Quality metrics for business process modeling

WIEM KHLIF, LOBNA MAKNI, NAHLA ZAABOUB, HANENE BEN-ABDALLAH

Mir@cl Laboratory,
Faculty of Economics and Management Sciences,
Sfax University, Tunisia.
{Wiem.Khlif, Lobna.Makni, Nahla.Haddar, Hanene.Benabdallah}@fsegs.rnu.tn

Abstract
Modeling business processes is vital when improving or automating existing business pocesses, documenting processes properly or comparing business processes. In addition, it is necessary to be able to evaluate the quality of a business process model, which in tern requires a set of quality metrics. Most of the works proposed to evaluate business process models deal with quality by adapting software metrics. This is possible, because software products and business processes software are quite similar. Our contribution in this paper consists in adapting object oriented software metrics to business process models. This adaptation is based on correspondences which we establish between BPMN (Business Process Modeling Notation) concepts and object oriented concepts. By adapting object oriented metrics, we aim to obtain new metrics which give us more information about the complexity of business processes, cohesion between process tasks and coupling between processes themselves.

Keywords: Business process modeling notation (BPMN), quality metrics, business process models, design quality, metric adaptation.

1. Introduction

Modeling business processes is necessary for an enterprise that desires to evaluate, improve, migrate to a different technological platform, automate, and/or document its business processes. Evidently, the quality of a business process model (BPM) highly influences the desired activity. This motivated several researchers to propose *metrics* to evaluate the quality of BPM.

In fact, the concept of metric was initially introduced to check software quality. According to [1], a quality measure is considered as a quantitative scale and a method that can be used in order to determine the value taken by a characteristic of a software product. Since software products and business process software are quite similar [2] [3], most of the works proposed to evaluate BPM deal with quality by adapting software metrics (*cf.,* [4] [5]). Our literature review revealed that the so-far proposed quality metrics ignored the similarities between the concepts of object-oriented software and BPMN (the Business Process Modeling Notation) [6], the standard notation for business processes.

After a brief review of the state of the art in software metrics adapted for BPM, this paper has a two-fold objective: First, it aims at presenting correspondences between the concepts of OO software and the concepts of BPMN. Secondly, it shows how these correspndances can be used to adapt two classes of common OO quality metrics for BPMN: coupling metrics and cohesion metrics.

2. Current metrics adapted to business processes

Several researchers adapted quality metrics from the software engineering domain. Similar to their classification in software engineering, the adaptated quality metrics can be also classified into three categories: coupling, cohesion and complexity.

2.1 Coupling metric adaptation

Coupling in business process models (BPM) focuses on how strongly the activities in a business process are related, or *connected*, to each other. An activity is connected to another activity if and only if they share one or more information elements. For a given activity, the coupling metric determines the number of activities related to it [7]. For a given BPM, its coupling metric equals to the number of interconnections between all its activities; in other words, it counts all pairs of activities in the BPM that are connected to each other. In addition, the degree of coupling depends on how complicated the connections are and also on the type of connections between activities (AND, OR, XOR). (For the mathematical definition of this metric, the redear is referred to [7].)

The coupling metric of an activity reflects how critical/important an activity is within a BPM. In fact, an activity with a high coupling metric value functionnaly determines a large number of activities in

the business process. Thus, its malfunctioning may cause several activities to malfunction; this in tern may jeopardize the overall business process functionalities. Such activities should be either avoided within a BPM, or treated with a special care, *e.g.,* by having a monitoring activity for it.

On the other hand, a BPM with a high coupling metric indicates a high level of informational dependency between its activities. Again, such a model produces a vulnerable process and one which maintainance is difficult, etc.

One limit of the coupling metric is that it does not give an indication about the reusability of a BPM. This quality information is important for design through reuse. A second limit is that focuses on data interchange and does not provide information about the activity dependency in terms of data usage. This limit is addressed by cohesion metrics.

