MISQ:
A Framework to Analyze and Optimize Web Service Composition in Business Service Networks

Seog-Chan Oh, Penn State University, USA

Dongwon Lee, Penn State University, USA

R. T. Kumara, Penn State University, USA

EXECUTIVE SUMMARY

A novel UML-based analytical modeling methodology named MISQ is presented for optimizing Web service composition in Business Service Networks. MISQ enables functional and temporal analyses at a high-level design stage so that Web service composition can be optimized systematically. Furthermore, MISQ provides an automatic generation of Web service implementations for improving productivity and reliability.

Keywords: simulation; UML; Web service composition

INTRODUCTION

In Business Service Networks (BSNs), by combining multiple, heterogeneous services, one can establish new value-added business processes for further applications. In particular, Web services has emerged as a popular means to describe the services that each vendor provides. Web services (W3C, 2002) is a piece of XML-based software interface that can be invoked over the Internet; it can be viewed roughly as a next-generation successor of CORBA or RPC technique. In such a setting, one of the key issues is how to generate, discover, compose, and optimize Web services that are of interest.

In this paper, we especially focus on the problem of optimizing Web service composition and propose a novel methodology—MISQ—as a solution. That is, we use UML to design agent-based business processes and two formal modeling schemes—Stochastic Process Algebra (SPA) and Generalized Stochastic Petri Nets (GSPN) (Ribaudo,

1995)—to analyze initial business processes design and to obtain optimized parameters. Finally, we propose to use the Business Process Execution Language for Web Service (BPEL4WS) (Andrews et al., 2003) as implementation artifacts for expressing the optimized business processes.

Example 1. Motivation

Consider a scenario in a BSN where the optimization of composed Web services is a crucial issue.

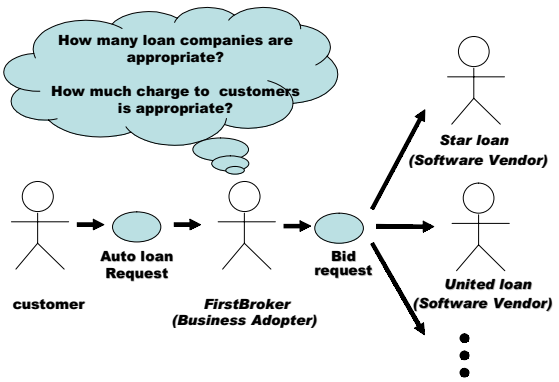
Suppose Bill opens an Internet-based auto loan brokerage company (FirstBroker), where he locates a loan with a low interest rate for customers who pay a nominal fee as a return. FirstBroker uses Web services from three loan companies: StarLoan, UnitedLoan, and BestLoan. Once FirstBroker gets a customer's inquiry, it sends bid requests to three loan companies, using their Web services, and forwards the lowest interest rate to the customer. Whenever FirtstBroker sends loan rate requests to the loan companies, FirstBroker has to pay a fee to each. That is, FirstBroker is a business adapter, and the three loan Web services are software vendors in the BSN.

Furthermore, a customer pays a fee to FirstBroker only if he or she is satisfied with the proposed rate and decides to make a contract with FirstBroker. In summary, Bill's profit model is the following:

Profit model = (# of accepted proposals by customers ´ charge per customer) – (# of loan rate requests ´ # of loan companies ´ charge per loan rate request).

Suppose Bill agrees to pay $1 for each loan rate request to loan companies, while charging $10 to customers who eventually accept the proposed rate. The business is initially booming, attracting a large number of customers due to the fact that customers do not have to pay for initial inquiries and pay $10 only afterwards. However, FirstBroker eventually files a bankruptcy, despite many customers submitting inquiries.

Figure 1. Use case of FirstBroker example



The scenario presented often occurs in combining and composing new services in BSNs, where a decision for parameters must be made to maximize profits. If Bill had chosen a correct number of Web services (i.e., loan companies) and a proper service charge to customers, possibly he still would have been in business.

Like the case of FirstBroker, early identification of optimal values through formal analyses is particularly desirable, since the costs of changing the design at a later stage are much higher (Marzolla, 2002). However, identifying optimal ones when multiple Web services are complicatedly interrelated is a challenging task, since in real applications, such parameters to consider can be many and non-trivial.

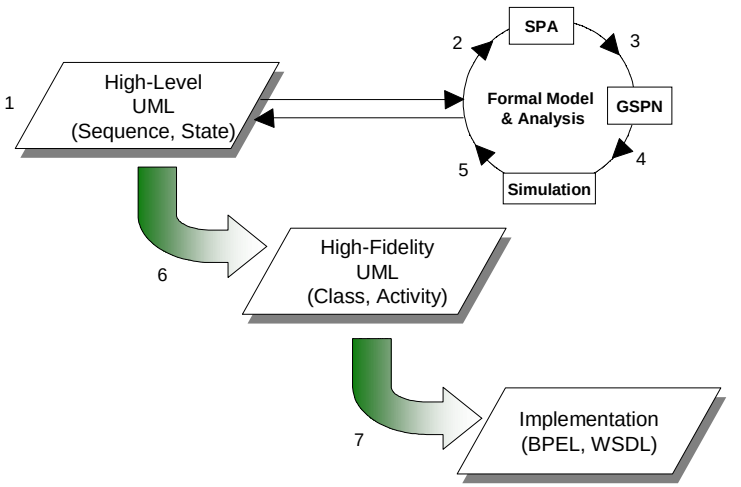
Therefore, there is an imminent need for the methodology that systematically and mechanically helps to model, analyze, and optimize Web service compositions. For this solution, we propose MISQ.

OVERVIEW OF MISQ

As illustrated in Figure 2, MISQ consists of analysis and implementation stages. Informally, the analysis stage runs as follows:

1. Design high-level UML diagrams, such as state and sequence diagrams.
2. Transform high-level UML designs into a formal model in Stochastic Process Algebra (SPA) model.
3. Transform SPA into Generalized Stochastic Petri-Net (GSPN) model using steps suggested in the previous research (Ribaudo, 1995).
4. Perform analysis via simulation.
5. Based on simulation results, identify optimal parameters and design. If needed, steps 2-4 may be repeated.

