Software Migration Project Cost Estimation using COCOMO II and Enterprise Architecture Modeling

Alexander Hjalmarsson[1], Matus Korman[1] and Robert Lagerström[1],

[1] Royal Institute of Technology, Osquldas v. 10, 10044 Stockholm, Sweden
alehja@kth.se
{matusk, robertl}@ics.kth.se

Abstract. Large amounts of software are running on what is considered to be legacy platforms. These systems are often business critical and cannot be phased out without a proper replacement. Migration of these legacy applications can be troublesome due to poor documentation and a changing workforce. Estimating the cost of such projects is nontrivial. Expert estimation is the most common method, but the method is heavily relying on the experience, knowledge, and intuition of the estimator. The use of a complementary estimation method can increase the accuracy of the assessment. This paper presents a metamodel that combines enterprise architecture modeling concepts with the COCOMO II estimation model. Our study proposes a method combining expert estimation with the metamodel-based approach to increase the estimation accuracy. The combination was tested with four project samples at a large Nordic manufacturing company, which resulted in a mean magnitude of relative error of 10%.

Keywords: Software migration estimation, Enterprise architecture modeling, Software engineering, Expert estimations.

1 Introduction

When having a software product portfolio spanning over hundreds of legacy systems, maintenance becomes a problem. Expensive hardware as well as lack of experienced developers in the environment drives the cost of maintenance each year. These legacy systems are often crucial to the businesses and cannot be phased out without proper replacement [1].

Even though new computing technologies have emerged on the market, a considerable amount of software still runs on legacy systems. It is estimated that around 200 billion lines of Cobol code are running in live operation and that 75% of the world's business data are processed in Cobol [2,3]. With an estimated shortfall in Cobol developers in the 2015-2020 timeframe, as the older generation leaves the workforce, it is imminent that migration from the legacy mainframes becomes a priority for many organizations [3]. There are many difficulties involved in the migration process. Understanding the design and functionality of the legacy systems may be troublesome due to the fact that many of these systems have poor, if any, documentation. Because of this, interaction from a system expert is often required [4]. These experts need to analyze the old systems to create accurate requirement specifications regarding technical functionality. This documentation is crucial for the developers and architects involved in the migration process.

Because of the importance of these systems the replacement often needs to suit both new business objectives while maintaining functionality for legacy systems that have not yet been migrated. These factors all come into play when estimating the cost of a migration software project. A case study made by [5] showed that as much as 72% of 145 studied maintenance projects used expert opinion as method for estimating software development costs. Another survey showed that out of 26 studied industrial projects 81% were based on expert estimates [6]. One of the problems with expert estimates is that these can be strongly biased and misled by irrelevant information, which can lead to over-optimism and inaccurate estimations. This often cause project over-runs and may be avoided with an unbiased estimation model [5].

There are claims that a combination of estimates from independent sources, preferably applying different approaches, will on average improve the estimation accuracy. Research has shown that a combination of model and expert estimates produces up to 16% better than the best single decision [7].

This paper proposes a metamodel based on the ArchiMate modeling language [8,9] combined with the COnstructive COst MOdel II (COCOMO II) [10]. In our case study we found that the estimation capabilities of the proposed metamodel together with expert estimation is acceptable. Therefore, we suggest that the metamodel should be used as a complement to expert estimations in order to provide more accurate assessment of migration projects.

The remainder of this paper is structured as follows: Section 2 describes COCOMO II; Section 3 presents enterprise architecture modeling; Section 4 describes the proposed estimation metamodel; Section 5 presents the case study; and Section 6 concludes the paper.

2 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 [10].

In the COCOMO II model, the final cost in person-months (*PM*s) is calculated as:

$$PM = A * Size^{E} \prod_{i=1}^{n} EM_i \, ; \qquad \textbf{(1)}$$

Where A is a calibration constant that depends on the organizations practices and the type of software migrated. E is a constant used to scale projects depending on size. E

reflects the fact that cost and size are not perfectly linear. *EMs* are so called Effort Multipliers.

2.1 Scale Factors

The constant *E* is derived using the following formula:

$$E = B + 0.01 \sum_{j=1}^{n} SF_j \ ; \qquad (2)$$

Where *SF*s are five scale factors. These are precedentedness, development flexibility, architecture/risk resolution, team cohesion, and process maturity. Boehm et al. [10] selected these five factors that describe economies or diseconomies of scale in software projects. This is based on the theory that depending on these variables, the productivity in the project can increase or decrease as it gets larger.

