Acta Cybernetica **22** (2015) 135–150.

Business Process Quality Measurement using Advances in Static Code Analysis

Gergely Ladányi*

Abstract

Business process models play an important role in the life of a company. Resemblances between software programs and business processes inspired several researchers to adapt software metrics from the field of static code analysis to help designers to build more effective and understandable processes. This paper aims to add recent advances in software quality measurement such as benchmarking and ISO/IEC 25010 standard based quality models to business process quality measurement. These techniques were proved to be very useful in software engineering both for managers and developers; moreover, they can be easily adopted to business process workflows. We focused on a specific type of flowchart called event-driven process chain (EPC), because in an EPC the activities are very often managed by software systems and our assumption is that the quality of these software systems affects the quality of the EPC itself. The presented business process quality model also uses the quality and test coverage metrics of these software systems besides business process metrics.

Keywords: business process, static code analysis, quality measurement

1 Introduction

A business process is a collection of activities that takes one or more kinds of input and creates an output that has value to the customer [17]. Since it has direct effect on the customer, it is really important to have an up-to-date knowledge about its weak and strong points. Using flowcharts like Event-driven process chain (EPC) is a widespread solution for modelling, analyzing, and redesigning business processes. EPC is commonly used because models described with it are flexible, easy to manage and understand. In this paper we focus on the quality of an EPC and then on the extension of this property to the entire process group hierarchy. The important parts of this approach are based on our earlier probabilistic software quality model, called Columbus QM [6]. It has been developed to evaluate maintainability of Java software systems; although, because of its generality it can be adopted to other languages [11] or business processes.

*University of Szeged, E-mail: lgergely@inf.u-szeged.hu

DOI: 10.14232/actacyb.22.1.2015.9

Earlier work [23] showed that the results of software quality measurement can be adopted to the problem of measuring the quality of business processes because an EPC is actually a simple program, which can be characterized with static code metrics like McCabe's cyclomatic complexity [15]. In this study the quality measurement of business processes is also based on the quality measurement of software systems. Our main contributions are listed in the following:

- If a task behind a function is managed by a software system, it is a reasonable assumption that the quality or test coverage of this software has a serious effect on the function and the EPC itself. First we approximated the weight of each function and then we used these values for aggregating the quality values of the employed software systems to the EPC level.
- In software quality measurement a quality model often uses code metrics as predictors and then aggregates these predictors to higher level characteristics by comparing them with a benchmark. We also created a quality model for business processes based on the ISO/IEC 25010 quality standard. As predictors we used simple metrics calculated from the EPC and the previously aggregated quality metrics of the functions.
- We explain our approach in more detail in a case study with the contribution of a middle-size software company. They were strongly involved in the design phase of the EPC quality model, they helped us to decide which metrics are important, and to what extent.

In this way we can support a company in many ways. Information about higher levels of the quality hierarchy provides a brief overview about the current state of the business configuration, which can support the decision-making of the managers. Quality of an EPC is valuable for designers; it helps them to choose from alternative configuration of systems and workflows. Finally the quality and test coverage of the applications have proved to be very useful for developers and quality assurance technicians. With this approach they can improve the maintainability of those software systems which have the most critical role in business processes.

The paper is organized as follows. In the next section we summarize some related studies. After a brief overview about the structure of the business hierarchies in Section 3 we describe our approach in details in Section 4. Next, in Section 5 we present a case study, which shows how our approach works in a real life situation. In Section 6 we collect some limitations and threats to validity. Finally, in Section 7 we close the study with conclusions.

2 Related work

A business process model which is represented by an EPC shows several similarities with a software product. In fact, business processes and software products have a similar structure: a program is composed of modules or classes, each module consists of statements and statements may contain variables and constants. In the

same way, a business process has activities, each of which is composed of elementary operations, and each operation uses one or more information to produce new information [14].

In the last decade a wide variety of business process metrics have been developed [13] to extend our knowledge about business processes. Vanderfeesten et. al [23] suggested to use the same five categories of software metrics: coupling [25, 24, 14], cohesion [20], complexity [7, 12], modularity [19, 10] and size [7]. We also used size and complexity metrics; however, our future plan is to extend our quality model with more powerful coupling and cohesion metrics as well.

