A Metric for Measuring BPEL Process Context-Independency

A. Khoshkbarforoushha, P. Jamshidi, A. Nikravesh, S. Khoshnevis, F. Shams
Automated Software Engineering Research Group
Electrical and Computer Engineering Faculty
Shahid Beheshti University GC
{a_khoshkbarforoushha, p_jamshidi, a_nikravesh, s_khoshnevis , f_shams}@sbu.ac.ir

Abstract

BPEL provides a workflow-oriented composition model for service-oriented solutions that facilitates the system integration through orchestration and choreography of services. In some cases, BPEL process designs can be highly complex owing to the vast number of services executed in global markets. Such heavy coupling and context dependency with partners in one side and the complicated structure of the processes on the other side, provoke several undesirable drawbacks such as poor understandability, inflexibility, inadaptability, and defects. Therefore, heavy context dependency should be avoided. This paper proposes a quantitative metric to measure BPEL process context-independency which lead SOA architect determines to the extent that a BPEL process is context-independent.

1. Introduction

The unpredictability of business processes requires maintainable workflow systems with the ability to dynamically adapt to the changing environment [7] [13]. This adaptation to the changing environment refers to the *flexibility* of the business processes.

The flexibility of a process is characterized by a capability to readily adapt to new, different, or changing requirements [4]. This adaptation requires the low degree of dependency (coupling) between business process and its changing environment. In this regard, solutions that do not exhibit low coupling might experience the following developmental difficulties [12]. Firstly, change in one module forces a ripple of changes in other modules. Secondly, modules are difficult to understand in isolation. Finally, modules are difficult to reuse or test because dependent modules must be included. Many researchers have attempted to develop a single metric that provides a comprehensive measure of software coupling [12]. Although dozens of coupling measures have been proposed, there are limited number of research and practical work, which provide metric for business process or composite service coupling measurement. In this regards, the major objective of this paper is to investigate BPEL process flexibility through introducing a novel quantitative metric to measure BPEL process context-independency. By adopting such a metric, SOA architect could determine to the extent that a BPEL process is flexible. This serves as a foundation for quality service-oriented solution design.

Context-independency is defined as the extent to which a BPEL process requires the knowledge of their surrounding environment. An environment, which a process needs to interact with and acquire or deliver the required information, consists of involved web services, clients, and resources.

Many efforts have been made by the service-oriented community to define standards and protocols in order to enable flexible business process management solutions. In this regard, technologies such as BPEL [20], WSCI and BPML are some of the main technologies which served to satisfy so called standardization. In fact, these technologies make it possible to define abstract business processes and then execute their instances. BPEL provides a workflow-oriented composition model for service-oriented solutions that facilitates the system integration through orchestration and choreography of services.

Business processes which are particularly under focus of this paper for context-independency exploration are, in fact, composite web services. The involved web services within a composite web service are coordinated and orchestrated through BPEL. Therefore, in BPEL, a business process is a coarse-grained web service (i.e. composite web service) executing a control flow to complete a business goal. The steps in the control flow execute activities that are centered on invoking partner services (e.g. Web services specified in WSDL) to perform tasks and return results back to the process.

The approach of the paper for measuring BPEL process context-independency is based on the coupling value analysis of a BPEL process to its partners (i.e. web services). The coupling value of a certain BPEL process is examined and quantified on the basis of measuring its interaction activities (e.g. <invoke>, <reply>, <receive>) within structured activities (e.g. <sequence>, <pick>).

2. Related work

While dozens of coupling measures such as content, external, and data coupling have been proposed [22] there are limited number of research and practical work, which provide metric for business process or composite service coupling measurement. Cardoso have introduced complexity analysis metrics to be used at design time to

978-1-4244-5299-6/09/$26.00 (c)2009 IEEE

evaluate the complexity of the process design before implementation actually takes place. The most significant work that can be mentioned is the cohesion and coupling metric proposed by Reijers [14], developed to analyze workflows. The paper includes an application of this heuristic in a realistic workflow process setting. In their previous paper [15], they compared the application of a simple workflow quality metric to various design dilemma's with the decisions of 14 experienced workflow designers. The outcomes matched, supporting the validity of the metric and the viability of the underlying idea. This metric, however, lacked facilities to handle conditional alternatives to achieve the same output, a construct very common in business processes and in administrative processes in particular [17]. Vanderfeesten [18] extends other work by specifically incorporating the effects of different types of connectors used on a process model's coupling level. Vanderfeesten [19] proposes a heuristic that offers guidance for the creation and evaluation of process designs. These heuristics can be used to select from several alternatives of the process design that is strongly cohesive and weakly coupled.

