to perform various kinds of analysis. This paper presents a set of instantiated architectural models for enterprise systems modifiability analysis, i.e. for assessing the cost of making changes to enterprise-wide systems. The instantiated architectural models detailed are based on 21 software change projects conducted at four large Nordic companies.

The enterprise architecture models are formalized using probabilistic relational models, combining regular entity-relationship modeling aspects with means to perform enterprise architecture analysis. The presented models contain objects such as change activities, architects and developers, systems and their components, documentation, infrastructure, and a change management process. Each object has a set of attributes related to it. For instance, a component object contains the attributes understandability, coupling, complexity, and size. These attributes are causally related to each other and provide the user with probabilistic analysis capabilities taking uncertainty into consideration.

The paper discusses the validity of the modifiability metamodel based on data collected in workshops and through surveys with both academia and industry (in total 110 experts were surveyed). The studied data shows that the metamodel seems to contain the appropriate elements. By employing the metamodel, data from 21 software change projects was collected. This data was used in order to validate the estimation capabilities of the metamodel, i.e. how well the metamodel estimates software change cost. The metamodel seems to produce estimates within a 75% accuracy in 87% of the time and has a mean accuracy of 88% (when considering projects of 2000 man-hours or more).

## References

Aggarwal, K., Singh, Y., Chhabra, J.K., 2002. An integrated measure of software maintainability. In: Proceedings of the Annual IEEE Reliability and Maintainability Symposium.

Allen, R., 1997. A Formal Approach to Software Architecture. Ph.D. Thesis, Carnegie Mellon University.

April, A., Abran, A., 2008. Software Maintenance Management. IEEE Computer Society/John Wiley & Sons.

Bass, L., Clements, P., Kazman, R., 1988. Software Architecture in Practice. Addison Wesley Longman/Software Engineering Institute.
Boehm, B., 1981. Software Engineering Economics. Prentice-Hall.
Boehm, B., Abts, C., Chulani, S., 2000. Software development cost estimation approaches a survey. Annals of Software Engineering 10, 177–205.
Chan, T., Chung, S.L., Ho, T.H., 1996. An economic model to estimate software rewriting and replacement times. IEEE Transactions on Software Engineering 22.
Chulani, S., Boehm, B., Steece, B., 1999. Calibrating software cost models using Bayesian analysis. IEEE Transactions on Software Engineering, 573–583.
Conte, S., Dunsmore, H., Chen, V., 1985. Software Effort Estimation and Productivity. Academic Press Inc..
Department of Defense Architecture Framework Working Group, 2007. DoD Architecture Framework, Version 1.5. Technical Report, Department of Defense, USA.
Fenton, N.E., Pfleger, S.L., 1997. Software Metrics: A Rigorous and Practical Approach. PWS Publishing Company.
Franke, U., Höök, D., König, J., Lagerström, R., Närman, P., Ullberg, J., Gustafsson, P., Ekstedt, M., 2009. EAF2 – a framework for categorizing enterprise architecture frameworks. In: Proceedings of the 10th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, pp. 327–332.
Friedman, N., Getoor, L., Koller, D., Pfeffer, A., 1999. Learning probabilistic relational models. In: Proceedings of the 16th International Joint Conference on Artificial Intelligence, pp. 1300–1309.
Granja-Alvarez, J.C., Barranco-Garcia, M.J., 1997. A method for estimating maintenance cost in a software project: a case study. Software Maintenance: Research and Practice 9, 161–175.
Grubb, P., Takang, A., 2003. Software Maintenance: Concepts and Practice. World Scientific, 2003.
Harrison, W., Cook, C., 1990. Insights on improving the maintenance process through software measurement. In: Proceedings of the IEEE Software Maintenance Conference.
IEEE Standards Board, 1990. IEEE Standard Glossary of Software Engineering Technology. Technical Report, The Institute of Electrical and Electronics Engineers.
International Organization for Standardization, 2001. Software Engineering – Product Quality. International Standard ISO/IEC TR 9126, International Organization for Standardization.
Jarzabek, S., 2007. Effective Software Maintenance and Evolution: A Reuse-Based Approach. Auerbach Publications, Taylor & Francis Group.
Jensen, F.V., 2001. Bayesian Networks and Decision Graphs. Springer-Verlag New York, Inc., Secaucus, NJ, USA.
Johnson, P., Ekstedt, M., 2007. Enterprise Architecture – Models and Analyses for Information Systems Decision Making. Studentlitteratur.
Johnson, P., Lagerström, R., Närman, P., Simonsson, M., 2007. Enterprise architecture analysis with extended influence diagrams. Information Systems Frontiers 9.
Jürjens, J., 2005. Secure Systems Development with UML. Springer, Berlin Heidelberg.
Kan, S., 2003. Metrics and Models in Software Quality Engineering, second ed. Pearson Education.
Kurpjuweit, S., Winter, R., 2007. Viewpoint-based meta model engineering. In: Enterprise Modelling and Information Systems Architectures (EMISA 2007).
Lagerström, R., 2007. Analyzing system maintainability using enterprise architecture models. Journal of Enterprise Architecture 3, 33–41.
Lagerström, R., Johnson, P., Närman, P., 2007. Extended influence diagram generation. In: Interoperability for Enterprise Software and Applications Conference.
Lagerström, R., Franke, U., Johnson, P., Ullberg, J., 2009a. A method for creating enterprise architecture metamodels – applied to systems modifiability analysis. International Journal of Computer Science and Applications 6, 89–120.
Lagerström, R., Johnson, P., Höök, D., König, J., 2009. Software change project cost estimation – a Bayesian network and a method for expert elicitation. In: International Workshop on Software Quality and Maintainability Proceedings.
Lagerström, R., Johnson, P., Ekstedt, M., submitted for publication. Architecture analysis of enterprise systems modifiability – a metamodel for software change cost estimation.
Laird, L., Brennan, C., 2006. Software measurement and estimation: a practical approach. IEEE Computer Society/John Wiley & Sons, Berlin.
Lankhorst, M., 2005. Enterprise architecture at work. Springer, Heidelberg.
Matinlassi, M., Niemel, E., 2003. The impact of maintainability on component-based software systems. In: Proceedings of the 29th IEEE EUROMICRO Conference "New Waves in System Architecture".
Matson, J., Barrett, B., Mellichamp, J., 1994. Software development cost estimation using function points. IEEE Transactions on Software Engineering 20, 275–287.
Moløkken-Østvold, K., Haugen, N.C., Benestad, H.C., 2008. Using planning poker for combining expert estimates in software projects. Journal of Systems and Software, vol. 81. Elsevier Science Inc., New York, NY, USA, pp. 2106–2117.
Neapolitan, R., 2003. Learning Bayesian Networks. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.
Oman, P., Hagemeister, J., Ash, D., 1992. A Definition and Taxonomy for Software Maintainability. Technical Report, Software Engineering Lab.
Pigoski, T., 1997. Practical Software Maintenance. John Wiley & Sons.
Putnam, L., Myers, W., 2003. Five Core Metrics. Dorset House Publishing.
Ross, J.W., Weill, P., Robertson, D., 2006. Enterprise Architecture As Strategy: Creating a Foundation for Business Execution. Harvard Business School Press.
Simonsson, M., 2008. Predicting IT Governance Performance: A Method for Model-Based Decision Making. Ph.D. Thesis, Royal Institute of Technology (KTH), Stockholm, Sweden.
Smith, D., 1999. Designing Maintainable Software. Springer.
Society of Automotive Engineers, 2009. Architecture Analysis and Design Language (aadl) Standard. Technical Report, Carnegie Mellon University.
The IT Governance Institute, 2007. Control Objectives for Information and Related Technology (COBIT) 4.1. Technical Report, The IT Governance Institute.
The Open Group, 2009. The Open Group Architecture Framework (TOGAF) – Version 9. The Open Group.
Winter, R., Fischer, R., 2007. Essential layers, artifacts, and dependencies of enterprise architecture. Journal of Enterprise Architecture 3, 7–18.
Zachman, J.A., 1987. A framework for information systems architecture. IBM Systems Journal, vol. 26. Riverton, NJ, USA, pp. 276–292.
Zachman, J., 2009. The Zachman Framework – The Official Concise Definition. <http://www.zachmaninternational.com> (accessed 16.06.09.).
Zuse, H.A., 1997. Framework of Software Measurement. Walter de Gruyter.

