A Metric for Composite Service Reusability Analysis

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

ABSTRACT
Reusability is one of the most prominent service principles that can reduce cost of solution ownership when design services for reuse. The concept of reuse is supported by and can enable a number of complementary service principles that makes its analysis complicated. However, well-engineered business capabilities could be encapsulates within a well-designed interface to produce more reusable composite services. There are several prescriptive guidelines help to ensure that your service candidates attain a balance of proper logic encapsulation and adherence to the standard description, but there is no quantitative metric to get ensured that a given composite service has and to what extent different versions of a composite service differ in reusability capability. This work is to propose a quantitative metric to measure the reusability of composite services. The approach is based on the analysis of Description and Logic Mismatch Probability of a composite service will be reused within potential solutions. By adopting this metric, service reusability analysis could be conduct quantitatively that leads to realize an optimized service-oriented solution in terms of its reusability.

Keywords
Service Reusability, SOA Metric, Composite Service, BPEL Process Reusability

1. INTRODUCTION
Many experts believe that reusability is inherent to Service-Oriented Architecture (SOA), and studies demonstrate that organizations regard reuse as the top driver for SOA adoption [1]. Likewise, in terms of the design of services reuse is acknowledged as the main purpose [2].

When a service encapsulates logic that is useful to more than one service consumer, it can be considered reusable. When designing services, it is important to be able to design them for reuse so that they can perform a given function wherever this function is required within an enterprise. In SOA literature, a business process is a coarse-grained web service (i.e. composite web service) executing a control flow (service logic) to complete a business goal.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
WETSoM'10, May 4, 2010, Cape Town, South Africa.
Copyright © 2010 ACM 978-1-60558-976-3/10/05…$10.00.

A composite service cannot be reused in potential context due to mismatches between requirements and the provided service in terms of its logic and description. The context is categorized to current solution that a service being reused and potential future solutions that it might be reused.

Service designers should keep in mind that any service they product can potentially become a reusable asset [3]. The designers should not exclusively focus on requirements of the initial consumers of a service, but rather should undertake more extensive business analysis in order to determine more complete requirements. There is a set of guidelines [4, 5, 6] that supplement the previous modeling process with some additional considerations. These guidelines help ensure that your service candidates attain a balance of proper logic encapsulation and adherence to standard description. Applying these prescriptive methods may be inconvenient, which will directly affect the work products that are being produced. On the other hand, automating the methods will reduce the labor time and eliminate the source of errors in applying the method. Developing quantitative measures is the first action toward the automation [7].

Moreover, from the viewpoint of service users, how they can detect which service is the most reusable among several services implementing the same specification, and how they can select them with higher reusability are key issues. It is necessary to measure the reusability of services in order to realize the reuse of them effectively.

This work proposes a quantitative metric to measure the reusability of service logic being reused within and across potential context. Our approach is based on measuring and analyzing logic and description mismatch and consolidating them to a metric formula for measuring the probability of a service will be reused within potential service-oriented solutions.

By adopting this metric analysis of service reusability could be conduct quantitatively that leads to have an optimized service-oriented solution in terms of its reusability and could help to realize the vision of service-orientation.

2. RELATED WORK
Software reuse has been the subject of extensive investigation in the context of object-oriented programming and component-based development [4]. We define service reuse as the ability to participate in multiple service assemblies (compositions) beyond those for which it was originally designed. Given this perspective, services must be composable to achieve good levels of reuse, i.e. service reuse is closely related to service composability. For example, a well-designed credit card verification service can be reused in a large number of payment applications, and therefore is highly reusable. It follows from the above discussion that the bias

towards coarse-grained (i.e. aggregated), message-oriented services, favored by most SOA practitioners makes achieving reuse difficult in practice as such services are not readily composable. To the best of our knowledge, several reports [5, 6, 7, 8] have been published on proposing techniques and methods to design reusable services; however, there is no research and practical work on measuring reusability of services by quantitative technical metric. The most related work [9] proposes a metric to measure context independency, which is a reusability factor of the services in a given context. The quality of the SOA solution will be taken into consideration by adopting technical metrics and used to guide the entire solution life cycle. The analytic data will be associated with the solution artifacts in order to optimize the selection and composition of solution patterns and architectural building blocks.