Studies about the usefulness of these metrics showed that they can be used as indicators for various problems in connection with business processes. Vanderfeesten et. al [24] found that strongly cohesive and weakly coupled processes will result in fewer errors during information exchanges and in more understandable activity descriptions. Mendling et. al [16] calculated several EPC metrics and checked them for structural correctness. They found that several metrics have a strong statistical connection with the occurrence of errors, and that most of the metrics increase or decrease error probability as expected.

Only a few studies deal with measuring maintainability of a business processes [9, 22]. The closest research to ours is the adoption of the Software Improvement Group (SIG) [2] quality model to business processes by Oktay Turetken [22]. They also created an ISO standard based quality model for business processes, which contains several nodes and metrics. The model and used metrics are promising, but in its current form it is just a suggestion, not an implementation yet. Our intention is also to extend our model with more metrics to achieve more precise results.

Benchmarking in software engineering is proved to be an effective technique to determine how good a metric value is. Alves et al. [3] presented a method to determine the threshold values more precisely based on a benchmark repository [8] holding the analysis results of other systems. The model has a calibration phase [4] for tuning the threshold values of the quality model in such a way that for each lowest level quality attribute they get a desired symmetrical distribution. They used the $\langle 5, 30, 30, 30, 5 \rangle$ percentage-wise distributions over 5 levels of quality. To convert the EPC metrics into quality indices we also used a benchmark with a large amount of EPC evaluations, but we applied it in a different way. During the calibration, instead of calculating threshold values we approximate a normal distribution function called benchmark characteristic (see Section 4), which is used to determine the goodness of the EPC with respect to a certain metric.

3 Overview

To determine the quality of a business configuration first we need to examine its structure. Each level of a business configuration will require a different kind of calculation.

- Business process: a collection of activities that takes one or more kinds of

input and creates an output that has value to the customer. They can not contain further processes.

- Business process group: container of business processes or other business process groups. Every group is contained by exactly one other group, except the root.

- Business configuration: a process group which is not contained by any other process group. This is the root of the process group hierarchy tree.

- Event-driven process chain (EPC): a flowchart used for modelling business processes. More precisely it is an ordered graph of events and functions with various connectors (AND, OR, XOR) that allows alternative and parallel execution of processes. It is very often used for modelling user interactions of an enterprise resource planning (ERP) application. In this way an EPC is in very close connection with the applications. A widespread solution for designing EPC is a framework called Architecture of Integrated Information Systems (ARIS) [21].

- Function: element of an EPC, it represents usually a user activity.

- Event: element of an EPC, it is just a state which serves as input and the output of a function.

- Connectors: these nodes represent the complex routing rules.

  - OR connector: triggers one, two or up to all of multiple branches.
  - AND connector: activates all subsequent branches in concurrency.
  - XOR connector: represents a choice between the alternative branches.

  An example EPC can be seen in Section 5 in Figure 4.

4 Approach

The structure of the evaluation approach follows the previously introduced hierarchical construction of the business processes. The evaluation starts with the most atomic part of the hierarchy, the functions and its applications and continues with the aggregation of their quality upwards to the business configuration (Figure 1). Calculation of the application and EPC quality uses the Columbus QM [6] approach. The aggregation between the hierarchical levels is done by average, but between the functions and the EPC we used a simulation technique to provide more realistic results. In the next sections we will explain in detail these levels of aggregation and calculation one by one.

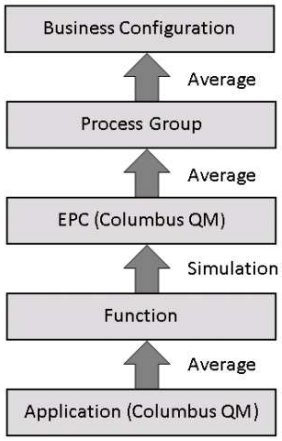

Figure 1: Hierarchical construction of the approach.