Figure 2. Overview of MISQ



The implementation stage is adopted from Amsden, et al. (2003) and is similar to the waterfall model of software development. It runs as follows:

1. Based on the optimized high-level design, produce high-fidelity, such as class and activity diagrams.
2. From the high-fidelity model, generate implementation artifacts.

MISQ contributes the following:

- A Petri-Net model for analyzing initial high-level, UML-based designs; the temporal and functional analysis for optimization can increase productivity and reliability of Web service-based software systems in BSNs.
- A methodology for seamless integration of several languages or modeling tools (e.g., UML, SPA, GSPN, WSDL, and BPEL) and a detailed example with a simulation result to illustrate the effectiveness of the proposed methodology.

RELATED WORK

Our research integrates three different streams of work: deriving analysis model from UML, deriving implementation artifacts from UML, and transforming models from SPA to GSPN.

To remedy lacks of verification and validation inherent in UML, some researches (Korenblat & Priami, 2003; Ribaudo, 1995) tried to translate UML into process algebra. The focus (Korenblat & Priami, 2003) is on a sequence diagram, where objects of the sequence diagram are considered as p-calculus processes, and messages are represented as actions among these processes.

Despite the inherent semi-formality, UML has a strong descriptive power for high-level modeling as well as high-fidelity modeling (Amsden et al., 2003). Among UML diagrams, state diagrams and sequence diagrams are sufficient to represent the high-level model. On the other hand, a high-fidelity model is capable of representing the details of implementation artifacts. Usually, a high-fidelity model can be expressed with class and activity diagrams of UML. The mapping from the high-fidelity model to corresponding implementation artifacts is provided using UML 1.4 profile and BPEL4WS (Andrews et al., 2003) as implementation artefacts (Amsden et al., 2003).

Comparisons between GSPN and SPA with different perspectives are given (Donatelli et al., 1995). In our proposal, we use both GSPN and SPA as an analysis model to optimize Web service composition.

As dynamic discovery of Web services and composition problem, run-time adaptability of a composed process is another research issue in this area. METEOR-S (Verma et al., 2004) project has addressed this issue for workflows. This allows the process designers to bind Web services to an abstract process, based on business and process constraints, and generate an executable process. Proteus (Ghandeharizadeh et al., 2003) was suggested as a framework that consumes a user request to compose a plan that incorporates available Web services and to execute a seamless plan. Finally, in both

METEOR-S and Proteus, the user defines a composition at design time, and in the latter, service adopting is processed at execution time by assembling available Web services. These compositions are called pattern-based; other researches also addressed this approach (Benatallah et al., 2002; Narayanan & Mcllraith, 2002).

Besides the dynamic composition approach, automatic composition of Web services is a challenging research problem. This is due to the difficulty of mapping user needs to a collection of correlated services, where their interim outputs can satisfy each other's input requirements and where the final deliverable meets the user demands. Finally, matching interfaces among Web services in the collection is the problem. If only syntactic matching is allowed, the problem can be formulated into a specially directed graph shortest path problem (Oh et al., 2005). On the other hand, semantic interface matching is expected to be crucial to automatically compose new services due to increasing numbers and the heterogeneity of available Web services. Interface-Matching Automatic Composition technique (Zhang, 2003) incorporates the use of WS ontology to find matching web services.

There is the emerging consensus that the ultimate challenge is to make Web services automatically tradable and usable by artificial agents in their rational, proactive interoperation on the next generation of the Web (Ermolayev et al., 2004). It may be solved by creating effective frameworks, standards, and software for automatic Web service discovery, execution, composition, interoperation, and monitoring (McIlraith et al., 2001). In the industries, only initial and partial solutions of the ultimate problem are provided. Existing de facto standards for Web service description (WSDL) (W3C, 2004), publication, registration, discovery (UDDI) (OASIS & UDDI, 2003), binding, invocation, and communication (SOAP) (W3C, 2003) provide merely syntactical capabilities and do not completely solve the ultimate challenge. More recent research and standardization activities of the DARPA DAML community resulted in offering semantic service markup language DAML-S (Ankolekar et al., 2002), based on the RDF platform.

MISQ METHODOLOGY

MISQ is based on various models (i.e., UML, SPA, GSPN, BPEL, WSDL) and transformation procedures between models. In the interest of space, here we only present a brief overview of SPA and GSPN.

Definition 1. SPA

Stochastic Process Algebra (SPA) is described by the following grammar (Donatelli et al., 1995):

$$P ::= Stop \mid (a, \lambda).P \mid a.P \mid P+P \mid P||_S P \mid P<\ddot{y}S \mid Q$$

where a variable P, Q, … denotes process variables, while S is a set of synchronization actions. The intuitive meaning of these elements is:

- Stop denotes the halting process.
- The process (a, λ).P models a delayed process that performs the action a with delayed rate λ and then behaves as process P.
- The process a.P models an immediate process that performs the action a without any delay and then behaves as process P.
- The choice operator '+' is used to model alternative behavior.
- The parallel operator '$\|_s$' models the parallel execution of two processes that have to synchronize in actions within the set of synchronizing actions S.
- The hiding operator '$\cdot$ ' is used for declaring actions as internal and often used to abstract away from internal events.

Definition 2. GSPN

Generalized Stochastic Petri-Net (GSPN) (Donatelli et al., 1995) is defined as a 5-tuple (PL, T, W, M_0, L), where:

- PL is a finite set of places.
- T is a finite set of transitions partitioned into two subsets: T_I (immediate) and T_D (delayed) transitions, where transitions $t \in T_D$ are associated with delayed rate λ.
- $W \in (PL \times T) \cap (T \times PL)$ is a set of directed arcs (i.e., flow relation).
- M_0: PL $\rightarrow \{0, 1, 2, \ldots\}$ is the initial marking.
- L: $T \rightarrow \Lambda$ is a labeling function, where › is a set of operation names.