2.2 Cohesion metric adaptation

Vanderfeesten *et al.* [3] adapted the cohesion metric as follows: The cohesion of an activity is the product of both the relation and information cohesion. The relation cohesion quantifies how much the different operations within one activity are related. It determines, for each operation of an activity, how many other operations it overlaps with by sharing an input or output.

On the other hand, the information cohesion focuses on all information elements that are used either as input or output by any operation within this activity. It determines how many information elements are used more than once in proportion to all the information elements used. Thus, it counts all information elements that appear in the intersection of a pair of operations, considering all pairs. To be normalized, this number is divided by the total number of information elements in the activity.

Another adaptation of the cohesion metric is the *cross connectivity* metric [8]. This adaptation aims to quantify the ease of understanding and the interplay of any pair of model elements. The term 'Cross-Connectivity' is chosen because the strength of connections between nodes is considered across all nodes in the model. As a result, the cross connectivity metric expresses the sum of the connectivity between all pairs of nodes in a process model, relative to the theoretical maximum number of paths between all nodes.

Overall, a BPM whose activities have high cohesion values indicates a good modular decomposition of its activities.

One advantage of cohesion metrics is that they can be used to determine the critical data (highly shared) as well as the sharing operations. Such information can be used to impose special treatements, like adding data distribution activities, etc. (For the mathematical definition of these metrics, the redear is referred to [3] and [8].)

2.3 Complexity metric adaptation

Complexity measures the simplicity and understandability of a design. In this quality perspective, several researches on business process metrics have been done, *cf.,* [4] [5].

Both [4] and [5] consider the adaptation of McCabe's cyclomatic number [9] as a complexity metric for business processes. The metric is called Control-flow Complexity (CFC) metric. The main idea behind this metric is to evaluate the number of possible states that have to be considered when a designer is developing a process.

In a BPM, splits introduce these states in processes. For XOR-splits, the control-flow complexity of an activity is simply the fan-out of the split connected to it; for OR-splits, the control-flow complexity is $2n$-1, where n is the fan-out of the split; finally, for an AND-split, the complexity is simply 1.

Mathematically, the CFC metric is additive. Thus, it is very easy to calculate the complexity of a BPM, by adding the CFC of all splits in the BPM. The greater the value of the CFC, the greater is the overall architectural complexity of a process.

A second adaptation of complexity is proposed by Cardoso *et al.* [4] by mapping business process elements to the set of primitive measures proposed by Halstead. With these primitive metrics, they introduce the notion of Halstead-based Process Complexity (*HPC*) metrics for estimating process length, volume and difficulty as follows:

Process Length: $N = n_1 * \log 2(n_1) + n_2 * \log 2(n_2)$

Process Volume: $V = (N_1 + N_2) * \log 2(n_1 + n_2)$

Process Difficulty: $D = (n_1 / 2) * (N_2 / n_2)$

Where:

- n_1 is the number of unique activities, splits and joins, and control-flow elements (such as sequence, switch, loop) of a business process;
- n_2 is the number of unique data that are manipulated by the process and its activities;
- N_1 and N_2 are process lengths derived from n_1 and n_2.

This adaptation views a process activity as a statement of a software program. It is used to derive another very simple metric that counts the number of activities (*NOA*) in a business process [4]. This second adaptation is analogeous to the Line of code (LOC) metric [10].

It should be noticed that the *NOA* metric characterizes only one particular view of size, namely the length; it

takes into account neither functionality nor complexity. Thus, a high NOA value may produce bad process designs with an excessive number of activities.

Another adaptation of the LOC metric not only maps activities to program statements, but also takes into account process control-flow elements (*i.e.,* control structures). This is a second metric (*NOAC*) proposed in [4] to count the activities and control-flow elements of a process.

On the other hand, the Henry and Kafura metric [11] is adapted to evaluate the complexity of processes in the following way [4]: The fan_{in} and fan_{out} can be mapped directly to inputs and outputs of activities. Activities are invoked when their inputs (fan_{in}) are available and the activities are scheduled for execution. When an activity completes its execution, its output data is transferred to the activities connected to it through transitions. Using this hypothesis, [4] proposes a metric called interface complexity (*IC)* of an activity which is defined as:

$$IC = length * (\text{number of inputs} * \text{number of outputs})^2$$

The advantages of the IC metric are that it takes into account data-driven processes and it can be calculated prior to implementation, during the design stage [4].