4.1 Function quality

The first level of the evaluation process is measuring the quality of the functions. The quality of the each function is calculated directly by the average quality of the applications which manage the function.

The evaluation of these applications was done by our earlier result called the Columbus QM [6]. The quality model in the Columbus QM approach describes the aspects and their weights which should be considered in the calculation. Figure 2 shows the applied quality model which is based on the quality characteristics defined by the ISO/IEC 25010 [1] standard. The white nodes in our approach represent source code metrics that can be readily obtained from the source code. The description of the metrics can be found in Table 1. The black nodes are defined by ISO/IEC 25010 [1] standard. The grey nodes are the inner nodes, they help us revealing the dependencies between the metrics and the nodes defined by the standard. The Columbus QM by comparing the current system with other systems from the benchmark at the end provides a goodness value for each node in the model. This goodness value is scaled into the [0,10] interval and basically means the proportion of all the systems within the benchmark whose goodness values are smaller.

Test coverage also has a key role in the approach. We used a test coverage tool called TestNavigator which instruments the given web application, and then it is capable to monitor the test coverage values in real time. We extended the concept of test coverage to a function. We calculated the overall number of Java methods and the overall number of covered Java methods of the systems together. The test coverage of an EPC function is the ratio of the covered Java methods.

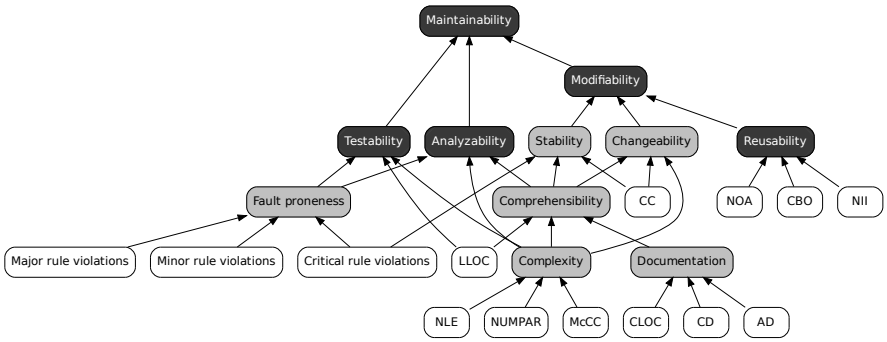

Figure 2: Java quality model

Table 1: The low-level quality properties of our Java model

Metric	**Description**
CC	Clone coverage. The percentage of copied and pasted source code parts, computed for the Java methods of the system.
NOA	Number of Ancestors. Number of classes, interfaces, enums and annotations from which the class is directly or indirectly inherited.
Critical rule violations	The number of critical rule violations in the Java method.
Major rule violations	The number of major rule violations in the Java method.
Minor rule violations	The number of minor rule violations in the Java method.
AD	Api Documentation. Ratio of the number of documented public Java methods in the class.
CLOC	Comment Lines of Code. Number of comment and documentation code lines of the Java method.
CD	Comment Density. The ratio of comment lines compared to the sum of its comment and logical lines of code.
LLOC	Logical Lines of Code. Number of code lines of the method, excluding empty and comment lines.
NUMPAR	Number of parameters in the Java method.
McCC	McCabe's Cyclomatic Complexity of the Java method. The number of decisions within the specified Java method plus 1, where each if, for, while, do...while and ?: (conditional operator) counts once, each N-way (switch) counts N+1 and each try block with N catch counts N+1.
NLE	Nesting Level Else-If. Complexity of the Java method expressed as the depth of the maximum embeddedness of its conditional and iteration block scopes, where in the if-else-if construct only the first if instruction is considered.
NII	Number of Incoming Invocations. Number of other Java methods and attribute initializations, which directly call the method.
CBO	Coupling Between Object classes. Number of directly used other classes (e.g. by inheritance, function call, type reference, attribute reference).