3. Is context-independency a necessity?

Regarding to context-independency definition, a context-independent system is expected to have the least coupling with its surrounding environment, in other words, context-independency is directly related to, and is synonymous with, loose-coupling.

Loose-coupling directly affects maintainability. It is an attribute of systems, referring to an approach to designing interfaces across modules to reduce the interdependencies across modules, and particularly, reducing the risk that changes within one module will create unanticipated changes within other modules [12]. In other words, loosely-coupled systems are least exposed to ripple effects caused by a change made or a fault/failure happened somewhere in the system. Therefore, this feature helps to have higher availability.

Loose-coupling provides extensibility to the design [10]. It specifically seeks to increase flexibility in adding modules, replacing modules and changing operations within individual modules. It also helps to reduce the overall complexity and dependencies, makes application landscape more agile and enables quicker change.

Loose-coupling also has effects on security. According to [6], with a loose-coupling approach, security can be designed as a set of services that are independent of any one application. In this way, security policy can be implemented once as a service and linked to each application that it applies to. This means that businesses can have better ways to ensure control over security across their own systems as well as the systems of their closest partners.

Additionally, testability and reusability are highly reduced when hidden, implicit dependencies exist due to the coupling between modules [12].

Very loosely-coupled systems have the added advantage that they tend to build more quickly. This is due to the low amounts of inter-module dependency. However, when making architectural decisions, one must carefully analyze the advantages and disadvantages of the level of coupling [5]. Loose-coupling is not universally positive. For example if systems are de-coupled in time using message-oriented middleware, it is difficult to also provide transactional integrity. Data replication across different systems provides loose-coupling (in availability), but creates issues in maintaining synchronization [10]. Some applications are tightly-coupled in nature and do not need high degrees of loose-coupling (like OLTP-style applications)[5]. Generally speaking, tighter coupling improves performance because of lessening the cost in the interfaces. There is always a trade-off between increasing many quality-attributes on one hand and the performance on the other [2]; in other words, loose-coupling helps to increase maintainability, reusability, security, etc. in cost of reducing performance.

4. BPEL process

In BPEL, a business process is a coarse-grained web service (i.e. composite web service) executing a control flow to complete a business goal. Each BPEL process consists of steps. Each step is called an activity. BPEL activities divided into Basic and Structured categories [5]. Basic activities are used for common tasks e.g. invoking a web service or manipulating data. The behavior of important basic activities is as follows:

- `<invoke>`: Invoking a web service.
- `<receive>`: Waiting to receive a message from the client.
- `<reply>`: Generating a response for synchronous operations.
- `<assign>`: Manipulating data variables.

Structured activities are used for arranging the structure of BPEL process. Structured activities can contain both basic and structured activities in order to implement complex business processes. The semantic and behavior of structured activities is as follow:

- `<sequence>`: Sequence for defining a set of activities that will be invoked in an ordered sequence
- `<flow>`: Defining a set of activities that will be executed in parallel.
- `<switch>`: Implementing branches.
- `<pick>`: Selecting one of a number of alternative paths.
- `<while>`: Defining the notion of loops.

Moreover, there are some other activities which are overviewed their behaviour, since in some sections of the paper we should have a basic understanding of their semantic:

- <partnerLinks>,<partnerLink>: Defining partners which are being interacted by composite service.
- <onMessage>, <onAlarm>: These activities are used within the <pick> construct to Capture events either message-based or time-based.

5. BPEL process context-independency

Context-independency is defined as the extent to which a BPEL process requires the knowledge of their surrounding environment. An environment, which a process needs to interact with and acquire or deliver the required information, consists of involved web services, clients, and resources. In our perspective, context-independency can be measured through coupling concept, as Vanderfeesten et al. [18] emphasize that the coupling measures the number of interconnections between the activities in a process model. On the other hand, a composite service is dependent to a context to the extent that is coupled with its web services, resources, and clients. Hence, service coupling measurement paves the way for context-independency measurement. Moreover, service coupling measurement provides a nontrivial insight into the communication overhead of discovered composite services with its related world. In this matter Papazoglou and van den Heuvel [11] state that the number of messages exchanged between a sender and addressee, that is communication protocol coupling, should be minimal.

