RC23822 (W0512-009) December 1, 2005
Computer Science

IBM Research Report

Proceedings of the International Workshop on Dynamic Web Processes (DWP 2005)

Amsterdam, The Netherlands, December 2005

[1]Kunal Verma, [1]Amit Sheth, [2]Michal Zaremba, [2]Christoph Bussler (Eds.)

[1]LSDIS Lab, USA
[2]DERI, Ireland

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Preface

These proceedings contain the papers accepted for presentation at the 1st International Conference on Dynamic Web Processes, DWP 2005, held at Amsterdam, the Netherlands, December 12, 2005 in conjunction with the Third International Conference on Service-Oriented Computing (ICSOC 2005).

This workshop intends to bring together researchers from various disciplines as well as industrial practitioners to explore the role of semantics in creating dynamic Web processes. As businesses transition to service oriented architectures, there is an emerging interest in businesses to begin exploring more dynamism in their business processes, to better react to the changing environments and to make their process more adaptive/agile.

The emphasis of this workshop in mainly on the following three topics - 1) formal modeling of different aspects of dynamism in Web processes, 2) execution environments for supporting dynamic and agile Web processes and 3) real world use cases that show the need for creating such models and environments.

The 8 research papers were carefully reviewed and selected from 13 submissions. This workshop also includes a panel entitled - "Semantic Web Services: Past, Present, and Future". John Domingue (Open University, UK), Amit Sheth (LSDIS Lab, University of Georgia, USA), Sheila McIlraith (University of Toronto, Canada) and Massimo Paolucci (NTT DoCoMo Euro Labs, Germany) will participate. The moderator of the panel is Michael Maxmilian from IBM Research, Almaden.

We would like to express our sincere gratitude to all authors for their high quality submissions. We would also like to thank all the members of the program committee for reviewing the papers. Special thanks also go to ICSOC workshop chairs - Frank Leymann and Winfried Lamersdorf and ICSOC general chairs - Francisco Curbera and Mike Papazoglou for their kind help and support in organizing the workshop.

December 2005

Kunal Verma, Amit Sheth
Michal Zaremba, Christoph Bussler
Organizers
DWP 2005

Organization

DWP 2005 is jointly organized by LSDIS Lab, Univeristy of Georgia, USA and DERI, Ireland in cooperation with the ICSOC workshop committee.

Organizers

General Chairs:	Amit Sheth (LSDIS Lab, USA)
	Christoph Bussler (Deri)
Program Chairs:	Kunal Verma (LSDIS Lab, USA)
	Michal Zaremba (Deri)

Program Committee

Nabil Adam, Rutgers University, USA
Rama Akkiraju, IBM Research, USA
Michael Altenhofen, SAP, Germany
Michael Brodie, Verizon, USA
Mark Burstein, BBN Technologies, USA
Jorge Cardoso, University of Madeira, Portugal
Pat Croke, HP, Ireland
Dipanjan Chakraborty, IBM Research, India
Umeshwar Dayal, HP, USA
Asuman Dogac, Middle East Technical University , Turkey
Prashant Doshi, LSDIS Lab, USA
Marlon Dumas, Queensland University of Technology, Australia
Wolfgang Gerteis, SAP(UK) Limited, United Kingdom
Michael Huhns, University of South Carolina, USA
Bill Karakostas, City University, London, UK
Adrian Mocan, DERI, Ireland
Marco Pistore, University of Twente, The Netherlands
Manfred Reichert, University of Twente, The Netherlands
Jianwen Su, University of California at Santa Barbara, USA
Vlamidir Tosic, Lakehead University, Canada
H. M. W. Verbeek, Eindhoven University of Technology, The Netherlands
Mathias Weske, Hasso-Plattner-Institute, Germany
Andreas Wombacher, University of Twente, The Netherlands

Table of Contents

User Preference Based Automated Selection of Web Service Compositions

Sudhir Agarwal and Steffen Lamparter

Institute of Applied Informatics and Formal Description Methods (AIFB),
University of Karlsruhe (TH), Germany.
{agarwal,lamparter}@aifb.uni-karlsruhe.de

Abstract. Semantically rich descriptions of web services enable automatic composition and matchmaking. End users as well as providers compose web services according to their needs. Since in general, there can be more than one possibilities (combinations of web services) to achieve a certain goal, a user has to decide which of the alternatives suits him the best. This leads to the requirement that the combinations of web services must be comparable. This can be achieved by aggregating web service attributes. Since the ranking of web service combinations depends on user's needs, capabilities and willingness for investment of resources, considering user preferences is important while calculating the rank of a web service composition. We show, how user preferences as well as aggregation information can be modeled in a formal way and how web service combinations can be ranked automatically based on this information.

1 Introduction

Composition of web services is performed by end users as well as web service providers. End users compose web services because (1) there may be no single web service that directly offers the desired functionality (2) a combination of web services may need less investment or capabilities than a single service. Providers compose web services in order to offer the composite service as a new web service. That is, the new composite web service acts as a mediator between an end user and other (component) web services.

However, there are in general more than one combinations of web services that can fulfill a given goal. That is, there is a need for decision support to help an end user to select one combination from many combinations. Such a decision support is only possible if the combinations are comparable. To compare two combinations of web services, the values of attributes of the component web services must be aggregated according to the structure of the composition. From the point of view of a provider aggregation is needed in order know how much he has to invest for the component web services so that he can decide how much he can demand for his composite service. In the following, we will not differentiate between end users and providers any further.

Consider a simple web service combination consisting of a sequence of web services w_1 and w_2 having price p_1 and p_2 respectively. Now, if a user wants to

know, how much the execution of the combination costs him, he needs to add p_1 and p_2. The values p_1 and p_2 are specified in the descriptions of the web services w_1 and w_2. Note, that in general, there is a need for machine support to perform such calculations, since (1) the number of web service combinations can be large, (2) the structure of the web service combinations can be complex, (3) a user may need to do such calculations very often. The most straightforward solution of our above problem would be, that the user inserts the descriptions of the web service combinations together with descriptions of the component web services and queries the knowledge base for the desired value, e.g. $p_1 + p_2$. This leads to the requirement, that the underlying user's description logic must support functions like addition, multiplication, minimum and maximum of numbers. In description logics theory, such functions are called aggregation functions and are first introduced in [1].

In this paper, our focus is on the selection phase. We make use of the techniques proposed in [1] to model aggregation information. In section 2, we will discuss some important non-functional properties of web services and show how they can be aggregated, when web services are composed. We show, how the aggregation information can be modeled as part of the ontology for describing web service combinations. In section 3, we show how user preferences can be modeled with fuzzy rules. In section 4, we will show how web services can be ranked based on the user preferences. We conclude in section 6 after discussing some related work in section 5.

2 Modeling Aggregation Information

While WSDL is commonly used to describe functional properties of a web service, there are currently no standards for describing non-functional properties of a web service and for describing composite web services. In general, every user can define his own ontologies to describe non-functional properties of his web service and to describe web service combinations. But, we believe, that in many cases users will use one of the already existing ontologies that are supported by their tools. Current ontologies for describing web service combinations do not allow to model the aggregation information about various attributes of component web services. Modeling aggregation information as part of a plan ontology is our main focus in this section. Having such aggregation information as part of description of a web service combination, values of various attributes of component web services can be aggregated automatically and a user only needs to specify his preferences to rank the plans.

2.1 Description of Non-Functional Properties

All those properties that are not absolutely necessary to be able to invoke a service and integrate the output are referred to as non-functional properties. In this section, we introduce the most common non-functional properties related to the direct usage of a web service. Note that only properties of the web service usage

itself and not of the product derived by means of a web service are discussed here.

Quality of Service. The most frequently discussed non-functional properties are quality of service attributes. These are attributes that define a minimal level of quality a service has to provide. They cover different aspects: The property *Locative Availability* defines a time frame a service is online, e.g. every day from 6 AM to 8 PM. Furthermore, an *Availability-Rate* can be defined that specifies the minimal percentage a service has to be available within a day, e.g. an Availability-Rate of 0.9 defines that a service has to be available at least 90% of a day. *Response Time* measures the duration between sending out the request and receiving a result from the service.

Price. A service price refers to the monetary amount that has to be paid by the requester to be allowed to use the service. Thus, the price is defined by an absolute *amount* and a specific *currency*. Furthermore, there may be *discounts* and *penalties* that can have influence on the price of a service.

Payment Method. To specify the price that has to be paid, properties like *Payment Instrument*, the *Duration*, and the *Charging Style* are required. Payment instrument refers to the type of payment, e.g. paying cash, by credit card, voucher, etc. The duration defines the time period in which the payment has to be completed. Moreover, services can be invoiced in different ways, which is specified by charging style (e.g. per invocation, for a certain time period).

Security. Security attributes allow to define *Identification Methods* as well as *Encryption Methods* that are supported by a service. Identification Methods refer to X509 [2], Kerberos [3], PIN input or similar systems. Encryption Methods defines the supported protocols for communication with a service.

Trust. There are various different ways to define trust levels with respect to web services. One way would be check for referrals by means of reputation systems. In this case a *Rating* attribute can be used to specify the level of trust within a certain integer range.

Privacy. Privacy statements basically describe what happens with the personal data of a requester once sent to a service. Such statements define, for instance, the allowed *Storage Period* of the data or if the service provider is allowed to disclose the data to third parties. This is expressed by the attribute *Disclosure*, which is valued by 'yes' or 'no'.

Of course, this list of non-functional properties is not exhaustive. A more detailed description of such properties can be found in [4].

2.2 Description of Web Service Combinations

Having discussed the non-functional properties an atomic web service, we now turn our attention to the description of web service combinations. We refer to such combinations as plans, processes or composite web services. We model four types of compositions, namely *sequence*, *parallel*, *choice* and *loop*. In this paper, our aim is to describe only static aspects of a composite web service. For doing so, we use standard description logic syntax and refer to [5] for details about the

semantics.

$$\texttt{Sequence} \sqsubseteq \texttt{WS} \sqcap \exists \texttt{component}.\texttt{WS}$$
$$\texttt{Parallel} \sqsubseteq \texttt{WS} \sqcap \exists \texttt{component}.\texttt{WS}$$
$$\texttt{Choice} \sqsubseteq \texttt{WS} \sqcap \exists \texttt{component}.\texttt{WS}$$
$$\texttt{Loop} \sqsubseteq \texttt{WS} \sqcap \exists \texttt{ws}.\texttt{WS} \sqcap \exists \texttt{times}.\mathbb{N}$$

Note, that we do not require to model the execution semantics of the various constructs in order to model aggregation information about the non-functional properties of the web service combinations. The roles `ws` and `times` are functional roles. That is, an instance of `Loop` can be related to only one instance of `WS` via the relation `ws` and to only one natural number via the relation `times`.

2.3 Modeling Aggregation Information

In section 2.1, we discussed some important non-functional properties of web services. While aggregating web service compositions only domain independent attributes are relevant.

	Sequence	Parallel	Choice	Loop
Locative Availability (LA)	*overlap*	*overlap*	*overlap*	LA of `ws`
Availability-Rate (AR)	$\prod$	min	min	*exp*((AR of `ws`), `times`)
Response Time (RT)	$\sum$	max	max	*mult*((RT of `ws`)), `times`)
Price Amount (PA)	$\sum$	$\sum$	max	*mult*((PA of `ws`), `times`)
Encryption Method (EM)	$\bigcup$	$\bigcup$	$\bigcup$	EM of `ws`
Identification Method (IM)	$\bigcup$	$\bigcup$	$\bigcup$	IM of `ws`
Rating (R)	min	min	min	R of `ws`
Storage Period (SP)	max	max	max	SP of `ws`
Disclosure (D)	$\bigvee$	$\bigvee$	$\bigvee$	D of `ws`

Fig. 1. Aggregation Functions

The function symbols used in figure 1 have their obvious meanings except the functions overlap, exp and mult. The function overlap determines the overlapping range of time periods $t_1, \ldots, t_n \in t$. The functions exp(x,y) and mult(x,y) calculate x^y and $x \times y$ respectively.