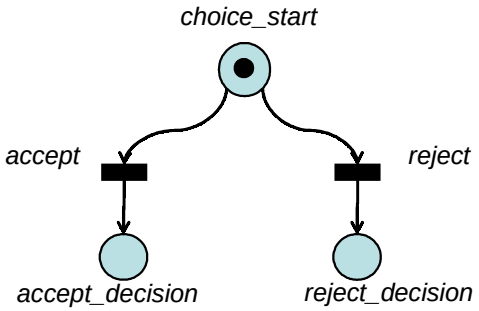
Example 2. SPN and GSPN

Consider the scenario of Example 1 again. A customer checks the proposal of FirstBroker and either accepts or rejects it. Since the customer chooses one behavior between two choices, we represent this process with *choice* operator of SPA, '+', as follows:

choice_decision := (accept + reject).

Similarly, we can map *choice_decision* into GSPN model, as shown in Figure 3. Here, the place *choice_start* with a token enables both *accept* and *reject* transition. If

Figure 3. The process choice_decision



accept transition is fired, the token switches places from *choice_start* to *accept_decision*. On the contrary, if *reject* transition is fired, the token switches places from *choice_start* to *reject_decision*.

Definition 3. MISQ Model

An MISQ Model is an 8-tuple (*DSequence*, *Agent*, *Protocol*, *DState*, *DClass*, *DActivity*, *SPA*, *GSPN*), where:

- *DSequence* is a sequence diagram with objects, behaviors, and messages among objects.
- *Agent* is a set of objects of *DSequence*. We denote each element of *Agent* as $a(i)$, with i being the position of the element (i.e., if $|Agent| = n$, $a(1)$ and $a(n)$ are the left-most and right-most objects in *DSequence*).
- *Protocol* is a set of protocols. An individual protocol is a set of messages between $a(i)$ and $a(j) \in A$, where $i < j$ and denoted as $prot(i, j)$.
- *DState* is a set of state diagrams. We denote each element of *DState* as $ds(i)$, which is the state diagram of $a(i) \in A$.
- *DClass* is a set of stereotyped class diagrams, such as *DClass-dependency*, *DClass-datatype*, *DClass-interface*, *DClass-protocol*, *DClass-process*. *DClass-dependency* defines the dependency relationship between each element in *Agent*. *DC-datatype* defines message contents and data classes as well as the relationship between message contents and data classes. *DC-interface* defines operations. *DC-protocol* defines roles of corresponding port type. *DC-process* defines internal variables and its ports connected to each element in *Agent*.
- *DActivity* defines activity diagrams for elements in *Agent*.

Next, we present several transformation procedures from one model to another in MISQ.

1. $a(i) \in A\ (i > 1)$ has communication with the left and right objects; that is, $prot(i\text{-}1, i) \neq \varnothing$ and $prot(i, i\text{+}1) \neq \varnothing$. For example, $a(1)$ has $prot(1, 2) \neq \varnothing$, and $a(n)$ has $prot(n\text{-}1, n) \neq \varnothing$.
2. For $prot(i, j)$, $|i - j| \leq 1$. That is, each object communicates only with its immediate neighbors.
3. $|A| \geq 2$. That is, there are at least two objects.

Now, we present three transformation procedures: (1) UML to SPA, (2) SPA to GSPN, and (3) UML to Implementation.

Procedure 1:

In Procedure, the given UML is recaptured into the SPA model. It has two steps. (Please see Procedure 1 on following page.)

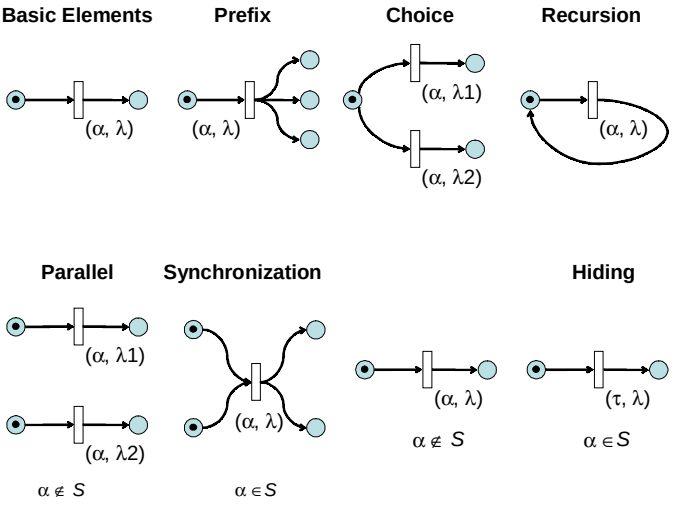
Procedure 1.

1. Building Atomic Processes
   1.1 Prepare *DSequence*, *Agent*, *Protocol*, *DState*.
   1.2 Create $APset = \{x \mid x \in SPA\} = \varnothing$.
   1.3 Set $i = 1$ and choose an $a(i) \in A$.
   1.4 Create an atomic process, $p(i) \in SPA$.
   1.5 Start transforming $ds(i)$ into $p(i)$. Transitions of $ds(i)$ are transformed to either *delayed* or *immediate* actions. If a transition does not have any temporal information, it becomes immediate action. '*a*'. Otherwise, λ is added and becomes the *delayed* action (a, λ).
   1.6 If any action branch exists, it is expressed by a non-deterministic choice: '+'.
   1.7 A sequence of transitions in $ds(i)$ corresponds to the sequence of actions in $p(i)$.
   1.8 $APset = \{p(i)\} \cap APset$.
   1.9 If $|APset| = |Agent|$; that is, all $ds(i) \in DState$ is transformed, then the procedure stops. Otherwise, increase i by 1 and go to step 1.3.
2. Building a Composite Process
   2.1 Create a process, $System \in SPA$ and $System := p(1)$. Increase i to 2.
   2.2 Choose $p(i) \in APset$.
   2.3 $System = System \|_S P(i)$; S, where S ? $prot(i\text{-}1, i)$.
   2.4 If $i = |APset|$; that is, all the $p(i)$ get combined into *System*, then stop. Otherwise, increase i by 1 and go to step 2.2.