4.1.1 Simulation

Instead of simple average we used a more realistic weighted sum model to aggregate the function quality values to the EPC level. To do this we need to estimate the

weights of each function, we need to provide a rank for them. Bakota et. al [5] to determine analytically the test cases for business processes generated every possible path from the model, which for a large, complex model can be too much. They used filter parameters specified by the model designer to reduce the number of possible paths. To avoid filter parameters and computational issues, we chose a heuristic algorithm because parallel threads and possible cycles in the EPC graph makes it hard to calculate these weights analytically in practice. Our algorithm is a special version of the random surfer algorithm, which is the basic idea behind the page rank algorithm as well [18]. Since each random surf starts from the starting point of the EPC, we will call each random surf as an execution of the EPC workflow. During the simulation we traversed the EPC one million times and counted how many times each function was called. This can be done relatively easily since an EPC is a simple directed graph only with special connectors. The traversing is different with each connector.

- OR: each branch is called with 0.5 probability, but at least one is called.
- XOR: exactly one random branch is called.
- AND: each branch has to be called.

At the OR and AND connectors a traversing thread can start numerous other threads. Possible cycles in the graph and parallel threads in the execution make us to use threshold values to the maximum traversing depth. This can guarantee that our algorithm will finished in a few seconds even for large complex EPC workflows. In the next step we normalize these counted values. Since after the normalization the values are in the $[0, 1]$ interval and their sum is one, they can be used as weights.

4.2 EPC quality

For qualifying an EPC we created a quality model (Figure 3) for EPC workflows. This quality model has three kinds of input, these are the following:

- business process metrics for EPC (Table 2).
- meta-information provided by the EPC designer about how important the EPC is and how frequently it is used.
- quality characteristics of the EPC calculated by the weighted sum of the function quality characteristics (function changeability, function testability, function stability, function test coverage).

Although these are important aspects about the criticality of an EPC, it is needs to be clarified the role of each input node in model. An EPC with many connections and functions makes it hard to change. Also an EPC with high McCabe's Cyclomatic Complexity and Control Flow Complexity makes it hard to change and test. Functional changeability represents the overall changeability of the functions which manage the EPC. The EPC is originally designed and often used for describing user

interactions of ERP systems. By changing the EPC very often we need to change the ERP systems as well. This is the reason why the functional changeability and every other functional quality characteristics affect its corresponding EPC quality characteristic. If an EPC is badly maintainable it not necessary critical, it is also needs to be considered how important the EPC is and how frequently it is used.

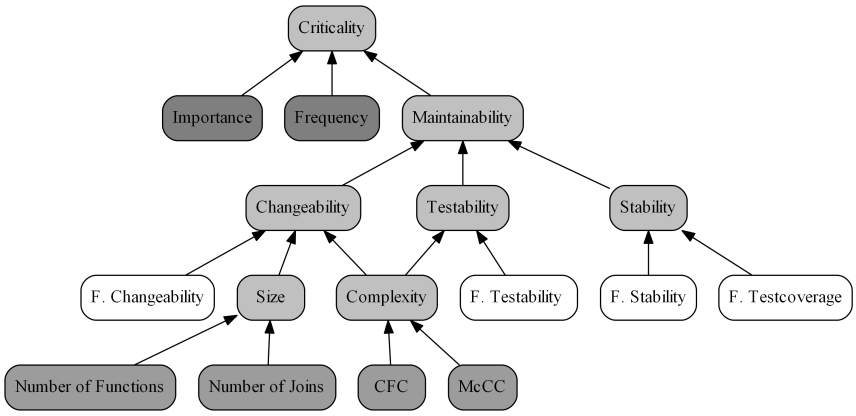

Figure 3: EPC quality model

Table 2: The EPC metrics in the model

EPC metric	**Description**
McCabe[15]	McCabe's Cyclomatic Complexity.
Number of Functions[7]	Number of functions in the EPC.
Number of Joins[7]	Number of joins in the EPC.
Control-Flow Complexity (CFC) [12]	Sum of the CFC of each join. CFC of the AND join is 1, for XOR it is the $fanout$, and for OR it is $2^{fanout} - 1$.