Now, we turn our attention to how the aggregation information as summarized figure 1 in can be modeled as part of the ontology for describing web service combinations that we described in section 2.2.

$$\texttt{Sequence} \sqsubseteq \prod_{i \in \{1,\ldots,n\}} P_{=}(a_i, \Sigma^S_{a_i}(\texttt{component} \circ a_i)) \tag{1}$$

where $\Sigma^S_{a_i}$ is an aggregation function for values of property a_i when composed in a sequence. E.g. a_i is *response time*, $\Sigma^S_{a_i}$ is equal to $\sum$.

$$\texttt{Parallel} \sqsubseteq \bigsqcap_{i \in \{1,\ldots,n\}} P_{=}(a_i, \Sigma^P_{a_i}(\texttt{component} \circ a_i)) \tag{2}$$

where $\Sigma^P_{a_i}$ is an aggregation function for values of property a_i when composed parallely. E.g. if a_i is *response time*, $\Sigma^P_{a_i}$ is equal to *max*.

$$\texttt{Choice} \sqsubseteq \bigsqcap_{i \in \{1,\ldots,n\}} P_{=}(a_i, \Sigma^C_{a_i}(\texttt{component} \circ a_i)) \tag{3}$$

where $\Sigma^C_{a_i}$ is an aggregation function for values of property a_i when composed as alternatives. E.g. if a_i is *response time*, $\Sigma^P_{a_i}$ is equal to *max*.

$$\texttt{Loop} \sqsubseteq \bigsqcap_{i \in \{1,\ldots,n\}} P_{=}(a_i, \Sigma^L_{a_i}(\texttt{ws} \circ a_i, \texttt{times})) \tag{4}$$

where $\Sigma^C_{a_i}$ is an aggregation function for values of property a_i when composed in a loop. E.g. if a_i is *response time*, $\Sigma^L_{a_i}$ is equal to *mult*.

3 User Preference Modeling

In many application domains, not only the membership of an individual to a set is nonrigid, but also the transition between the memberships of an individual from one set to another is smooth. Consider, for example, *height* of a human. Small children grow, but when do they stop to be small? So the transition from short humans to humans of average height is rather smooth and not crisp. Such kinds of knowledge can be encoded using techniques from fuzzy logic.

Vague knowledge, i.e. rules based on fuzzy logic, are also important from the perspective of evaluating values of attributes that have very complex dependencies with other attribute values. Such rules play an important role in application domains, where a good approximation of the desired value of an attribute is acceptable. For example, consider the controlling of a train. It is desired, that when a train arrives at station, it halts at a certain fixed position. However, calculating how exactly the brake should be applied at what position in which speed so that the passengers can still sit comfortable etc. is difficult. Considering that it is acceptable if the train stops a small distance before or after the mark, automatic control of the train is much easier.

Fuzzy logic, first introduced by Zadeh in [6, 7], provides answers to both the problems. On the one hand, the fuzzy sets allow to model vague memberships of individuals to sets. On the other hand, fuzzy IF-THEN rules allow to evaluate good approximations of desired attribute values in a very efficient way [6, 7].

In this section, we will show how user preferences can be modeled with fuzzy IF-THEN rules. We begin with the modeling of fuzzy membership functions and show how the membership of an individual to a fuzzy membership function can be calculated inside an appropriate description logic reasoner. Then, we model fuzzy IF-THEN rules and show how the degree of fulfillment of a rule by an individual can be calculated.

3.1 Modeling Fuzzy Membership Functions

Figure 2 shows the linguistic terms *fast*, *medium* and *slow* modeled as membership functions for a linguistic variable *Response Time*.

Now let $\mathbb{R}_{[0,1]}$ denote the set of real numbers between 0 and 1. For a concept v and a linguistic term t, we define a membership function μ_t^v as a finite and non-empty set of points[1] (x, y) in $\mathbb{R} \times \mathbb{R}_{[0,1]}$, where x is a number representing[2] an individual of the concept v. We will define next a concept *Point*. For this purpose, we introduce two concrete functional roles x and y, which assign the first respectively second coordinate to the point. So we define the concept *Point* as $Point \sqsubseteq \exists x.\mathbb{R} \sqcap \exists y.\mathbb{R}_{[0,1]}$.

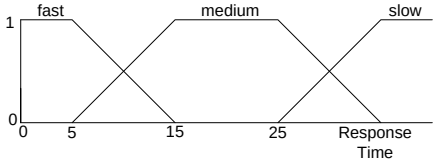


Fig. 2. Example Membership Functions

Similarly, we define a concept μ that denotes the set of membership functions. We do this by means of a (non-functional) role p which assigns points to individuals, as $\mu \sqsubseteq \exists p.Point$[3].

For a concept v, being viewed as a linguistic variable and having linguistic terms $t_1, \ldots, t_n$, we add n instances $\mu_{t_1}^v, \ldots, \mu_{t_n}^v$ of μ with corresponding roles and points. We interpret the set of points associated with some $\mu_{t_i}^v$ as a piecewise linear function. For a concept v being viewed as a linguistic variable, we denote the set of its linguistic terms by v^*.

We refer to [8] for complete modeling of the information needed to calculate the membership of a given individual to a given fuzzy set. As an example, the response time of 12 time units is medium with degree 0.7 and fast with degree 0.3 considering the fuzzy sets from figure 2.

3.2 Modeling Fuzzy Rules

A *fuzzy IF-THEN rule* consists of an IF part (antecedent) and a THEN part (consequent). The antecedent is a combination of terms, whereas the consequent is exactly one term. In the antecedent, the terms can be combined by using fuzzy conjunction, disjunction and negation. A *term* is an expression of the form $X = T$, where X is a linguistic variable and T is one of its linguistic terms.

Since terms are the elementary building blocks of a fuzzy rule, we start with modeling terms. As described above, a term consists of two parts, a linguistic variable and a linguistic term. So, we model a concept *Term* as $Term \sqsubseteq \exists r.\top \sqcap \exists f.\mu$, where the roles r and m are functional roles assigning linguistic

[1] I.e. we allow only membership functions which are piecewise linear.

[2] Identifying individuals with real numbers simply serves to make computations simpler, though it may appear to be counterintuitive in some cases.

[3] $\mu \sqsubseteq \geq 2\, p.Point$ would be more precise if qualified number restrictions are available.

variable resp. linguistic term. Terms can be combined via conjunction, disjunction and negation to term expressions. Further, a term expression is fulfilled by an individual to a certain degree. So, we define concept *TermExp* and extend the definition of the concept *Term* as follows:

$$\begin{aligned} \mathit{TermExp} &\sqsubseteq \exists \mathit{degree}.\mathbb{R}_{[0,1]} \\ \mathit{Term} &\sqsubseteq \mathit{TermExp} \end{aligned}$$

A rule has an antecedent and a consequent. The antecedent is a term expression and the consequent is a term. Further, a rule has a degree to which it is fulfilled by an individual. So, we define a concept *Rule* as[4]

$$\mathit{Rule} \sqsubseteq \exists \mathit{antecedent}.\mathit{TermExp} \sqcap \exists \mathit{consequent}.\mathit{Term} \sqcap \exists \mathit{degree}.\mathbb{R}_{[0,1]}.$$

3.3 Calculating the Degree of Fulfillment of a Rule

Since terms are the basic building blocks of a rule, the degree of fulfillment of a rule depends ultimately on the degrees of fulfillment of the terms occurring in the rule. Now, an individual connected to a term via the role r fulfills the term with the same degree as the corresponding value of the membership function the term is connected with via the role m. We model this by extending the concept *Term* with the axiom $\mathit{Term} \sqsubseteq P_{=}(\mathit{degree}, r \circ m_f)$.

We can calculate the degree of fulfillment of a term expression according to the semantics suggested by Zadeh in [6, 7], which can be summarized as follows.[5] Given two membership functions μ_A and μ_B, $(\mu_A \wedge \mu_B)(a) = \min\{\mu_A(a), \mu_B(a)\}$, $(\mu_A \vee \mu_B)(a) = \max\{\mu_A(a), \mu_B(a)\}$ and $(\neg\mu_A)(a) = 1 - \mu_A(a)$.

So, we define the concept $\mathit{TermExp}_{\wedge}$, $\mathit{TermExp}_{\vee}$ and $\mathit{TermExp}_{\neg}$ as follows, introducing also the corresponding roles[6].

$$\begin{aligned} \mathit{TermExp}_{\wedge} &\sqsubseteq \mathit{TermExp} \sqcap \exists \mathit{conjunct}.\mathit{TermExp} \sqcap P_{=}(\mathit{degree}, \min\{\mathit{conjunct} \circ \mathit{degree}\}) \\ \mathit{TermExp}_{\vee} &\sqsubseteq \mathit{TermExp} \sqcap \exists \mathit{disjunct}.\mathit{TermExp} \sqcap P_{=}(\mathit{degree}, \max\{\mathit{disjunct} \circ \mathit{degree}\}) \\ \mathit{TermExp}_{\neg} &\sqsubseteq \mathit{TermExp} \sqcap \exists \mathit{operand}.\mathit{TermExp} \sqcap P_{=1-}(\mathit{degree}, \mathit{operand} \circ \mathit{degree}) \end{aligned}$$

The predicate $P_{=1-}(a, b)$ is true iff $a = 1 - b$. min and max are aggregate functions for the concrete domain $\mathbb{R}$.

To interpret a fuzzy IF-THEN rule, we need an interpretation for the implication. In general, one can have a different interpretation of the implication for every rule, which is particularly important when the application domain requires the use of weighted rules. Here, we use a universal interpretation π of the implication in all the rules. However, we do not fix π any further. In most of the cases, it is equal to minimum. So,

$$\mathit{Rule} \sqsubseteq P_{\pi}(\mathit{degree}, \mathit{antecedent} \circ \mathit{degree}, \mathit{consequent} \circ \mathit{degree}),$$

[4] The roles *antecedent* and *consequent* are functional roles.
[5] Certainly, other T-norms and T-conorms could be used.
[6] The relation *operand* is a function role.

where P_π is a ternary predicate from the concrete domain $\mathbb{R}$ and represents the interpretation of the implication function π. That is, for given $a, b, c \in \mathbb{R}$, $P_\pi(a, b, c)$ is true iff $a = \pi(b, c)$.

3.4 Modeling User's Preference as Fuzzy Rules

We view preferences as the information that describes the constraints on the properties of an individual in order to be accepted for further consideration. We specify different levels of acceptance with fuzzy membership functions as described above.

In the web service compositions scenario, the individuals are concrete web service compositions. We model user preferences with fuzzy IF-THEN rules. The IF part contains membership functions of the various properties of an individual (e.g. those listed in figure 1) and the THEN part is one of the membership functions of a special concept called *Rank*. Intuitively, a fuzzy rule describes which combination of attribute values a user is willing to accept to which degree, where attribute values and degree of acceptance are fuzzy sets, i.e. vague. Note, that a user has to define at most as many rules as there are degrees of acceptance that he/she wants to differentiate. We believe, that in practice, the number of such categories will not be large. An example fuzzy IF-THEN rule can be

```
IF RT = fast and PA = cheap THEN Rank = high.
```

4 Automated Plan Selection

Let o represent the concept that represents the acceptance and let o be categorized in k categories represented by $g_1 \dots g_k$. Further, there exists k rules $R_1, \dots, R_i, \dots, R_k$, where R_i has g_i as conclusion. In the following, we calculate the ranking r of an individual a with respect to objective o according to the FITA principle (First Inferencing Then Aggregation). The other alternative for interpreting fuzzy rules is FATI (First Aggregation Then Inferencing). It has been shown in [9] that the two principles are equivalent.

4.1 FITA

Consider a rule base containing n rules of the form $F_1 \rightarrow G_1, \dots F_n \rightarrow G_n$. In FITA, first each rule is interpreted. That is, for each x, y and each rule i, the value of $\pi(F_i(x), G_i(y))$ is calculated. Now, for a given x, the inference step is performed for each rule. Again, the inference operator can be different for different rules. However, we use a universal inference operator for all the rules and call it κ — in many practical cases, κ is equal to minimum. In general, the inference operator κ is some function that maps the square $[0, 1]^2$ to $[0, 1]$. Performing an inference step for a given x and a given rule i means calculating $\kappa(F(x), \pi(F_i(x), G_i(y)))$, where F is some fuzzy set describing the membership function for a given situation. Having performed the inferencing step for all the

n rules, an aggregation step is performed to obtain a single value from n values. For this purpose, an aggregation operator $\alpha : [0,1]^n \rightarrow [0,1]$ is needed. The most common aggregation operator is maximum. However, we do not fix α any further. In the aggregation step,

$$\alpha(\kappa(F(x), \pi(F_1(x), G_1(y))), \ldots \kappa(F(x), \pi(F_n(x), G_n(y))))$$

is calculated. We define a concept *FITA* as:

$$\begin{aligned} \mathit{FITA} \sqsubseteq & \prod_{i\in\{1,\ldots,n\}} \exists \mathit{rule}_i.\mathit{Rule} \sqcap \prod_{i\in\{1,\ldots,n\}} \exists x_i.\mathbb{R}_{[0,1]} \sqcap \exists \mathit{output}.\mathbb{R}_{[0,1]} \sqcap \\ & \prod_{i\in\{1,\ldots,n\}} P_\kappa(x_i, \mathit{rule}_i \circ \mathit{degree}, x \circ m_f) \sqcap P_=(\mathit{output}, \alpha\{x_i\}) \end{aligned}$$

where α is an aggregate function and P_κ is a ternary predicate on the concrete domain $\mathbb{R}_{[0,1]}$. $P_\kappa(a,b,c)$ is true iff $a = \kappa(b,c)$.