Procedure 2:

In this procedure, the SPA model is transformed into a Petri-Net-based GSPN graphical model for easier manipulation. As shown in Figure 4, it is generally known that any SPA model can be represented as a GSPN model, and details of such translations can be found in Korenblat and Priami (2003) and Donatelli, et al. (1995). In our proposal,

Figure 4. Mapping SPA operations into GSPN



the approach introduced in the previous research (Ribaudo, 1995) is used. Due to lack of space, the entire procedure cannot be described.

Procedure 3:

Once the high-level UML design has been optimized in the GSPN model, finally, Web service implementation can be generated in this procedure. We use the methods (Amsden et al., 2003), but can use other implementation-specific methods for this procedure (e.g., from UML to CORBA).

1. Based on optimized system specifications obtained in Procedure 2, *DClass-dependency*, *DClass-datatype*, *DClass-interface*, *DClass-protocol*, *DClass-process*, and *DActivity* are drawn.
2. *DClass-dependency* maps to an XML namespace import in WSDL. *DClass-datatype* maps to message types and data types in WSDL. *DClass-interface* maps to operations types in WSDL. *DClass-protocol* maps to port and service link types in WSDL. *DClass-process* and *DActivity* map to process definitions in BPEL.

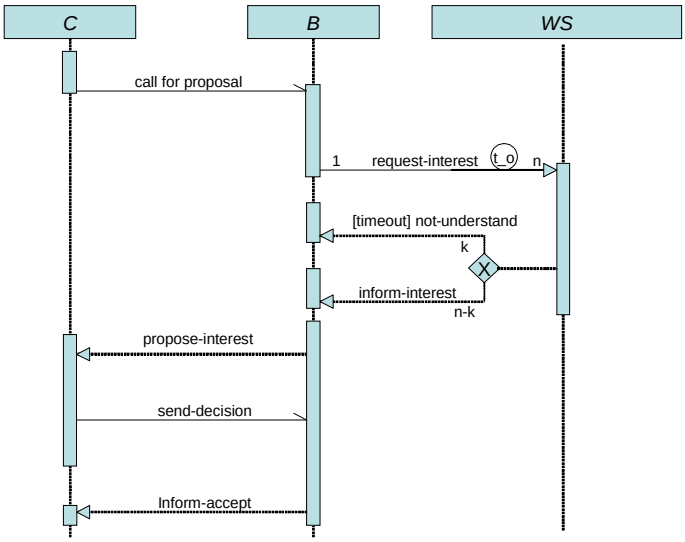
ILLUSTRATIVE EXAMPLE

In this section, let us demonstrate how to optimize Web service composition using the MISQ methodology. Table 1 summarizes notations used in this example.

Table 1. Notations for the example

Notation	Meaning
C	Customer, C's interarrival time follows exp(λ)
B	Brokerage Web service
WS	A set of auto loan Web services, $ws_j \in WS$, $1 \le j \le n$. We assume that $1 \le n \le 4$
Rate(*ws*)	A loan rate returned from $ws \in WS$, uniform(5, 6) is followed
t_o	Time-out until *B* waits for *Rate*(*ws*)
WS(*S*)	A set of Web services, $WS(S) \subset WS$, that successfully send a loan rate before *t_o*
WS(*F*)	A set of Web services, $WS(F) \subset WS$, that fail to send a loan rate before *t_o*
Min(*Rate*)	Smallest $Rate(ws_i)$, $\forall\ ws_i \in WS(S)$
$Fee(ws_i)$	Service fee that *B* pays to $ws_i \in WS(S)$
Fee(*B*)	Service fee that *C* pays to *B*
AR	*Accept rate*, $AR = exp\{-\sigma \times (Min(Rate) - 5)\} - 2^{(Fee(B)-10)}/2^{10}$, where σ is a preference parameter
PT	Profitable throughput $PT = \lvert C \rvert \times AR$

Figure 5. Sequence diagram of the example



PT is exponentially decreased as *Min*(*Rate*) increases (i.e., customers will not accept the offer if the rate is high) and is also decreased in proportion of $2^{(Fee(B)-10)}/2^{10}$ as *Fee*(*B*) increases (i.e., customer will not accept the offer if the service charge to *B* is high). *Accept rate* expresses *C*'s purchasing intention, whose parameters could be selected based on real-market surveys. Here, however, we simply use parameters *exp* and 2 in the interest of time.

Scenario

Consider the following scenario:

1. *C* seeks an auto loan with minimum interest rate, sends an inquiry to *B* (*C* has no direct access to *WS*).
2. *B* relays *C*'s request to each $ws_j \in WS$.
3. *ws* calculates and returns its $Rate(ws_j)$ to *B*.
4. The communication between *B* and ws_j is asynchronous with the timeout, *t_o*. After *t_o*, *B* does not wait for $Rate(ws_j\ s)$ anymore. B must pay $Fee(ws_j)$ to successful ws_j who returns $Rate(ws_j)$ within *t_o*.
5. *B* sends *Min*(*Rate*) to *C*.
6. If *C* accepts *Min*(*Rate*), *C* pays *Fee*(*B*) to *B*. Otherwise, *B* cannot charge *Fee*(*B*) on *C*.

Figure 5 illustrates the sequence diagram of the scenario.

Applying MISQ to the Example

We want to maximize the expected profit of *B*, who is a business adopter in the context of BSNs. Thus, the objective function, *Z*, representing the expected profit of *B*, can be $Z = Fee(B) \times (PT) - Fee(ws) \times |WS(S)| \times T$. If $Z \geq 0$, *B* makes a profit. *Z* is directly proportional to *PT*. If |*WS*| increases, *PT* is likely to increase, because *C* has a better chance to obtain lower *Min*(*Rate*), but *B* has to pay more fees to increased |*WS*(*S*)|. Meanwhile, if *Fee*(*B*) decreases, *PT* may increase, since a low service charge can attract more *C* to accept the offer, but *B*'s profit decreases.