3. BPEL PROCESS STRUCTURE

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. Therefore, in this paper we use BPEL process, composite service, or service interchangeably. Each BPEL process consists of steps. Each step is called an activity. BPEL activities divided into Basic and Structured categories [10]. 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.

4. BPEL PROCESS REUSABILITY

Reusability is the degree to which a thing can be reused [11]. Service reusability is the key determinant factor for identification of optimally granular services, since its proved role in saving cost of development, and maintenance. While significant research has been devoted to reusability concept in object-oriented design [12] there is no theoretical or empirical metric for service reusability measurement.

Unlike components, services are realized at a higher-level of abstraction directly supporting business processes [13], so it seems measuring their reusability is relatively straightforward. However, reusability in service oriented area is not an isolated concept and architects should decide about that, while other contexts and projects are contemplated. This means, an architect must analyze whether the given service can be reused in other business processes or contexts.

The term context in this paper is categorized to current and potential ones. Reusability in current and potential contexts makes sense if we consider the point that every identified, specified, realized and finally implemented service, particularly composite services, should be reused in other possible SOA solutions. Other possible assumptions could be those cases where services are going to be published and utilized in a broader range and new business ventures that take advantage of the reuse capabilities and reusable assets. Based on this assumption, the reusability of service must be calculated with respect to the current and potential reusability.

A given composite service cannot be reused in potential context due to mismatches between requirements and the provided service.

Definition 1: (Mismatch). Mismatch refers to the point that a given composite service cannot be matched with other context requirements.

In this regard, there are two kinds of mismatches including Description mismatch, and Logic mismatch.

Definition 2: (Description Mismatch). Description mismatch refers to the mismatch between the requirements and the description of the given composite service i.e. WSDL.

When a certain service consumer either a service or an SOA architect wants to use the given service based on their requirements, it may not be reused due to mismatch in WSDL including data types and messages of service operations. Of course, some solutions could help an architect utilize a service with mismatch in description; however it is not straightforward and requires some modifications in the base service or using some converters. Therefore, description mismatch probability calculation of a composite service can give an insight about its reusability to the service designer.

Definition 3: (Logic Mismatch). Logic mismatch refers to mismatch between the requirements and the logic of given composite service. Composite service logic is utilized through the flow control of basic and structured activities within it.

In fact, services are composed and orchestrated in accordance with business rules [14]. This means, the structure and arrangement of web services within a composite one is imposed by business rules. Regarding different contexts may enjoy dissimilar business rules, consequently, a service cannot be reused in potential contexts for the sake of disparate business rule which impose disparate structure and control flow i.e. logic mismatch.

To illustrate the point, consider manageEmergencyPatient() service. This service consists of seven steps which is coordinated through `<sequence>` activity. In some hospitals the sequence of emergency patient management is similar to the Fig. 1. That is, after receiving and processing client request, the patient is

admitted and in the next step the payment verification scheme is launched. However, in the other hospitals it would be possible that the patient should be paid in advance in order to be provided with medical care and services. Therefore, the given structure may mismatch with the business rule in some hospitals and following the manageEmergencyPatient() service cannot be reused entirely. Consequently, the mismatch probability, which is provoked by how a service is coordinated and arranged with structured activities, would be taken into account.

Fig 1. ManageEmergencyPatient process

Definition 4: (Mismatch Probability). Mismatch probability (MMP) refers to the probability that a given composite service cannot be matched with other context requirements. Obviously, probability is a numerical measure of the likelihood of an event relative to a set of alternative events.

In the following subsections, both description and logic mismatch probability will be calculated.

4.1 Description mismatch analysis

Each BPEL process is a web service and needs a WSDL document. A WSDL document is just a simple XML document that contains set of definitions to describe a web service including either atomic or composite one. A WSDL document describes a web service using major elements, table 1.

Table 1: Major elements of WSDL document.

Element	Defines
<types>	The data types used by the web service.
<message>	The messages used by the web service.
<portType>	The operations performed by the web service.
<binding>	The communication protocols used by the web service.

