Towards an Estimation Model for Software Maintenance Costs

Irene Buchmann
Archimetrica
Wiesbaden, Germany
irene.buchmann@archimetrica.com

Sebastian Frischbier
Databases and Distributed Systems Group
Technische Universität Darmstadt
Darmstadt, Germany
sebastian.frischbier@dvs.tu-darmstadt.de

Dieter Pütz
IT Service Management
Deutsche Post AG
Bonn, Germany
d.puetz@deutschepost.de

Abstract—Today there is no best practise method available to effectively estimate the maintenance costs of historically grown large-scale software landscapes. Most cost estimation models are either not generalizable due to highly specialized scenarios or too abstract to be implemented in practice. In this paper we introduce a multi-level approach to create transparency, estimate costs realistically based on current spending and establish a method for sustainable cost-control. At the heart of our approach is the deduction of meaningful indicators for estimating current and future maintenance efforts. We present the first version of a statistical cost estimation model being implemented at Deutsche Post MAIL as a baseline for contract negotiations with providers.

Index Terms—Software maintenance; software measurement; prediction models; cost estimation; regression models; case study.

I. INTRODUCTION

Deutsche Post AG is Europe’s largest postal provider, processing mail and parcel in Germany as well as worldwide. Deutsche Post MAIL division delivers about 70 million letters and 2.5 million parcels to 29 million households in Germany each working day [1]. To provide logistical services at this scale, the underlying processes are supported by an extensive and heterogeneous application landscape consisting of more than 150 applications. These applications are developed, operated and maintained by several external providers, costing several hundred million euros each year. Software maintenance represented 26% of the total IT budget in 2009. Hence IT Service Management (ITSM) at Deutsche Post MAIL is tasked with managing the efforts of the external providers effectively.

The management of software maintenance in the outsourced environment of Deutsche Post comprises four areas:

1) Adapting and adding new functionality to existing applications: With requirements, ownership or technology changing, functionalities have to be altered or added to existing applications. The spectrum of change ranges from adding new interfaces or substituting frameworks to merging whole applications. While the operational advantages and drawbacks of those decisions are often recognized, their impact on future maintenance efforts is often not clear and easily quantifiable.

2) Planning maintenance for new applications: Currently there is no good method available to calculate the maintenance costs for applications currently under development. One common rule of thumb is to estimate the maintenance costs as x% of the development costs. This approach does not take into account the systems’ real maintainability and therefore is only a rough estimation.

3) Sourcing strategies: Providers are sometimes tempted to offer dumping prices to win the contract without being able to deliver at these conditions. To avoid the consequences of such dumping (e.g. hidden additional costs, complete drop out of the chosen provider due to overloading) indicators and reference values for assessing each bid are needed.

4) Managing call for bids: Although competitive bidding is a proven instrument for achieving results with a good value for money ratio, the process of bidding itself is too costly to apply it comprehensively. Thus tools and indicators for identifying those applications gaining the most from bidding are needed.

To address these areas in both prediction and assessment, transparency of maintenance costs is essential. In historically grown application landscapes, such as Deutsche Post MAIL, this is difficult due to the following reasons:

1) The line between efforts for development, operations and maintenance is often blurred due to lacking or conflicting definitions.

2) Non-uniform data due to: a) non-standardized contracts (i.e. different contract periods, daily rate or services included); b) several information sources based on different data models and semantics covering separate aspects of application characteristics; c) changing application-ownership over time due to mergers, acquisitions, transitions or reorganization.

3) A provider’s proposition for maintaining a given application is based on an individual pricing model that does not always reflect the genuine effort.

The situations described above point out the need for objective metrics and methods to: (a) characterize and compare different applications in terms of maintainability; (b) assess intended changes; (c) compare bids of different providers.

We believe that there are three tasks at the heart of this matter: First, identify factors having a meaningful impact on
maintenance cost to allow a reliable prediction. Second, define metrics to measure these factors and derive key performance indicators (KPI). Third, construct an arithmetic cost estimation model that is easy to understand by all stakeholders involved and is based on these metrics.