A composite service, which is realized through BPEL, communicate and interact with involved partners (i.e. web services, clients, etc) by means of <invoke>, <reply>, <receive>, <onAlarm>, and <onMessage> activities. We refer to these activities as interaction activities in the rest of the paper.

We assign the coupling value of 1 for each of the interaction activities, since they impose simple interaction within a composite service.

Let

C_{ia}: refers to the coupling value of interaction activities (ia).

$$C_{ia} = 1$$

Nevertheless, the coupling of a composite service with its partner becomes significant on the condition that the interaction activities incorporated within the structured activities. For instance, <flow> construct may contains four invoke activities which should be executed in parallel. Thus, we need to compute the coupling in terms of the structured activities.

The coupling of interaction activities are computed as they are incorporated into structured ones in the following subsections.

5.1. Sequence activity

A <sequence> activity is used to define activities that need to be performed in a sequential order. In BPEL, <sequence> construct is represented as follows:

```
<sequence standard-attributes>
  standard-elements
  activity+
</sequence>
```

The coupling of BPEL process with <sequence> activity is calculated as follows:

$$C_{<\text{sequence}>} = n$$

n is the number of interaction activities within a sequence.

In fact, the sequence construct does not impose any further coupling to the interaction activities.

5.2. Switch activity

To express conditional behavior, the <switch> activity is used. It consists of one or more conditional branches defined by <case> elements, followed by an optional <otherwise> element. The case branches of the switch are considered in alphabetical order. The switch activity has the following structure:

```
<switch standard-attributes>
  standard-elements
  <case condition="bool-expr">+
    activity
  </case>
  <otherwise>?
    activity
  </otherwise>
</switch>
```

The coupling of BPEL process with <switch> activity is computed as follows:

$$C_{<\text{switch}>} = \sum_{i=1}^{n} \left(\frac{1}{n} * m \right)_i$$

Where

- n is the number conditions of switch construct.
- m is the number of interaction activity within condition i.

In our view, coupling of interaction within a switch construct is related to the probability of the occurrence of each of its conditions. Thus, the coupling of switch construct is calculated as the summation of the probability of each condition occurrence multiply with the number of interaction activities within that condition.

5.3. Pick activity

The <pick> activity is used to wait for the occurrence of one of a set of events and then perform an activity associated with the event. The structure of this construct is as follows:

```
<pick createInstance="yes|no"? standard-
attributes>
  standard-elements
```

```
  <onMessage partnerLink="ncname"
portType="qname"

operation="ncname" variable="ncname"?>+
    <correlations>?
      <correlation set="ncname"
initiate="yes|no"?>+
    </correlations>
    activity
  </onMessage>
  <onAlarm (for="duration-expr" |
until="deadline-expr")>*
    activity
  </onAlarm>
</pick>
```

The coupling of BPEL process with <pick> activity is computed as follows:

$$C_{<\text{pick}>} = \sum_{i=1}^{n} \left(\frac{1}{n} * m \right)_i$$

Where

- n is the number of <onAlarm> and <onMessage> constructs.
- m is the number of interaction activities within <onAlarm> or <onMessage> constructs.

The semantic and behaviour of pick construct is similar to switch activity so that we treat in a same way as we did for the coupling of composite service (i.e. BPEL process) with switch activity.

5.4. Flow activity

The <flow> activity provides concurrent execution of enclosed activities and their synchronization. The syntax of flow activity is as follows:

```
<flow standard-attributes>
  standard-elements
  <links>?
    <link name="ncname">+
  </links>
  activity+
</flow>
```

The coupling of BPEL process with <flow> activity is computed as follows:

$$C_{<\text{flow}>} = n$$

Flow construct provide a kind of parallel interactions with the partners. The activities within a flow construct are triggered at same time, while they are terminated at different moments. Therefore, the numbers of interactions with the partners are not constant during the time, but the total interactions are still *n*.

