Complexity Metrics for Measuring the Understandability and Maintainability of Business Process Models using Goal-Question-Metric (GQM)

Abdul Azim Abdul Ghani Koh Tieng WeiGeoffrey Muchiri Muketha Wong Pei Wen
Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, 43400 UPM Serdang, Selangor.

Summary
Business Process Models (BPMs), often created using a modeling language such as UML activity diagrams, Event-Driven Process Chains Markup Language (EPML) and Yet Another Workflow Language (YAWL), serve as a base for communication between the stakeholders in the software development process. In order to fulfill this purpose, they should be easy to understand and easy to maintain. For this reason, it is useful to have measures that can provide us adequate information about understandability and maintainability of the BPM. Although there are hundreds of software complexity measures that have been described and published by many researchers over the last few decades, measuring the complexity of business process models is a rather new area of research with only a small number of contributions. In this paper, we provide a comprehensive report on how existing complexity metrics of software were adapted in order to analyze the current business process models' complexity. We also proposed a Goal-Question-Metric (GQM) framework for measuring the understandability and maintainability of BPMs.

Keywords:
Complexity Metrics, Business Process Modeling and Analysis, Goal-Question-Metric

1. Introduction

One of the main purposes for developing BPM is to support the communication between the stakeholders in the software development process [4]. To fulfill this objective, the models should be easy to understand and easy to maintain. If we are interested to create a model that is easy to understand and easy to maintain, at first we have to define what is understandability and maintainability: Boehm defines understandability as the degree to which the purpose of the system or component is clear to the evaluator, while in the later case, the IEEE Standard Computer Dictionary defines maintainability as the ease with which a software system or component can be modified to correct faults, improve performance, or other attributes, or adapt to a changed environment. To a certain extent, we may conclude that the measurements should tell us whether the model is easy or difficult to understand and, we may conclude from the metrics that the model should be re-engineered, for example by decomposing it into simpler modules.

Manuscript received May 5, 2008
Manuscript revised May 20, 2008

There are hundreds of software complexity metrics measures that have been described and published by a significant number of researchers. Metrics were designed to analyze software such as imperative, procedural, and object-oriented programs [5]. For example, the most fundamental complexity measure, the number of lines of code (LOC), simply counts the lines of executable code, data declarations, comments, and so on. While this measure is extremely simple, it has been used successfully for the purposes like predicting the error rate, estimating development and maintenance costs. However, to our best knowledge, there is almost no published work where the metrics were created particularly for measuring the complexity of BPM. Most of the works are transfer and adaptation of quality metrics from software engineering domain to business processes [6].

In this paper, we discuss in the most comprehensive way on how a set existing complexity metrics of software were modified and adapted by researchers to provide useful information on complexity of the BPM. This paper is organized as follows. In Section 2, various efforts of works on complexity metric for BPM are reported. Then we have summarized the metrics from literature and their usage for analyzing the complexity of BPM in Section 3. This is followed by our proposed GQM-based metrics for measuring the understandability and maintainability of BPM. Conclusions and future works are drawn in Section 5.

2. Related Work

To our knowledge, there is no much published work about complexity analysis of BPM. The authors of [5] have surveyed several contributions from neighboring disciplines on how complexity can be measured. They have gathered insight from software engineering, cognitive science, and graph theory, and discussed to what extent analogous metrics can be defined for business process models. In order to demonstrate that these metrics serve their purpose, they plan to carry out several empirical validations by means of controlled experiments. These experiments will involve more than 100 students

from the 3 Universities from Netherlands, Austria and Portugal. The collected data will be analyzed using statistical methods to verify the degree of correlation between students' perception of complexity of processes and the proposed metrics. In their small experiment that involved 19 graduates students was conducted and tested if the control-flow complexity of a set of 22 business processes could be predicted using Control Flow Complexity (CFC) metrics. The analyzed result shows that CFC metric is highly correlated with the CFC of processes. They concluded that the metric can be used by business process analysts and process designers to analyze the complexity of processes and, if possible, develop simpler processes.