2.2 Effort Multipliers

COCOMO II [10] contains seventeen so called Effort Multipliers (*EM*). These cost drivers affect the software development project in either positive or negative way. The *EM*s are divided into four categories: product factors, platform factors, project factors and personnel factors. They each have a different set of factors within their respective category. The product factors are; required software reliability (RELY), database size (DATA), product complexity (CPLX), developed for reusability (RUSE), and documentation match to life-cycle needs (DOCU). The platform factors are; execution time constraint (TIME), main storage constraint (STOR), and platform volatility (PVOL). The personnel factors are; analyst capability (ACAP), programmer capability (PCAP), personnel continuity (PCON), applications experience (APEX), platform experience (PLEX), and language and tool experience (LTEX). The project factors are; use of software tools (TOOL), multisite development (SITE), and requirement development schedule (SCED).

3 Enterprise Architecture Modeling

Enterprise architecture analysis has emerged during the last decade as an approach to assess different types of non-functional requirements in a company. Migration projects are common projects in an enterprise today, thus including cost estimation for these projects with enterprise architecture could appeal to architects. Research in the area has proposed a framework of enterprise architecture analysis using ArchiMate and a computational model "The Predictive, Probabilistic, Architecture Modeling Framework" (P^2AMF) [11]. P^2AMF can enable calculation on entities in for instance an ArchiMate model. This framework will be the basis of the metamodel used to enable COCOMO II estimations.

3.1 ArchiMate

ArchiMate is a modeling language intentionally resembling the Unified Modeling Language (UML) [8,9]. The reason of using ArchiMate as the basis of graphical notation framework is due to its generality, making it possible to extend existing metamodels with change project estimation as well as providing a solid ground for future adaptions.

The ArchiMate language consists of three core concepts, namely the active structure, passive structure, and behavioral elements. The passive structure elements are elements on which behavior is performed while the active structure is the entity performing the behavior. These concepts are then specialized in each of the three layers specified in ArchiMate [8,9]; the business layer that offers products and services to external customers, the application layer that supports the business layer with application services which are realized by software applications, and the technology layer containing the infrastructure services needed to run applications, realized by computers, communication hardware and system software. The classes found in ArchiMate is for instance; business process, software application, and infrastructure service.

3.2 The Predictive, Probabilistic, Architecture Modeling Framework (P2AMF)

The Predictive, Probabilistic Architecture Modeling Framework (P^2AMF) is a generic framework for system analysis [11] based on OCL and used in order to describe expressions in the Unified Modeling Language (UML). P^2AMF is fully implemented in the Enterprise Architecture Analysis Tool (EAAT) [12,13]. The framework has been utilized to calculate the formulas in the COCOMO II model accordingly.

The end result of this would be that the algorithmic formula used in the model would have a probability distribution indicating the probable cost range of the project rather than a specific mean value. This, in combination with the ArchiMate language, provides a strong basis for using the P^2AMF for cost estimation. However, due to space limitations we have not made use of the probability distributions in this paper.

4 The Proposed Estimation Metamodel

This section presents the metamodel for migration project cost estimation. The metamodel is heavily influenced by COCOMO II [10] and the previously proposed metamodel by [14] and [15]. The most relevant parts of COCOMO II are included in the metamodel proposed while Lagerström's previous work has served as an influence and guideline for the metamodel construction and is thus left out of this description.

ArchiMate is in general used to describe the layers in enterprises' architectures and to for example show what applications are used in what business processes. ArchiMate is tailored for describing as-is and to-be scenarios [8,9]. In this paper we present a specialization of ArchiMate that handles project specific factors. The project specific metamodel elements are then combined with the regular ArchiMate metamodel classes to calculate the migration cost estimate.

The combined metamodel contains the seventeen effort multipliers as well as the five scale factors in a combination. The metamodel differentiates between the three ArchiMate layers as well as the new project specific metamodel classes (see Fig. 1): the business layer (in red) contains the class "*Personnel*;" the application layer (in green) contains the classes "*ApplicationComponent*," "*ApplicationFunction*," and "*ApplicationService*;" the infrastructure layer (in yellow) contains the class "*InfrastructureService*;" and the project entities (in blue) are "*SoftwareDevelopmentProcess*," "*SoftwareDevelopmentProject*," "*Activity*," "*Change*," and "*EffortDivisor*."