In summary, although some researchers proposed using software metrics to evaluate business process designs, the number of publications on concrete metrics and applications in the business process domain is still small and only of a very recent date. We also note that object oriented metrics have not been adapted to business process models despite the similarities that exist between the latters and object oriented software. In the next section, we propose to adapt object oriented measures to business processes.

3. Correspondences between BPMN and object oriented software

A business process model which is modeled by EPC (Event-Driven Process Chain), Petri nets, activity diagrams or BPMN manifest several similarities with software [2][3]. In fact, business processes and software products have a similar compositional structure: a program is composed of modules or classes, each module consists of statements and each statement contains variables and constants. In the same way, a business process has activities each which is composed of elementary operations and each operation uses one or more information to produce new information [2] [3].

Based on these similarities, we determined a set of correspondences between the Business Process Modeling Notation (BPMN) concepts [6] and the object oriented software concepts. The choice of BPMN is justified by the fact that this formalism defines an OMG standard notation for modeling business processes. However, our correspondences can be adjusted to deal with EPC, or Petrinets.

Table 1 summaizes our proposed correspondences between object oriented software concepts and BPMN concepts.

Table 1: Correspondances between BPMN and object oriented software core concepts.

	Object oriented software	***BPMN Notation***
Static view	Class/package	Process, sub-process
	Method	Task
	Variable/ Constant	Data object
	Comment line	Annotation
	Interface of a class	Interface of a process/sub process: the set of tasks in a process which send or receive a flow message.
	Local data in a class.	Process tasks data objects: data objects related to process tasks by associations.
	Data used by a class.	Data object used by process tasks: data objects associated with message flows going into tasks in the process.
Dynamic view	Method invocation	Reception of a sequence flow or a message flow by a task.

As illustrated in Table 1, we map a class to a process (or sub process) in the business process domain. In particular, we map a composite class to a multi-level process that contains sub-processes. These sub-processes may be reused independtly of their containing process.