On the other hand, Jorge Cardoso also proposed a CFC metric to be used during the design of processes. It can be used to evaluate the difficulty of producing Business Process Execution Language (BPEL) process design before implementation. In addition, they also investigate the complexity concept to avoid a vague use of the term "complexity" in the workflow designs. They have presented several complexity metrics that have been used for number of years in adjacent fields of science and explain how they can be adapted and used to evaluate the complexity of workflows. An empirical validation of the CFC metric was carried out [11].

Irene Vanderfeesten [6] claims that modeling and designing business processes without the aid of metrics to question the quality or properties of their models will lead to a lower understandability and higher maintenance costs, and perhaps inefficient execution of the processes in question as a result of simple processes being modeled in a complex and unsuitable way. The authors have elaborated on the importance of quality metrics for business modeling. It presents a classification and an overview of current business process modeling and it gives an example of the implementation of these metrics using the ProM tool where it can be used to study process models implemented in more than eight languages.

Since it is not yet clear which metrics are needed in order to guide and increase the quality of model design, Jan Mendling [7] has proposed a density metric inspired by social network analysis in order to quantify the complexity of an EPC business model on a scale between zero and one. They have considered minimum and maximum number of arcs for a given set of function, event, and connector nodes. In addition, they also test the EPC density metric in combination with simple metrics of size for its capability to predict errors in the SAP reference model. While the significance of density is promising, the experiment reveals that there are further metrics needed in addition to density.

In the first place, we agreed to [5] and believed intuitively that if there are some similarities between software programs and business processes as shown in Table 1, then business management systems should have similar characteristics as other software programs. However, a specialized quality evaluation of metrics by completely understanding the difference between the business process management enterprise software and other software products was carried out in [13]. Surprisingly, their results show that business process management is the software focused on different processes which is something different from other systems and the presented quality evaluation metrics reflecting the characteristics of business process management, process-based software based on the ISO/IEC 9126 model which is the standard quality evaluation metric.

Table 1: Similarities between software programs and business processes

Software Program	**Business Process**
Module/ Class	Activity
Method/ Function	Operation
Variable/ Constant	Data element

A measurement framework based on the GQM paradigm was proposed in [1]. It is generally applicable to any business process and supporting software system after its instantiation. The collaborative software environment WebEv, Web for the Evaluation, is also proposed for facilitating the collection and elaboration of the required measures. To our best understanding on GQM, we have suggested a few metrics for measuring the understandability and maintainability of BPM in Section 4.

Volker Gruhn [15] has adopted the cognitive complexity measure to estimate the comprehension effort for understanding software. Overviews about factors that have an influence on the complexity of control flow of a BPM and metrics that can be used to measure these factors were discussed in [16]. However, no formal validation on the proposed metrics has been discussed.

3. Adaptation of Complexity Metrics in Business Process Model

In this section, we will analyze and summarize the adaptation of software engineering complexity metrics which are applied in business process models into 5 categories: size, complexity, structure, comprehensiveness and modularization. We believed all the following metrics are helpful in business process design and modeling.

3.1 Size of the Model

The most fundamental and easiest complexity measurement for software program is the LOC count which represents the program size. The basic of the LOC measure is that program length can be used as a predictor of program characteristics such as errors occurrences, reliability, and ease of maintenance. In business processes, we can derive a very simple metric that merely counts the number of activities (NOA) in a business process. It should be noted that NOA metric does not take into account of functionalities in this case and it is not language dependent as the original LOC metric.

Another adaptation of the LOC metric is it also takes into account process control-flow elements. If we can consider the processes are well structured, then we can simply count the control structures corresponding to splits, since it is explicitly known that a corresponding join exits. Several adaptation of LOC metric was derived in [5].

3.2 Complexity of the Model

The cyclomatic number, introduced by McCabe, is the most widely used measurement in software program. It calculates from the control flow graph and measures the number of linearly-independent paths. The cyclomatic number indicates that the program is easy to understand and modify. Cardoso [5] has suggested a complexity measure for BPMs which is a generalization of McCabe's cyclomatic number. The CFC metric was based on the analysis of XOR-splits, OR-splits and, AND-splits control statement. The main idea behind the CFC metric defined by Cardoso is the number of mental states that have to be considered when designer develops a process.

