Towards a maintainability model for business processes

Citation for published version (APA):
Turetken, O. (2013). Towards a maintainability model for business processes: Adapting a software maintainability model. In I. Vanderfeesten, & D. Fahland (Eds.), 1st International Workshop on Communicating Business Process and Software Models (CPSM) (pp. 1-4). Institute of Electrical and Electronics Engineers. DOI: 10.1109/CPSM.2013.6703083

DOI:
10.1109/CPSM.2013.6703083

Document status and date:
Published: 01/01/2013

Document Version:
Accepted manuscript including changes made at the peer-review stage

Please check the document version of this publication:
• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.

Download date: 31. Jul. 2019
Towards a Maintainability Model for Business Processes

Adapting a Software Maintainability Model (Position Paper)

Oktay Turetken
Information Systems Group, School of Industrial Engineering
Eindhoven University of Technology (TU/e)
Eindhoven, Netherlands
o.turetken@tue.nl

Abstract—This position paper introduces the initial steps towards a business process maintainability model. The proposed model is grounded on a software maintainability model, which has been empirically validated and currently being used in practice. The paper briefly discusses the structure of the model, options for the base measures that can be applied in the business process domain, and challenges and future research for the construction and validation of the proposed model.

Keywords—business process maintainability; maintainability model; software maintainability, business process quality metrics

I. INTRODUCTION

With the presumption that the amount of effort needed to maintain a software system is related to the technical quality of the source code of that system, several approaches and models that are based on internal quality metrics have been developed to quantify the maintainability of software systems (see [1] for a survey). These metrics have been successfully used as a guidance to improve software designs to make them easier to maintain.

Software quality metrics have long been exploited in the business process management (BPM) field due the commonalities shared by software applications and (business) processes. In the recent years, several studies have been performed on measuring the quality of BPs ([2], [3], [4], [5]). However, despite several base metrics available in the literature, work on the maintainability of BP models is limited.

Adopting the maintainability definition in the software domain [6], we can define BP maintainability as “the ease with which a single or a collection of business processes can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment”.

Existing research typically focuses on the quality property of a single process model (e.g., [7], [8], [9]). However, changes in the business environment would likely to impact a collection of interrelated process models rather than merely a single model. Understanding the relationships between the process models to better analyze the impact of a change and to ensure the integrity of the entire collection is maintained is utmost important, as it constitutes a key part in the maintenance process. Therefore, in quantifying the maintainability of BP models, we should also consider the maintainability of an entire collection of business process models. This requires the integration of both intra-process and inter-process metrics and indicators both over the process and the collection level.

In this paper, we discuss the applicability of a software maintenance model to indicate the maintainability of a collection of interrelated business process models. We have chosen the SIG (Software Improvement Group) Maintainability Model [10] as the basis, since:

- It has been empirically validated ([10], [11]).
- Unlike many other maintainability models, it allows for the analysis of root causes of poor performance and guides improvement efforts.
- It is pragmatic and relatively easy to use and automate.
- It has a clear reference to an agreed standard—i.e. the ISO/IEC 9126 [12].

II. SIG MAINTAINABILITY MODEL

The ISO/IEC 9126 standard for software product quality [12] distinguishes three perspectives on product quality: internal quality, external quality, and quality-in-use. Internal quality represents the totality of characteristics of the software product from an internal view. Source code metrics serve as good indicators to determine internal quality. External product quality refers to the quality when the software is executed. Quality-in-use represents the user’s view of the quality of the software product when it is used in a specific environment and a specific context of use.

For internal and external quality, the ISO/IEC 9126 categorizes software quality attributes into six characteristics, which are further subdivided into 20 sub-characteristics [12]. One of the six main characteristics is maintainability, which is further broken down into analyzability, changeability, stability and testability.

The SIG maintainability model is a pragmatic model that operationalizes the quality definitions of ISO/IEC 9126 by mapping a selection of source code metrics to the maintainability characteristic and its sub-characteristics [11]. Being based on static analysis of source code, this model
measures maintainability under the perspective of internal product quality. It combines a set of source code metrics into a single rating of the maintainability of a software product.

A. Mapping Source Code Metrics to Product Properties

The SIG Model has a layered structure (Fig. 1) for measuring and rating the quality of a software system in terms of the quality characteristics of ISO/IEC 9126. The first layer consists of the measures based on the source code analysis performed on the software system. The analysis involves basic metrics, such as Lines of Code (LOC), duplicated LOC, McCabe complexity metric, parameter counts and dependency counts. These metrics are programming language independent and collected on the level of command lines, units (e.g. methods or functions) and modules (e.g. files or classes).