5.5. While activity

A <while> activity is used to define an iterative activity. The iterative activity is performed until the specified Boolean condition no longer holds true. In BPEL, the representation of while construct is as follows:

```
<while condition="bool-expr" standard-
attributes>
  standard-elements
  activity
</while>
```

The coupling of BPEL process with <while> activity is computed as follows:

$$C_{<\text{while}>} = N_i * n$$

Where

- n is the number of interaction activities within while construct.
- N_i refers to the number of loop iterations.

Process coupling with its partners is calculated as the number of <while> iterations multiply with the number of interaction activities within that.

5.6. BPEL process context-independency measurement

There are some assumptions which are observed in our formulations. For a given composite service we compute all kinds of interactions, since in our perspective it shows the extent to which a composite service coupled with its certain partner. As a result, synchronous and asynchronous relations are considered implicitly.

Now, based on the above formulations we are able to measure BPEL coupling and context-independency as follows:

Let:

- P_{CM}: refers to process coupling measurement.
- j: refers to the number of structured activities.
- P: is constant to 1.
- P_{CIM}: refers to process context-independency measurement.
- C_{ia}: refers to the coupling value of interaction activities (ia).
- C_i: refers to the coupling values which is imposed by i structured activities.
- k: refers to the number of interaction activities which are not surrounded in any structured activity.

$$P_{CM} = \sum_{1}^{k} C_{ia} + \sum_{1}^{j} C_i \quad (1)$$

$$P_{CIM} = \frac{p}{P_{CM}} \quad (2)$$

Eq. (1) is the summation of service coupling with its partners which is provoked by interaction activities. Based on what was discussed, the relationship between service coupling and service context-independence is reverse, such that service context-independency is measured via Eq. (2).

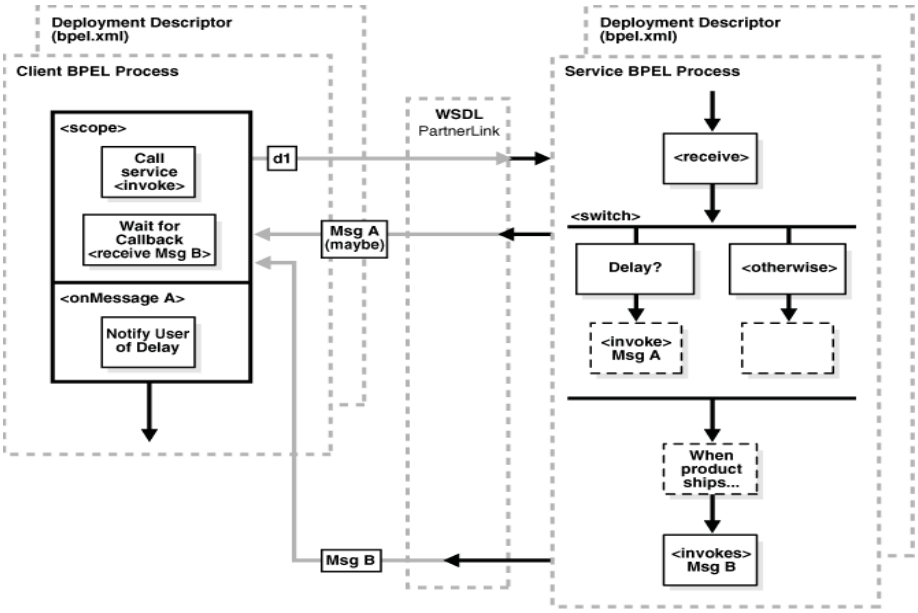

Figure 1: One Request, a Mandatory Response, and an Optional

6. Interaction patterns

Every possible metric for BPEL process coupling and following context-independency measurement must be evaluated against various kinds of interaction patterns. As specified in [9], there are ten interaction patterns as follows:

- One-Way Message
- Synchronous Interaction
- Asynchronous Interaction
- Asynchronous Interaction with Timeout
- Asynchronous Interaction with a Notification Timer
- One Request, Multiple Responses
- One Request, One of Two Possible Responses
- One Request, a Mandatory Response, and an Optional Response
- Partial Processing
- Multiple Application Interactions

In the same way, [1] enumerated 13 patterns in four categories including Single-transmission bilateral interaction patterns, Single-transmission multilateral interaction patterns, Multi-transmission interaction patterns, Routing patterns.