The measure of Halstead is another measure of software complexity. The measures were developed as a means of determining a quantitative measure of complexity based on a program comprehension as a function of program operands and operators. Cardoso [5] has suggested to map business process elements to the set of primitive measures proposed by Halstead. With these primitive measures, Cardoso has introduced the notion of Halstead-based process Complexity (HPC) measures for estimating process length, volume, and difficulty. According to the author, HPC measures do not require in-depth analysis of process structures, they can predict rate of errors and maintenance effort, easy to calculate and applicable for most process modeling languages.

On the other hand, J. Mendling [7] has found out that Adaptation of McMabe cyclomatic metric has no impact on the odds of an error in a BMP model and, include HPC as well did not provide proper distinction between size and complexity. J. Mendling has defined the density metrics to calculate the minimum and maximum number of arcs for a given set of function, event, and connector nodes. The results tested for this metric is mixed. Advantages and limitations are both included in the test result for this metric. The density metric capable to predict errors in the SAP reference model, positive impact on error probability on a significance level better than 99% but the density metric and size together are not sufficient to explain the variance of errors.

3.3 Structure of the Model

Gruhn [16] claims that model that contains greater nesting depth implies greater complexity. Here shows that the nesting depth value has its impact onto the structured related complexity metrics. The nesting depth of an element implies number of decisions in the control flow that are necessary to reach this element.

Some models might be having a numbers of control flows that needed to go through to get a decision in order to come to final outcome. The model with nested XOR-splits and XOR-joins might be more complex and harder to understand than almost linear model, but the CFC for both models might be same. For this reason, the author [16] also has suggested to use the nesting depth metric to get the nesting depth value and add the value to the CFC in order to measure the complexity of BMP.

One of the differences between a well structured model and not well structured model is the splits or joins. In the well structured model, splits or joins are contained completely within the control structure whereas the not well structured model may have a jump out of the control block. Not being well structured in BPM informally means that a misfit between the split and join connectors exists.

The author in [8] also has suggested to use the split-join-ratio to calculate the misfit between the split and join connectors in BPM as what the knot count metric that is been used to calculate the jumps out of and into a structured control flow for software programs. The metric says to be too simple to measure the unstructured model due to the unstructured in one part of the BPM that results a high value for split-join-ration can be corrected by simply adding another unstructured element into the model which has too small split-join-ratio. The research for this metric is still under preparation. Due to this metric is too simple to measure unstructured model.

3.4 Comprehensiveness of the Model

According to [5], cognitive complexity is related to cognitive psychology that aims at studying, among other things, thinking, reasoning, and decision making. The understanding of cognitive complexity is to divide the memory into long term and short term memory. The short term memory limit the duration of storage to less than about 30 seconds whereas the long term memory can last can last as little as 30 seconds or as long as decades. The chunk of processes that can be captured and stored by a short term memory would be determined as meaningful. The structure of the BPM has to be taken into account when it is measured by cognitive weight due to the model may have cancellation or other concepts. Gruhn claims that the cognitive weight is still needed to have further research for its usage as the basic idea in BPM.

Process patterns are examples that show how to connect activities together to solve a common problem. In software complexity, a good design pattern helps to improve code quality, understandability and maintainability [16]. The author [12] recognizes the anti-patterns. Anti-patterns are specific repeated practices that appear initially to be beneficial, but ultimately result in bad consequences that outweigh the hoped-for advantages [Wikipedia]. If the anti-pattern has been found in coding, this is a sign of a bad programming [16]. The author [16] also says that uncovering the anti-patterns in BPM should be useful in order to define whether the model has a good modeling style.

3.5 Modularization of the Model

Modular modeling of business process is supported by almost all BPM languages. By dividing a BPM into modular sub-models we can increase their understandability and also lead to smaller, reusable models for future maintenance.