On the second layer, the source code measures are aggregated and mapped onto ratings for product properties at the level of the entire software product, such as volume, duplication, and unit complexity. The ratings are in unitless ordinal scale ranging from 1 to 5 (stars). Using such a rating scheme facilitates communication and benchmarking of the product quality at the entire software product level.

For volume and duplication properties, LOC-based metrics are translated into ratings based on predefined threshold values. For example, a system with less than 3% code duplication is considered to be well designed and rated as ‘*****’ (five stars).

For the mapping to the remaining product properties, the model applies a risk profiling technique. Accordingly, the system is partitioned volume-wise into four risk categories: low, moderate, high and very high-risk. For example, a software unit is considered to have a ‘high-risk’ in terms of unit complexity if McCabe number is between 21 and 50. Once the risk level for each unit is determined, first, an aggregation of complexities per unit is performed by counting for each risk level what percentage of lines of code falls within units categorized at that level. For example, if, for a system with 10,000 LOC, the high-risk units together amount to 1200 LOC, then the aggregate number computed for that risk category is 12%. Next, given the risk profile of a system for a particular product quality property, the rating for that property can be computed based on a mapping schema. Following the same example, a system with 10% to 15% of code categorized as high-risk is rated with ‘***’ (two stars out of five).

The thresholds used by the model are founded on the statistical analysis of a set of representative systems [10], expert opinion and works in the literature [13].

B. Mapping Product Properties to ISO/IEC 9126

At the third layer, the model has ISO/IEC 9126 maintainability sub-characteristics that are conceptually associated with product properties. Basically, to arrive at a system-level score, a weighted average is computed of each product property rating that is relevant according to the cross association between product properties and sub-characteristics (as depicted in Fig. 1). For instance, changeability sub-characteristic is associated with duplication, unit complexity and module coupling product properties. A system with ‘***’ for duplication, ‘****’ for unit complexity and ‘****’ for module coupling, will have the weighted average of ‘****’ for changeability sub-characteristic. Details regarding the mapping scheme in the SIG model are available in [11] and [13].

C. Model Validation

For the validation of the model, empirical studies have been conducted to demonstrate a statistical correlation between the quality of software products as measured by the SIG model and the speed at which software issues (of both defect and enhancement type) are solved by the development and/or maintenance teams of these software products [10], [11]. The studies indicated a significant predictive effect of the model for the efficiency of development and maintenance tasks, such as resolution of defects and implementation of enhancements.

III. ADOPTING THE SIG MODEL FOR BUSINESS PROCESSES

There are several challenges in adopting the SIG model to BP maintainability. At this stage, two of these challenges can be considered as vital for an effective adoption. First, representative and well-established measures of BP models should be applied for the replacement of the sources code measures used in the SIG model. SIG uses several thresholds derived from the analysis of a large set of representative software systems, expert opinions and related literature. So, the second challenge is to establish such threshold values for the process models, which requires extensive empirical studies and evidence that is currently vastly limited in the BPM domain (particularly when compared with the maturity of the measurement field in the software engineering domain).

The first challenge is a prerequisite for the efforts to cope with the second one. Therefore at this stage, we focus on the first challenge of selecting and applying the most effective base metrics of BP models to represent the product properties. Here, the product that we consider as a subject matter is a collection of BP models that are interrelated (e.g. through control flow, message exchange, aggregation). We should also note that, in the following discussions, a software unit (module, method, function, etc.) translates into a process model for our purposes.

The remaining paragraphs discuss the base measures that can be applied in representing the properties of a BP model collection (i.e. volume, duplication, unit complexity, unit size, unit interfacing, and module coupling).

![Fig. 1. SIG Maintainability Model [10] maps source code measurements onto ISO/IEC 9126 [12] quality characteristics.](Image)
A. Volume

The total size of a system is a typical factor that has a heavy influence in any measure of maintainability. This is why it is usually revealed as a significant predictor in such models [1]. The larger the system, the more effort it requires for the maintenance, as it becomes harder to understand and analyze. This view is also supported in the BPM domain [14].

Although there are several variations, the predominant metric used to quantify the size of a process model is the total number of activities ([15], [16], [17], [18]). Mendling et al. also proposes the total number of nodes and the diameter metrics (length of the longest path from a start node to an end node in the process model) as alternatives for the size (volume) [3]. Following the SIG approach, the overall volume can be an aggregation of the sizes of units (single process models).

B. Duplication

Unlike volume, duplication is hardly considered as a measure in software maintainability models [1], which makes SIG model peculiar from that perspective. A certain degree of duplication is considered typical for systems; however, unnecessary amounts are damaging to its maintainability [19].