The proposed metric for context-independency measurement remarkably support these patterns. This is due to the fact that our metric is on the basis of BPEL activities. In other words, the introduced metric computes every single interaction since every single interaction which is realized with interaction activities, are contemplated. Moreover, as discussed earlier such an approach make us to take synchronous and asynchronous relations into account.

To illustrate the point, we apply the metric to one of the complicated patterns that is *one Request, a Mandatory Response, and an Optional Response* pattern, figure 1.

In this type of interaction, the client sends a single request to a service and receives one or two responses. Here, the request is to order a product online. If the product is delayed, the service sends a message letting the customer know. In any case, the service always sends a notification when the item ships.

In the Service BPEL process side we have two structured activities. The first one is `<sequence>` (a BPEL process will have the top-level element which is usually `<sequence>`). The second one is `<switch>` construct which is inside of the `<sequence>` activity. Within these structured activities there are four interaction activities including one `<receive>` and three `<invoke>` constructs. After such analysis, we are able to measure process context-independency via the proposed formulations. For instance in this pattern the context-independency of Service BPEL Process is the value of 0.33 (i.e. P_{CIM}= 0.33). In the same manner, all kinds of the patterns are supported via our metrics.

7. Scenario

We apply the proposed metric to fulfilPatientMedicalTest() composite service which is presented at different versions; Figure 2 and Figure 3. For readability reasons, the BPEL processes are represented through Business Process Modelling Notation (BPMN) [8]. To clarify the calculations, the required basic and structured BPEL activities have been attached to the BPMN diagrams, even though it is possible to generate BPEL execution definitions from BPMN processes [16]. Moreover, we ignore some basic or structured activities to oversimplify our example (e.g. a BPEL process will have the top-level element which is usually `<sequence>` or `<flow>`).

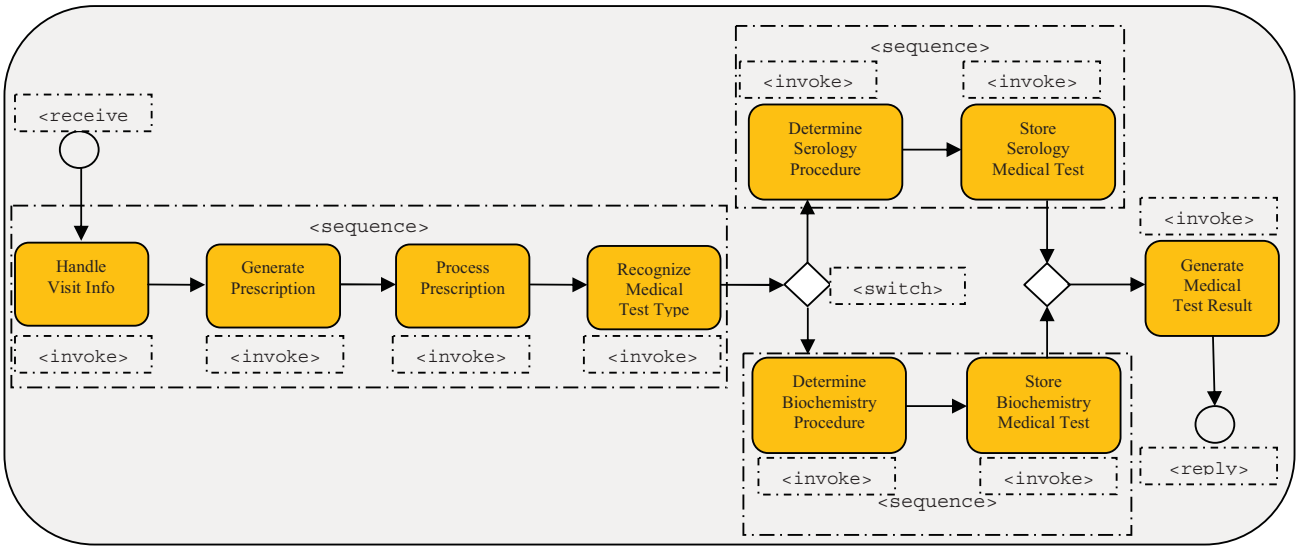

Figure 2. fulfilPatientMedicalTest () BPEL process (First version).