A client will usually invoke an operation on the BPEL process to start it. With the BPEL process WSDL, we specify the interface for this operation. We also specify all message types, operations, and port types a BPEL process offers to other partners. However, the source of description mismatches come from operation message number and their data types.

The <types> element encloses data type definitions that are relevant for the exchanged messages. For maximum interoperability and platform neutrality; WSDL uses XML schema syntax to define data types. The XML schema distinguishes between primitive, derived, and complex data types [15].

Primitive data types: Primitive data types are those that are not defined in terms of other data types. Currently, XML schema defines 19 primitive data types, figure 2.

Derived data types: Derived data types are those that are defined in terms of other data types. There are three kinds of derivation: by restriction, by list, and by union. The XML Schema has 25 derived types build-in, figure 2.

Complex data types: XML Schema provides a flexible and powerful mechanism for building complex data structures from its simple data types. You can create data structures by creating a sequence of elements and attributes. You can also extend your defined types to create even more complex types. XML Schema has three compositor elements that allow constructing complex data types from simpler ones: sequence, choice and all. The behavior of these compositor elements is defined in table 2.

Table 2. Complex type compositor elements behavior.

Element	Complex Type Behavior
Sequence	All the complex type's fields must be present and in the exact order they are specified in the type definition.
All	All of the complex type's fields must be present but can be in any order.
Choice	Only one of the elements in the structure can be placed in the message.

If neither a *sequence* element, an *all* element, nor a *choice* is specified, a sequence is assumed.

Description mismatch probability calculation is done as follows.

Let.

- MP: refers to match probability.
- MMP: refers to mismatch probability.
- MP_P: refers to match probability of input/output parameters data types.
- MMP_{CD}: refers to mismatch probability of a complex type.
- MP_{SD}: refers tomatch probability of service description.
- MMP_{SD}: refers to mismatch probability of service description.
- d: refers to the total number of defined or built in primitive or derived type in XML schema. That is, in case of primitive type, d has the constant value of 19 and in case of derived type, d has the constant value of 25.

- *l*: refers to the number of primitive, derived or complex data type of a given operation parameters.
- *n*: refers to the number of simple or even another complex type within Sequence, Choice, or All element.
- O_j: refers to the number of service operations.

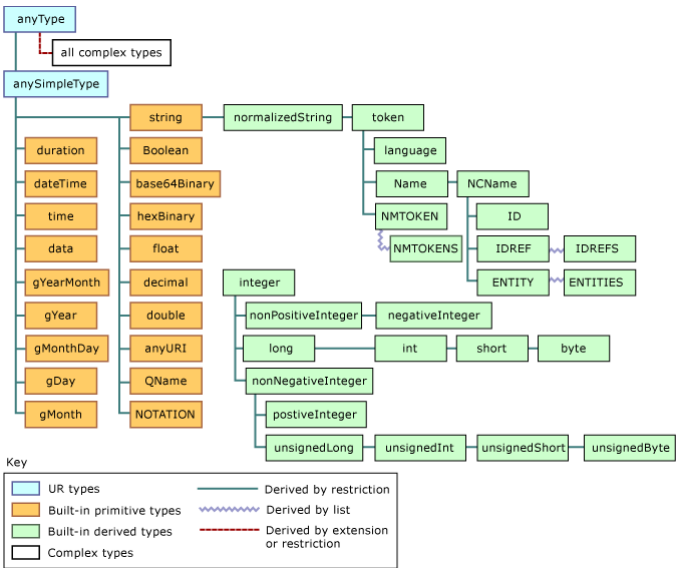

Figure 2. XML built in data type hierarchy [16].

$$MP_P = \begin{cases} \prod_{j=1}^{l}\left(\frac{1}{d}\right)_j & primitive type \\ \prod_{j=1}^{l}\left(\frac{1}{d}\right)_j & Derived type \\ \prod_{j=1}^{l}\left(MP_{CD_j}\right) & Compelex type \end{cases} \quad (1)$$

$$MP_{CD} = \begin{cases} \left(\frac{1}{n!}\right)\times\left(\prod_{i=1}^{n}\left(\frac{1}{d}\right)_i\right) & Sequence \\ \left(\frac{1}{2^n}\right)\times\left(\prod_{i=1}^{n}\left(\frac{1}{d}\right)_i\right) & Choice \\ \left(\prod_{i=1}^{n}\left(\frac{1}{d}\right)_i\right) & All \end{cases} \quad (2)$$

$$MP_{SD} = \left(\prod_{k=1}^{m}\left(MP_P\right)_{O_k}\right) \quad (3)$$

$$MMP_{SD} = \left(1 - \left(\prod_{k=1}^{m}\left(MP_P\right)_{O_k}\right)\right) \quad (4)$$