4.2 Defuzzification

The goal of a DL query is to determine the value of an instance. FITA delivers the membership of an arbitrary instance of the target concept to the goal fuzzy concept according to the compositional rule of inference. That is, if we have a sufficient number of instances of the target concept, we can calculate for each instance its membership to the goal fuzzy concept. This way, we obtain a set of points $(x, \mu_T(x))$, where x is an arbitrary instance of the target concept and μ_T is the target fuzzy concept.

However, the goal of query answering is to determine an instance of the target concept. This is done by interpreting the set of points $(x, \mu_T(x))$ as an area in $\mathbb{R}^2$ and defuzzifying this area. One of the most common defuzzification methods is the so called *center of gravity* method, where the geometrical center of gravity of a given area is calculated. The desired instance is then equal to the value of the x-coordinate of the center of gravity of the area. Hence, the desired instance ω can be calculated by the following formula:

$$\omega = \frac{\int x \cdot \mu(x) dx}{\int \mu(x) dx} \tag{5}$$

To model the defuzzification process, we define a concept *DefuzInfo* as: $\mathit{DefuzInfo} \sqsubseteq \exists \mathit{fita}.\mathit{FITA} \sqcap \exists x \sqcap \exists \mathit{prod}.\mathbb{R} \sqcap P_{mul}(\mathit{prod}, x, \mathit{fita} \circ \mathit{output})$.

Finally, consider a concept $C \sqsubseteq \exists w.W$. For a given instance θ of C, to determine an instance ω such that $w(\theta, \omega)$ holds while considering fuzzy rules, we extend the definition of the concept C as follows:

$$C \sqsubseteq \exists \mathit{di}.\mathit{DefuzInfo} \sqcap P_=(w, x) \sqcap P_{div}(x, \mathit{sum}\{\mathit{di} \circ \mathit{prod}\}, \mathit{sum}\{\mathit{di} \circ \mathit{fita} \circ \mathit{output}\}). \tag{6}$$

We use the already existing instances of the concept W as the arbitrary instances for determining the area that is defuzzified. If no instances of W are

available in the knowledge base, we can always insert some instances which do not need to be in any relation with instances of other concepts. The number and value of such instances depends on the application domain, more precisely on the width of the range (subset of $\mathbb{R}$) and on the value of dx in equation 5.

4.3 Calculation of Ranking of Web Service Compositions

In the above sections, we have explained how an attribute value of an individual that has complex dependencies on other attribute values of the same individual can be calculated automatically, where the dependencies are modeled as fuzzy rules. Coming back to our big picture described in section 1, we have the situation that a user has many web service compositions, each of them fulfilling user's goal, and the user needs to select one of them for execution. That is, we have many web service compositions and we wish to calculate rank for each of them based on user's preferences.

We calculate the rank of a web service composition by setting C equal to WS and w equal to $rank$ in equation 6. Further, we set range of $rank$ equals to $Rank$. That is, $WS \sqsubseteq \exists rank.Rank$. The rank of a web service composition is then the value of the attribute $rank$. Finally, we perform this step for each web service composition. Since, we have already modeled the calculation of aggregation of various attribute values in equations 1, 2, 3 and 4, they are automatically considered in the calculation of the overall ranking.

4.4 Example

Consider two web services w_1 and w_2. Suppose both the web services w_1 and w_2 use the web services c_1, c_2 and c_3 with the only difference that w_1 uses them in sequence whereas w_2 calls them in parallel. Suppose the response times of c_1, c_2 and c_3 are 3, 4 and 5 time units resp. Suppose, a user is only concerned with the response time of the web services and has the following preferences modeled as fuzzy rules

```
  IF RT = fast THEN Rank = high
IF RT = medium THEN Rank = average
  IF RT = slow THEN Rank = low
```

The response times rt_1 and rt_2 of w_1 and w_2 are calculated with equations 1 and 2 as 12 and 5 time units resp. Degrees of fulfilments of the rules by w_1 are 0.3, 0.7 and 0 resp. and by w_2 1.0, 0 and 0 resp. (cf. figure 2). Considering, "low", "medium" and "high" as fuzzy sets for the concept "Rank", step FITA yields one area for each web service. For w_1 this area is the maximum of the areas for the sets low, medium and high chopped at 0, 0.7 and 0.3 and for w_2 the maximum of of the areas chopped at 0, 0 and 1.0. Now, the center of gravities g_1 and g_2 of both the areas are calculated in the defuzzification step. g_1 lies left to g_2 which means w_1 is ranked lower than w_2, which corresponds to the intuition.

5 Related Work

There are quite a few efforts for modeling web processes. OWL-S [10] and BPEL4WS [11] are the most widely known. OWL-S claims to have formal semantics and thus added value as compared to XML based approaches like BPEL4WS. However, none of them allows to model aggregation information, which is necessary to reason about composite web services. We believe, that OWL-S and similar ontologies can become more useful by using our approach to enable formal specification of aggregation information and thus allowing reasoning about properties of web service combinations and making them comparable.

There are a few approaches that have investigated how various attributes of workflows or composite web services can be aggregated [12–14]. Our approach builds on the existing works, since it uses the insights gained from the mentioned works. The mentioned works does not provide a mechanism how the aggregation information can be modeled as part of a process ontology. We have shown, that ranking and thus plan selection is a possible use case for aggregation of web service attributes. We have also shown how rankings can be calculated and plan selection can be automated. Most of the existing approaches for automatic selection e.g. [15] either consider only atomic services or they are not based on user preferences.

6 Conclusion and Outlook

In this paper, we presented a generic approach for modeling aggregation information for various web service attributes as part of an ontology for describing composite web services. We have also shown, how user preferences can be modeled as fuzzy rules. Consequently, with the techniques presented in this paper, web service combinations can be compared with each other and ranked according to the user preferences.

In our previous works [16], we have developed a tool for automatic composition of web services. Currently, a user has to go through all the generated plans and decide manually, which plan he wishes to execute. We intend to extend this tool by a plan selection component based on the approach presented in this paper. The tool will allow a user to enter his preferences and present the user a list of generated plans sorted by rank. On the basis of which the user can decide more easily which of the plans, he wishes to execute. Further, a user can define a priori a threshold for rank of a plan, which he wishes to be executed automatically.

In this paper, we suggested to select composite services based on a ranking calculated with respect to user preferences. However, this "take it or leave it" principle may be sometimes economically inefficient, since possible negotiations yielding higher benefits for both sides are not considered [17]. In future, we wish to investigate the possibility of integrating multi attributive negotiations that facilitates Pareto-optimal allocations into our system.

Acknowledements

This work was funded by the Federal Ministry of Education and Research (BMBF), the German Research Foundation (DFG), and the European Union in scope of the Internetökonomie project SESAM, the Graduate School Information Management and Market Engineering and the IST project SEKT under contract IST-2003-506826.

References

1. Baader, F., Sattler, U.: Description Logics with Concrete Domains and Aggregation. In Prade, H., ed.: Proceedings of the 13th European Conference on Artificial Intelligence (ECAI-98), John Wiley & Sons Ltd (1998) 336–340
2. OpenSSL: Openssl x509. (http://www.openssl.org/docs/apps/x509.html)
3. MIT: Kerberos. (http://web.mit.edu/kerberos/www/)
4. O'Sullivan, J., Edmond, D., ter Hofstede, A.: Formal description of non-functional service properties. Technical report, Queensland University of Technology (2005)
5. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F., eds.: The Description Logic Handbook: Theory Implemenation and Applications. Cambridge University Press (2003)
6. Zadeh, L.A.: Fuzzy Sets. Information and Control **8** (1965) 338–353
7. Yager, R.R., Ovchinnikov, S., Tong, R.M., Nguyen, H.T., eds.: Fuzzy sets and applications - Selected Papers by L. A. Zadeh. John Wiley & Sons, New York, NY, USA (1987)
8. Agarwal, S., Hitzler, P.: Modeling Fuzzy Rules with Description Logics. In: Proceedings of Workshop on OWL Experiences and Directions, Galway, Ireland (2005)
9. Temme, K.H., Thiele, H.: On the correctness of the principles of FATI and FITA and their equivalence. In: IFSA 95 - Sixth International Fuzzy Systems Association World Congress. Volume II. (1995) 475–478
10. Coalition, D.S.: DAML-S: Web Service Description for the Semantic Web. In: ISWC2002: Ist International Semantic Web Conference, Sardinia, Italy. LNCS, Springer (2002) 348–363
11. et al., T.A.: Business Process Execution Language for Web Services. Technical report, Bea Systems, IBM Corp., Microsoft Corp., SAP AG, Siebel Systems (2003)
12. Cardoso, J., Sheth, A., Miller, J., Arnold, J., Kochut, K.: Quality of Service for Workflows and Web Service Processes. Journal of Web Semantics **1** (2004)
13. Cardoso, J., Sheth, A.: Semantic e-Workflow Composition. Journal of Intelligent Information Systems **21** (2003) 191–225
14. Jaeger, M.C., Rojec-Goldmann, G., Mühl, G.: QoS Aggregation in Web Service Compositions. In: Proceedings of IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE-05), China, IEEE Press (2005) 181–185
15. Liu, Y., Ngu, A.H., Zeng, L.Z.: Qos computation and policing in dynamic web service selection. In: WWW Alt. '04: Proc. of 13th Int. WWW Conf. on Alternate track papers & posters. (2004) 66–73
16. Agarwal, S., Handschuh, S., Staab, S.: Annotation, Composition and Invocation of Semantic Web Services. Journal of Web Semantics **2** (2005) 1–24
17. Bichler, M.: An experimental analysis of multi-attribute auctions. Decision Support Systems **29** (2000)

RFT: a useful model for dynamic web processes

Justin O'Sullivan and David Edmond

Business Process Management Group
Centre for Information Technology Innovation
Queensland University of Technology
GPO Box 2434
Brisbane QLD 4001 (Australia)
justin@service-description.com

Abstract. Whilst existing web processes can be considered dynamic from a service requestor perspective, the service provider must advertise their service description into a catalogue. This results in a static description of a service that must be general enough to cater for all requestors undertaking service discovery. We feel that dynamic web processes should extend to the service descriptions advertised by the service provider. To that end, we believe that the Request for Tender (or RFT) provides a useful model for dynamic web processes. RFTs result in a dynamic service interaction lifecycle for both the service requestor and provider, in particularly creating an environment where responses can be tailored to a specific service requestor's needs.

1 Introduction

The process of tendering is currently used internationally by both businesses (large and small) and government in the procurement of goods and services. It is our assertion that this process (sometimes referred to as a Request for Tender or RFT) is a useful analogy for the service lifecycle. An overview of the tendering process is presented in [1]. We believe that the commonly referenced publish-find-bind model [2] of services not representative of how business is conducted. Under the publish-find-bind model a service provider publishes the description of their service, requestors use a catalogue to find the service description(s) and then bind to the service that they feel is most appropriate. It is our opinion the service provider should not be required to publish a "one size fits all" description into a catalogue.

In contrast, the tendering process provides an opportunity for the prospective purchaser to state its requirements, assess potential providers through the receipt of responses, evaluate responses by subjecting them to specific criteria, and award a contract to the winning tender. A variation on this process is referred to as a "Request for Expressions of Interest" (REOI). This normally involves the publishing of an REOI document. A short list of potential tenderers is selected from the responses to the REOI. These parties are subsequently invited to

provide more formal offers. REOI also provides an opportunity for a service requestor to determine the level of interest that exists with respect to their stated needs.

The rest of the paper is structured as follows. In section 2 we provide an insight into our view of the tendering process as an analogy for the service lifecycle. Next, in section 3, we present a discussion of how the non-functional properties of services could enhance the tendering process. We have previously highlighted the need for a comprehensive taxonomy for the non-functional properties of services [3]. Such a taxonomy is presented in detail in [4]. The formal taxonomy is also available as a set of navigable models from `http://www.service-description.com/`. Research into the application of these non-functional properties to the tendering process should be considered in its early stages. Finally, we present our conclusions in section 4.

2 Tendering analogy

As previously stated, we believe that tendering acts as a useful analogy for the service lifecycle (i.e. service discovery, negotiation and invocation). We use Figure 1 to present an overview of the tendering process. In this figure we have denoted four major steps within the process: (1) the publishing of the RFT, (2) discovery of the RFT by potential providers, (3) submission of responses, and (4) awarding of a contract. We have purposely not depicted the evaluation of responses to the RFT.

Automation of the tender process (sometimes referred to as "e-tendering") currently facilitates the publishing and retrieval of tender documents. It doesn't provide a format that enables RFTs or tender responses to be created dynamically through the use of a common vocabulary, or to be reasoned about in some way. Some interesting tender sites in Australia are TenderLink [5] and AusTender [6]. We advocate such a common vocabulary and we believe that the non-functional properties outlined in [4] constitute a basis for such a language.