**Robert Lagerström** received his MSc degree in computer science from the Royal Institute of Technology in late 2005. In early 2006 he started his PhD at the department Industrial Information and Control Systems at the Royal Institute of Technology in Stockholm, Sweden. His topic of research as a PhD-student is Enterprise Architecture and Systems Modifiability. Robert is responsible for the courses "Industrial Information Systems, System Technology" and "Industrial Information Systems, Case Studies". In addition to that he supervises master thesis students. Robert was a member of the Swedish Chapter committee of INCOSE (International Council on Systems Engineering) up until 2009. He is responsible for the Software Metrics network at the Swedish Computer Society. Robert has written a number of academic publications in the field of Enterprise Architecture and Modifiability, also he is a co-author of the book Enterprise Architecture: Models and Analyses for Information Systems Decision Making. Robert is partner and consultant at Management Doctors, a Swedish IT-management consultancy firm.

**Pontus Johnson** is Professor and Head of the Department of Industrial Information and Control Systems at the Royal Institute of Technology (KTH) in Stockholm, Sweden. Active at the department are 25 researchers and PhD students focusing particularly on the analysis of architectural models of information systems and their context. He is secretary of the IFIP Working Group 5.8 on Enterprise Interoperability, technical coordinator of the FP7 Viking project, organizer, program committee member and associate editor of several international conferences, workshops and journals. Pontus supervises a number of PhD students. In his research he has much contact with Swedish corporations and organizations in the form of research projects, master thesis projects, seminars and consultations. He has written a book with the title Enterprise Architecture: Models and Analyses for Information Systems Decision Making, available in many book stores. He received his MSc from the Lund Institute of Technology in 1997 and his PhD and Docent title from the Royal Institute of Technology in 2002 and 2007. He was appointed professor in 2009.

**David Höö**k received his MSc degree in electrical engineering from the Royal Institute of Technology in 2008. The same year he started his PhD studies at the department of Industrial Information and Control Systems at the Royal Institute of Technology in Stockholm, Sweden. The topic of his research project as a PhD student is data quality in maintenance management. David is partly responsible for the course "Management of projects" held at the department. In addition to that he supervises master thesis students as well as students within the courses "Industrial Information Systems, System Technology" and "Industrial Information Systems, Case Studies". David has written a number of academic publications in the field of maintenance management as well as within the field of Enterprise Architecture.