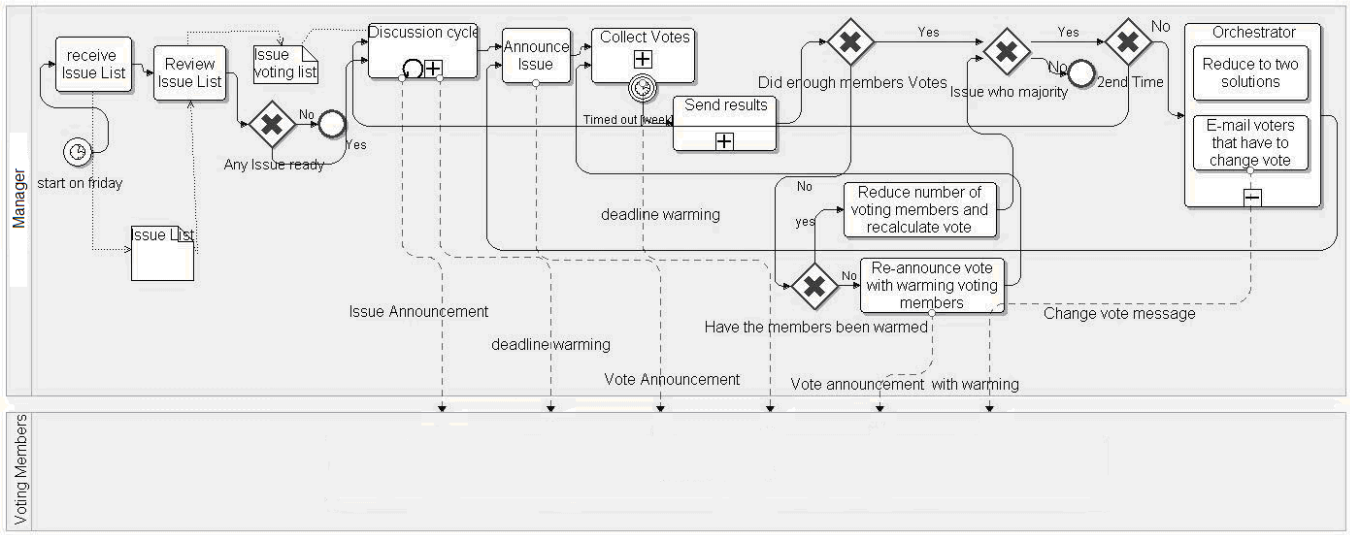

Figure 1. E-Mail Voting Process

In addition, a simple class and its methods are mapped respectively to a simple process containing tasks as properties. Note that a task can be either a humain or an automated one. In addition, a class has local data used by its methods; it also may use data coming from other classes. Thus, we map local data to data objects generated or used by the tasks of a process while data used by a class correspond to data objects associated with message flows arriving to the process tasks.
Moreover, all public methods determine the class interface. By applying this concept to the BPMN formalism, the process/sub process interface will be defined by the set of tasks in a process which send or receive a message flow.
Finally, comment in a software product corresponds to the annotation in BPMN.

4. Quality metrics for BPMN

In this section, we show how the previous mappings of OO software engineering to BPMN concepts can be used to adapt the coupling and cohesion metrics for business process models. To do so, we use a modified version of the e-mail Voting Process [6] modeled with BPMN in Figure 2.

4.1. Coupling metrics for BPMN

Among the various coupling metrics, we adapted four metrics: IC [12], EC [12], RFC [13] and LD [14].

4.1.1 Imported and exported coupling

In the software engineering domain, two types of coupling have been defined [12]:

- IC (Imported Coupling) which counts, for each class C, all interactions in which C *uses* another class.
- EC (Exported Coupling) which counts, for each class C, all interactions in which C *is used*.

According to our correspondence rules (Table 1), we adapt these metrics in the business modeling domain as follows:

- ICP (Imported Coupling of a Process): counts, for each (sub-) process, the number of message/sequence flows sent by either the tasks of the (sub-) process or the (sub-) process itself.
- ECP (Exported Coupling of a Process): counts, for each (sub-) process, the number of message/sequence flows received by either the tasks of the (sub-) process or the (sub-) process itself.

Let us consider the simple sub-process "Discussion Cycle" of our example (Figure 2): It sends a sequence flow to the "Announce Issues" task and two message flows to the process "Voting members". Thus, its ICP is equal to 3. In addition, this sub-process receives two sequence flows: one from the gateway "Any issues ready" and another from the unnamed sub process. Thus, the ECP of "Discussion Cycle" is equal to 2.
Note that a process with high ICP value highly dependents on several external services offered by other processes. This might increase delays, costs and error probability. In addition, a process with a high ECP has a considerable influence on the whole model since a multitude of processes depends on its services. In other words, problems encountered in the business process may be caused by a fault in this influencial process.

4.1.2 Response for class coupling

Examining coupling metrics in the software engineering domain, we noticed that the response for a class (RFC) metric [14] focuses on the coupling in terms of control flows. We call the adapted version of this metric response for a process (RFP) in the

business domain. We compute it as follows: RFP = |RS| where RS is the set of all responses of a process:

$$RS = \{T_j\} \cup \{R_i\},$$

where $\{R_i\}$ is the set of tasks invoked by a task *i* in the process and $\{T_j\}$ is the set of all tasks *j* in the process.

Let us consider the "Orchestrator" process which contains in parallel the two tasks: "Reduce to two solutions" and "E-mail voters that have to change vote". Its set of responses (RS) contains the following tasks:

RS = {"Reduce to two solutions", "E-mail voters that have to change vote"}
U
{"Voting Members", "Announce Issues"}

Thus, the RFP of "Orchestrator" is equal to 4.