We do not consider that the existing approach of a service provider publishing a detailed service description to a catalogue as being sufficiently flexible for the various service requestor's needs. Service descriptions (or advertisements) attempt to become a "one size fits all" description of the service(s) offered by the provider. Our preference is for the service requestor to initiate the discovery process (through publishing the RFT) and for service providers to respond to each RFT of interest. This approach is only achievable when both the functional and non-functional requirements for the service can be specified by the service requestor, and the response can be confirmed or further refined, either functionally or non-functionally. Criticism that this approach results in a state where the service requestor is unaware if a RFT response is being prepared by a service provider(s). The shortened period of initial responses to a REOI would provide a solution to this criticism. We advocate the use of a language that supports such a level of description.

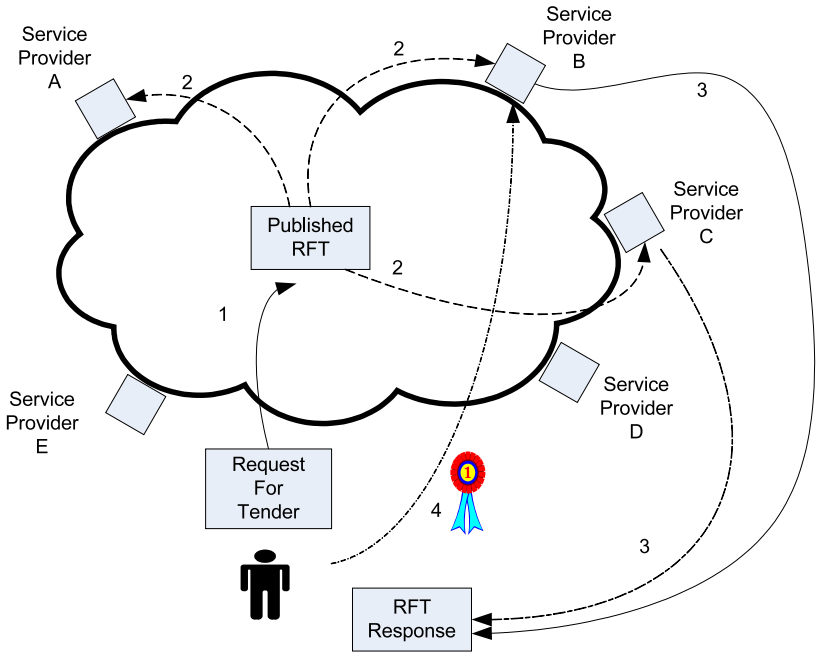


Fig. 1. The tender process.

Why take this approach? The primary advantage is that unlike current discovery mechanisms the service descriptions of a provider (i.e. the party that responds to the RFT) are both tailored to the specific request, and published in response to a request (thus avoiding the issue of out-of-date service descriptions). This in turn removes the necessity for a catalogue, it just requires a well-known space for advertising RFTs. This approach would appear to be amenable to the notion of Triple Space Computing [7]. Service providers would also be capable of undertaking dynamic service compositions to meet the requirements stated in tender documents. The tender process also provides a level of transparency for all the service providers. They are usually then aware of all the information that is pertinent to the tender.

As discussed, we believe that the analogy of the RFT provides a useful model for dynamic web processes. It is a slight departure from the existing publish-find-bind model, as the requestor does the publishing, with the provider undertaking the discovery (or finding). With a view to using this type of model with dynamic web processes, we have begun to investigate the content of the interactions that occur between parties under this model. One area of previous research that appears applicable to the proposed model is outlined in [3]. RFTs are rich in non-functional properties. These properties assist service providers with discovery of appropriate RFTs, in guiding how to respond to an RFT, and with providing details within the response to the RFT.

3 Non-functional properties in tendering

After having reviewed a significant number of tender documents we believe that there is a clear distinction between the non-functional properties available in tender documents. It would appear that they can be categorised into two areas:

- Those non-functional properties that that relate to the conditions of tendering (i.e. the temporal and locative requirements for submitting a response, laws applicable to the RFT, late response conditions and procedures, and the publisher of the RFT).
- Those non-functional properties that relate to the service delivery outlined within the tender (or an associated appendix). This refers to items such as the location of provision, length of provision, agreed response times, and the warranty period.

The alignment of the RFT model and our previous non-functional property work should be considered "early stage" research. We provide some initial findings below.

Numerous non-functional properties are provided within RFT documents that relate to the conditions of tendering. These include but are not limited to the following:

- Publisher of the RFT, RFT title, and identifier for the RFT.
- The tender lifecycle (dates for issuing the date, closing dates for respondents, selection/evaluation dates, and service commencement).
- The temporal and locative requirements for submitting a response to the RFT (including where and when to perform enquiries with respect to the RFT).
- Definition of terms used within the RFT.
- Pre-conditions to the tender response, selection criteria for response.
- Rights of the party providing the RFT.
- Laws applicable to the RFT (e.g. privacy, auditing, and freedom of information) and applicable jurisdiction(s).
- Retention of intellectual property rights.
- Late response conditions and procedures.
- Tender correction handling procedure.
- Miscellaneous - the bearer of tender response costs, the tender response validity period, general tender submission conditions (including reasons for excluding particular tenders), currency to be used when stating prices, exclusion/inclusion of taxes within prices, consideration for part or joint tenders available, and the language that the response must be provided in.

We are more interested in the non-functional properties that are stated within the requirements section of the RFT. These include but are not limited to the following:

- Location of provision (e.g. Street addresses).

- Ability to perform the services outside normal working hours (e.g. 8am to 4pm).
- Ability to state that you can meet the agreed response time(s). This would need to be stated as specific temporal durations, instants or intervals.
- Length of provision/term of the contract - Examples include the ability to state periods such as 3 years, or alternatively concepts such as 3 years plus two one year options. Provision may also be stated in the form of a temporal interval (e.g. 7:30am to 6:30pm seven days a week including public and other holidays).
- Length of period that a price is valid for - Normally a temporal interval or a temporal duration.
- Discounts that apply to various methods of payment - For example, specific instrument types that receive a discount, or alternatively an early payment discount.
- Warranty periods.
- Term of the contract (e.g. 3 years).
- Related business (e.g. Business "X" is a subsidiary of business "Y").

It is the introduction of a taxonomy for these latter non-functional properties (i.e. those relating to the functional requirements) that will enable the tendering process to be further automated. It does so by enabling the issuer of the RFT to state their functional requirements in conjunction with their non-functional requirements. These descriptions can then be reasoned over by the service provider who formulates a response to the RFT. A requestor could be considered to provide a partial instantiation of the models presented in [4]. This is advertised to some well-known location. Responses can then be a refinement of the originally instantiated models.

4 Conclusion

The current approach to service discovery involves service requestors using catalogues to determine service providers who are capable of meeting their requirements. We believe that a better approach is to model the service lifecycle on RFTs. This enables service requestors to receive responses that are specific to their requirements. It subsequently results in a catalogue free service environment. Dynamic web processes should be dynamic for all parties concerned. We consider the use of RFTs as a slight departure from the existing publish-find-bind model of service interaction. The approach is advantageous as it is currently used by both business and government, and more importantly, is specifically targeted to the needs of the requestor.

References

1. Standards Australia: Code of Tendering - AS 4120 (1994)
2. Gottschalk, K., Graham, S., Kreger, H., Snell, J.: Introduction to Web services architecture. IBM Systems Journal **41** (2002) 178–198

3. O'Sullivan, J., Edmond, D., Hofstede, A.t.: What's in a service?: Towards accurate description of non-functional service properties. Distributed and Parallel Databases Journal - Special Issue on E-Services **12** (2002) 117–133
4. O'Sullivan, J., Edmond, D., Hofstede, A.H.t.: Formal description of non-functional service properties. Technical FIT-TR-2005-01, Queensland University of Technology, Brisbane (2005) Available from `http://www.citi.qut.edu.au/about/research_pubs/technical/non-functional.jsp`, accessed on 15-Feb-2005.
5. Tenderlink.com Pty Ltd: Tenderlink (2005) Available from `http://www.tenderlink.com/`, accessed on 01-Aug-2005.
6. Federal Government of Australia: AusTender - The Australian Government Tender System (2005) Available from `http://www.tenders.gov.au/`, accessed on 01-Aug-2005.
7. Bussler, C.: A Minimal Triple Space Computing Architecture. In: 2nd WSMO Implementation Workshop, Innsbruck, Austria, Digital Enterprise Research Institute (2005)

Towards a Formal Model for Agile Service Discovery and Integration

Hagen Overdick, Frank Puhlmann, and Mathias Weske

Hasso-Plattner-Institute for IT Systems Engineering
at the University of Potsdam
D-14482 Potsdam, Germany
{overdick,puhlmann,weske}@hpi.uni-potsdam.de

Abstract. As the fundamental web services technologies are becoming mature, web service composition, orchestration and choreography are gaining increasing attention. The movement from static interactions between already known partners as in BPEL to dynamically discovered and agile business partners is irresistible facing ever changing environments while aiming for specifically optimized collaborations. However, the techniques and models currently used lack fundamental formal foundations making them inadequate especially for modeling non-functional aspects. As a step toward the vision of dynamically discovered and agile processes, this paper proposes a formal approach to unambiguously define the syntax of service orchestrations and choreographies by representing key elements of service–oriented computing in a process algebra, the π-calculus. The results include a formal description of correlations in the context of service choreography as well as a formal representation of an orchestration pattern derived from BPMN and BPEL. The results provide a better understanding of service-based processes in terms of a formal algebra that will open the door for automated discovery and binding of potential business partners via service equivalence and mobility.

1 Introduction

As the fundamental web services technologies are becoming mature, web service composition, orchestration and choreography are gaining increasing attention. The movement from static interactions between already known partners as in BPEL [1] to dynamically discovered and agile business partners is irresistible facing ever changing environments while aiming for specifically optimized collaborations. However, the techniques and models currently used lack fundamental formal foundations making them inadequate especially for modeling non-functional aspects.

In this paper we try to remedy this situation by proposing a formal approach to unambiguously define service orchestrations and choreographies. We have recently shown that the π-calculus, a process algebra, is capable to formally specify all workflow patterns [2]. That is a major advance in contrast to established formal approaches like workflow nets [3, 4]. Furthermore, the π-calculus

supports the concept of link passing, that allows the representation and analysis of dynamic process structures, as can be found in service oriented environments. This paper extends our previous work on using the π-calculus for representing workflow patterns to other key elements of service oriented computing. Workflow patterns represent the orchestrational view of services. The proposed formal and unambiguous characterizations of processes in service-oriented environments are useful for a precise understanding of these processes, as well as enabling further research on automated discovery and binding via service equivalence and mobility [5, 6].

The paper is organized as follows. Section 2 introduces the π-calculus as well as formal workflow modeling by example, thus representing the state-of-the-art. Section 3 discusses the representation of services in the π-calculus, including correlations, invocation and data flow. Section 4 describes a formal service orchestration by example, using the results from section 3. It furthermore introduces a behavioral pattern named *Event-based Rerouting*, which is not contained in the workflow patterns collection so far [7]. The paper concludes with a short summary and discussion of related work.

2 The π-calculus

The π-calculus is a process algebra intended to describe mobile systems [8]. Mobile systems are made up of components which communicate and change their structure as a result of interaction. The core concepts of the algebra are processes and names. A π-calculus process is an entity which can communicate with other processes by the use of names. A name is a collective term for existing concepts like links, pointers, references, identifiers, etc. Names could be unbound (global) or bound to specific processes, i.e. they have a scope. The scope of a bound name can be dynamically expanded or reduced during the lifetime of the system by communicating names between processes. As this paper has a limited size, we can only introduce the notation of the π-calculus that will be used. Further details can be found in [8–11].

Syntax. The π-calculus consists of an infinite set of process identifiers denoted as $\mathcal{K}$ and another infinite set of names denoted as $\mathcal{N}$, where names define links. The processes are defined as:

$$P \;::=\; M \mid P|P \mid \mathbf{v}zP \mid !P \;.$$

The composition $P|P$ is the concurrent execution of P and P, $\mathbf{v}zP$ is the restriction of the scope of the name z to P, which is also used to generate a unique, fresh name z and $!P$ is the replication operator that satisfies the equation $!P = P \mid !P$. M contains the summations of the calculus:

$$M \;::=\; \mathbf{0} \mid \pi.P \mid M + M$$

where $\mathbf{0}$ is inaction, a process that can do nothing, $M+M$ is the exclusive choice between M and M, and the prefix $\pi.P$ is defined by:

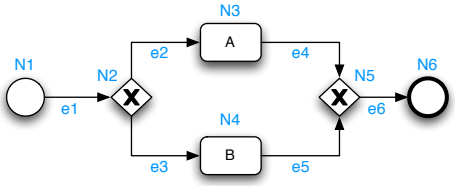


Fig. 1. A simple BPMN process.

$$\pi \quad ::= \quad \overline{x}\,\langle y\rangle \;\mid x(z) \mid \tau \mid [x=y]\pi\ .$$

The output prefix $\overline{x}\,\langle y\rangle\,.P$ sends the name y over the name x and then continues as P. The input prefix $x(z)$ receives any name over x and then continues as P with z replaced by the received name (written as $\{^{name}/_{z}\}$). The unobservable prefix $\tau.P$ expresses an internal action of the process, and the match prefix $[x=y]\pi.P$ behaves as $\pi.P$, if x is equal to y.