Note that there are two trade-off relations for which we need to find the optimal values as follows:

Figure 6. State diagrams of the example

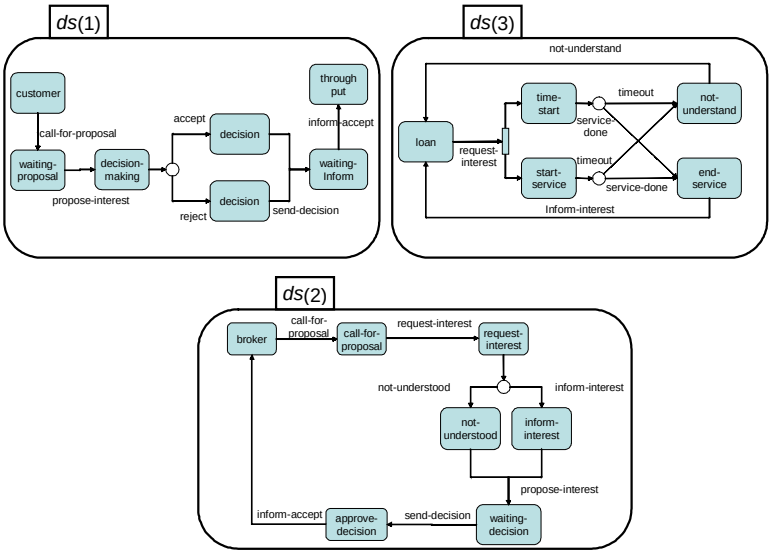
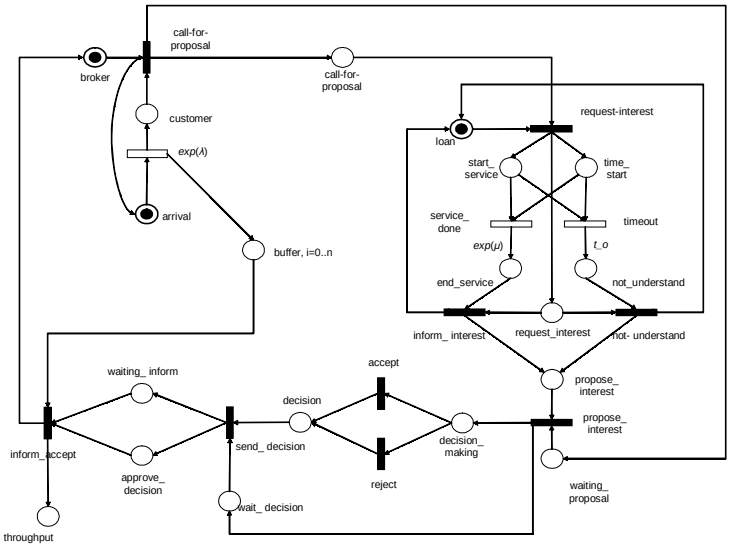


Figure 7. GSPN of system



- $|WS| = n$: How many Web services of loan companies are economical for *B* to use?
- *Fee*(*B*): How much service charge to customers is appropriate?

Since we assumed $1 \le n \le 4$, we apply MISQ analysis starting with $n = 1$ and can repeat the analyses by increasing n by 1. If $n = 1$, *Agent* = {*a*(1), *a*(2), *a*(3)}, where *a*(1) is *C*, *a*(2) is *B*, and *a*(3) is each ws_j of *WS*. Similarly, *DSequence* = {*ds*(1), *ds*(2), *ds*(3)}, where *ds*(1), *ds*(2), and *ds*(3), as shown in Figure 6. *Protocol* = {*prot*(1,2), *prot*(2,3)}, where *prot*(1,2) ={call for proposal, propose-interest, send-decision, inform-accept} and *prot*(2,3) = {request-interest, not-understand, inform-interest}.

Building Atomic and Composition Processes.

We first can build the following atomic processes:

- *customer*:= call-for-proposal; propose-interest; (accept + reject); send-decision; inform-accept; *throughput*.
- *broker*:= call-for-proposal; request-interest; (not-understand + inform-interest); propose-interest; send-decision; inform-accept; *broker*.
- *loan*:= request-interest; ((timeout, *t_o*);not-understood + (service-done, ¼);inform-interest); *loan*.

In addition to the original atomic processes, we can add two more processes—arrival and buffer—for collecting analysis data as follows:

- *arrival* := (gen, »); call-for-proposal; *arrival*
- *buffer*(*i*):= (gen, »); *buffer*(*i*+1) + inform-accept; *buffer*(*i*-1), where $i \geq 1$

Next, based on the aforementioned atomic processes, we build the composite process of *System* as follows:

- *System'* := *customer* $||_{prot(1,2)}$ *broker*$||_{prot(2,3)}$ *loan*• (*prot*(1,2) ∩ *prot*(2,3))
- *System* := (*System'* $||S_1$ *arrival*) $||S_2$ *buffer*• ($S_1 \cap S_2$) where S_1 = {call-for-proposal}, S_2 = {gen , propose-interest}

Transforming SPA into GSPN

Through the SPA to GSPN procedure, the composite process System in SPA is transformed into GSPN, as shown in Figure 7.

Simulation of GSPN

We conducted simulations for four experimental cases: |*WS*| = 1, 2, 3, and 4. We assumed that $1/\lambda = 1/\mu = t\_o = 4$ hours, $\sigma = 5$, and Fee(wsj) = $1. GSPN model simulation was done using HPSim (Anschuetz, 1999), and the result analysis was conducted with MS Visual Basic and Excel. Simulation time was set to the same as *B*'s life cycle: 10,000 hours.

As shown in Figure 8, The optimal setting for the scenario occurs when |*WS*| = 4, *Fee*(*B*) = $16, with the expected profit of *B* being $3,373.

