

# Using Architectural Models to Predict the Maintainability of Enterprise Systems

Robert Lagerström\*, Pontus Johnson  
Department of Industrial Information and Control Systems  
Royal Institute of Technology (KTH)  
{robertl, pj101}@ics.kth.se

## Abstract

*Modern software systems are highly interconnected and have been under constant change for many years. IT decision makers find it difficult to predict and plan change projects due to the complexity of the enterprise systems. Thus, a large proportion of projects with the purpose of changing a software system environment fail, i.e. they tend to take longer time and cost more than expected. This paper suggests enterprise architecture as an approach to model software systems and their environment. An enterprise architecture metamodel for maintainability modeling and analysis is presented. IT decision makers can use this metamodel in order to make cost predictions and do risk analysis for their change projects.*

Index Terms: Modeling, Maintenance, Prediction Methods

## 1. Introduction

Business environments today progress and change rapidly to keep up with evolving markets. Most business processes are supported by information systems and since the business processes change the information systems need to be changed as well in order to continue supporting the processes. These modifications can vary from adding a functional requirement in one system to implementing a service oriented architecture for the whole enterprise.

An essential issue with today's software systems is that many of them are interconnected, thus a change to one system may cause a ripple effect among other systems. Also, numerous systems have been developed and modified during many years and to make further changes to them require a lot of effort from the organization since lots of the previous changes may have been done ad hoc. Such problems pose questions

for IT decision makers such as: Is there enough documentation describing the systems and has the documentation been updated correctly after each change? Is the source code easy to understand? Which systems are interconnected?

Enterprise architecture is an approach to enterprise information systems management that relies on models of the information systems and their environment. Instead of modifying the enterprise information system using trial and error, a set of models is proposed to predict the behavior and effects of changes to the system. The enterprise architecture models allow reasoning about the consequences of various scenarios and thereby support decision making.

What constitutes a "good" enterprise architecture model is contingent on the purpose the model is intended to fill, that is what type of analysis it will be subjected to. In the case of analyzing maintainability, it is necessary that the models contain information on maintainability, e.g. component source code understandability, documentation quality, change management process maturity etc.

The purpose of this paper is to present an enterprise architecture metamodel for maintainability analysis. The metamodel is intended to be employed in decision situations when prioritizing and choosing change projects. Models based on this metamodel enable the decision maker to: (1) make predictions of the maintainability, i.e. change cost, for different software projects and (2) retrieve information for risk analysis in the change projects.

The remainder of the paper is delineated as follows; section 2 describes the approach of enterprise architecture analysis. The following section, section 3 presents the metamodel for maintainability analysis. Section 4 exemplifies the use of the metamodel. Finally, section 5 summarizes the paper.

## 2. Enterprise Architecture Analysis

The purpose of using enterprise architecture models and conducting analysis of these is to facilitate rational decision making regarding software systems at an enterprise [1]. The paper *A Tool for Enterprise Architecture Analysis* by Johnson et al. [2] presents a notation called abstract models. An abstract model is a kind of metamodel extended with attributes, used when modeling for analysis of different qualities like maintainability. The following subsections present the notation of abstract models and how these can be used in enterprise architecture analysis.

### 2.1 Abstract Models

An abstract model is an enterprise architecture metamodel containing entities and entity relations, augmented with attributes and attribute relations.

Entities are fundamental parts found in most metamodels. Entities represent the objects of interest when modeling, e.g. systems, components, persons, or processes. Entities in abstract models are similar to classes found in UML.

Entity relations connect two entities, e.g. “documentation describes system” or “person is a resource of process”. Entity relations also state the multiplicity of the relationship between the entities, e.g. that one person can be the resource of zero or more processes.

Attributes of an abstract model represent variables related to the entities. UML also have attributes related to entities, but the attributes in abstract models differ from the attributes in UML. Here attributes are discrete random variables that may assume values from a finite domain, e.g. {true, false}, {high, medium low}, or {1, 2, 3, 4, 5}. The attributes represent concepts such as maintainability, complexity, quality of documentation, number of lines of code, cost, or expertise.

Attributes of an abstract model can relate to other attributes with attribute relations. There are two types of relations between attributes in an abstract model, causal relations and definitional relations.

The causal relation indicates a probabilistic dependence between two attributes, in the same way as the causal relations in Bayesian networks [3]. For example “component size affects system complexity” or “high developer expertise has a positive influence on maintenance cost”. This type of relation is graphically represented as a grey arrow.

It is important that the attributes of an abstract model are well-defined. If they are not well-defined, it will be impossible to determine whether there is in fact any truth to the causal relation between the attributes

or not. The definitions of some attributes are considered uncontroversial. However, there is a substantial number of attributes without a common agreement of their definitions. In order to manage these attributes a second type of relation is introduced in abstract models; the definitional relation, graphically represented as a grey arc with a diamond at the end. The definitional relation enable attributes of an abstract model to be defined in terms of other attributes. E.g. “change activity cost is defined as change activity learning cost, development cost, and communication cost”. More information about definitional relations between attributes can be found in [1].