Except business process metrics every other characteristics are in the $[0, 10]$ interval, so they can be used as goodness values in the aggregation process of the quality model. To convert these four EPC metrics to goodness values we used the approach introduced in Section 4 again, but this time for EPC workflows instead of software systems. The Columbus QM compares the current EPC with several other EPC from a benchmark and at the end it provides a goodness value for each EPC metric in EPC quality model (Figure 3). To get quality of the higher level nodes along the edges we used linear combination of the lower level nodes weighted by expert votes provided by the software company.

4.3 Dealing with missing data

The approach needs to be prepared if some kind of data is missing from the EPC quality model. Very often it is not possible to measure the test coverage of the used application, for example if the application is open-source and we have no information about how we could test it properly. Another scenario when the business process is not managed by any software system. When a node is not available in the EPC quality model we redistribute its weights. For example if the Function Changeability node is not available, its weight will be 0 and the weight of the Size and the Complexity node will rise with the half of the weight of Function Changeability node.

4.4 Process group and business configuration quality

Since the business configuration is just a special business group, their evaluation process can be defined with the same recursive expression: quality of a process group is the average of the process groups and EPC workflows in the current process group. Weighted sum is not necessary here, because their importance and frequency have been already taken into consideration in the EPC quality model.

5 Case Study

We tested our approach by modelling and measuring business processes of a medium-sized software company. These processes describe how a software product is being developed, tested and managed in this company. In the business configuration there are four business groups.

- Software development: the business processes in this group describe how to define, start and close a software development project (e.g. *Closing software development project*).
- Specification: it is dealing with collecting and accepting software development requirements (e.g. *Accepting and closing the specification*).
- Implementation: it controls the agile software development (e.g. *Daily scrum*).
- Release test: it handles the testing and releasing of the release candidate version of the system (e.g. *Release*).

The business processes in this business configuration are not large or too complex. The largest business process is *Reporting and fixing code faults*, it contains 14 events and 12 functions, it also contains a circle if a fault was not fixed properly the process starts from the beginning. In the next section we will present the results of the evaluation process.

5.1 System quality

Altogether 19 EPC workflow were designed and 10 were connected to at least one software system. Actually all of them are somehow related to at least one software system but not all of them were written in Java; for example, the most frequently used software system, Redmine (issue tracking system) was written in Ruby. In Table 3 we collected the basic metrics for the Java systems. There are several well-known open-source Java systems from 69,416 to 1,318,626 total logical lines of code (TLLOC). The proprietary software systems were created by the software company. This is the reason why we could ask them to execute their test cases and measure the test coverage of their application; these values can be seen in Table 4. Since the used tool is only capable of measuring test coverage of web applications, and we did not have any information about the test coverage of the open source system like Eclipse we could not measure their test coverage.

The quality of these systems was measured by the Columbus QM, these values are presented in Table 4 as well.

Table 3: Descriptive statistics of the analyzed systems

System	**TLLOC**	**Number of classes**	**Type**
EclipseJDT	1,318.626	7,828	Open-source
Jenkins	106,561	2,609	Open-source
PMD	69,416	1328	Open-source
Selenium	132,532	2,087	Open-source
BusinessProcessManager	14,014	222	Proprietary
CloneManager	7,333	120	Proprietary
MagicTestManager	35,972	581	Proprietary
ProductDashboard	1,180	19	Proprietary
SourceAudit	9,639	152	Proprietary
TestNavigator	40,951	708	Proprietary

5.2 Business process quality

To evaluate an EPC, first of all we need to run the simulation to approximate the weight of each function. In Figure 4 there is a simple EPC from the case study for analyzing a software and do some refactoring to raise its quality above a threshold value and eliminate blocker rule violations and clone smells.

Function quality After one million random executions the weights converged in the case study for every EPC. The weights of the EPC presented in Figure 4 were 0.250 for every function, except the functions in the XOR connector, their weights were 0.125. This is what we expected, because every function will be executed in each simulation except the functions in connectors, only one of them will be traversed because of the XOR connector.