High-Fidelity UML and Implementation

Once we acquire optimal parameters for the auto loan example, we can build *DClass-dependency*, as shown in Figure 9. Similarly, we also can generate *DClass-datatype*, *DClass-interface*, and *DClass-protocol*, as shown in Figure 10. Those models map into a WSDL file. Furthermore, we also can build *DClass-process* in Figure 11

Figure 8. Profit according to |WS| and Fee(B)

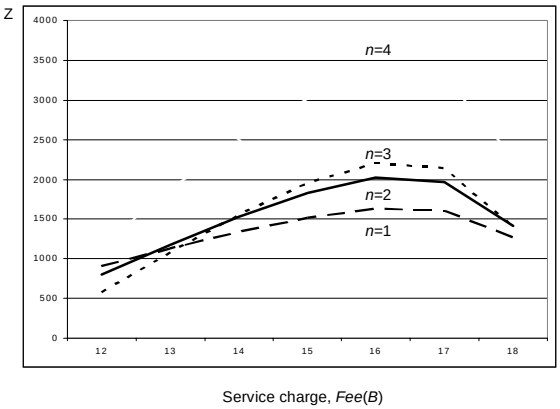


Figure 9. Dependency diagram of the example

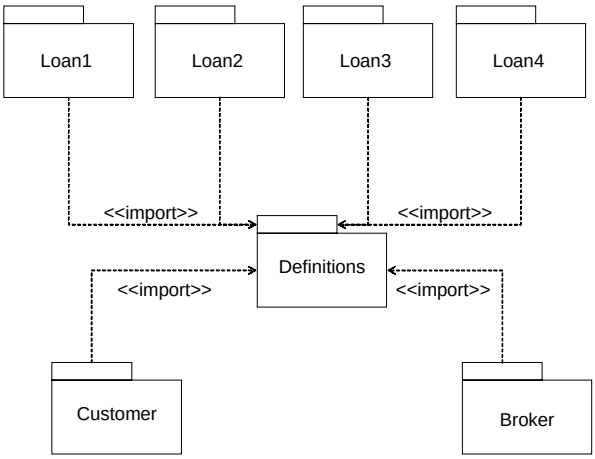
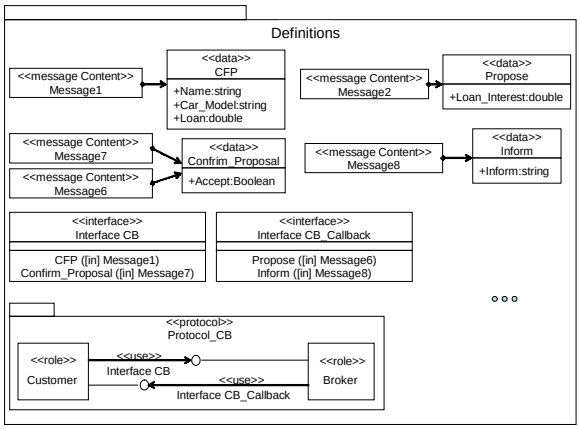


Figure 10. Definitions package of the example



and *DActvity* in Figures 12, 13, and 14. Those models map into a BPEL file. Some part of implementation codes of WSDL and BPEL are illustrated in Figures 15 and 16, respectively.

Figure 16 illustrates the BPEL of the example, which imports the WSDL and orchestrates Web services, including customer and four loan Web services. The main body of the BPEL is <process>, which can be divided into two parts, such as the process type definition and the process activity definition.

As shown in Figures 15 and 16, WSDL has the important role in BPEL, since the BPEL process is established, based on the service model, which is defined by WSDL. In

Figure 11. Broker package of the example

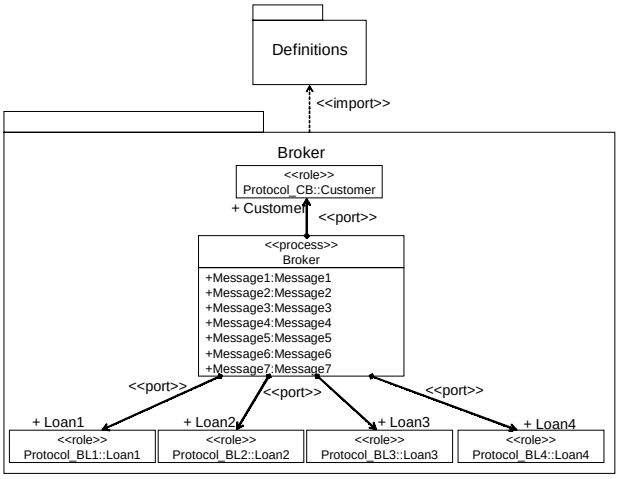


Figure 12. Activity diagrams of the example(1)

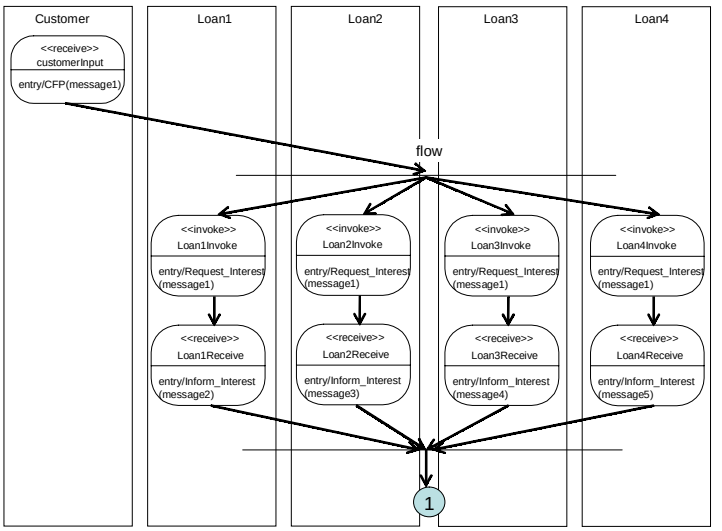


Figure 13. Activity diagrams of the example(2)

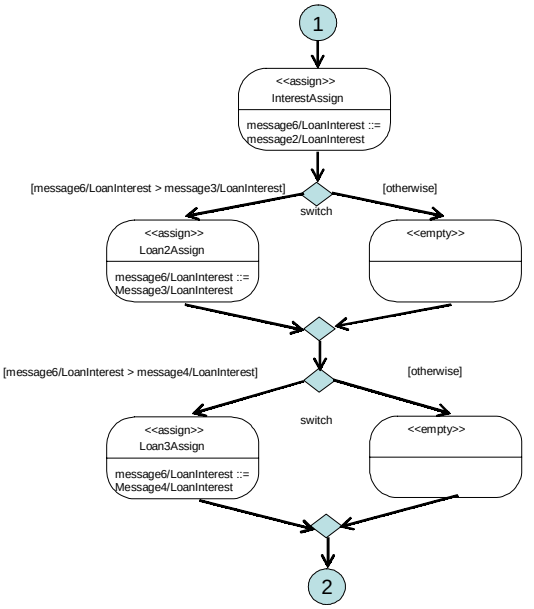


Figure 14. Activity diagrams of the example(3)