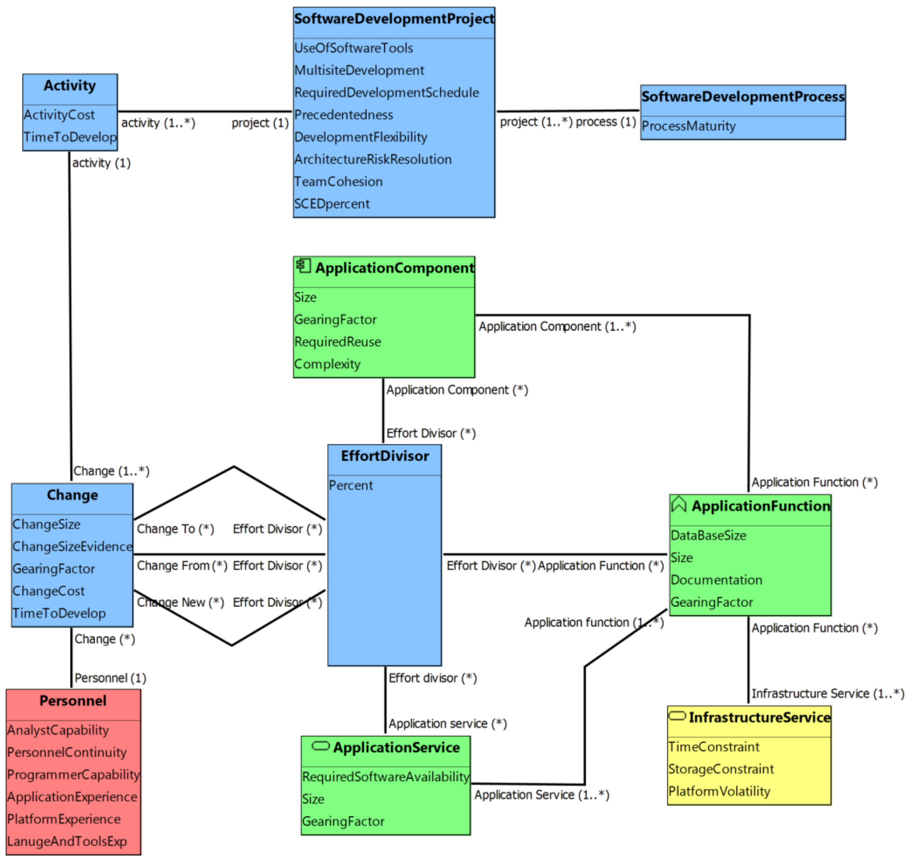

Fig. 1. The proposed metamodel for software migration cost estimation.

5 The Case Study

Our study was conducted at a large Nordic manufacturing company. The data points used in order to validate and calibrate the metamodel are projected as having been closed during the last six months and satisfy the constraint of having > 2000 SLOC produced in the project. The data was collected through interviews with managers, developers, and architects in the projects. Project reports were also used to validate the information elicited and as a source of the project costs (effort in person-hours/man-months). In total we looked at four different migration projects. Due to space limitation we provide some more details regarding Project B (below) before presenting the analysis and results. The complete study can be found described in [16].

5.1 Project B

This project was initiated for the purpose of replacing an old application with a new one running on the company's standardized platform with included support and development agreements. The old application was based on old technology and could not run on modern PC's such as the ones based on the x64 architecture. The software is used to determine variables of the propeller shaft used in vehicles produced by the company. It is only used by the experts in the area and the old application did only run on one PC. Overall, the project was deemed successful. Deviations in the project schedule occurred due to the complexity in the algorithms that were implemented. The project utilized a software development method working iteratively in sprints with demonstrations to customers after each of the sprints. The project had an 18% overrun of the estimated budget due to new requirements added to the migrated version of the software, which increased the scope of the project. The size of Project B was straight forward as it only consisted of migrating one application. The project resulted in 5,500 SLOC developed with the .NET platform. Table 1 presents the data for Project B.

Table 1. Data for Project B.

Scale Factors	**Rating**	**Effort Multipliers**	**Rating**
PREC	NOMINAL	RELY	LOW
FLEX	LOW	DATA	VERY HIGH
RESL	HIGH	CPLX	VERY HIGH
TEAM	VERY HIGH	RUSE	VERY HIGH
PMAT	HIGH	DOCU	VERY HIGH
		TIME	NOMINAL
Factors:		STOR	NOMINAL
	$E = 1.0582$	PVOL	NOMINAL
	$EM_{project} = 0.774$	ACAP	VERY HIGH