The equation 1 calculates match probability of both input and output parameters data types. As specified earlier, Twenty-five derived types are defined within the specification itself, and further derived types can be defined by users in their own schemas. We refer to those derived types out of the 25 built in ones as complex types. Since, complex data types include some primitive data types with *Sequence* or *Choice* behavior; for example, hence their match/mismatch probability must be calculated in terms of these behaviors, equation 2.

To illustrate more, we have to emphasize that the events of occurring one of the possible data types for each of the input/output parameters are *independent events*. Based on probability theory, assuming independence (or that the mismatch of either parameter data type is not influenced by the success or failure of the other parameter data type), the service description mismatch probability can be calculated through probability calculation of the description not matching, or the matching of the service description and then, the probability of service description mismatch is simply 1 (or 100%) minus the match probability, equation 3.

This question may arise that "Did the metric take the number of parameters as an important source of mismatch into account?". The answer is positive. In fact, provided we accept that the more parameters lead to the more description mismatch probability, by increasing the number of parameter result in more type mismatch which is calculated through MP_p.

4.2 Logic mismatch analysis

In addition to description mismatch concept, a BPEL process cannot be reused in potential contexts for the sake of disparate business rule which lead to disparate structure and control flow i.e. logic mismatch. For the purpose of logic mismatch probability, firstly we have to calculate match probability of each structured activities within a composite service (i.e. BPEL process). In this regard, the following subsections explore how match probability of BPEL process constructs is calculated.

4.2.1 <Sequence> match probability

A `<sequence>` activity is used to define activities that need to be performed in a sequential order. The match probability (MP) of `<sequence>` activity is calculated as follows:

$$MP_i = \left(\frac{1}{n!}\right) \quad (5)$$

- *n:* is the number activities within a sequence.
- *i:* refers to `<sequence>` construct.

From our viewpoint, the match probability involved in a sequence of activities is insignificant as all the activities are invoked in sequence. Indeed, the match probability is directly dependent on the different ways of arranging the activities in a sequence i.e. permutations. Thus, in case of '*n*' activities, we had *n*! as a total number of sample space members. Based on the probability theory, the match probability of a service with one sequence activity computed via the equation 4.

However, there is an exception in considering `<sequence>` activity in a BPEL code. It is expected that the `<sequence>` activity which is used upon for synchronous invocation of web

service operations is not taken into account, since the permutation is not possible.

4.2.2 <Switch> match probability

The <switch> activity is used to express conditional behavior. 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 match probability of <switch> activity is calculated as follows:

$$\mathrm{MP_i} = \left(\frac{1}{2^{\mathrm{n}}}\right) \quad (6)$$

- *n:* is the number of conditions.
- *i:* refers to <switch> construct.

The <switch> activity consists of an ordered list of conditions specified by a <case> element followed by one optional <otherwise> element. In our perspective, switch activities can be changed based on its conditions. Each condition in a switch activity can or cannot be changed according to business rule. We compute the contingency of this behaviour via the power set of the number of conditions, i.e. 2^n.

4.2.3 <Pick> match probability

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 match probability of <pick> activity is computed as follows:

$$\mathrm{MP}_i = \left(\frac{1}{2^{\mathrm{n}}}\right) \quad (7)$$

- *n:* is the number of events.
- *i:* refers to <pick> construct.

For the purpose of calculating match probability of <pick>activity we should treat in the same way as we did for switch activity, since the semantic behaviour behind them is equivalent. However, in the above equation 'n' indicates the number of events which is cached through <onAlarm> and <onMessage> activities.