Throughout this paper, upper case letters are used for process identifiers and lower case letters for names. Two pre–defined, static names $\top$ and $\bot$ denote true and false. More additional process identifiers and names that represent special functions are introduced later on. Furthermore defined processes from the original paper on the π-calculus are used for parametric recursion, that is $A(y_1, ..., y_n)$ [8].

The abbreviation $\sum_1^m(M)$ is used to denote the summation of m choices; e.g. $\sum_1^3(M_i) = M_1 + M_2 + M_3$. $\prod_1^m(P)$ is used to denote the composition of m parallel copies of P, e.g. $\prod_1^3(P) = P \mid P \mid P$. Also, $\{\pi\}_1^m$ denotes m subsequent executions of π, e.g. $\{\pi\}_1^3 = \pi.\pi.\pi$. All abbreviations could be used with an indexing variable, e.g. $\prod_{i=1}^3(d_i(x)) = d_1(x) \mid d_2(x) \mid d_3(x)$. Round brackets are used to define the ordering of a process definition. Given $\tau.P$ for instance, P might be expanded to $M + M'$ by using the summation rule from the π-calculus grammar. To avoid ambiguity, round brackets are put around the expanded symbol, e.g. $\tau.(M + M')$ instead of $\tau.M + M'$.

Semantics. The behavior of the π-calculus is defined by a reduction relation, $\longrightarrow$, on processes. The essence is captured in the axiom $(\overline{x}y.P_1 + M_1)|(x(z).P_2 + M_2) \longrightarrow P_1|P_2\{^{y}/_{z}\}$. The axiom states that whenever two processes can communicate, they will communicate and all other capabilities are rendered void. There exists other axioms, such as for structural congruence, which allow the conversion of π-calculus processes [11].

Representing Workflows in the π-calculus. We introduce the application of the π-calculus with a rather simple example shown in figure 1. The notation of the example is BPMN and the process will be mapped to π-calculus expressions. The process consists of two tasks A and B, which are placed between two XOR-gateways. As a first mapping task, all flow objects, i.e. events, gateways, tasks,

are assigned an unique π-calculus process identifier. The start event is assigned to $N1$, the first XOR-gateway to $N2$, the task A to $N3$, the task B to $N4$, the second XOR-gateway to $N5$, and the end event to $N6$. All sequence flows are mapped to a unique π-calculus name from $e1$ to $e6$ (see figure 1).

By referring to the generic structure for π-calculus processes that represent basic workflow activities [2], we can derive the next steps:

$$\{x_i\}_{i=1}^{m}.\{[a=b]\}_1^n.\tau.\{\overline{y_i}\}_{i=1}^{o}.\mathbf{0}\ . \tag{1}$$

Each basic activity consists of pre- and postconditions for routing the control flow as well as an unobservable action τ that represents the functional perspective of the activity. The precondition is split into two part: (1) $\{x_i\}_{i=1}^{m}$ denotes that the activity waits for m incoming names, and (2) $\{[a=b]\}_1^n$ denotes n additional guards that have to be true to execute the activity. The postcondition denotes the triggering of o outgoing names.

By using the definition from equation 1, we can define the π-calculus process $N1$ for the start event from the example:

$$N1 = \tau_{N1}.\overline{e1}.\mathbf{0}\ .$$

$N1$ has no preconditions; therefore the responding part from equation 1 is omitted. The functional part is represent by τ_{N1}, where the index $N1$ denotes that this τ belongs to the process $N1$. The postcondition of $N1$ signals $e1$, which will trigger the process $N2$.

The process $N2$ represents an XOR-gateway which routes the control flow depending on some conditions. $N2$ has the name $e1$ as a precondition and either $e2$ or $e3$ as postconditions. However, the generic structure for basic workflow activities from equation 1 only supports serial AND-split triggering by calling the names $y_1 ... y_o$ sequentially. A modified version of the structure that supports XOR, OR, as well as AND splits is as follows:

$$\{x_i\}_{i=1}^{m}.\{[a=b]\}_1^n.\tau.\prod_{i=1}^{o}[c=\top]\overline{y_i}.\mathbf{0}\ . \tag{2}$$

For each possible postcondition, a parallel process part is introduced by $\prod_{i=1}^{o}[c=d]\overline{y_i}.\mathbf{0}$. Each part has a match prefix which either enables (true) or disables the postcondition (false). If the match prefix is true, the corresponding name is signaled. All other parts are discarded. A π-calculus process that behaves like an OR/XOR-split as a postcondition is then defined by i.e. $[c=\top]\overline{y_1}.\mathbf{0} \mid [d=\top]\overline{y_2}.\mathbf{0}$. If no match prefixes overlap, the split behavior is XOR, otherwise OR. An XOR split could also be realized by using the summation operation ($\sum$) instead of concurrency. That would also allow non-deterministic choices. An AND-split simply has no match prefixes at all: $\overline{y_1}.\mathbf{0} \mid \overline{y_2}.\mathbf{0}$.

By using the definition from equation 2, $N2$ can be defined:

$$N2 = e1.\tau_{N2}.([c_1=\top]\overline{e_2}.\mathbf{0} \mid [c_2=\top]\overline{e_3}.\mathbf{0})$$

As the business process diagram from the example contains no conditions, we assume two global names c_1, and c_2 that hold the expressions. Per definition of the exclusive choice pattern, we assume c_1 and c_2 to be disjoint; i.e. if $c_1 = \top \Rightarrow c_2 = \bot \wedge c_1 = \bot \Rightarrow c_2 = \top$.

The π-calculus processes for the tasks A and B from the example are straightforward as they simply contain one pre- and postcondition each:

$$N3 = e2.\tau_{N3}.\overline{e4}.\mathbf{0}, N4 = e3.\tau_{N4}.\overline{e5}.\mathbf{0} \ .$$

The XOR-join gateway $N5$ is rather simple, as we can adapt the pattern simple merge from [2]. As the simple merge pattern has only one precondition per definition (i.e. $d.\tau_d.D'$), we need an additional π-calculus process, which triggers $N5$ if either $e4$ or $e5$ are signaled:

$$N5 = \mathbf{v}x(e4.\overline{x}.\mathbf{0} + e5.\overline{x}.\mathbf{0} \mid x.\tau_{N5}.\overline{e6}.\mathbf{0}) \ .$$

The left hand side of the definition contains the additional process. It generates a fresh name x, which is used to signal the precondition of the XOR-join, which is contained at the right hand side. As the name x is unique to $N5$, only $e4$ or $e5$ could trigger the precondition and enable $\tau_e 6$.

The last process, $N6$, is now trivial:

$$N6 = e6.\tau_{N6}.\mathbf{0} \ .$$

3 Representing Services in the π-calculus

We have shown that the few concepts of the π-calculus are readily suited to model workflows, which can also be used to describe complex orchestrations in the context of service–oriented architectures. Indeed, all workflow patterns described in [7] can be formalized with basic π-calculus expressions, as shown recently [2]. However, so far we have only discussed how the routing of activities, i.e. web service calls, can be realized in the π-calculus in a straightforward way. This discussion will be continued in the next section. First, the invocation and providing of services in the π-calculus will be introduced.

3.1 Invoking Services

As with activities, services are also invoked by names in the π-calculus. In the BPMN notation, message flow is used for this purpose, whereas sequence flow controls the routing within a process. We define two distinct subsets from the set of names, $\mathcal{N}_\mathcal{S} \subset \mathcal{N}$ that contains all names used for service communication, and $\mathcal{N}_\mathcal{C} \subset \mathcal{N}$ that contains all names used for control flow routing. The intersection of $\mathcal{N}_\mathcal{S}$ and $\mathcal{N}_\mathcal{C}$ has to be empty: $\mathcal{N}_\mathcal{S} \cap \mathcal{N}_\mathcal{C} = \varnothing$. To enable different kinds of service invocations, we furthermore extend the τ prefix of the π-calculus with a placeholder denoted as $\square$. Each τ inside an activity definition that contains service invocations is then replaced by the placeholder.

Correlations. A common problem in the area of service–oriented computing is the description of correlations between service invokers and service providers. Usually, some kind of correlation identifier is placed inside each request and reply [12]. The invoker as well as the provider have to take care to match all requests. In the π-calculus, the unique identifier of a request is also the channel used for reply from the service. By merging these two concepts, a clear representation of the correlations is straightforward. A new unique identifier is a π-calculus name from the set of $\mathcal{N}_\mathcal{S}$ which is created with the $\mathbf{v}$ operator. The application is described below.

Synchronous Invocation. A synchronous invocation of a service contains an outgoing message to the service as well as an incoming response:

$$\Box = \mathbf{v}c(\overline{w}\langle c\rangle.c.\tau) \ .$$

A service that is bound to $w \in \mathcal{N}_\mathcal{S}$ is invoked by $\overline{w}$ with a fresh name that acts as a correlation identifier as well as a response channel. The process holds until the response is signaled over the name c. Thereafter, it continues as τ. The whole placeholder process part $\Box$ can then replace a τ in a process definition to represent a service invocation, e.g. $N3$ from the previous section is extended to $N3 = e2.((\mathbf{v}c)\overline{w}\langle c\rangle.c.\tau_{N3}).\overline{e4}.\mathbf{0}$.

Asynchronous Invocation. The asynchronous invocation of a service is achieved by separating the transmission and the receipt of an invocation into different processes. A placeholder $\Box_1$ is defined by $\overline{w}\langle c\rangle.\tau$ and a second one as $\Box_2 = c.\tau$. The two placeholders can be placed in two different processes which must share the fresh name c.

By reconsidering the service invocation descriptions, it shows that all service invocation in the π-calculus is indeed asynchronous. This is essential, as the name of the response channel must be transmitted to the service in order to work. The notation of synchronous invocation simply means that there is no additional activity between the invocation and response.

3.2 Providing Services

The π-calculus provides an easy notation for describing state preserving services, where the correlation is bound to a unique π-calculus name. A π-calculus process that acts as a service is defined by:

$$SERVICE = !w(c).\cdots.\bar{c}.\mathbf{0} \ .$$

Each time a request for $SERVICE$ is received via the global name w, the service is replicated (comparable to instantiated). The service can be as complex as required (represented by $\cdots$). After all computation has been done, the response is sent. As service invocations in the π-calculus are generally asynchronous, there is no differentiation between a synchronous and asynchronous service. The service always requires a response channel to work.

Interestingly, by using a unique name for the correlation as well as response channel, the service also holds the state between complex operations. As the name is unique and bound to the invoker, there can be no confusion. An example is a service that can be invoked again with the unique name, those interacting in a complex choreography:

$$SERVICE = !w(c).\cdots.\overline{c}.c.\cdots.\overline{c}.\cdots \ .$$

When $SERVICE$ receives an invocation at the unique name c, it knows exactly which replication instance is triggered, if that instance exists. As only the invoker has access to c, no confusion is possible. A concrete, exemplary client/service choreography can be denoted as follows:

$$\begin{aligned} SERVICE &= !w(c).\tau_{S1}.\overline{c}.c.c.\tau_{S2}.\overline{c}.\mathbf{0} \\ \square_{CLIENT} &= \mathbf{v}c(\overline{w}\langle c\rangle.c.\tau_{C1}.\overline{c}.\tau_{C2}.\overline{c}.c.\tau_{C3}) \ . \end{aligned}$$

The client signals an initial invocation to $SERVICE$ with a fresh name c by $\overline{w}\langle c\rangle$. $SERVICE$ then spawns of a new instance by replication which does some internal computation represented by τ_{S1} and afterward replies on the channel c. The client in response computes something denoted by τ_{C1} and sends two requests over the name c. The service then executes τ_{S2} and responds again on the channel c. After the client received the last response, it does some internal computation τ_{C3} and finishes.

3.3 Supporting Data

One important characteristics of service invocations is the transmission and reception of messages. Messages, or arbitrary data-structures, are also represented by names in the π-calculus. A name thereby holds a reference to a π-calculus process which represents a data structure. For the ease of representation in this paper, we simply denote the type of the data structure with a colon and an XML-type. For instance, $mess : string$ denotes a name $mess$ that references a process that contains a data structure for text strings. By using the polyadic extension of the π-calculus, we can group different names and transmit or receive them with one output or input prefix. An example that invokes a service at the name $s \in \mathcal{N}_S$ with a parameter of the type string and receives a response containing a double as well as a date is denoted as:

$$\square = \mathbf{v}c(\overline{s}\langle c, request : string\rangle.c(rate : double, validuntil : date).\tau \ ).$$

4 π-calculus Orchestration Refinements by Example

Now, enough background is given to show the mapping of a more complex example. Again, the process is visualized with the BPMN (figure 2). To make the

example more interesting, the modeled process consists of several workflow patterns, namely sequence, exclusive choice, simple merge, deferred choice as well as a new pattern called *Event-based Rerouting*. The example describes a process orchestration with web service interaction, both synchronous and asynchronous.