$EM_{product} = 2.23$	PCAP	VERY HIGH
$EM_{platform} = 1$	PCON	VERY HIGH
$EM_{personell} = 0.32$	APEX	HIGH
$EM_{tot} = 0.55$	PLEX	HIGH
Actual	LTEX	HIGH
$PM_{act} = 17.42$	TOOL	HIGH
$SLOC = 5{,}500$	SITE	VERY HIGH
	SCED	NOMINAL

5.2 Validation Method

The validation consists of measuring the accuracy of the model. The accuracy is measured by using the Mean Magnitude of the Relative Error (*MMRE*) and the Magnitude of the Relevant Error (*MRE*) [17].

$$MMRE = \frac{1}{n}\sum_{i=0}^{n} MRE_i \ ; \qquad (3)$$

$$MRE = \frac{|E - \hat{E}|}{E} \ ; \qquad (4)$$

Where E is the actual result and $\hat{E}$ is the estimate.

A model has an acceptable accuracy level if 75% of the projects' estimations are higher or equal to 75% [17]. This is called the prediction quality (*PRED*) and has been used frequently when comparing models and methods within the area of software estimation [14,18]. The prediction quality formula (formula 5) where n is the complete set of projects and k is the amount of projects that have greater or equal accuracy as q.

$$PRED(q) = \frac{k}{n}. \qquad (5)$$

An acceptable accuracy level for a model can be denoted *PRED*(0.25) = 0.75, meaning that 75% of the projects shall be within 25% of the actual result.

5.3 Accuracy

Even before calibration the model conforms rather well to the data gathered. The two largest projects, Project A and C are within the predictive quality margin of 25% (16% and 4%). Project B is not estimated accurately and has a *MRE* of 44%. The model underestimates the effort needed for the project which partly may be because of the additional effort needed due to the problems found in the old application that was migrated.

Compared to the expert estimates the model produces competitive estimates. In the table the mean relevant error has been computed with four different measures. These are the model and expert estimates as well as two combinations of them. The two combinations are the result of the optimal combination between model and expert estimates for the specific purpose. Optimal predictive quality (*Opt. pred*) ensures that all projects are within 25% of the real effort outcome. The optimal mean relevant error (*Opt. MRE*) uses the combination that gives the lowest average *MRE* for the projects.

Table 2. Results before calibration.

Measured in hours				**MRE**			
Project	**Model**	**Estimate**	**Real**	**Model**	**Expert**	**Opt. pred**	**Opt. MRE**
A	14794	9700	12700	16%	24%	15%	0%
B	1494	2140	2648	44%	19%	25%	34%
C	7748	6500	8060	4%	19%	16%	10%
D	1315	918	1209	9%	24%	17%	6%
			Mean MRE	**18%**	**22%**	**18%**	**12%**

Opt. pred is using 24% model and 76% expert. *Opt. MRE* is using 59% model and 41% expert. Table 2 shows that heading for the optimal predictive quality in the model would lower the mean magnitude of relevant error, while the optimal *MRE* achieves a very good mean magnitude of relevant error. From the result it also can be seen that by combining the expert judgments with the model both increases the predictive quality as well as the *MMRE*. This is in line with previous research [7].

5.4 Calibration

Calibrating COCOMO II with organizational specific data typically results in better estimates [10]. One way of calibrating COCOMO II to existing project data is by using the multiplicative constant *A* (see [10,16] for the exact calibration equations). The local calibration usually improves the prediction accuracy due to the use of subjective factors in the model. Further, the lifecycle activities in the projects covered by COCOMO II may differ from the ones in the particular organization [10].

The calibration resulted in an increased value of the multiplicative constant *A* used in the effort estimation from 2.94 to 3.23. As can be seen in

Table 3, the calibration yields a lower *MMRE* for the model estimation. This is because the calibration is minimizing the sum of squared residuals in log space rather than the *MRE*. *Opt. pred* was achieved using 31% model and 69% expert, while *Opt. MRE* was achieved by using 46% model and 54% expert.

Table 3. Results after calibration.

Measured in hours				MRE			
Project	**Model**	**Estimate**	**Real**	**Model**	**Expert**	**Opt. pred**	**Opt. MRE**
A	16250	9700	12700	28%	24%	8%	0%
B	1640	2140	2648	38%	19%	25%	28%
C	8510	6500	8060	6%	19%	12%	8%
D	1445	918	1209	19%	24%	11%	4%
			Mean MRE	**23%**	**22%**	**14%**	**10%**

6 Discussion and Conclusions

The results of the case study validates that the combination of COCOMO II with the ArchiMate modeling language works as predicted and that the model estimates are on par with the managers at the case study company. The combination between model and expert estimates performs far better than single selections of model or expert estimations. Without calibration, optimal *MMRE* strategy achieved a *MMRE* of 12% with *PRED*(.25) = 75%. When adding the constraint of *PRED*(.25) = 100%, the *MMRE* rose to 18% which was slightly better than the expert estimates (22%) and on par with the model (18%).