4.2.4 <Flow> match probability

The <flow> activity provides concurrent execution of enclosed activities and their synchronization. The match probability of <flow> activity is zero, since the activities within a flow are processed on the condition that the order of execution is not defined. This means, different versions of a same business rule does not affect the flow control of <flow> construct.

$$\mathrm{MP}_i = 0 \quad (8)$$

- *i:* refers to <flow> construct.

4.2.5 <While> match probability

A <while> activity is used to define an iterative activity. The iterative activity is performed until the specified Boolean condition no longer holds true. The match probability of <while> construct has the value of 0.5, as the match can be occurred at Boolean condition of the activity:

$$\mathrm{MP}_i = \frac{1}{2} \quad (9)$$

- *i:* refers to <while> construct.

4.2.6 Logic mismatch measurement

In previous subsections, the match probability of all kinds of structured construct was calculated. Since a BPEL process may have more than one of them at the same time, it is expected to compute the match probability in case of more than one structured activities in a composite service. Thus, logic match probability of a composite service with any number of structured constructs is calculated through equation 10.

$$\mathrm{MP}_{SL} = \left(\prod_i^n MP_i\right) \quad (10)$$

Where

- *i*: refers to specific structured construct.
- *n*: refers to the total number of structured constructs within a certain BPEL process.
- MP_{SL}: refers to the Service Logic (SL) match probability.

Obviously, logic mismatch probability of a composite service with any number of structured constructs is calculated via equation 11.

$$\mathrm{MMP}_{SL} = \left(1 - \left(\prod_i^n MP_i\right)\right) \quad (11)$$

Where

- MMP_{SL}: refers to the Service Logic (SL) mismatch probability.

4.3 BPEL process reusability measurement

Now, we are able to measure the BPEL process reusability. In this regard, firstly we have to calculate a BPEL process total mismatch probability that is both description and logic mismatch probability. A BPEL process total mismatch probability is as follows:

$$\mathrm{MMP_S} = \left(1 - \left[\left(MP_{SD}\right) \times \left(\mathrm{MP}_{SL}\right)\right]\right) \quad (12)$$

Where

- MMP_S: refers to mismatch probability a service (i.e. BPEL process).
- MP_{SL}: refers to the Service Logic (SL) match probability.
- MP_{SD}: refers tomatch probability of service description.

Based on the calculation of BPEL process total mismatch probability, BPEL process reusability can be computed as follows:

Let:

- R_c: refers to the numbers a BPEL process reused in current context.
- R_p: refers to potential reusability of a BPEL process in potential contexts.