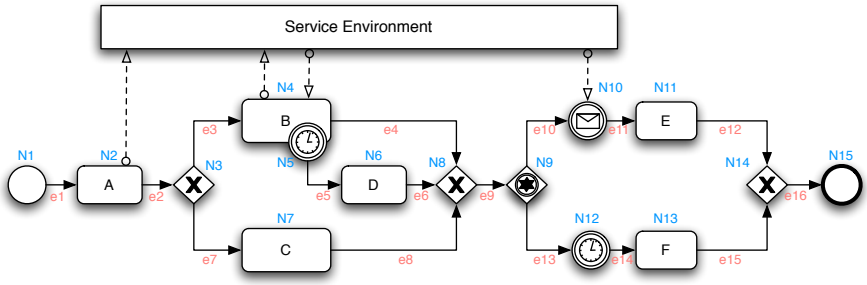


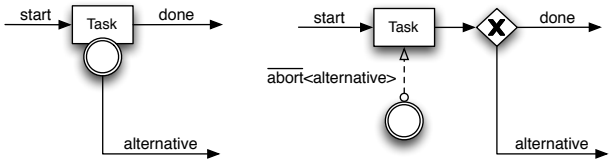
Fig. 2. A BPMN process consisting of several patterns.

The process starts with task A making an asynchronous call to a web service. Thereafter either task B or task C are executed. Task B contains a synchronous web service invocation and has an attached intermediate timer event, which interrupts task B after a certain amount of time has passed and B still has not finished, and continues the execution with task D. After task B, C, or D have been executed, a deferred choice is made. If an answer from the asynchronous call of task A is received before another timeout is reached, task E is executed, otherwise task F.

4.1 Event-based Rerouting

Before the actual mapping of the example process (figure 2), let us first go back to the tasks, as they are modeled in BPMN. As shown in figure 3(a), all tasks have an implicit hook for intermediate events, either directly attached as shown, or logically attached, i.e. via a surrounding sub-process. On occurrence of such an event, the control flow is immediately rerouted (*alternative* instead of *done* in figure 3(a)). This is currently not captured by any workflow pattern, as the rerouting may take place, before the completion of the activity modeled by the task. This is especially true in the SOA context, as web services cannot be canceled easily. Without this pattern, e.g. a timeout for a synchronous web service call can not be modeled. As mentioned above, we propose the name *Event-based Rerouting*.

Looking at the equations 1 and 2, one quickly realize that this immediate rerouting is not supported. Consequently, we need to adapt our equation. Continuing the procedure of expressing each BPMN-node as a π-process, the intermediate event will be expressed by a π-process as well. This event-process



(a) BPMN-Task with event (b) Mapping to π-calculus

Fig. 3. Intermediate events in BPMN and their mapping to π-calculus

can send the task an alternative route via the *abort* name, which is immediately taken. Figure 3(b) tries to visualize how we propose to model a $Task$ with the π-calculus as shown in the following. As before, the $\square$ denotes a placeholder for the actual process description. As $\square$ can simply be expanded, no new semantics are introduced by this abbreviation.

$$\begin{aligned} TASK(start, done, abort, \square) =& start.TASK_{EXECUTE} \\ &+ abort(alternative).\overline{alternative}.\mathbf{0} \end{aligned} \tag{3}$$

Equation 3 defines a task via the process $TASK$, having the names *start*, *done*, and *abort* as parameters, plus the actual implementation denoted by $\square$. $TASK$ waits either for a signal on *start* to start the execution $TASK_{EXECUTE}$ or receiving an *alternative* name on *abort*, which is immediately signaled.

$$TASK_{EXECUTE} = (\mathbf{v}continue)(TASK_{ABORT} \mid TASK_{IMPL}) \tag{4}$$

$TASK_{EXECUTE}$ (equation 4) defines a private name *continue* and executes $TASK_{ABORT}$ and $TASK_{IMPL}$ in parallel.

$$\begin{aligned} TASK_{ABORT} =& abort(alternative).\overline{alternative}.\overline{continue}\langle\bot\rangle.\mathbf{0} \\ &+ \overline{continue}\langle\top\rangle.\mathbf{0} \end{aligned} \tag{5}$$

$TASK_{ABORT}$ (equation 5) tries to receive an *alternative* name over *abort* to signal $\overline{alternative}$ and send $\bot$ via $\overline{continue}$. Alternatively, $TASK_{ABORT}$ sends $\top$ via $\overline{continue}$. The reduction semantics of the π-calculus guarantees that the decision can not be made until either *continue* is read by $TASK_{IMPL}$ (see below) or $\overline{abort}$ is written by a different process, i.e. an event node.

$$TASK_{IMPL} = \square.continue(flag).([flag = \top]\overline{done}).\mathbf{0} \tag{6}$$

$TASK_{IMPL}$ (equation 6) first executes the activity represented by $\square$. It then reads *continue*. If no read on *abort* occurred by this time, $TASK_{ABORT}$ is now

able to send $\top$ over $\overline{continue}$, otherwise a $\bot$ is transmitted. A $\top$ represents normal execution, so iff $\top$ is read, $\overline{done}$ is written. Here, $\bot$ represents the occurrence of an intermediate event, which is already handled by that time, thus the process simply ends.

4.2 Mapping to π-calculus

Now, let us start the mapping of the example process (figure 2) to π-calculus. Again, our approach is to model each BPMN-node as an individual process.

First, the gateways:

$$\begin{aligned} N3 =& e2.\tau_{N3}.([c_{e3} = \top]\overline{e3} \mid [c_{e7} = \top]\overline{e7}) \\ N8 =& \mathbf{v}x(e4.\overline{x}.\mathbf{0} \mid e6.\overline{x}.\mathbf{0} \mid e8.\overline{x}.\mathbf{0} \mid x.\overline{e9}.\mathbf{0}) \\ N14 =& \mathbf{v}x(e12.\overline{x}.\mathbf{0} \mid e15.\overline{x}.\mathbf{0} \mid x.\overline{e16}.\mathbf{0}) \end{aligned}$$

All gateways, except for the deferred choice, are expressed with equation 2, just as before.

Next, the trivial tasks:

$$\begin{aligned} N6 =& TASK(e5, e6, env_{ABORT}, \tau_{N6}) \\ N7 =& TASK(e7, e8, env_{ABORT}, \tau_{N7}) \\ N11 =& TASK(e13, e14, env_{ABORT}, \tau_{N11}) \\ N13 =& TASK(e14, e15, env_{ABORT}, \tau_{N13}) \end{aligned}$$

The trivial tasks ($N6, N7, N11, N13$) are represented by our new $TASK$ process, abstractly executing a τ. The name env_{ABORT} represents a name provided by the environment to signal a global abort, e.g. the shutdown of the workflow engine.

As we just introduced the concept of environmental names, let us now look at the start- and end-event of the process:

$$\begin{aligned} N1 =& TASK(env_{START}, e1, env_{ABORT}, \tau_{N1}) \\ N15 =& TASK(e16, env_{DONE}, env_{ABORT}, \tau_{N15}) \end{aligned}$$

These are trivial tasks as well, but interact with the environment via env_{START} and env_{DONE}. These names provided the integration to a workflow engine to signal the start of the execution as well as the completion between the process and the workflow engine.

Next, the asynchronous web service invocation in $N2$:

$$N2 = TASK(e1, e2, env_{ABORT}, \overline{w_{req1}}\langle w_{resp1}\rangle.\tau_{N2})$$

The implementation follows the explanation from section 3.1. The web service is called via $\overline{w_{req1}}$ and the unique response name w_{resp1} is passed along. All of this is encapsulated within a $TASK$ process.

$N4$ is a synchronous web service invocation with an attached intermediate timeout event $N5$:

$$N4 = TASK(e3, e4, abort_{N5}, \overline{w_{req2}}\langle w_{resp2}\rangle.w_{resp2}.\tau_{N4}).\mathbf{0}$$
$$N5 = env_{TIMEOUTN5}.\overline{abort_{N5}}\langle e5\rangle$$

As before, the response name w_{resp2} is passed along the service call. The only difference to $N2$ is the immediate read on the response name. Also, note that the abort-name is different than before to allow for a communication between $N5$ and $N4$ in case the timeout gets triggered. The actual timeout is once again triggered by the environment ($env_{TIMEOUTN5}$). This is an application of the *Event-based Rerouting* pattern described before.

Last but not least, the deferred choice $N9$:

$$CHOICE_{N9,N10,N12} = e9.(w_{resp2}.\overline{e11}.\mathbf{0} + env_{TIMEOUTN12}.\overline{e14}.\mathbf{0})$$

Notice, that the nodes $N10$ and $N12$ are not modeled explicitly, but as part of the $CHOICE_{N9,N10,N12}$ process, hence the name. Again, we model the actual timeout as a signal by the environment. The simplicity of the equation is yet another good example of the expressiveness of the π-calculus.

5 Conclusion

In this paper, we have sketched how the π-calculus can be used in the service–oriented domain. Starting from our work on workflow pattern in π-calculus, the representation, orchestration, and choreography of services has been discussed. Interestingly, the π-calculus concept of mobility, which is based on communicating names that can be used as interaction channels, proved to be very useful to formally represent unique correlations. Furthermore, we introduced formal representations for service invocation, both for the client and the provider. In section 4 the orchestration of services was refined by introducing a formal representation of a pattern known from notations like BPMN or BPEL. We named it *Event-based Rerouting* pattern and introduced the precise semantics of a task containing it. Finally, the formalization of an example containing the new pattern as well as another common pattern in service–oriented orchestrations, a deferred choice modeled with an event–based gateway in the BPMN, has been discussed.

Related Work. Lucas Bordeaux and Gwen Salaün wrote a survey about using process algebra for web services [5]. They argued that the formal reasoning capabilities of process algebras are well suited in the service–oriented domain. Especially (relaxed) equivalence properties for services participating in a choreography and some kind of soundness for orchestration can be formally proved by using process algebra. They also concluded that the name passing notation of the π-calculus is definitely of interest in the context of web services.

Further discussions about behavioral compatibility of web–services can be found in [13]. L.G. Meredith and Steve Bjorg wrote a journal article about using mobile process algebra in the context of formal service descriptions where they emphasized the use of behavioral types as a new kind of service discovery mechanisms [6]. There have also been investigations on extending the π-calculus for representing and reasoning about long–running transactions in component–based distributed applications like web–service platforms [14]. A more practical approach of using CCS [15] to formalize web service choreography can be found in [16].

References

1. BEA Systems, IBM, Microsoft, SAP, Siebel Systems: Business Process Execution Language for Web Services Version 1.1. (2003)
2. Puhlmann, F., Weske, M.: Using the Pi-Calculus for Formalizing Workflow Patterns. In van der Aalst, W., Benatallah, B., Casati, F., eds.: BPM 2005, volume 3649 of LNCS, Berlin, Springer-Verlag (2005) 153–168
3. van der Aalst, W., van Hee, K.: Workflow Management. MIT Press (2002)
4. van der Aalst, W.M.P., ter Hofstede, A.H.M.: YAWL: Yet Another Workflow Language (Revised version. Technical Report FIT-TR-2003-04, Queensland University of Technology, Brisbane (2003)
5. Bordeaux, L., Salaün, G.: Using Process Algebra for Web Services: Early Results and Perspectives. In Shan, M.C., Dayal, U., Hsu, M., eds.: TES 2004, volume 3324 of LNCS, Berlin, Springer-Verlag (2005) 54–68
6. Meredith, L., Bjorg, S.: Contracts and Types. Communications of the ACM **46** (2003) 41–47
7. van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B., Barros, A.: Workflow patterns. Distributed and Parallel Databases **14** (2003) 5–51
8. Milner, R., Parrow, J., Walker, D.: A calculus of mobile processes, Part I/II. Information and Computation **100** (1992) 1–77
9. Milner, R.: The polyadic π–Calculus: A tutorial. In Bauer, F.L., Brauer, W., Schwichtenberg, H., eds.: Logic and Algebra of Specification, Berlin, Springer-Verlag (1993) 203–246
10. Milner, R.: Communicating and Mobile Systems: The π-calculus. Cambridge University Press, Cambridge (1999)
11. Sangiorgi, D., Walker, D.: The π-calculus: A Theory of Mobile Processes. Paperback edn. Cambridge University Press, Cambridge (2003)
12. Hohpe, G., Woolf, B.: Enterprise Integration Patterns. Addison-Wesley, Boston (2003)
13. Bordeaux, L., Salaün, G., Berardi, D., Mecella, M.: When are Two Web Services Compatible? In Shan, M.C., Dayal, U., Hsu, M., eds.: TES 2004, volume 3324 of LNCS, Berlin, Springer-Verlag (2005) 15–28
14. Bocchi, L., Laneve, C., Zavattaro, G.: A Calculus for Long–Running Transactions. In Najm, E., Nestmann, U., Stevens, P., eds.: FMOODS 2003, volume 2884 of LNCS, Berlin, Springer-Verlag (2003) 124–138
15. Milner, R.: Communication and Concurrency. Prentice Hall, New York (1989)
16. Brogi, A., Canal, C., E.Pimentel, Vallecillo, A.: Formalizing Web Service Choreographies. In: Proceedings of First International Workshop on Web Services and Formal Methods. Electronic Notes in Theoretical Computer Science, Elsevier (2004)

Issues of Access Control in Cross-organizational Workflows

Andreas Wombacher

University of Twente,
7500 AE Enschede, The Netherlands
A.Wombacher@utwente.nl

Abstract. Service Oriented Architectures facilitate the autonomous design of state dependent services, where the behavior can be represented as a workflow. Service composition constructs cross-organizational workflows in a decentralized way, where the service providers have limited knowledge about the trading partner's usage of data provided to them. Due to the decentralized nature of composite services the control and enforcement of data must also be performed in a decentralized way, that is, control rules assigned by one party must be enforced by an environment which is out of the control of the party itself. This paper investigates access control in decentralized cross-organizational workflows an a first approach is sketched.