Note that, the larger the RFP is, the greater the complexity of the process is: In deed, if a large number of tasks can be invoked in response to a message, then the process becomes complixe and requires a greater level of understanding.

4.1.3 Locality of data-based coupling

Another coupling metric we see adaptable is locality of data (LD) [14]. This metric links data from the activity (process or sub process) to the total data used by this activity. The adapted metric, we call *locality of data activity* (LDA), for an activity (sub process or task) with *n* tasks can be expressed mathematically as follow:

$$LDA = \frac{\sum_{i=1}^{n} |L_i|}{\sum_{i=1}^{n} |DT_i|}$$

where DT_i ($1 \leq i \leq n$) is the set of data associated to task T_i within the activity, and L_i ($1 \leq i \leq n$) is the set of data produced by other activities and used by a task T_i the activity.

Let us consider the "Review Issue List" task in Figure 2; this task uses the "Issue List" data which is produced by the "Receive Issue List" task. Therefore, the LDA of the "Review Issue List" task is computed as follows:

$$LDA = \frac{\sum_{i=1}^{1} |\{"IssueList"\}|}{\sum_{i=1}^{1} |\{"\text{IssueList}", "\text{IssueVotingList}"\}|} = \frac{1}{2}$$

Note that activities with a high data locality are more self-sufficient than those with a low data locality. Hence, they are more adapted to reuse and easier to test.

4.2 Cohesion metrics for BPMN

Various metrics in software engineering and especially object oriented ones have focused on cohesion [13]. On the basis of these works, we propose adaptations of the well known metrics TCC and LCC [15].

4.2.1 Tight Class Cohesion

Tight class cohesion (TCC) [15] counts for each class the percentage of method pairs that are directly *related*. Two methods are directly related if they both use either directly or indirectly a common instance variable. An instance variable is used directly by a method M, if the instance variable appears in the body of the method M. An instance variable is used indirectly by a method M, if the instance variable is directly used by a method M' that is either directly or indirectly invoked by M.

More specifically, TCC for a class is computed as follows:

$$TCC = NDC / NP$$

where *N* is the number of public methods in the measured class; NP is the maximum number of public method pairs: NP = [N * (N – 1)] / 2; and NDC the number of direct connections between public methods. Then TCC is defined as the percentage of method pairs, which are directly related.for the measured class.

We adapt the TCC metric as follows: for a process with *N* (> 1) public tasks (*i.e.,* tasks contained within its interface and which are connected to exterior activities/tasks), we compute its NSP as the maximum number of public task pairs:

$$NSP = [N * (N – 1)] / 2$$

and its NSPDC as the number of direct connections between its public tasks. The adapted TPC metrics, which we call *Tight Process Cohesion* (TPC), to be the percentage of task pairs directly related:

$$TPC = NSPDC / NSP$$

Two tasks are directly related if they both use (directly or indirectly) a common data. A data is used directly by a task T, if it is produced by this task T; a data is used indirectly by a task T, if it is directly used by a task T that receives directly or indirectly a sequence/message flow from the task T.

In our running example, the TPC metric is not applicable since none of its processes has more than one public task.

Note that a TPC equal to 0 means that that the tasks within the measured process are not directly related. This is the worst cohesion scenario.

4.2.2 Loose Class Cohesion

Loose class cohesion (LCC) is a second type of class cohesion used in OO software engineening. LCC counts, for each class, the percentage of method pairs either directly or indirectly related:

$$LCC = NIC / NP.$$

where NIC is the number of direct or indirect connections between the public methods of the measured class, and NP is the maximum number of public method pairs in the measured class.

Our adaptation for business processes (which we call *Loose Process cohesion* (LPC)), counts the percentage of task pairs, which are either directly or indirectly related:

$$LPC=NSPC/NSP$$

where NSPC is the number of direct or indirect connections between the tasks of the measured process.

In our running example, this metric is not applicable since all none of its processes has more than one public task (N >1).

Note that, similarly to TPC, a high LPC is the best quality scenarion; it means that there are several tasks directly or indirectly related.