$$R_p = R_c \times \left[1 - MMP_S\right] \quad (13)$$

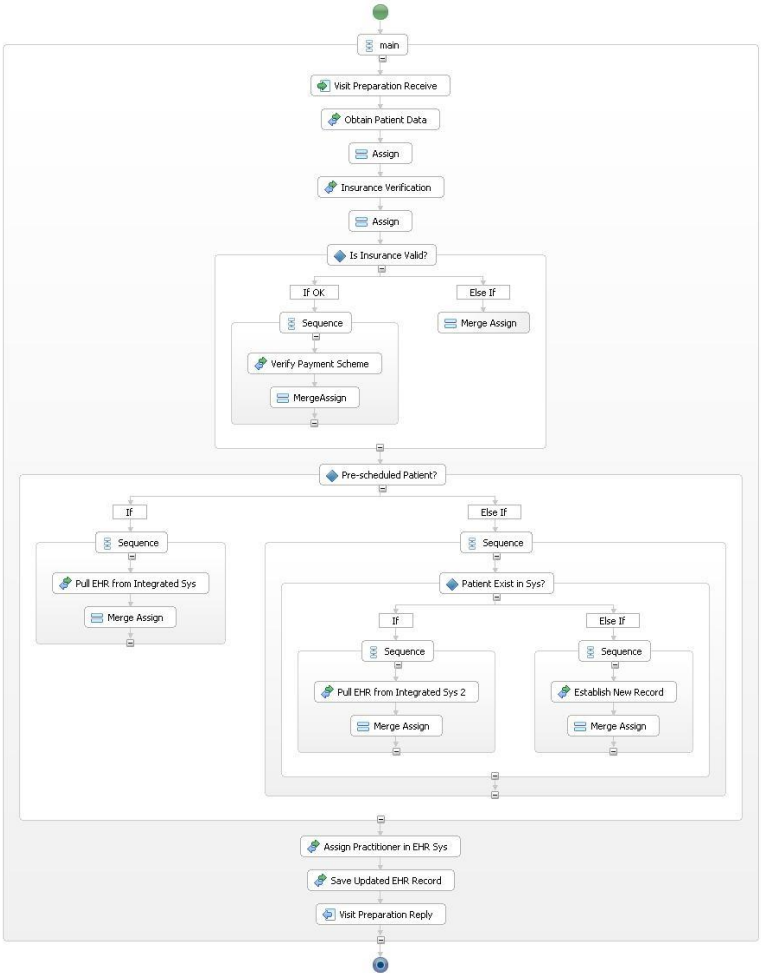

Figure 3: Visit Preparation BPEL process.

Equation 13 computes the potential reusability of a given a BPEL process with respect to the mismatch probability of a BPEL process in the prospect contexts.

However, if one takes a closer look at the mismatch probability concept, it can be deduced that the coarse-grained services has the higher mismatch probability and following less reusability and vice versa. Such a deduction is correct and it can be confirmed via cohesion concept.

Dhama (1995) [17] defines a relationship between functional cohesion and generality of a typical module as follows:

$$Functional\ Cohesion\ \alpha\ \frac{1}{Generality}$$

The above relationship implies that when the number of functionality (i.e. generality) decreases, the module cohesiveness becomes stronger. Besides, more cohesion leads to better reusability [18]. Likewise, Feuerlicht and Lozina (2007) in [4] state that "there is an inverse relationship between service granularity and service reusability; as the scope of functionality implemented by a given service increases, the potential for its reuse diminishes." As a result, it can be inferred that the mismatch probability concept is theoretically correct building block to constitute service potential reusability metric.

5. SCENARIO

In order to elaborate more on how the proposed metric works, we apply it to one of the important healthcare processes that is Visit Preparation, figure 3. This process was adopted from one of the IBM SOA solution in healthcare organization [19]. We also apply some assumptions to the scenario to make it as simple and clear as possible. For readability reasons, the BPEL processes are modeled and represented through Eclipse BPEL Designer.

The BPEL process is triggered by receiving visit preparation request. After getting patient information, insurance verification is invoked. When the verification completed, the process check the patient status including he has any Electronic Health Record (EHR) in the integrated system or not. In the next step the qualified practitioner is assigned to EHR system. After that, EHR record is updated. Finally the result of the visit preparation has been generated and replied. Besides, the given BPEL process has two operations, in which table 3 contains their corresponding input and output parameters.

Table 3: operations of given example and their corresponding types.

Service Operations	Parameters	Types
Visit Preparation Receive	PatientID	Primitive
Visit Preparation Reply	VisitNote	Complex

Table 4: The results of Reusability value measurement and corresponding experts' judgment.

BPEL Process Name	Potential Reusability Value (R_p)			Max(R_p)	Experts' consensus	Success/ Failure
	Version1	Version 2	Version 3			
Issue Letter Of Guarantee	0.000018	0.001041	0.0000325	Version 2	Version 2	✓
Discard Letter Of Guarantee	0.000108	0.000108	0.001302	Version 3	Version 3	✓
Prepare local branch Balance sheet	0.000520	0.002604	-	Version 2	Version 2	✓
Prepare total Balance sheet	0.002604	0.041666	-	Version 2	Version 2	✓
Centers performance measurement	0.008333	0.041666	-	Version 2	Version 1	×
Prepare foreign branch Balance sheet	0.002083	0.010416	-	Version 2	Version 2	✓
Fulfill patient medical test	0.000086	0.010416	0.04160	Version 3	Version 3	✓
Emergency Patient Visit Preparation	0.005208	0.020833	-	Version 2	Version 1	×

In accordance with the provided information about the scenario, we are able to measure its potential reusability.