Discovery of duplicate code is fairly straightforward, while it can be challenging in the BPM domain. Existing work on determining a degree of duplication between process models is limited. The measures proposed by Dijkman et al. [20] can be exploited to quantify different aspects of duplication through the notion of process model similarity. Likewise, techniques probed by Ekanayake et al. [21] for retrieving clusters of approximate BP clones taking into account measures such as cluster quality, should be considered.

C. Unit Complexity

The complexity is a well-studied property of systems, which is frequently referred as factor influencing several aspects of maintainability [22], [23]. Various metrics have been proposed in the BPM field to quantify the complexity of process models. For example; Cardoso proposes the Control Flow Complexity (CFC) metric for measuring control-flow complexity of process models [24], Vanderfeesten et al. [25] introduces the cross-connectivity metric, that aims to capture the cognitive effort to understand the relationship between any pair of process model. The study in [26] surveys other important works on the complexity of process models, including [3], [27], and [15].

D. Unit Size

Apart from the overall size (volume) of the BP model collection and complexity per unit (a single business process model), the size of individual units may also explain its level of maintainability. It is fairly intuitive that larger the units, more difficult it is to analyze and test them. The metrics we discussed in section A are readily applicable for quantifying the unit size.

E. Unit Interfacing

In the SIG model, the unit interfacing is measured in terms of the number of parameters declared in the interface of each unit. It is assumed that a larger interface means more parameters to keep track of, which in turn influences maintainability, in particular the stability. Unit interfacing would render into the interfacing of a process model, i.e., the number (and possibly complexity) of data input/outputs that are exchanged with other process models.

The existing literature on the measurement of business process quality considers mostly the control-flow aspect of process models and emphasizes rarely on the data/information aspect of process models (that incorporate data elements usually referred as: work products, messages, business objects, information elements, etc.). The metrics proposed by Canfora et al. [17] (such as NDWPI - the number of input dependences of the work products with the activities in the process) and Rolon et al. [28] offer constructs to define metrics to quantify the unit interfacing property of process models.

F. Coupling

Coupling is another important property that frequently appears among the factors influencing the maintainability of systems [1], [22]. It measures the number of interconnections among software units. SIG model uses the number of incoming invocations for each module (a delimited group of units, e.g., class or file).

Canfora et al. [17] defines activity coupling metric as a ratio of activities to the number of precedence dependences between activities. The coupling metric developed by Reijers et al. [29] (which is further enhanced in [7]) also take the data aspect into account. It counts the overlap of data elements for each pair of activities using a static description of the product. Two activities are ‘coupled’ if they contain one or more common data elements. In addition to these metrics that consider activity coupling, our model should also consider the coupling of process models and take works such as [30] into account for this purpose.

IV. DISCUSSION AND FUTURE WORK

A BP maintainability model would be of interest to different BPM stakeholders, both in practice and research. It will provide guidance to BPM tool vendors for integrating such metrics, and presenting aggregated views from different perspectives. Quantifying the maintainability on the model and repository levels would help particularly business process designers and practitioners- to get insight on the quality of the process models in general, understand deficiencies as well as strengths, and pinpoint parts where improvements are possible. The maintainability model can also be used to predict the efficiency of modeling and maintenance tasks, such as resolution of defects and implementation of enhancements. The predictive power of these models can be further improved by taking several other factors, such as BP designer characteristics (e.g., experience, knowledge level), modeling notation and tool, etc. into consideration.

Nevertheless, the development and the validation of the model require research in several directions. The applicability of current BP quality metrics should be thoroughly analyzed to make sure that they accurately represent the BP properties. New BP metrics may have to be proposed for a better coverage. Additional quality properties such as structuredness (which is validated to play a part in the understandability of BP models [31]) can be incorporated to enhance the correlation.
The empirical studies should be conducted; first, to establish the threshold values and values used for risk profiling; and second, for the overall validation of the model. This requires carefully designed multiple case studies and/or experiments that involve process repositories of different characteristics being subject to diverse types of maintenance tasks. The experiments can be conducted to analyze the effectiveness and efficiency of the maintenance tasks (structurally similar to the studies such as [32], [8]) and may employ statistical analysis techniques to uncover possible correlations between the effectiveness and efficiency of maintenance tasks and the BP quality metrics as measured by the BP maintainability model. In determining thresholds for metrics, Mendling et al. [9] proposes threshold values for diverse BP quality metrics for discriminating models of high and low error probability. Similarly, Sánchez-González et al. [33] proposes threshold values for complexity of BPs. These values as well as the statistical methods used for determining such thresholds can be adapted in future experiments.

The proposed structure of the BPM maintainability model is based merely on internal process quality metrics and does not consider the role of cognitive physiology on the understandability of process models. Future work should also consider underpinnings on the cognitive dimension by following studies such as [34].

REFERENCES