When two attributes relate to each other a change to the value of the source attribute is expected to result in a change in the target attribute. The probability of the values of a target attribute is consequently dependent on the values of its source attributes. This is represented in conditional probability matrices.

The conditional probabilities are an important factor in the presented approach of performing analysis using enterprise architecture models. Thus, based on information on some attributes an expected value of other attributes can be calculated. E.g. measuring component size and developer expertise one can estimate the maintainability of a system.

## 3. Abstract Model for Maintainability

This section presents an enterprise architecture metamodel in the form of an abstract model, c.f. Figure 1. The model is to be employed in software system maintainability analyses for change project cost predictions, i.e. to predict the number of man hours needed to make a certain change. Using the model can also provide the IT decision maker with information for change projects risk analysis.

The presented abstract model is influenced by and based on earlier suggested maintainability frameworks, scientific papers, and books. Such as the work done by Oman et al. [4], Matinlassi et al. [5], Aggarwal et al. [6], Granja-Alvarez et al. [7], Chan et al. [8], and Grubb et al. [9].

The maintainability metamodel presented in this paper is an evolution of previous presented work on the subject of enterprise architecture modeling for maintainability analysis [10].

### 3.1 Entities and Attributes of the Model

IEEE defines maintainability as the ease with which a software system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment [11]. We interpret

ease as the cost of making the modification. Instead of focusing on a single software system or component, as many have done before, this metamodel intends to support maintainability of enterprise systems, i.e. enterprise wide systems of systems. It is no longer appropriate to neglect the fact that with modern system architectures, i.e. systems that are highly coupled together, there are new problems occurring. Problems that need to be taken into account when planning for change.

The abstract model focuses on modeling the systems and the surrounding system environment involved in or affected by the modifications implemented in a *change project*. A change project could typically be a project for adding functionality to systems already in use or integrating a newly bought system with other systems at the enterprise. The main attribute of the abstract model is the *cost* attribute in the entity change project, i.e. the models based on this abstract model are supposed to be used for predicting the cost of a specific change project. The cost of a change project is measured as the number of man hours needed to make the change.

All changes related to software systems need to be formally managed in a controlled manner, this includes changes to be logged, assessed and authorized prior to implementation. For change projects the *change management process* is of most importance. This process need to be *mature* in order to provide the proper support for a project. The maturity attribute in the abstract model is measured by assessing document maturity, metrics maturity, activity maturity, and number of assigned responsibilities.

The entity *change organization* refers to the organizations implementing the software system changes, that is the parties involved in the project. Such as the customer, the vendors, consultants etc. The change organization attributes in the abstract model concerns *the number of architects and developers* involved in the project.

*Architects* are the people in the change project who design and modify the architecture of the enterprise system. *Developers* are the people in the change project who program and modify the source code of the different components in the systems. The entities architect and developer has the attributes *expertise* and *time on project*. Expertise may be measured in terms of change project experience, source code language experience, and system knowledge. Time on project refers to the point in time a person enters the project, more time on a project may increase the level of expertise.

System documentation is one way for architects and developers to understand the systems, the components,