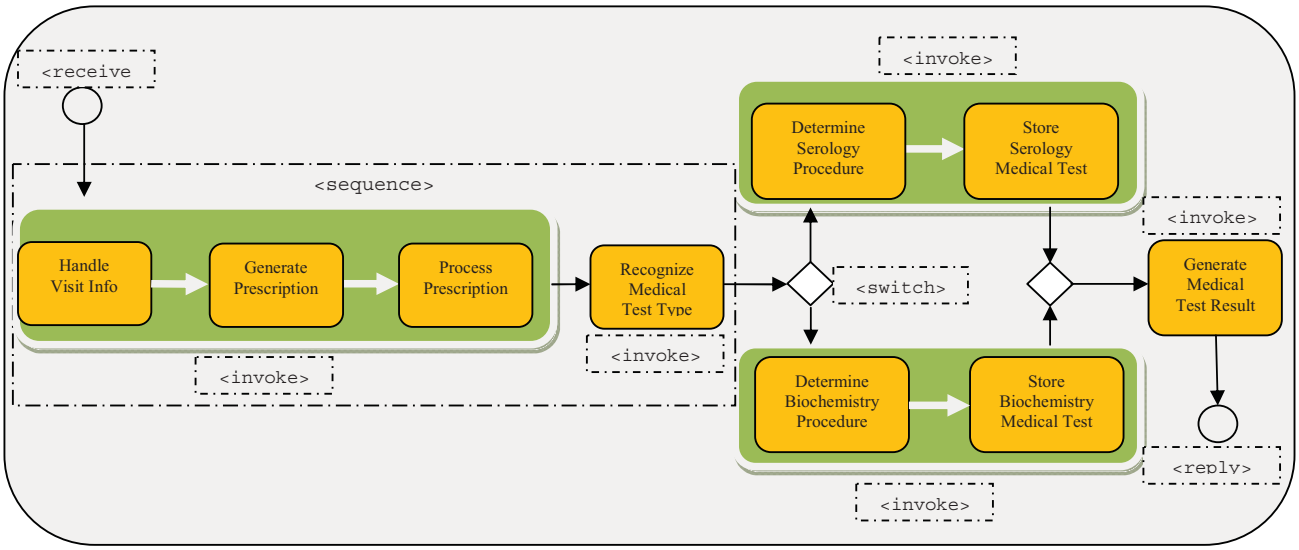

Figure 3. fulfilPatientMedicalTest () BPEL process (Second version).

These BPEL processes are derived from the business processes of Iran Hospitals and Laboratories. The first version of the fulfilPatientMedicalTest() composite service consists of nine activities. The service is triggered by receiving the information of patient's visit. After storing visit information the physician prescription has been generated. In the next step the prescription is processed and required test has been recognized. Based on the type of test that is biochemistry or serology the procedure of test is determined and following the required information is stored. Finally the result of the test has been generated and delivered to the patient.

These two versions are actually the different designs of the same BPEL Process. Indeed, the difference between them arises from the diverse granularity of their partners. In other words, the second version is more modular than the first one.

- Context-Independency:

  Context-independency of the first version:

$$P_{CM} = 1+4+\left[(0.5*2)+(0.5*2)\right]+1+1=9$$

$$P_{CIM} = \frac{1}{9} = 0.11$$

  Context-independency of the second version:

$$P_{CM} = 1+2+\left[(0.5*1)+(0.5*1)\right]+1+1=6$$

$$P_{CIM} = \frac{1}{6} = 0.16$$

Table 1: The results of Context-independency value measurement and corresponding experts' judgment.