5. Conclusion and perspectives

Quality design metrics can help designers making their modeling decisions judiciously. To define metrics to examine the quality of a business process model, we first overviewed existing quality metrics and then we explored the adaptation of various OO metrics for BPMN. In particular, we focused on adapting coupling and cohesion metrics for BPMN. Indeed, a good quality model is one whose processes are loosely coupled and its tasks are highly cohesive.

Our future work focuses on two main axes: 1) establishing relationships between metrics and quality dimensions of business process models; and 2) checking the proposed metrics through empirical studies.

References

[1]: ISO/FCD 9126-1.19, Software qualiy characteristics and metrics, *Information Technology* Part 1: Quality characteristics and sub characteristics, ISO/FCD 9126-1. 1998.

[2]: Reijers .H.A., and Vanderfeesten.I, Cohesion and Coupling Metrics for Workflow Process Design. In: Desel, J., Pernici, B., and Weske, M., editors, *Business Process Management* (BPM 2004).

[3]: Vanderfeesten.I, Reijers.H.A, van der Aalst W.M.P, Evaluating Workflow Process Designs using Cohesion and Coupling Metrics, Technische Universiteit Eindhoven, *Department of Technology Management.*

[4]: Cardoso.J, Mendling.J, Neuman.J, Reijers.H.A. (2006), A discourse on complexity of process models, In: Eder, J.; Dustdar, S. et al, editors, BPM 2006 workshops. *Lecture Notes in Computer Science* 4103, Springer-Verlag, Berlin, pp. 115-126.

[5]: Gruhn.V, and Laue.R, Complexity metrics for business process models, In: Witold Abramowicz and Heinrich C. Mayer, editors, *9th international conference on business information systems* (2006), vol. 85 of Lecture Notes in Informatics, pp. 1-12.

[6]: Business Process Modeling Notation, V1.1, OMG *Available Specification OMG* Document Number: formal/2008-01-17 Standard document.

[7]: Vanderfeesten.I, Cardoso.J, Reijers.H.A, A weighted coupling metric for business process models, Technische Universities Eindhoven, *Department of Technology Management*, PO Box 513, 5600 MB Eindhoven, The Netherlands, University of Madeira, Department of Mathematics and Engineering.

[8]: Vanderfeesten.I, Reijers.H.A, Mendling.J, van der Aalst.W.M.P, Cardoso.J, On a quest for good process models: the cross-connectivity metric. *Advanced Information Systems Engineering (Proceedings 20th International Conference*, CAiSE'08, Montpellier, France, June 18-20, 2008).

[9]: McCabe T.J, A Complexity Measure, *IEEE Transactions on Software Engineering*, 2(4), pp. 308-320, 1976

[10]: Fenton N.E., Pfleeger S.L., Software Metrics: A Rigorous and Practical Approach, PWS *Publishing Company*, Boston, USA, 2ème edition, 1997.

[11]: Henry .S, Kafura.D, Software structure metrics based on information-flow. *IEEE Transactions on Software Engineering*, 7(5):510–518, 1981.

[12]: Briand.LC, Daly .JW, Porter.V, Wüst.J, A Comprehensive Empirical Validation of Design Measures for Object-Oriented Systems. 5th International Software Metrics Symposium (METRICS 1998), IEEE Computer Science, 43–53.

[13]: S.R.Chidamber, C.F.Kemerer, Authors' Reply to: comments on « A metrics suite for object oriented design », *IEEE Transactions on software Engineering*, 21(3), p.265, March 1995.

[14]: M. Hitz and B. Monastery, « Measure Coupling and Cohesion in Object-Oriented Systems ». *Proceedings of International Symposium* on Applied Corporate Computing (ISAAC'95), October 1995. (pp 24, 25, 274, 279).

[15] Bielak.J., Maccamy.R.C, Zeng.X, Stable coupling method for interface scattering problems by combined integral equations and finit elements, J, vol. 119, 1995, p. 374–384.