Henry and Kafura [14] proposed a metric based on the impact of the information flow in the program' structure. The technique suggests identifying the number of calls to module (i.e. the flows of local information entering: Fan-In) and identifying the number of calls from a module (i.e. the flows of local information leaving: Fan-Out). This metric can be used in the same way for analyzing BPMs. If a sub-model of a BPM has a high structural complexity according to the fan-in/fan-out metric, they will be difficult to use and are most likely poorly designed. The high value for fan-in will achieve by the module been called and used by other module and high fan-out is caused by the module called to use or import the other modules [16].

4. GQM-based Complexity Metrics

Although some researchers have proposed individual or sets of isolated metrics such as [9], they do not give us guidelines on how to choose a particular metric for a particular situation. Several models have been proposed to address this problem. In the two subsections that follow, we compare two previous approaches with GQM, and then give a GQM example that derives understandability and maintainability metrics for business process models.

4.1 Approaches for Deriving Metrics

A powerful approach called Goal-Attribute-Measure (GAM) has been proposed [20]. It is based on Norman Fenton's [18] ideas where measurement objects are identified as products, processes and resources. To derive measures in GAM, identify measurement customers and their goals, and then identify a set of target attributes, their driving attributes, and measurement objects. Attributes are then divided into directly measurable sub attributes from which a set of measures are defined [20, 21]. In GAM, the scope of goals is on measurement objects while focus is on the structuring and definition of attributes.

In a related study [17] a popular approach called the Balanced Scorecard (BSC) is proposed. It originated from strategic management and is used by top management to provide a measure on how the organization is progressing towards its strategic goals [17]. BSC provides four perspectives namely, financial perspective (shareholders' view), customer perspective (value-adding view), internal perspective (process-based view), and learning and growth perspective (future view).

The first step in deriving BSC measures starts with the analysis of the mission and vision of the organization. This is followed by the definition of goals for financial and other perspectives. The next step defines drivers that will help achieve the goals. Finally, indicators for each driver are defined [1, 19].

One of the most goal-focused and most widely used of all other measurement approaches is the Goal-Question-Metric (GQM) [1-3, 19]. A GQM a team defines project goals and a set of questions to achieve each goal. Next, the team develops metrics to address each question. GQM's questions and metrics are similar to BSC's drivers and indicators respectively except they have a different scope. A detailed description of the differences and similarities between BSC and GQM can be found in [19].

The initial GQM paradigm was too flexible, and could easily generate unnecessarily large sets of metrics. Several different improvements have been proposed to address

this problem. In [18] it is argued that GQM should be combined with process maturity level of the organization in order to determine the most appropriate metrics. For example, if an organization is at maturity level 1, only baseline measures can be collected since most of the characteristics of objects to be measured are ill-defined. On the other hand, a richer set of measures can be collected at higher maturity levels where processes are well defined [18].

In [3] a prioritization step has been incorporated into GQM to reduce the generated metrics to a bare minimum. The problem with prioritization is that it has the side effect of a stripped-down GQM tree which may fail to address certain perspectives of the project. Table 2 below compares GAM, BSC and GQM approaches for defining metrics.

Table 2: Comparison of GAM, BSC, and GQM approaches

Approach	Architecture	Scope	Focus
GAM	Goal Attribute Measure	Measurement object	Attribute structuring & definition
BSC	Goal Driver Indicator	Organization	Driver definition
GQM	Goal Question Metric	Project	Question definition

As can be seen from the table above, GAM is best used to measure specific objects; BSC is used when measuring organizational progress even outside IT scope; and GQM is used when measuring a software project as a whole.

4.2 Metrics for Understandability and Maintainability

In this paper, we define understandability as the ability to easily manage business flow without additional explanation. Case example: Can users easily understand some functions such as “Undo” or “Resubmit”? We also define maintainability as the ability to agilely change business process. Case example: How easy is it to change business process in runtime?