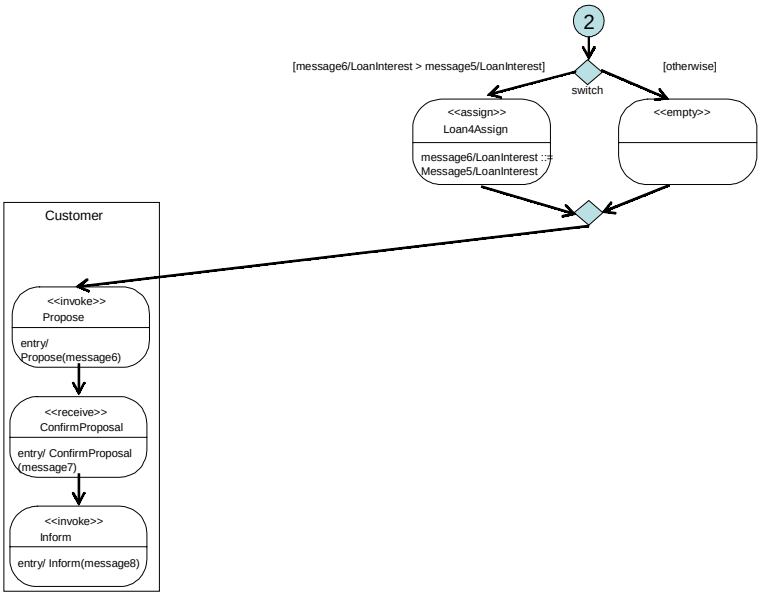


Figure 15. WSDL of the example

```
<?xml version="1.0"?>
<definitions name="Broker" ... >
<types>
 <element name = "CFP">
        <seqeunce>
                <element name="Name" type="string">
                <element name="Car_Model" type="string">
                <element name="Loan" type="int">
        </sequence>
 <element>
...
</types>
<message name="Message1">
        <part name="parameters" element="CFP"/>
</message>
...
<portType name="Interface CB">
        <operation name="CFP">
                <input message="Message1"/>
        </operation>
        <operation name="Confirm_Proposal">
                <input message="Message7"/>
        </operation>
</portType>
...
<serviceLinkType name="Protocol_CB">
        <role name="Customer">
                <portType name="Interface CB_Callback"/>
        </role>
        <role name="Loan">
                <portType name="Interface CB"/>
        </role>
</serviceLinkType>
...
</definitions>
```

the WSDL, the two key concepts—process and partner—are modeled as WSDL services. A BPEL process reuses the definition of WSDL, which can be deployed in different ways and in different scenarios. For instance, as shown in Figure 11, the loan broker as well as four loan companies can reuse the same WSDL file, but they use it in different scenarios. The BPEL process model has a limitation that it can conduct peer-to-peer interaction between services described in WSDL. For this peer-to-peer interaction, as shown in Figure 15, the WSDL model defines messages and portTypes. Within its portTypes, the interactions among Web services are defined as operations where the corresponding messages are used as arguments (Shen, Yang & Lawani, 2004).

Like a flow chart, BPEL provides two types of primitives: unit and control. For the unit primitives, as shown in Figure 16, BPEL uses <invoke>, <receive>, <reply>, and <wait>, and combines them to make more complex process units. For the control primitives, it uses structural activities, such as <flow>, <sequence>, <switch>, <pick>, and so forth.

Figure 16. BPEL of the broker

```
<process name ="Broker" ...>
<partners name="Customer" serviceLinkType="Protocol_CB"
        partnerRole="Protocol_CB:Customer"
        myRole="Protocoal_CB:Broker"/>
</partners>
...
<receive name="customerInput" partnerLink="Customer"
portType="Interface CB"    operation="CFP" variable="Message1" ... />
<flow>
<sequence>
        <invoke name="Loan1Invoke" partnerLink="Loan1"portType="Interface BL1"
      operation="CFP" variable="Message1" ... />
        <recevie name="Loan1Invoke" partnerLink="Loan1"    portType="InterfaceBL1 Callback"
operation="Propose"
        variable="Message3" ... />
</sequence>
...
</flow>
<assign name="InterestAssign" >
        <copy>
        <from variable="message2" portion="LoanInterest" />
      <to variable="message6" portion="LoanInterest" >
        <copy/>
</assign>
<switch>
        <case condition="message6/LoanInterest >
        message3/LoanInterest">
                <assign name="Loan2Assign" >
                        <copy>
                        <from variable="message6" portion="LoanInterest" />
                        <to variable="message3" portion="LoanInterest" >
                        <copy/>
                </assign>
        <otherwise>
                <empty />
</switch>
...
<invoke name="Propose" partnerLink="Customer" portType="Interface CB callback " operation="Propose "
variable="Message6" ... />
<receive name="ConfirmProposal" partnerLink="Customer" portType="Interface CB " operation="ConfirmProposal
" variable="Message7"
... />
<invoke name="Inform" partnerLink="Customer" portType="Interface CB callback" operation="Inform "
variable="Message8" ... />
</process>
```

In BPEL, there are partnerLinkTypes, which characterize the services with which the corresponding business process communicates. BPEL allows for maintaining data for later use during the interaction phase in the business process. A process definition is made of an activity, a series of partners and containers with specific correlation sets, the definition of fault handlers, and compensation handlers (Verm, 2004). Interaction implemented in BPEL is specified, based on message exchange between Web services. It is necessary if these messages are predefined in WSDL, where operations also must be defined if they will use those messages (Shen, Yang & Lawani, 2004).