1 Introduction

Service Oriented Architectures, and as a concrete technology of it Web Services, are getting more and more popular. In particular, a service is a functional component, which is described by a service interface and hosted and maintained by a service provider. The functional components in current Web Service implementation examples are mainly stateless, thus represent a single request and response of the component. However, more realistic functional components turn out to be state dependent, that is, require several request and response interactions. Thus, state dependent services expect a certain behavior of their service requester and vice versa. Therefore, the behavior of a service itself and the service requester can be represented as a workflow and the invocation of a service represents a bilateral cross-organizational workflow. The scenario evolves to a multi-lateral cross-organizational workflow in case the service requester invokes more than a single service or the invoked service itself invokes further services.

A special characteristic of Service Oriented Architectures is that service requesters and service providers are acting autonomously, that is, they are not sharing a common authority. As a consequence, the services are developed and maintained by the service provider autonomously, which also includes the behavior of the services. Therefore, also the corresponding workflows are developed and maintained autonomously, which requires a checking of cross-organizational workflow properties in a decentralized way. Potential properties are the syntactical and semantical compliance of used messages, the deadlock-freeness of the cross-organizational workflow, or the compliance of Quality of Service and security properties. In the following we will focus on security properties,

because the lack of security support is a major draw-back for the composition of state dependent services. In particular, there are data and task related security properties. The data related security properties constrain the access to the data provided to a trading partner with regard to users and purposes. The task related security properties expresses constraints on who is allowed to perform a certain task. In either case the aim is to keep as much control/knowledge about the cross-organizational workflow as possible. A classical means to express such constraints is access control which will be used in the following.

Cross-organizational workflows are well known in the workflow community and there exist approaches for access control. However, these approaches are based on a top-down design of the workflows, where the specification of the access control is done based on the knowledge of the complete cross-organizational workflow model by applying access control models like e.g. task based access control [1–4]. The cross-organizational workflow model is then split into public workflows representing the part of the cross-organizational workflow which is performed by an individual party. Public workflows are further specialized into private workflows containing the integration of back-end systems. However, the assigned access control policies remain unchanged. Public workflows are an abstraction of private workflows, where internal activities and mission critical information is omitted. The decentralized execution of the cross-organizational workflow, finally, does not require a centralized access control enforcement, but can be realized in a decentralized way (like e.g. [5]).

Opposed to the top-down design, the Service Oriented Architecture supports a bottom-up design, where a cross-organizational workflow is established from services implementing public and private workflows. The cross-organizational workflow can be constructed by combining the public workflows. However, in case there is no centralized coordinator in the cross-organizational workflow the workflow model itself is never instantiated. Therefore, the cross-organizational workflow property checking can not rely on the knowledge of the cross-organizational workflow but requires a decision making based on the public workflows and their corresponding security policies. The aim is to investigate which security properties of a cross-organizational workflow can be decided in a decentralized way. In this paper, we investigate a task based and a data based approach of explicating access control constraints, which turn out not to be applicable in a bottom-up approach. Further, we propose to use a combined approach and illustrate issues, which will be addressed in future work.

We continue with a discussion of related work in Section 2, describe an example in Section 3, and discusses problems with cross-organizational access control in Section 4. In Section 5 the approach is sketched and some issues are presented. The paper is summarized in Section 6.

2 Related Work

Workflows can be classified in accordance to [6] as communication based and task based workflow models. A communication based workflow model provides a description of the effects of receiving messages on an internal state as well as the potential ordering of messages exchanged with communication partners. The theoretical foun-

dation of these approaches is Speech Act Theory [7, 8], which has e.g. been applied in agent communication languages [9]. The extension of an agent approach to cross-organizational workflows is straight forward. However, due to the lack of a control flow, the design of a workflow is quite challenging because at every state in the workflow all potential messages have to be considered as a potentially following action. Further, the specification is quite hard to read and understand compared to task based approaches. Therefore, we further stick to the task based approaches.

A task based workflow aims to separate control logic and logic contained in the tasks, where a task is either performed by an information system or by a human [10]. Example task based workflow models are Finite State Automata (FSA) [11], Place/ Transition Nets (P/T-Nets) [12–14], Coloured Place/Transition Nets (CP/T-Nets) [15], Workflow Nets (WF-Nets) [16], message sequence charts [17], statecharts [18, 19], or flowcharts [20, 21]. The construction of cross-organizational workflows based on these workflow models can be further classified into capacity sharing, chained execution, subcontracting, and loosely coupled interactions [22]. The workflows derived from a Service Oriented Architecture are considered to be loosely coupled interactions and there exist approaches to decide deadlock freeness of these workflows in a more or less centralized way [23–26]. Since Finite State Automata are the simplest formalization and a decentralized decision making has been proposed for them, we stick to it in the following.

Access control is the mechanism by which users are granted access to resources, operations, or data within computer systems, according to their identity (established by authentication) and associated privileges (established by authorization). Access control has been studied extensively and applied to all kind of research domains, like e.g. data bases, cooperative environments, and workflows. In particular, access control is a generic concept which can be applied to data as well as to tasks. Mainly it grants permissions to a user via an assignment of roles to improve the maintainability of the access control, like e.g. in role based access control [27–29]. More complex models support the enabling and disabling of permissions by relating permissions to a purpose, like e.g. in task based access control [3, 4].

In addition, access control can be used to design specific security policies like e.g. separation of duties [30, 31], but this is done based on the knowledge of the underlying workflows [28] or a commonly agreed structure of roles and permissions [32, 33]. We are not aware of an investigation of security properties based on decentralized maintained access control policies.

3 Example

An example of a cross-organizational workflow is depicted in Figure 1. The representation is based on Finite State Automata notation, where states are represented as circles, state changes, that is, transitions are represented as arcs connecting states being labeled with the message exchanged during a state change. The message has the structure of $S\#R\#msg\_name$ where S represents the sender and R the receiver of a message, while msg_name is the name of the message. The execution of a workflow is initiated

by a start state represented by a circle with a small incoming arrow and is completed in a final state represented by a circle with a thick line.

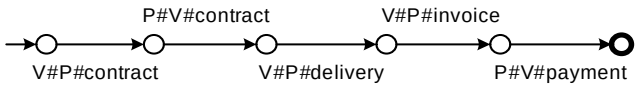


Fig. 1. Example cross-organizational workflow

The depicted cross-organizational workflow describes a simple buying process, where a Vendor V and a Purchasing Department P are first signing a contract followed by the Vendor delivering the good and an invoice to the Purchasing Department, which finally pays the Vendor.

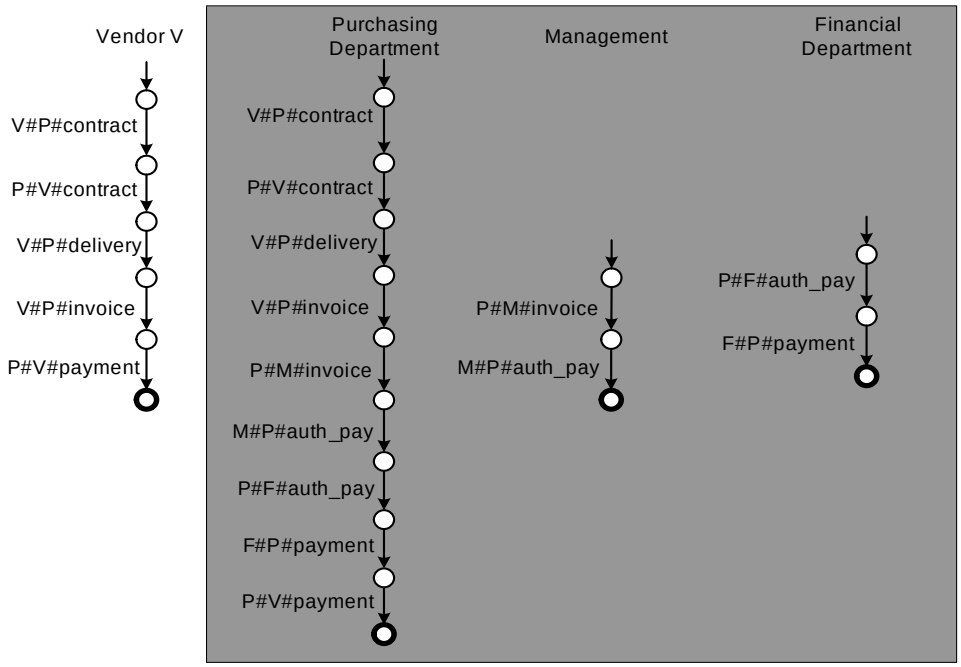


Fig. 2. Example public workflow

This cross-organizational workflow can be split into several public workflows as depicted in Figure 2. In particular, the Purchasing Departments public process covers not only the direct interaction with the Vendor, but also represents the interaction with the

Management M being required for approval of the payment and the Financial Department F performing the payment. Be aware, that the public process of the Purchasing Department implements a separation of duties on the approval and the performing of the payment. The Purchasing Department, the Management, and the Financial Department are all belonging to a single legal entity each representing a department. In Figure 2 this is indicated by the gray box clustering the three parties.

The purchasing process starts with the Purchasing Department receiving a $contract$ message sent by the Vendor, which has to be signed and returned by the Purchasing Department. Afterwards the Vendor delivers the goods ($delivery$ message) and the invoice ($invoice$ message) to the Purchasing Department. The Purchasing Department then sends the invoice to the Management asking for an approval of the payment ($auth\_pay$ message), which is then forwarded to the Financial Department to perform the payment, where the payment confirmation statement ($payment$ message) is first send to the Purchasing Department, which forwards it to the Vendor.

4 Basic forms of access control

In the following, we will investigate the two extremes of applying access control to tasks and data respectively. Potential issues are illustrated based on the example introduced in the previous section.

4.1 Tasks

As stated above a cross-organizational workflow is based on a set of public workflows derived from the involved party's private workflows. Each private workflow is modeled by a task based workflow model, that is, applied to Finite State Automata the states in an automaton. Since the coordination of the private workflows in accordance to the cross-organizational workflow is done by exchanging messages, tasks are uniquely assigned to a single private workflow and no task is shared with several private workflows.

Further, access control constraints are specified for each task in a private workflow implementing a local security policy. For example in a role based access control [27], the constraints represent which user can take over which role (like e.g. administrator, manager, or employee), and which role is allowed to perform which action (like e.g. monitor or process) on a specific task. Local security policies are set up and maintained locally and it is quite unlikely that a party accepts requests for changing local security policies from external parties. As a consequence, the cross-organizational workflow security policy is a combination of the local policies without providing potential for negotiations.

Let's assume that local policies are implemented correctly and working properly for all private workflows, then due to the fact that tasks are not shared by several private workflows the local security policies can also be applied to the cross-organizational workflow. Since the tasks are considered to be disjoint there is no potential of conflicting local policies, thus the cross-organizational policy can be considered to be conflict-free, if the users of each private workflow are considered to be disjoint, that is, there is no user who belongs to two organizations.

In case this disjointness is not valid, local security policies could be violated in cross-organizational workflows. This can happen in case of a single user is involved in more than one private workflow of the cross-organizational workflow, e.g., a service provider contributing more then one service/private workflow. However, also in case each service/private workflow contributing to the cross-organizational workflow is realized by a different service provider it could still be that a single person may represent a user for each of the service providers again violating the disjointness criteria. Thus, a major issue with regard to checking cross-organizational security properties is on checking the disjointness of users, service providers, and legal entities being involved in private workflows.