The remainder of this paper is structured as follows: Section II reviews related work both in academia and industry. In Section III we introduce our proposed approach to identify meaningful indicators. The results of our analysis as well as the key elements of our statistical cost estimation model are presented in Section IV. In Section V we discuss our findings and provide an outlook.

II. BACKGROUND: ESTIMATION MODELS FOR SOFTWARE MAINTENANCE

A. Types of Models

Depending on the perspective of either provider or consumer, estimation models serve quite different purposes:

1) **Cost Estimation Model.** To assess offerings for software maintenance the service consumer may use a stand-alone cost estimation model. Combining meaningful and objective criteria for cost drivers allows a rough prediction of maintenance costs as a baseline for negotiation. Thus the generalized form is Cost estimation = \( f(\text{factors influencing maintenance costs}) \) with \( f \) being an algebraic function to combine and weight individual metrics describing meaningful factors.

2) **Pricing Model.** The service provider uses a pricing model to determine the price level of the proposed maintenance service. A simple pricing model consists of two main components: \( \text{price} = \text{cost estimation} + \text{profit margin} \). The profit margin depends on many parameters (e.g., pricing politics, general market position, negotiation situation). The real cost estimation is hard to estimate.

In consequence, a realistic cost estimation model serves both service provider and service consumer. In search for available best practices we turned to reviewing related work both in academia and industry.

B. Related Work

In literature, the field of estimating software maintenance costs is not as widely covered as, for example, estimating development costs. Fortunately we could rely on valuable contributions such as [2]–[15]. Especially the studies by Sneed [11], [12] and Jørgensen [7] provided us with a sound basis for our approach by identifying a) the kind of factors being critical to the success of a maintenance operation and b) evaluating the precision of different types of arithmetic models. Nevertheless, most models and approaches proposed were either not easily generalizable due to highly specialized scenarios, too abstract to implement, or not meeting our requirements. In particular, the challenge to predict maintenance costs for a huge number of heterogeneous applications turns out to differ much from estimating maintenance cost/benefit for a single application or a single system being under development.

The former challenge requires quite different applications to become comparable while the latter allows focusing more on the characteristics of an application. As most of the studies analyzed what makes software hard and expensive to maintain we focused on examining cost drivers to abstract from a single system. The main cost drivers identified in the literature are: a) size of application; b) complexity of application; c) number of components (e.g., modules, databases, programming languages, frameworks); d) interfaces to surrounding systems; e) changes (frequency and intensity); f) developers experience; g) code quality.

This categorization is a good first step. However, there is no generalizable approach available to identify those factors and their metrics for a heterogeneous application landscape.

III. EMPLOYED APPROACH

We introduce a multi-level approach to manage software maintenance costs in heterogeneous application landscapes. Our approach consists of the following three phases: i) create data transparency; ii) examine current spending on maintenance; iii) optimize cost/benefit sustainably. At the heart of phases two and three we propose a cost estimation model based on key indicators identified by statistically analyzing the dataset created in phase one.

A. Phase I: Create Data Transparency

Although consistent data is assumed quite often in theoretical work, assembling consistent data proves to be a constant challenge in practice. The steps followed to create data transparency are:

1) Define the tasks inherent to the four categories of maintenance according to ISO/IEC 14764 and [16], [17]: corrective, adaptive, perfective and preventive.
2) Standardize contracts and KPI reporting.
3) Track efforts based on standardized KPI reporting.

B. Phase II: Examine Current Spending on Maintenance.

1) Identify main influencing variables for cost prediction and estimation.
2) Relate variables to estimate current spending.
3) Create a cost estimation model.

C. Phase III: Optimize Cost/Benefit Sustainably

1) Use the cost estimation model to identify high cost applications and providers.
2) Identify hotspots in the application landscape (e.g., low quality applications with many incidents and problems due to software defects).
3) Select providers based on performance.