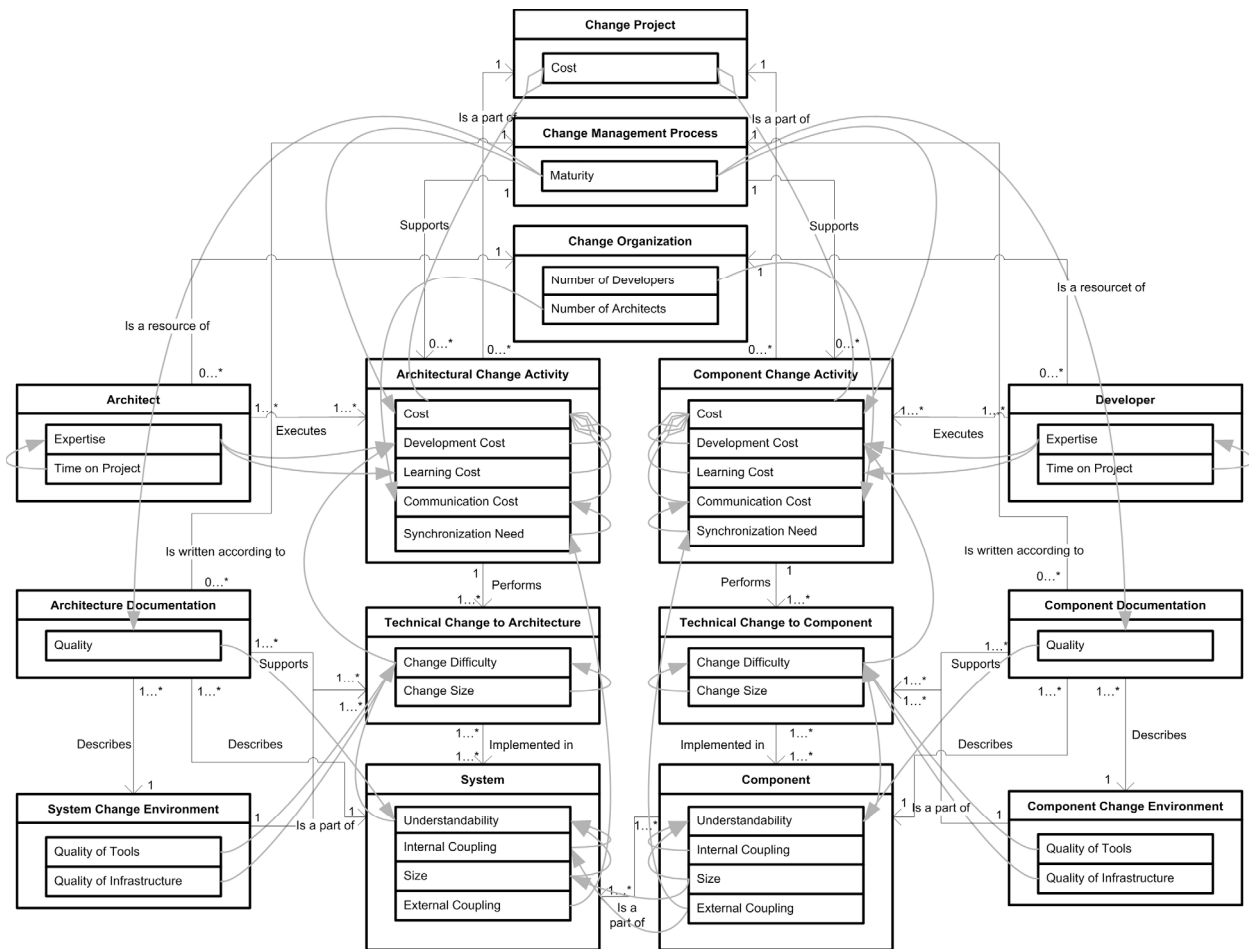
and the environment. Therefore, the *architecture documentation* and the *component documentation* must be of high *quality*, e.g. the documentation must be available, complete, accurate, consistent, and readable. Typical documents could be system rationales, requirements and design specifications, test plans, and data dictionaries.

The *system environment* and the *component environment* contain *tools*. The available tools have the intention of making the modification work easier. Although, this require the tools to be of high *quality* e.g. standardized, well known, and easy to use. The system and component environment also includes infrastructure such as platforms. *Infrastructure quality* can also be measured in terms of standardization level.

Change projects are divided into *architectural change activities* and *component change activities*. Architectural change activities are the activities concerning changes on an architecture level, i.e. involving several systems or components. While, component change activities concern changes to a single component. Both types of change activities have the attributes *cost*, *development cost*, *learning cost*, *communication cost*, and *synchronization need*. The more systems and components involved in a change activity the higher the need of synchronization. The need of synchronization in turn affects the communication cost, which together with the learning cost and development cost defines the cost of the change activities. All costs are measured in number of man hours.

The change activities perform *technical changes to the architecture and the components*. The technical change entities have two attributes, *change difficulty* and *change size*. Change size for architecture is measured in number of components involved in the change and change size for components is measured in number of lines of code involved in the component change. The change size affects the difficulty of implementing the change.

Technical changes are implemented in either a *system* or a *component*. There are four attributes contained in these entities, *understandability*, *internal coupling*, *size*, and *external coupling*. Component external coupling concern the relations to other components, this affects the system internal coupling, while system external coupling concerns the relations to other systems. Component size is measured in number of lines of code, and system size is an aggregate of all its components number of lines of code. Component internal coupling is the interdependence within the component.



**Figure 1. The maintainability abstract model for change project cost predictions.**

The abstract model for change project cost predictions is presented in Figure 1.

#### **4. Modeling and Analyzing Maintainability - A Fictional Example**

This section illustrates the employment of the presented maintainability abstract model as a prediction tool in a decision situation at a fictional company.