Table 4: Quality statistics of the analyzed systems

System	Maintainab.	Changeab.	Testab.	Stability	Test coverage
EclipseJDT	3,34	2,91	3,16	2,71	-
Jenkins	6,03	6,24	6,32	6,53	-
PMD	5,84	5,71	6,23	6,24	-
Selenium	5,5	5,78	5,69	5,46	-
CloneManager	7,8	6,79	7,47	6,95	55%
BPM	6,26	5,89	6,91	6,34	42%
MTM	7,48	7,09	7,75	6,94	23%
ProductDashboard	6,94	6,94	6,26	8,46	59%
SourceAudit	6,26	5,99	6,14	6,49	69%
TestNavigator	6,08	5,8	6,56	5,97	37%

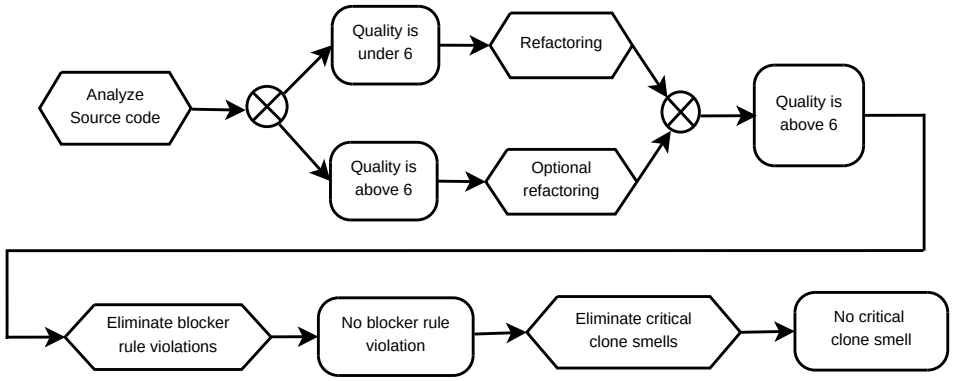

Figure 4: Quality EPC

To go further, first we need to know which system was attached to which function. We collected this information in Table 5. Based on this table we calculated the function quality of each function in the EPC. For example the function maintainability of function Analyze is: AVG(6.03 + 5.84 + 7.8 + 6.94 + 6.26)=6.574. By a weighted sum we calculated the function quality characteristics on EPC level using the weights and quality of the functions.

Metrics The EPC metrics were also calculated for every EPC in the business configuration. Since we can only aggregate goodness values in the [0, 10] interval, we used the Columbus QM to compare the current EPC metrics with other EPC metrics from a benchmark. The metric values and the calculated goodness values for the sample EPC can be seen in Table 6.

Table 5: Attached software systems

System	**Analyze**	**Refactoring**	**Opt. refactoring**	**Elim. rule viol.**	**Elim. clone smells**
EclipseJDT		X	X	X	X
Jenkins	X				X
PMD	X				
Selenium					
CloneManager	X				X
BPM					
MTM					
ProductDashboard	X				
SourceAudit	X				
TestNavigator					

Table 6: The metric values for the EPC presented in 4.

Metric	**Value**	**Goodness value**
McCabe	2	4.93
Number of Functions	5	3.58
Number of Joins	1	5,71
Control-Flow Complexity (CFC)	2	3.58

Meta-information When an EPC was created we asked the designer to estimate the Frequency and Importance meta-information of the business process. For example the EPC presented in Figure 4 has an Importance of 7 and Frequency of 10, since it is executed every time when a developer commits a new version of their software, and it could reveal potential errors in the code.

After every lower level node is calculated, we can aggregate their goodness values to the higher level nodes. We used the described weighted linear combination to this, the edges of the model was weighted by the company. As the result of the aggregation process we evaluated every EPC in the business configuration. The criticality of the EPC in the example is 3.41, it means this EPC is more critical than the average. The distribution of the 19 EPC criticality can be found in Figure 5. 13 from 19 EPC has greater criticality than 4, so most of them are close to the

average in the criticality point of view. The rest six EPC, which has a criticality below 4 needs to be investigated. It is recommended to reorganize these business processes in a less critical way.

Final aggregation of process groups was done by simple average and we got 4.95 for the criticality of the business configuration. This means that the overall criticality of this business configuration is a slightly worse than the average.