One question that might arise is: Why combining EA and COCOMO II and not only use COCOMO II? As we see it, there is a strength of using EA models as input together with project specific data. ArchiMate as-is and to-be models that already contain information can easily be re-used for every software migration project and the project specific information is the only part that needs to be up-dated. Also, many companies today struggle with maintaining their EA models since new projects alter the as-is architecture continuously. With this approach one could align the as-is and to-be models with all the on-going projects and automatically update the models once the projects are finished. Also, for architects it provides an instrument to work with when creating to-be models and assessing if future scenarios are appropriate for change projects.

In this paper we have presented a metamodel for software migration project estimation. The metamodel was constructed based on metrics from COCOMO II, modeling elements from ArchiMate, and an analysis engine of P^{2}AMF. The metamodel was tested in four cases at a large Nordic manufacturing firm. Our results show that the metamodel itself performs rather well but as COCOMO II suggests it performs even better when calibrated with data from the company under analysis. In software cost estimation research it has been shown that model estimates and expert estimates complement each other in a good way and that the combination often outperforms the two approaches. This was also the case in our study. Therefore, we conclude that our proposed metamodel is useful, especially after company specific calibration and in combination with expert estimates.

References

1. Bennet, K.: Legacy Systems - Coping with Stress. IEEE Software 12(1), 19--23 (1995)
2. Datamonitor: COBOL - Continuing to Drive Value in the 21st Century. London: Datamonitor (2008)
3. Barnett, G.: The Future of the Mainframe. London: Ovum (2005)
4. Bisbal, J., et al.:A Survery of Research into Legacy System Migration. Dublin: Trinity College Dublin (1997)
5. Kitchenham, B., Pfleeger, S. L., McColl, B., Eagan, S.: An Empirical Study of Maintenance and Development Estimation Accuracy. Journal of Systems and Software 64(1), 57--77 (2002)
6. Molokken, K., Jørgensen, M.: A Review of Software Surveys on Software Effort Estimation. Empirical Software Engineering, 223—230 (2003)
7. Blattberg, R. C., Hoch, S. J.: Database Models and Managerial Intuition: 50% model + 50% manager. Management Science 36(8), 887—899 (1990)
8. The Open Group: ArchiMate 1.0 Specification. http://pubs.opengroup.org/architecture/archimate-doc/ts_archimate/
9. Lankhorst, M.: Enterprise Architecture at Work - Modelling, Communication and Analysis. Springer. Third Edition (2013)
10. Boehm, B., et al.: Software Cost Estimation With Cocomo II. New Jersey: Prentice Hall (2000)
11. Johnson, P., Ullberg, J., Buschle, M., Franke, U., Shahzad, K.: P^2AMF - Predictive, Probabilistic Architecture Modeling Framework. In: Proc. of International IFIP Working Conference on Enterprise Interoperability Information, Services and Processes for the Interoperable Economy and Society (2013)
12. The Enterprise Architecture Analysis Tool, www.ics.kth.se/eaat
13. Buschle, M., Ullberg, J., Franke, U., Lagerström, R., Sommestad, T.: A Tool for Enterprise Architecture Analysis using the PRM Formalism. Information Systems Evolution, 108--121 (2011)
14. Lagerström, R., Johnson, P., Höök, D.: Architecture Analysis of Enterprise Systems Modifiability: Models, Analysis, and Validation. Journal of Systems and Software 83.8, 1387--1403 (2010)
15. Österlind, M., Lagerström, R., Rosell, P.: Assessing Modifiability in Application Services using Enterprise Architecture Models - A Case Study. In: Proc. of Trends in Enterprise Architecture Research and Practice-Driven Research on Enterprise Transformation. Springer Berlin Heidelberg (2012)
16. Hjalmarsson, A.:Software Development Cost Estimation using COCOMO II based Meta Model. Master Thesis. The Royal Institute of Technology, Stockholm, Sweden. XR-EE-ICS 2013:005 (2013)
17. Conte, S. D., Dunsmore, H. E., Shen, V. Y.: Software Effort Estimation and Productivity. Advances in Computers. Academic Press, Inc, 1--59 (1985)
18. Kemerer, C. F.: An Empirical Validation of Software Cost Estimation Models. Communications of the ACM 30.5, 416--429 (1987)