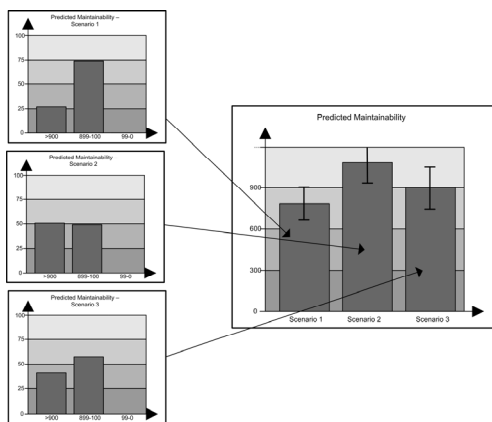
As a result of almost a decade of mergers and acquisitions on the deregulated European electricity market, EURON is today one of Europe's largest energy companies, acting in all parts of the electricity value chain. The company has a large amount of IT systems required to ensure a non-interrupted flow of electricity within their distribution network. There is a CIO making decisions related to IT; the CIO is constantly involved in, and supervises, the evolution of the enterprise's information systems.

EURON's geospatial network system, GeoNet, needs some modifications due to changed requirements. One possible change, scenario 1, is to integrate the facilities record application in GeoNet with the network calculation system Netto. The CIO wants to compare the effort of this change, scenario 1, with other possible changes, scenario 2 and 3, in order to choose the change requiring the least effort. Before the change project starts the CIO would also like to get a good information base for project planning. The CIO and a project manager choose to use the maintainability abstract model proposed in this paper.

When collecting information for the models, one important issue that must be taken into consideration is the credibility of the source and the gathered information. Low credibility in several parts of the model may lead to high uncertainty in the final result, making it difficult for the CIO to make a rational decision. Often it is very difficult and expensive to collect perfectly credible information. Since the attributes in the abstract model are based on the

Bayesian networks formalism this credibility variation can be handled.

The created model based on the maintainability abstract model provides the CIO and the project manager with valuable information; both the expected cost of the change projects (scenarios) and what parts of the projects that are most likely to have a large influence on the cost. As we can see in Figure 2, scenario 1 has the lowest predicted maintainability cost. It is now possible for the CIO to choose scenario 1 or to elaborate with the entities in the models in order to try to lower the costs more, for example involving other developers at the company, hiring new personnel, improving the change management process, or finding better tool support for the project.



**Figure 2. The predicted maintainability.**

## 5. Summary

This paper presented an enterprise system maintainability analysis metamodel in the form of an abstract model. The abstract model consists of entities with accompanying attributes that can be used to create enterprise architecture models. The information contained in these models support quantitative analysis of maintainability. IT decision makers employing the model will be able to make predictions of the change cost for different software projects and retrieve information for risk analysis in these change projects. Thus, enabling IT decision makers to prioritize and choose change projects in their project portfolios.

## 6. References

- [1] P. Johnson and M. Ekstedt, *Enterprise Architecture – Models and Analyses for Information Systems Decision Making*, Studentlitteratur, 2007.
- [2] P. Johnson, E. Johansson, T. Sommestad, and J. Ullberg, "A Tool for Enterprise Architecture Analysis", In *Proceedings of the 11<sup>th</sup> IEEE International Enterprise Computing Conference (EDOC'07)*, Annapolis USA, 2007.
- [3] F. Jensen, *Bayesian Networks and Decision Graphs*, Springer-Verlag, 2001.
- [4] P. Oman, J. Hagemester and D. Ash, "A Definition and Taxonomy for Software Maintainability", SETL Report #91-08 TR, University of Idaho, Moscow, ID 83843, 1992.
- [5] M. Matinlassi and E. Niemelä, "The Impact of Maintainability on Component-based Software Systems", In *the Proceedings of the 29th EUROMICRO Conference "New Waves in System Architecture" (EUROMICRO'03)*, 1089-6503/03, IEEE, 2003.
- [6] K. Aggarwal, Y. Singh, and J.K. Chhabra, "An Integrated Measure of Software Maintainability" In *the 2002 Proceedings Annual Reliability and Maintainability Symposium*, 0-7803-7348-0/02, IEEE, 2002.
- [7] J.C. Granja-Alvarez and M.J. Barranco-García, "A Method for Estimating Maintenance Cost in a Software Project: A case study, Software Maintenance", *Research and Practice*, Vol. 9, 161-175, John Wiley & Sons, Ltd, 1997.
- [8] T. Chan, S.L. Chung and T.H. Ho, "An Economic Model to Estimate Software Rewriting and Replacement Times", In *the IEEE Transactions on Software Engineering*, Vol. 22, No. 8, 0098-5589/96, 1996, IEEE.
- [9] P. Grubb and A. Takang, *Software Maintenance – Concepts and Practice*, Second Edition, World Scientific Publishing, 2003.
- [10] R. Lagerström, "Analyzing System Maintainability Using Enterprise Architecture Models", In *Journal of Enterprise Architecture*, 2007.
- [11] IEEE Standard Glossary of Software Engineering Terminology, IEEE Std 610.12-1990, ISBN 1-55937-067-X, 1990.