- Description match probability:

PatientID is has String dataType. Contrary, VisitInfo is a complex type with sequence behavior in it. VisitInfo consists of three primitive types including (ID: String, VisitDate: String, Medications: String)

$$MP_{PatientID} = \frac{1}{19} = 0.052$$

$$MP_{VisitNote} = \frac{1}{3!} \times \left(\prod_{1}^{3} \frac{1}{19} \right) = 0.16 * 0.0001 \approx 1.6 \times 10^{-5}$$

$$MP_{SD} = 0.052 \times (1.6 \times 10^{-5}) \approx 8.32 \times 10^{-7}$$

- Logic match probability:

$$MP_{SL} = \left(\frac{1}{6!} \right) \times \left(\frac{1}{2^2} \times \frac{1}{2^2} \times \frac{1}{2^2} \times \frac{1}{1!} \times \frac{1}{1!} \times \frac{1}{1!} \times \frac{1}{1!} \times \frac{1}{1!} \right) = 2.17 \times 10^{-5}$$

- Total mismatch probability:

$$\mathrm{MMP_S} = (1 - (\mathrm{MP_{SD}} \times MP_{SL})) \approx 1 - 3.47 \times 10^{-10}$$

- BPEL process potential reusability:

$$R_p = 1 \times [1 - \mathrm{MMP_S}] = 3.47 \times 10^{-10}$$

Finally, potential reusability of Visit Preparation service calculated with the assumption that the current reusability of the service is the value of 1 (i.e. $R_{C=}1$). In accordance with the value of R_p, it seems the under discussion scenario has little potential reusability. This is why the workflow is too coarse-grain and provide large chunk of functionality. Therefore, its mismatch probability become significant and its reusability in potential contexts get reduced.

6. EVALUATION AND DISCUSSION

In order to empirically validate the proposed metric, further experiments using experimental model need to be carried out. In this regard, to gain confidence in our theoretical works, a controlled experiment has been conducted in which we take 8 BPEL processes from two projects including TJT Core-banking processes at international sector and also the business processes of healthcare organization that is EHR project. These processes were designed in more than one version. These versions are different from each other in terms of their designs. This means, each versions of the same BPEL process may have some sort of modularity, since the encapsulation of functionality further supports the reuse of services [20].

These composite services were evaluated by use of our metric and also examined qualitatively by experts for business process. The two evaluations then were compared to demonstrate that the model is a valid characterization of BPEL process reusability.

Table 4 contains 19 versions of BPEL processes and their corresponding reusability values. The maximum of these values indicates the most reusable variant, which is denoted by *Max(R_p)* column Regarding these versions are examined and analyzed by business process management experts, table 4 also embodies the selected versions, which seem to be more reusable comparing to the others.

According to the results, we obtained indications of a positive evaluation of our metric from the experts view points. As table 4 indicates, the consensus provided by the participants when choosing the best process design was in favor of our metric. However, as indicated in the table 4, there are 2 contradictions between the expert selection and the metric value. In our point of view, such a contradiction may arise from the fact that experts employ the most aspects of the quality when selecting the alternatives, nevertheless, the proposed metric only measure one aspect of them that is reusability. Nonetheless, to empirically validate the adopted technical metrics and the method itself, further experiments need to be carried out.

Another important issue that we have to emphasize is the proposed metric is expected to be used in analysis and design phase of service-oriented solution development lifecycle, before any implementation actually take place.

7. CONCLUSION AND FUTURE DIRECTIONS

Reuse has long been a holy grail of the IT industry. The initial idea focused on reusing code through class libraries or code templates. However, within an enterprise, code reuse often has less far-reaching impact than the reuse of runtime components. It is important to point out that it is a key benefit of an SOA that it contributes to runtime reuse. Remembering that a composite service is a choreographed group of services whose interface is published in some agreed-upon format.

In this article, we proposed a metric for composite service reusability measurement. Our approach is based on the analysis of logic and description mismatch. We then consolidated these measures to a metric formula for quantifying the probability of a service will be reused within potential service-oriented solutions. 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 reusability.

The last point that needs to be explored and answered is about the meaning of a given reusability metric, for instance, what is the significance of the R_p of 0.00104. In fact, determining the upper and lower bound of R_p provide a remarkable insight about the obtained results. This answer will be given from empirical results only when we get involved and collecting data from extensive practical projects.