No	BPEL Process Name	Context-Independency Values			Max(P_{CIM})	Experts' consensus	Success/Failure
		Version 1	Version 2	Version 3			
1	*Issue Letter Of Guarantee*	0.133	0.143	0.118	Version 2	Version 2	✓
2	*Discard Letter Of Guarantee*	0.133	0.125	0.117	Version 1	Version 2	×
3	*Prepare local branch Balance sheet*	0.130	0.154	-	Version 2	Version 2	✓
4	*Advance Payment Guarantee*	0.111	0.143	0.166	Version 3	Version 3	✓
5	*Prepare total Balance sheet*	0.153	0.154	-	Version 2	Version 1	×
6	*Centers performance measurement*	0.142	0.166	-	Version 2	Version 1	×
7	*Ad-hoc Services*	0.099	0.222	0.154	Version 2	Version 2	✓
8	*Prepare foreign branch Balance sheet*	0.132	0.154	-	Version 2	Version 2	✓

We assume that the client that invokes our process is different from the one who receives the expected reply or output of the service.

As the computations denote, context-independency for first version of fulfilPatientMedicalTest() process is less than the second one. As discussed earlier the flexibility is the function of context-independency and complexity. As a result, fulfilPatientMedicalTest() composite service (i.e. BPEL process) is much more flexible if it is designed with the structure like the second version.

However, one could question that why the composite service version 2 is in more appropriate level of context-independency. In fact, the second version is more modular and its modules are functionally cohesive [11]. This means partners of composite service version 2 is related to one specific problem (i.e. test), whereas the version 1 is concerned with more than one problem (i.e. test, prescription processing, etc). Such cohesiveness of a module contributes lots of software qualities such as flexibility, maintainability, reusability, reliability, etc [3]. Therefore, after context-independency analysis architects can reengineer the BPEL process to reduce the complexity and increase flexibility, if it is in need.

8. Evaluation of the metric

In order to empirically validate the metrics that have been described, further experiments using experimental models need to be carried out [21]. In this regard, to gain confidence in our theoretical works, we conducted a controlled experiment in which we take 8 BPEL processes from two projects including TJT Core-banking (TCB) processes at international sector and ad-hoc services at IKRF relief foundation of Iran.

These processes were designed in more than one version. These composite services were evaluated by use of our metric and also examined qualitatively by three experts for business process. The two evaluations then were compared to demonstrate that the model is a valid characterization of BPEL process context-independency. The table 1 denotes the results of our experiment.

Table 1 contains 20 versions of BPEL processes and their corresponding context-independency values. The maximum of these values indicates the most context-independent variant, which is denoted by *Max(P_{CIM})* column. Regarding these versions are examined and analysed by business process management experts, table 1 also embodies the selected versions, which seem to be more context-independent comparing to the others.

We obtained indications of a positive evaluation of our metric from the experts involved in the case study. As table 1 shows, the consensus provided by the participants when choosing the best process design were in favour of our metric. However, as indicated in the table 1, we have 3 mismatches between the expert selection and our metric measurement. This contradiction arises from the fact that experts employ the most aspects of the quality when selecting the alternatives, nevertheless, our metric only measure one aspect of them that is context-independency. Regarding that the proposed metric is expected to be used in service-oriented solution development lifecycle, firstly, we have validated it through a survey, which actually take place before implementation phase. In order to empirically validate the adopted technical metrics and the method itself, further experiments need to be carried out.

9. Conclusions and future directions

Today's turbulent business atmosphere causes enterprise's processes to change dramatically. In order to meet these modifications, we need to design business processes, which are broadly realized with service oriented computing technologies such as web services, BPEL, etc, as flexible as possible. Therefore, it is important to develop measures to analyze the context-independency of processes

so that they can be reengineered to reduce the complexity and increase flexibility.

In this article, we proposed a metric for BPEL process context-independency measurement. Our approach for measuring BPEL process context-independency is based on the coupling value analysis of a BPEL process to its partners. The coupling value of a certain BPEL process is examined and quantified on the basis of measuring its interaction activities within structured activities. This paper also studied a certain BPEL process in detail which exhibit how the proposed metric works. Additionally, we conducted an experiment based on the processes of two running enterprise projects. The initial results demonstrate the metric is a valid characterization of BPEL process context-independency.

However, in a broader range we need to continue with two distinct approaches for further validation. Regarding that the proposed metric is expected to be leveraged at analysis and design phase of SOA development before implementation actually takes place, in the first fold, they can be validated through data collection from experienced practitioners who have engaged with extensive projects. The second fold is concerned with data collection and analysis after service implementation. This means, gathering information after implementation, open the way to validate metric empirically. Work is also planned to implement a toolset to automate the computation of the metric.