In this section we apply GQM to the twin goals of understandability and maintainability. Our aim is to generate the complete set of metrics that can help an organization to measure all attributes of these two goals from the perspective of user satisfaction, usability, functionality and reliability. We demonstrate this in our example below:

G1 To analyze a business process with the aim to evaluate its comprehensibility from user point of view

Q1.1 How easy is it to read the model?
- *M1.1.1 No. of symbols and formulas used*
- *M1.1.2 Type of structures used*
- *M1.1.3 No. of unstructured statements*

Q1.2 To what extent is it convenient?
- *M1.2.1 Types of standards used*

G2 To analyze a business process with the aim to evaluate its changeability from manager point of view

Q2.1 How complex is the process?
- *M2.1.1 No. of activities/ services in the process*
- *M2.1.2 LOC (for an executable language like BPEL)*
- *M2.1.3 CFC*
- *M2.1.4 No. of modules in process or sub-system*
- *M2.1.5 Fan-in and Fan-out*

After generating a list of metrics, numeric formulas are then developed for each metric. Before use, the metrics should be validated through empirical experiments that test the correlation between a metric and the attribute being measured. For example in [11] a CFC validation experiment was conducted with the hypothesis that there is a significant correlation between the CFC metric and the subject’s rating of the control-flow complexity of processes. We, however, did not validate our GQM metrics example above because a controlled experiment is beyond the scope of this paper and has to be done in future research.

5. Conclusion and Future Research

In this paper, we have surveyed and reported few findings from software engineering, particularly in complexity metrics, and we gather opinion from researchers as to what extent analogous metrics can be defined for business process models. Table 3 summarizes the results from our survey: adaptation of complexity metrics of software program in BPMs.

Table 3: Complexity Metrics for software and BPMs

Software Complexity Metric	**BPM complexity Metric**	**Usage of the metric in BPM**
Lines of code	Number of activities	Simply count the number of activities for the model.
Cyclomatic number	CFC	Measure the number of control flow in the model.
Nesting depth	Nesting depth	Defined the structured of the model. Higher nesting depth value indicates more complex.
Knot-count	Split-join-ratio	To define whether the model being well structured
Cognitive Weight	Cognitive Weight	Measure the understandability of a model
Anti-pattern	Anti-pattern	To uncover the bad modeling style in BPM model.
Fan-in/Fan out	Fan-in/Fan out	To define good or bad modularization of a model.

GQM ensures that each metric has a purpose, and no metrics are defined without a purpose. We provided a GQM-based complexity metrics for handling the understandability and maintainability of BPM in Section 4.

In our future research, we intend to investigate and extend the GQM approach to enable it to generate the minimum set of metrics that is also complete and that doesn't lead to the problem of a stripped-down GQM tree. Also, most of the adapted metrics on BPM have not been validated, and we therefore plan to conduct validation experiments as future work.

References

[1] Lerina Aversano, Thierry Bodhuin, Gerardo Canfora, Maria Tortorella, (2004). "A Framework for Measuring Business Processes based on GQM". Proceeding of the 37th Hawaii International Conference on System Sciences @ 2004 IEEE.

[2] Victor Basili, G. Caldiera and H. D. Rombach, "The Goal Question Metric Approach". Encyclopedia of Software Engineering, Wiley, 1994.

[3] Berander, P. and P. Jonsson. "A Goal Question Metric Based Approach for Efficient Measurement Framework Definition". ISESE'06. ACM. 2006.

[4] Bill Curtis, Marc I. Kellner and Jim Over, (1992). "Process Modeling". Communications of the ACM. Vol35(9). pp.75-90.

[5] Jorge Cardoso, J. Mendling, G. Neumann, H.A. Reijers, (2006). "A Discourse on Complexity of Process Models", BPI'06 - Second International Workshop on Business Process Intelligence, In conjunction with BPM 2006, Vienna, Austria, 5-7 September, 2006. J. Eder, S. Dustdar et al. (Eds.): BPM 2006 Workshops, LNCS 4103, pp. 115–126, 2006. Springer-Verlag, Berlin, Heidelberg, 2006.

[6] Irene Vanderfeesten, Jorge Cardoso, Jan Mendling, Hajo A. Reijers, Wil van der Aalst, (2007). "Quality Metrics for Business Process Models". Workflow Handbook 2007, WfMC, Layna Fischer (Ed.), Lighthouse Point, FL, USA, Future Strategies Inc., pp. 179-190. ISBN: 0-9777527-1-2.