Work will be continued to implement a toolset as a metric suite to automate computation, analysis, and optimization of the solution artifacts appropriateness based on the proposed metrics.

8. REFERENCES

[1] Hurwitz, J., Bloor, R., Baroudi, C.: Thinking from Reuse SOA for Renewable Business (2006) http://www.hurwitz.com/PDFs/IBMThinkingfromReuse.pdf

[2] Herr, M., Bath, U., Koschel, A.: Implementation of a Service Oriented Architecture at Deutsche Post MAIL. In: Zhang, L.-J., Jeckle, M. (eds.) ECOWS 2004. LNCS, vol. 3250, pp. 227–238. Springer, Heidelberg (2004)

[3] Artus, D., J., 2006, SOA realization: Service design principles. IBM Developer Works, http://www-128.ibm.com/developerworks/webservices/library/ws-soadesign/

[4] Feuerlicht, G., Lozina, J., 2007, Understanding Service Reusability, in the Proceedings of the 15th International Conference Systems Integration, June 10-12, Prague, Czech Republic, ISBN 978-80 245-1196 2, 144-150.

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

[6] Erl, T., Service-oriented architecture: concepts, technology, and design, Prentice Hall, 2005

[7] P. Jamshidi, M. Sharifi, S. Mansour: To Establish Enterprise Service Model from Enterprise Business Model. IEEE SCC (1) 2008: 93-100

[8] P.Jamshidi, S.Khoshnevis, R.Teimourzadegan, A.Nikravesh, A. Khoshkbarforoushha, F.Shams, "ASSM: Toward an Automated Method for Service Specification", IEEE Asia-Pacific Services Computing Conference (APSCC'09), Biopolis, Singapore

[9] A. Khoshkbarforoushha, P. Jamshidi, S. Khoshnevis, A. Nikravesh, F. Shams, A Metric for Measuring BPEL Process Context-Independency, IEEE International Conference on Service-Oriented Computing and Applications, 2009 (SOCA'09), Taipei, Taiwan. DOI: 10.1109/SOCA.2009.5410260

[10] Juric, M., B., Mathew, B., Sarang, P., 2006, Business Process Execution Language for Web Services, Packt Publishing, Birmingham, B27 6PA, UK, Second edition.

[11] Frakes, W., Terry, C., 1996, Software reuse: metrics and models. ACM Comput. Surv. 28, 2, DOI= http://doi.acm.org/10.1145/234528.234531, 415-435.

[12] Poulin, J., S., 1992, Measuring Software Reusability, 5th Annual Workshop on Software Reuse, WISR5. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.25.9571&rep=rep1&type=pdf

[13] Zimmermann, O., Krogdahl, P., Gee, C., 2004, Elements of Service-Oriented Analysis and Design, available at: http://www.ibm.com/developerworks/library/ws-soad1/

[14] Yu, J., Han, Y.B., Han, J., Jin, Y., Falcarin, P., Morisio, M., 2008, Synthesizing service composition models on the basis of temporal business rules. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 23(6): 885-894 Nov.

[15] Cardoso, J., 2005, About the Data-Flow Complexity of Web Processes, 6th International Workshop on Business Process Modeling, Development, and Support: Business Processes and Support Systems: Design for Flexibility. The 17th Conference on Advanced Information Systems Engineering (CAiSE'05), Porto, Portugal, 67-74.

[16] http://msdn.microsoft.com/en-us/library/ms256131.aspx

[17] Dhama, H., 1995, Quantitative Models of Cohesion and Coupling in Software, Journal of Systems and Software, vol. 29, no. 4, 65 74.

[18] Barnard, J., 1998, A new reusability metric for object-oriented software, Software Quality Journal 7, 35–50.

[19] http://www.ibm.com/developerworks/webservices/library/ws-soa-bddhealth/

[20] K. Kaschner, N. Lohmann, Automatic Test Case Generation for Interacting Services, Fourth International Workshop on Engineering Service-Oriented Applications (WESOA 2008), ICSOC 2008 International Workshops, Sydney, Australia, December 1st, 2008.