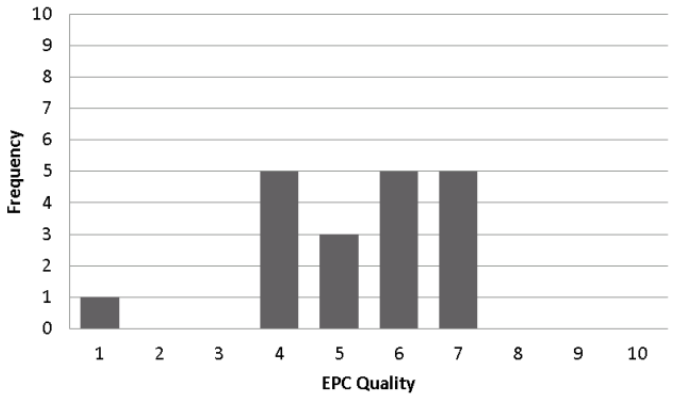

Figure 5: Quality distribution of the evaluated 19 EPC in the case study.

6 Threats to validity, limitations

Altogether the results are promising, but we have to highlight the limitations of our approach as well. A serious limitation is that our approach is not general in every aspect.

- The implementation is working only with EPC workflows, other business process formats are not supported yet. Currently it is possible to import ARIS EPC workflows to our model, but they are not compatible with our approach perfectly. Although the method itself is general, it could be used for other formats, but our tool was designed specially for EPC workflows.

- If the EPC is not using any Java software system, we can not calculate and use quality values extracted from the software systems. In this case the approach will calculate only with frequency, importance and the EPC metrics. However, our C++, RPG and Python analyzer tools are near completion, after they are ready we can extend our approach with these languages as well.

- The benchmark which was used in the case study is based on a middle-sized software company. It can not be used to evaluate the processes of a bank system for example. We need to create domain specific benchmarks with large number of EPC workflows.

During the evaluation process we had to approximate different kinds of values because we could not calculate or collect them in an exact way.

- When we calculated the weights of each function we used a simulation technique which is based on random choices. To approximate the real values we would had to collect long term statistical information about the usage of the given business process when it is created or every time it has been changed. This would make us impossible to help designers right at the design phase, because they would need to use the business processes for months before we could provide them useful information.

7 Conclusions and future work

Our intent was to bring static code analysis and business process quality measurement closer. We used metrics, benchmarking, ISO/IEC 25010 standard based quality models in our bottom-up approach. In a case study we used our approach in a middle-size software company and the results showed that there is a potential in our method. Managers and decision makers can easily access information about the criticality of their business processes. Moreover, the approach provides technical details why a business process is critical or hard-to-maintain. It also helps designers to decide which EPC workflow is too complex and which application should be replaced or updated.

In the future, we would like to implement more powerful metrics. We would also like to create more domain specific benchmarks, which would extend the usability of the approach. Finally, the main future work is to generalize our tool to support several other business process workflow formats.

Acknowledgements

This research was supported by the Hungarian national grant GOP-111-11-2011-0038.

References

[1] *ISO/IEC 25000:2005. Software Engineering – Software product Quality Requirements and Evaluation (SQuaRE) – Guide to SQuaRE.*

[2] Software Improvement Group.
`http://www.sig.eu/en/`.

[3] Alves, Tiago L., Ypma, Christiaan, and Visser, Joost. Deriving Metric Thresholds from Benchmark Data. In *Proceedings of the 26th IEEE International Conference on Software Maintenance (ICSM 2010)*, 2010.

[4] Baggen, Robert, Schill, Katrin, and Visser, Joost. Standardized Code Quality Benchmarking for Improving Software Maintainability. In *Proceedings of the Fourth International Workshop on Software Quality and Maintainability (SQM 2010)*, 2010.

[5] Bakota, Tibor, Beszédes, Arpád, Gergely, Tamás, Gyalai, Milán Imre, Gyimóthy, Tibor, and Füleki, Dániel. Semi-automatic test case generation from business process models. In *11th Symposium on Programming Languages and Software Tools and 7th Nordic Workshop on Model Driven Software Engineering (SPLST09 & NW-MODE09)*, pages 5–18. Citeseer, 2009.