D. Results and Methodology

The results of phase I are:

1) A unified definition of maintenance efforts.
2) Standardized contracts with KPI reporting for software maintenance.

Phase II results in:
1) A set of factors having a significant impact on maintenance costs.
2) A set of meaningful metrics to measure those factors.
3) An arithmetic cost estimation model based on these metrics.

Phase III results in a set of applications with problems and performance of providers.

We propose a cyclic procedure to incrementally adjust the results of both phases II and III as they tend to change over time. Each iteration of our refinement cycle consists of six steps:

1) Measure indicators, identify and interpret outliers. Try to expand and complete your dataset to solidify further statistical analysis.
2) Compute correlations between indicators and price as well as cross-correlations.
3) Select indicators with best combination of correlation to price, confidence and sample size. Use suitable statistical instruments (e.g. cluster analysis, correlation analysis, regression analysis, factor analysis) for further investigation and information aggregation whenever applicable.
4) Construct linear and non-linear models based on these selected indicators.
5) Evaluate new models in comparison to existing models.
6) Keep models with best fit.

IV. CASE STUDY: CONSTRUCTING A COST ESTIMATION MODEL FOR SOFTWARE MAINTENANCE

We evaluated our proposed approach by implementing it as far as phase III at Deutsche Post MAIL. In this section we briefly describe findings and challenges of this ongoing work.

A. Preparatory Work

As preparation for deriving meaningful factors we started with phase I as presented in Section III by defining software maintenance tasks. We distinguished between basic maintenance (error correction) and auxiliary services (e.g. additional services such as supplementary monitoring, consulting).

Based on this definition and a derived task catalogue we implemented a standardized contract scheme with corresponding KPI reporting for software maintenance. Thus covering more than 150 applications with uniform KPI reports we proceeded with tracking efforts in both basic maintenance and auxiliary services. Having separated all standardized additional tasks from the core task of error correction, basic maintenance costs remained as a solid non-transparent cost block. As basic maintenance still accounted for 62% of all maintenance costs in 2009 we proceeded with analyzing this cost pool further in phase II using statistical analysis.

B. Identified Factors and Metrics

Our dataset covered a total of 136 applications, excluding several applications with too specialized characteristics. An initial set of 25 metrics was identified, representing all three categories of factors presented in Section II (i.e. effort-based, product-based and pricing politics). The initial set consisted of both standard metrics (e.g. number of reported software defects, number of problems, resolution time, lines of code, backfired function points, number of middleware components, number of modules, daily rates) as well as metrics specific to Deutsche Post BRIEF (e.g. application complexity, conformity to technical guideline, level of standardization, code quality).

Having eliminated redundant and interdependent indicators we ranked the remaining metrics based on the number of available pairwise complete observations for each metric. We selected those metrics with highest correlation to the basic maintenance effort, the lowest p-value and a dataset size of $n \geq 22$. The metrics conforming to these restrictions are shown in Table I together with their associated p-value, the correlation coefficient to basic maintenance effort (BME), the number of pairwise complete observations (OBS) as well as the respective category of factor (COF): i) average number of reported software defects (RSD); ii) number of programming languages used (PL); iii) service level (SLA); iv) level of standardization (LST); v) number of interfaces (INT); vi) median of backfired function points (BFP); average number of hotfixes (AHF). Metrics with too few observations were discarded, but we tried to account for them by using other related metrics.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Corr(BME, X)</th>
<th>OBS</th>
<th>P-Value</th>
<th>COF</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSD</td>
<td>0.6390</td>
<td>83</td>
<td>1.47883E-10</td>
<td>EB</td>
</tr>
<tr>
<td>PL</td>
<td>0.8736</td>
<td>27</td>
<td>1.29558E-09</td>
<td>PB</td>
</tr>
<tr>
<td>SLA</td>
<td>0.6408</td>
<td>44</td>
<td>2.11583E-06</td>
<td>PB</td>
</tr>
<tr>
<td>LST</td>
<td>0.7208</td>
<td>30</td>
<td>4.79785E-06</td>
<td>PB</td>
</tr>
<tr>
<td>INT</td>
<td>0.5511</td>
<td>52</td>
<td>1.89962E-05</td>
<td>PB</td>
</tr>
<tr>
<td>BFP</td>
<td>0.5581</td>
<td>41</td>
<td>0.000122764</td>
<td>PB</td>
</tr>
<tr>
<td>AHF</td>
<td>0.6307</td>
<td>23</td>
<td>0.000953836</td>
<td>PB</td>
</tr>
</tbody>
</table>