[7] Jan Mendling, (2007). "Testing Density as a Complexity Metric for EPCs". Vienna University of Economics and Business Administration Augasse 2-6, A-1090Wien, Austria.

[8] Jan Mendling, M.Moser, G.Neumann, H.M.W. Verbeek, B.F.van Dongen, and W.M.P .vander Aalst "A Quantitative Analysis of Faulty EPC in the SAP Reference Model" Vienna University of Economics and Business Administration Augasse2-6, 1090 Vienna, Austria.

[9] Jorge Cardoso, (2006). "Complexity Analysis of BPEL Web Processes". Software Process: Improvement and Practice Journal @ 2006 John Wiley & Sons, Ltd.

[10] Jorge Cardoso, (2006). "Approaches to Compute Workflow Complexity". Department of Mathematics and Engineering University of Madeira, 9050-390 Funchal, Portugal

[11] Jorge Cardoso, (2006). "Process Control-flow Complexity Metric: An Empirical Validation" IEEE International Conference on Services Computing (IEEE SCC 06), Chicago, USA, September 18-22, 2006. pp. 167-173, IEEE Computer Society. ISBN: 0-7695-2670-5

[12] Juha Gustafsson (2000). "Metrics Calculation in Maisa"

[13] Lee Yeong Seok, Bae Jung Hyun, Shin Scok Koo, (2005). "Development of Quality Evaluation Metrics for BPM (Business Process Management) System". Proceedings of the Fourth Annual ACIS International Conference on Computer and Information Science (ICIS'05) @ 2005 IEEE.

[14] S. Henry, D. Kafura, (1981). "Sofware Structure Metrics based on Information-flow" IEEE Transactions on Software Engineering, 7(5), pp.510 – 518

[15] Volker Gruhn, Ralf Laue, (2006). "Adopting the Cognitive Complexity Measure for Business Models".

Proceedings of the 5th IEEE Int. Conf. on Cognitive Informatics (ICCI'06) @ 2006 IEEE.
[16] Volker Gruhn, Ralf Laue, (2007). "2 Approaches for Business Process Model Complexity Metrics". Technology for Business Information System, pp.13-24 @ 2007 Springer.
[17] Maris Martinsons, Robert Davison, Dennis Tse, (1999). "The Balanced Scorecard: a Foundation for the Strategic Management of Information Systems". Decision Support Systems 25 (1999), pp. 71-78.
[18] Norman Fenton and Shari. L. Pfleenger, (1996). Software Metrics: A rigorous and Practical Approach, 2nd edition, IT Publishing Company.
[19] Luigi Buglione and Alain Abran, "Balanced Scorecard and GQM: what are the differences?" (2000). FESMA-AEMES Software Measurement Conferences.
[20] A.T. Nilsson and J. L. Rise, (1996). "Performance Measurements, Procedure to Design Measures – Goal Attribute Measure (GAM)", Ericsson Quality Institute, LME/Q-93:332, Rev.E, (Internal Publication).
[21] Jorma Hirvensalo, (2003). "Quality Measurement and the Utilization of Measurement Results in a Software Development Process", Doctoral Dissertation, Helsinki University of Science and Technology.

Abdul Azim Abdul Ghani is an Associate Professor cum the Dean of the Faculty of Computer Science and Information Technology, University Putra Malaysia. He obtained his PhD in computer science from University of Strathclyde, Scotland. His research interests are software metrics and software quality.

Koh Tieng Wei received the first degree in computer science in 2003 and the MS degree in software engineering in 2006 from the Putra University, Malaysia. Currently, he is pursuing a PhD degree with his research work related to object-oriented software sizing measure.

Geoffrey Muchiri Muketha received a Bachelor of Science degree in Information Science from Moi University, Kenya, in 1995 and a Master of Science degree in Computer Science from Periyar University, India in 2004. He is currently pursuing his PhD in Software Engineering from Universiti Putra Malaysia, Malaysia. His research interests are Business Process modeling and measurement.

Wong Pei Wen received the Bachelor of Computer Science in 2007 in University Putra Malasia and now she is pursuing the Master of Science degree in Computer Science from University Putra Malasia.