[6] Bakota, Tibor, Hegedűs, Péter, Körtvélyesi, Péter, Ferenc, Rudolf, and Gyimóthy, Tibor. A Probabilistic Software Quality Model. In *Proceedings of the 27th IEEE International Conference on Software Maintenance (ICSM 2011)*, pages 368–377, Williamsburg, VA, USA, 2011. IEEE Computer Society.

[7] Cardoso, Jorge, Mendling, Jan, Neumann, Gustaf, and Reijers, Hajo A. A discourse on complexity of process models. In *Business process management workshops*, pages 117–128. Springer, 2006.

[8] Correia, José Pedro and Visser, Joost. Benchmarking Technical Quality of Software Products. In *Proceedings of the 15th Working Conference on Reverse Engineering (WCRE 2008)*, pages 297–300, Washington, DC, USA, 2008. IEEE Computer Society.

[9] Ghani, Abd, Azim, Abdul, Koh, Tieng Wei, Muketha, Geoffrey Muchiri, and Wong, Pei Wen. Complexity metrics for measuring the understandability and maintainability of business process models using goal-question-metric (gqm). *International Journal of Computer Science and Network Security*, 8(5):219–225, 2008.

[10] Gruhn, Volker and Laue, Ralf. Complexity metrics for business process models. In *9th international conference on business information systems (BIS 2006)*, volume 85, pages 1–12, 2006.

[11] Hegedűs, Péter. A probabilistic quality model for c# – an industrial case study. *Acta Cybernetica*, 21(1):135–147, 2013.

[12] J., Cardoso. Business process control-flow complexity: Metric, evaluation, and validation. In *International Journal of Web Services Research (IJWSR)*, volume 5(2), pages 49–76, 2008.

[13] Khlif, Wiem, Makni, Lobna, Zaaboub, Nahla, and Ben-Abdallah, Hanene. Quality metrics for business process modeling. In *WSEAS International Conference on APPLIED COMPUTER SCIENCE (ACS 09). Genova: WSEAS Press*, pages 195–200, 2009.

[14] Khlif, Wiem, Zaaboub, Nahla, and Ben-Abdallah, Hanene. Coupling metrics for business process modeling. *International Journal of Computers*, 4(4), 2010.

[15] McCabe, Thomas J. A complexity measure. *Software Engineering, IEEE Transactions on*, (4):308–320, 1976.

[16] Mendling, Jan, Neumann, Gustaf, and Van Der Aalst, Wil. Understanding the occurrence of errors in process models based on metrics. In *On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS*, pages 113–130. Springer, 2007.

[17] Michael Hammer, James Champy. *Reengineering the Corporation: A Manifesto for Business Revolution.* Harper Business, 1993.

[18] Page, Lawrence, Brin, Sergey, Motwani, Rajeev, and Winograd, Terry. The pagerank citation ranking: Bringing order to the web, 1999.

[19] Reijers, Hajo and Mendling, Jan. Modularity in process models: Review and effects. In *Business Process Management*, pages 20–35. Springer, 2008.

[20] Reijers, Hajo A and Vanderfeesten, Irene TP. Cohesion and coupling metrics for workflow process design. In *Business Process Management*, pages 290–305. Springer, 2004.

[21] Scheer, August-Wilhelm. *Architecture of Integrated Information Systems: Foundations of Enterprise Modelling.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1992.

[22] Turetken, Oktay. Towards a maintainability model for business processes.

[23] Vanderfeesten, Irene, Cardoso, Jorge, Mendling, Jan, Reijers, Hajo A, and van der Aalst, Wil. Quality metrics for business process models. *BPM and Workflow handbook*, 144, 2007.

[24] Vanderfeesten, Irene, Reijers, Hajo A, and Van der Aalst, Wil MP. Evaluating workflow process designs using cohesion and coupling metrics. *Computers in Industry*, 59(5):420–437, 2008.

[25] Vanderfeesten, Irene TP, Cardoso, Jorge, and Reijers, Hajo A. A weighted coupling metric for business process models. In *CAiSE Forum*, 2007.