C. Constructing Regression Models

The selected metrics cover both effort-based and product-based factors (c.f. Table I), factors related to pricing policy could not be measured. Regarding the type of model to be used, Jørgensen already showed that multiple regression models achieve the most accurate predictions [7]. Thus we constructed several linear and multiplicative regression models to test different types of interdependencies. In addition, we simulated decreasing effects by dampening each factor with $\Theta, \Psi, \Psi < 1$.

We strongly believe that both size and complexity of an application influence the maintenance costs significantly based on the actual rate of defects. Thus we based our models $LM_1$-$LM_2$ on: i) basic maintenance effort (BME); ii) number of reported software defects (RSD); iii) number of programming languages (PL) and iv) backfired function points (BFP):
As Table II shows, a multiplicative model of damped factors (MM2) outperforms both simple linear and multiplicative combinations.

<table>
<thead>
<tr>
<th>Model</th>
<th>R²</th>
<th>adj.R²</th>
<th>P-Value</th>
<th>Deg. Freedom</th>
</tr>
</thead>
<tbody>
<tr>
<td>LM1</td>
<td>0.7745749</td>
<td>0.8039782</td>
<td>2.309465e-07</td>
<td>20</td>
</tr>
<tr>
<td>LM2</td>
<td>0.7646609</td>
<td>0.7953573</td>
<td>4.299913e-07</td>
<td>20</td>
</tr>
<tr>
<td>MM1</td>
<td>0.6684998</td>
<td>0.6829129</td>
<td>6.509045e-07</td>
<td>22</td>
</tr>
<tr>
<td>MM2</td>
<td>0.8308293</td>
<td>0.8381845</td>
<td>3.630063e-10</td>
<td>22</td>
</tr>
</tbody>
</table>

**D. Evaluation**

We propose the following explanation: The basic maintenance effort depends on the number of reported software defects multiplied by the average effort to locate and adjust an error within the application. The latter effort depends on both the size and complexity of the application. Moreover, the additional marginal effort for handling an additional reported software defect or backfired function point decreases when already operating with either a large number of reported software defects or huge and complex applications. Thus, dampening each factor is appropriate. Backfired function points are an implementation-independent indicator for the size of an application. The number of programming languages used within an application act as an adequate indicator for complexity. The average number of reported software defects is a good indicator for maintenance effort. Our model covers the three critical contributions (size, complexity and effort) using three readily measurable yet representative parameters.

**V. Conclusion and Outlook**

In this paper we presented a multi-level approach to identify meaningful factors for estimating and controlling software maintenance costs in a large historically grown heterogeneous application landscape. Therefore, we first categorized the challenges IT Service Management has to face when governing extensive software maintenance efforts and pointed out the lack of a generalizable approach in practice as well as in theory. We evaluated our proposed approach by implementing it at Deutsche Post MAIL and presented our findings here. The resulting multiplicative regression model for cost estimation combines both effort-based as well as product-based factors and consists of: i) average number of reported software defects; ii) number of programming languages; iii) median of backfired function points.

As part of our future work we will improve our cost estimation model by extending the data pool to cover additional applications, and further cleansing existing data. Long term Deutsche Post MAIL is interested in cooperating with service providers in the development of future pricing models.

**Acknowledgment**

We would like to thank Alejandro Buchmann and Elke Walz for valuable discussions and support as well as Michael Chromik for doing feasibility studies by applying our approach to a small preliminary dataset as part of his bachelor’s thesis.

**References**