References

[1] A. Barros, M. Dumas, A. ter Hofsted, Service Interaction patterns; Joint initiative by SAP and Queensland University of Technology, co-funded by Queensland State Government. http://math.ut.ee/~dumas/ServiceInteractionPatterns/patterns.html

[2] Bass, Clements, Kazman, Software Architecture in Practice, Second Edition, Addison Wesley , 2003.

[3] Bowen, T. P., Post, J. V., Tsai, J., Presson, P. E., Schmidt, R. L., Software Quality Measurement for Distributed Systems, Guidebook for Software Quality Measurement, RADC-TR-83-175, Vol. II, Final Technical Report, Rome Air Development Center, Air Force Systems Command, Griffis Air Force Base, NY, 1983.

[4] Cardoso, J., 2007, Complexity Analysis of BPEL Web Processes, Software Process Improvement and Practice, Wiley InterScience; 12, 35–49.

[5] Erl, T., Service-oriented architecture: concepts, technology, and design, Prentice Hall, 2005Juric, M., B., Mathew, B., Sarang, P., 2006, Business Process Execution Language for Web Services, Packt Publishing, Birmingham, B27 6PA, UK, Second edition.

[6] Hurwitz,J., Bloor, R., Baroudi, C and Kaufman, M., Service Oriented Architecture for dummies, Wiley, 2007.

[7] P. J. Kammer, Gregory Alan Bolcer, Richard N. Taylor, Mark Bergnam, “Techniques for Supporting Dynamic and Adaptive Workflow” Computer Supported Cooperative.

[8] OMG, 2006, Object Management Group: Business process modeling notation specification, available at: http://www.bpmn.org/

[9] Oracle BPEL Process Manager Developer's Guide 10g Release 2 (10.1.2); BPEL processes common interaction patterns; available at: http://download.oracle.com/docs/cd/B14099_19/integrate.1012/b14448/interact.htm

[10] James Douglas Orton and Karl E. Weick, Loosely Coupled Systems: A Reconceptualization, Academy of Management Review 15 (2):203-223 1990

[11] Papazoglou, M., P., van den Heuvel, W., J., 2006, Service-Oriented Design and Development Methodology, Int'l Journal of Web Engineering and Technology (IJWET).

[12] Pressman, R.S., Software Engineering, a practitioner's approach, 5th Edition, McGrawHill, 2001.

[13] M. Reichert and S. Rinderle; On Design Principles for Realizing Adaptive Service Flows with BPEL; Proc. EMISA. GI Lecture Notes in Informatics, LNI P-95, 2006, pp. 133-146.http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.88.2255&rep=rep1&type=pdf

[14] H.A. Reijers and I.T.P. Vanderfeesten. Cohesion and Coupling Metrics for Work-flow Process Design. In J. Desel, B. Pernici, and M. Weske, editors, International Conference on Business Process Management (BPM 2004), volume 3080 of Lecture Notes in Computer Science, pages 290-305. Springer-Verlag, Berlin, 2004.

[15] H.A. Reijers. A Cohesion Metric for the Definition of Activities in a Workflow Process. Proceedings of the Eighth CAiSE/IFIP8.1 International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design 2003, pages 116-125, 2003.

[16] Stephen, A., W., 2006, Using BPMN to Model a BPEL Process, IBM Corp., United States, available at: http://www.bpmn.org/Documents/MappingBPMNtoBPELExample.pdf

[17] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow Patterns. Distributed and Parallel Databases, 14(1):5-51, 2003.

[18] Vanderfeesten, I., Cardoso, J., Reijers, H., A., 2007, A weighted coupling metric for business process models, The 19th International Conference on Advanced Information Systems Engineering (CAiSE Forum),11-15.

[19] Vanderfeesten, I., Cardoso, J., Reijers, H., A., Evaluating workflow process design using cohesion and coupling metrics.

[20] WS-BEPL. 2005. Business Process Execution Language for Web Services, http://www-128.ibm.com/developerworks/library/specification/ws-bpel/.

[21] Zelkowitz MV, Wallace DR. 1998. Experimental models for validating technology. IEEE Computer 31(5): 23–31.

[22] Zuse H, Software Complexity Measures and Models, de Gruyter, NY, 1990.