CONCLUSION

The MISQ systematically optimizes Web service composition to identify the optimal values, such as the number of ideal Web services, maximum throughput, and so forth. There are several future research directions. In addition to simple value optimization, more functional analysis (e.g., deadlock detection or security flaw detection) can be greatly benefited by MISQ. Also, considering real-time IT provisioning and adoption enabled by BSNs, more dynamic optimization is a challenging goal. For instance, optimizing the dynamic workflow (Kim et al., 2003) of Web service components can greatly benefit both software vendors and business adopters. Toward this scenario, discovering, dynamically composing, and optimizing large-scale (e.g., in the range of 1,000 to 10,000) Web services is a challenging problem. We approached the problem by viewing Web services compositions as a graph search problem (Oh et al., 2005). What has been presented in this chapter is thus complementary to the graph search based Web services composition research.

In the near future, we plan to combine the ideas (Oh et al., 2005) and that of MISQ to accomplish truly dynamic Web service composition methodology.

REFERENCES

Amsden, J., Gardner, T., Griffin, C., & Iyengar, S. (2003). *Draft UML 1.4 profile for automated business processes with a mapping to BPEL 1.0.* IBM.

Andrews, T., et al. (2003). *Business process execution language for Web services (BPEL) 1.1.* OASIS.

Ankolekar, A., et al. (2002). DAML-S: Web service description for the Semantic Web. *Proceedings of the International Semantic Web Conference*, Sardinia, Italy.

Anschuetz, H. (1999). HPSim copyright © 1999-2001. Retrieved from http://www.winpesim.de/petrinet/e/hpsim_e.htm

Benatallah, B., Dumas, M, Sheng, Q. Z., & Ngu, A. H. (2002). Declarative composition and peer-to-peer provisioning of dynamic Web services. *IEEE International Conference on Data Engineering*, San Jose, CA.

Donatelli, S., Hermanns, H., Hillston, J., & Ribaudo, M. (1995). *GSPN and SPA compared in practice: Quantitative modelling in paralle systems*. Springer.

Ermolayev, V., Keberle, N., Plaksin, S., Konoenko, O., & Terziyan, V. (2004). Towards a framework for agent-enabled semantic Web service composition. *International Journal of Web Service Research, 1*(3).

Ghandeharizadeh, et al. (2003). Proteus: A system for dynamically composing and intelligently executing Web services. *Proceedings of the ICWS*, Las Vegas, NV.

Kim, J., et al. (2003). Web services and BPEL4WS for dynamic eBusiness negotiation processes. *Proceedings of the International Conference on Web services.*

Korenblat, K., & Priami, C. (2003). *Extraction of p-calculus specifications from UML sequence and state diagrams* [technical report DIT-03-007]. University of Trento.

Marzolla, M. (2002). *Simulation-based performance evaluation of software archiecture described in UML*. Universita Ca Foscari di Venezia.

McIlraith, S. A., Son, T. C., & Zeng, H. (2001). Semantic Web services. *IEEE Intelligent Systems Magazine. Special Issue on the Semantic Web*, *16*(2).

Narayanan, S., & McIlraith, S. A. (2002). Simulation, verification and automation composition of Web services. *Proceedings of the 11th International World Wide Web Conference (WWW2002)*, Honolulu, Hawaii.

Oh, S., On, B., Larson, E. J., & Lee, D. (2005). BF*: Web services discovery and composition as graph search problem. *Proceedings of the IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE)*, Hong Kong, China.

Ribaudo, M. (1995). Stochastic petri net semantics for stochastic process algebra. *Proceedings of the 6th International Workshop on Petri Nets and Performance Models*.

Shen, J., Yang, Y., & Lalwani, B. (2004). Mapping Web services specifications to process ontology: Opportunities and limitations. *Proceedings of the 10th IEEE International Workshop on Future Trends in Distribution Computing Systems (FTDCS04)*.

Simple object access protocol (SOAP) 1.2. (n.d.). Retrieved March 10, 2005, from http://www.w3.org/TR/2003/REC-soap12-part1-20030624/

UDDI 3.0 technical white paper. (n.d.). Retrieved March 10, 2005, from http://uddi.org/

Verma, K., Sivashanmugam, K., Sheth, A., Oundhakar, S., & Miller, J. (2004). METEOS WSDI: A scalable infrastructure of registries for semantic publication and discovery of Web services. *Journal of Information Technology and Management*.

Virdell, M. (2004). Business processes and workflow in Web services world. *IBD DeveloperWorks*. Retrieved March 10, 2005, from http://www-128.ibm.com/developerworks/webservices/library/ws-work.html

W3C Web services activity. (n.d.). Retrieved March 10, 2005, from http://www.w3c.org/2002/ws/

Web services description language (WSDL) 2.0. (n.d.). Retrieved March 10, 2005, from http://www.w3.org/TR/2004/WD-wsdl20-20040803/

Zhang, R., Arpinar, B., & Aleman-Meza, B. (2003). Automatic composition of semantic Web service. *Proceedings of the International Conference on Web Services (ICWS03)*, Las Vegas, NV.

Seog-Chan Oh has been a PhD student in industrial and manufacturing engineering since 2002. He earned a BS and an MS from Dongguk University (1993 and 1996, respectively). Before he began his PhD studies, he worked as an IT consultant for seven years at Daewoo Information Systems. His research interests include Web service composition and optimization, AI, MAS and Web intelligence. He has a Certified Professional Engineer of Information Management from the Korean government.

Dongwon Lee has been an assistant professor at the Pennsylvania State University, School of IST, since 2002. He earned a BS from Korea University (1993), an MS from Columbia University (1995), and a PhD from UCLA (2002), all in computer science. Before he began his PhD

studies, he worked at AT&T Bell Labs (1995-1997). His research interests include database and data modeling, XML and Web analysis, and bibliometrics. He has (co-)authored about 30 scholarly articles in conferences or journals.

Soundar R. T. Kumara is distinguished professor of industrial engineering with joint appointments in the Department of Computer Science and Engineering and the School of Information Systems and Technology. His research interests are in intelligent systems research with an emphasis on sensor-based equipment monitoring and diagnosis, software agents, and complexity in supply chains. He has more than 150 publications. He has won several awards and is an elected member of the International Institution of Production Research (CIRP).