Applying this issue to the example depicted in Figure 2 the cross-organizational workflow will provide the intended separation of duties on authorizing a payment and performing the payment as long as the Management and the Financial Department workflow are performed by different individuals. However, checking on individuals in a Service Oriented Architecture is nearly impossible, thus we should focus on checking whether they are realized by different service providers. This check can easily be achieved by publishing the service providers during the decentralized establishment of the cross-organizational workflow as e.g. proposed in [34].

4.2 Data

In a cross-organizational workflow the private workflows are coordinated by exchanging messages, thus, an exchange of data. Access control, like e.g. Role Based Access Control, can be applied to data where it specifies which user is assigned to which role (like e.g. administrator, manager, or employee), and which role can perform which action (like e.g. read, write) on the data. Since a party is providing data to another party for coordination purposes, the party originating or owning the data wants to restrict the usage and distribution of the data within the cross-organizational workflow and beyond this. As a consequence, the owner has to state explicitly constraints on the access of these data, which is passed on together with the data. This approach is known from distributing digital content [35] and has been applied e.g. in organizational studies under the notion of delegation of rights [32, 33].

However, to express the delegation of access rights a common understanding of the roles and the permissions as a basis of the cross-organizational policy specification is required. Further, the delegation approach is independent of the control flow of the cross-organizational workflows. Thus, maintaining the task list of a user the required permissions on data used in the task must be collected and must be compared with the permissions delegated/granted by the owner of these data. In case the permissions are not covered by the delegation the task can not be executed. The issue here is the required commonly agreed model of roles and permissions, which is contradicting the autonomous creation and maintenance of services.

In Figure 3 we elaborate on the interaction of the Purchasing Department and the Management public processes. In particular, we extended messages by permissions granted for a role on data annotated in square brackets. The Purchasing Department uses the roles *P* for a member of a Purchasing Department and *ext* representing external

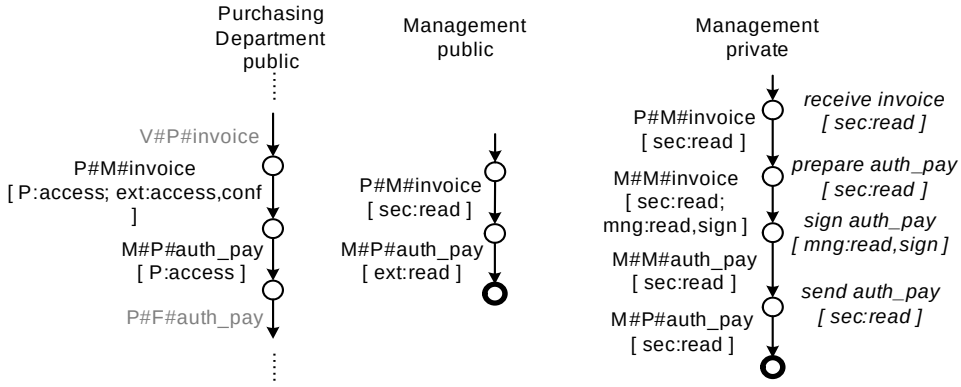


Fig. 3. Example private workflow

roles. The corresponding permissions are *access* for accessing data and *conf* representing a confirmation of data. The Management uses the roles *sec* for secretary and *mng* for manager, and the permissions *read* and *sign*. Be aware that the two local policies are based on different role models and use different permissions. Although, there is a quite obvious mapping of permissions for a human reader, an automated mapping of permissions is hard to achieve. As a consequence, the pure data driven approach is not applicable in this example because the Management is neither able to interpret the role *ext* nor the permissions *access* and *conf*.

5 Approach

Based on the investigation performed in the previous section, the outcome is that

- an access control purely on tasks does not support the prevention of unintended usage of data exchanged for coordination purposes, and that
- an access control purely based on data is impractical due to the requirement of having a commonly agreed understanding of roles and permissions.

Thus, the aim must be to come up with a form of access control, which provides less guarantees but is applicable in the addressed cross-organizational scenario. Hence, this kind of access control can not prevent the unintended use of data, but allows to reduce the risk of such fraudulent usage by checking properties of the cross-organizational workflow in a decentralized way. However, a certain level of trust between the trading partners is required, which is realistic at least in business scenarios, where the aim is to perform not only a single transaction but a lot of them, thus, a good business relationship is required.

We propose to base the approach on a mixture of the above mentioned two general approaches. Each private workflow has its local security policy constraining the access to data and tasks. Since the coordination of private workflows in a cross-organizational

workflow is based on coordinating tasks by exchanging messages (that is data), the message and its corresponding access rights are contained in the public and private workflows respectively. As a consequence the local policies of the parties share the point of the sent and received message relating the two policies to each other. That means in an e.g. role based access control model that the permissions of a role on the data specified in the sending public workflow can be related to the permissions of a role on the same data specified in the receiving public workflow. Be aware that there is no common understanding of all permissions and roles required, but that a shared understanding is required of only the particular roles and permissions relevant for this particular message. Thus, in the sending case certain permissions are granted, while in the receiving case certain permissions are requested. As a consequence, the direction of the message exchange influences the mapping, i.e., the mapping is not necessarily symmetric.

Based on this relation with regard to a single message we learn something of how the different policies are inter related by considering the relations derived for the different messages used for coordination. In particular, if the relation turns out to be bijective there is a high probability that the policies of the trading partners are based on the same level of granularity. Further, in case the relation is symmetric with regard to sending and receiving of messages this indicates a high probability that the bijectively assigned permissions have similar meaning under the assumption of the least privilege principle [36].

Besides of these first observations a more thorough investigation and a formalization of the approach is required, which will be performed in future work. During this investigation the following issues have to be investigated in more detail:

Issue 1 *What can be derived from the bilateral relations?*

In case the relation is bijective there is a high probability that the local security policies are based on the same level of granularity. Besides of this optimal case the local security policies can be based on different levels of granularity resulting in other properties of the relation. This difference in granularity introduces a lack of differentiation of how to deal with data and provides less specific understanding on how a trading partner is dealing with the provided data. From the preliminary investigation, the worst case is a many to many relation preventing any statement on the security policies applied on handling the data.

Figure 3 depicts public workflows of the Purchasing Department and the Management. The relations established there indicate that for the *invoice* message the Purchasing Department grants the Management access and confirmation permissions expressed as $[ext : access, conf]$, while the Management public workflow indicates the need of read permissions for the secretary indicated by $[sec : read]$. With regard to the $auth\_pay$ message the Management grants read permissions ($[ext : read]$) to the Purchasing Department, which requires access permissions for the Purchasing Department ($[P : access]$). As a consequence, the relation maps *access* and *conf* representing Purchasing Department's permissions to *read* of the Management. As a consequence, the relation is a one to many relation, because there is no differentiation of *access* and *conf* permissions on the Management side. Changing the public workflow of the Management as depicted in Figure 4 results in a mapping of *access* and *conf* of the Purchasing

Department to *read* and *sign* of the Management provided by the management role mng. Thus, the mapping turns into a bijective mapping representing similar granularity of the local policies. The public Management workflow used in this case could be constructed from the private Management workflow (see Figure 3) by collecting all the permissions required to access the data after receiving it. Thus, the second issue addresses the information contained in the public workflow.

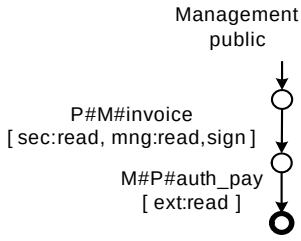


Fig. 4. Example Mangement public workflow

Issue 2 *Which security policy information of the private workflow must be contained in the public workflow?*

In general, a private workflow represents the concrete specification of a process including branching conditions, internal tasks, etc., which are not revealed to trading partners since it contains mission critical information. As a consequence, an abstraction of the private workflow, the public workflow, is provided to trading partners. With regard to security policies, the above example illustrates the difference on which access rights could be provided in a public workflow derived from the same private workfow. The public Management workflows depicted in Figure 3 and 4 provide different security policy information. However, the less information is provided the less specific the analysis of the relations will be and the more information is provided the more mission critical information might be revealed. The issue is to find a good trade off and a formal method to derive the public workflow as an abstraction of the private workflow including the local security policy.

Based on the sketched approach it is possible to derive some properties of the mapping of local security policies in a bilateral case. However, we are investigating multilateral scenarios and it is known from distributed systems that bilateral properties are hardly sufficient to derive multi-lateral properties. Thus, future work will address how to extend bilateral properties to derive properties of the cross-organizational workflow.

Issue 3 *Which cross-organizational workflow security properties can be derived?*

In [26] the principle of propagation of additional information affecting bilateral properties has been applied to derive multi-lateral properties based on bilateral ones. A similar approach called gossiping has been applied in [37] for investigating semantic

mappings, which is quite comparable to the mappings of permissions as discussed in this paper. We expect to get reasonable results by applying these approaches to decide cross-organizational security properties, that is, decide whether data is likely to be used in a fraudulent way in a cross-organizational workflow.

6 Conclusion and Future Work

In this paper we discussed the access control in a cross-organizational workflow by sketching an approach and raising several issues. In particular, we propose to bilaterally relate autonomous, local security policies and derive from these mappings information on the local security policy's granularity. The quality of the derivable properties mainly depends on the information provided in the local security policy. Finally, the issue of extending local properties to cross-organizational properties has been raised and two directions are indicated. Future work will investigate the identified issues.

References

1. R. K. Thomas and R. S. Sandhu. Towards a task-based paradigm for flexible and adaptable access control in distributed applications. In *NSPW '92-93: Proceedings on the 1992-1993 workshop on New security paradigms*, pages 138–142. ACM Press, 1993.
2. Ravi S. Sandhu and Roshan K. Thomas. Conceptual foundations for a model of task-based authorizations. In *Proceedings of the IEEE Computer Security Foundations Workshop (CSFW)*, pages 66–79. IEEE Computer Society, 1994.
3. Gaby Herrmann and Guenther Pernul. Towards security semantics in workflow management. In *HICSS '98: Proceedings of the Thirty-First Annual Hawaii International Conference on System Sciences-Volume 7*, page 766. IEEE Computer Society, 1998.
4. Sejong Oh and Seog Park. Task-role-based access control model. *Information Systems*, 28(6):533–562, 2003.
5. Myong H. Kang, Joon S. Park, and Judith N. Froscher. Access control mechanisms for inter-organizational workflow. In *SACMAT '01: Proceedings of the sixth ACM symposium on Access control models and technologies*, pages 66–74. ACM Press, 2001.
6. D. Georgakopoulos, M. Hornick, and A. Sheth. An Overview of Workflow Management: From Process Modelling to Workflow Automation Infrastructure. *Distributed and Parallel Databases*, 3(2):119–153, April 1995.
7. J. L. Austin. *How to do things with words*. Oxford University Press, 1965.
8. Kent Bach and Robert M. Harnish. *Linguistic Communication and Speech Acts*. The MIT Press, Cambridge, Massachusetts, 1979.
9. FIPA. Foundation for intelligent pysical agents. FIPA spefcifications. http://www.fipa.org, 2004.
10. C. Mohan. Workflow management in the internet age. In Witold Litwin, Tadeusz Morzy, and Gottfried Vossen, editors, *Proceedings of the Second East European Symposium on Advances in Databases and Information Systems (ADBIS)*, pages 26–34. Springer LNCS 1475, Sept 1998.
11. J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison Wesley, 2001.
12. C. A. Petri. *Kommunikation mit Automaten*. Schriften des Institutes fur Instrumentelle Mathematik, Bonn, 1962.

13. James L. Peterson. *Petri Net Theory and the Modeling of Systems*. Prentice-Hall, 1981.
14. Hartmann J. Genrich. Predicate/Transition Nets. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Petri Nets: Central Models and Their Properties, Advances in Petri Nets 1986 Part I*, volume 254 of *Lecture Notes in Computer Science*, pages 207–247. Springer-Verlag, Berlin, Germany, 1987.
15. K. Jensen. *Coloured Petri Nets*. Springer Verlag, Heidelberg, 1992.
16. W.M.P. van der Aalst and Kees van Hee. *Workflow Management - Models, Methods, and Systems*. MIT Press, 2002.
17. International Telecommunication Union. Message sequence charts. ITU-T Recommendation Z.120, Nov 1999.
18. D. Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8(3):231–274, June 1987.
19. D. Harel and A. Naamad. The STATEMATE semantics of statecharts. *ACM Transactions on Software Engineering and Methodology*, 5(4):293–333, 1996.
20. P. Grefen, K. Aberer, Y. Hoffner, and H. Ludwig. CrossFlow: Cross-organizational workflow management in dynamic virtual enterprises. *International Journal of Computer Systems Science & Engineering*, 15(5):277–290, Sep. 2000.
21. J. Klingemann, J. Wäsch, and K. Aberer. Adaptive outsourcing in cross-organizational workflows. In *Proceedings of the the 11th Conference on Advanced Information Systems Engineering (CAiSE)*, pages 417–421, Heidelberg, Germany, June 1999.
22. W.M.P. van der Aalst. Interorganizational workflows: An approach based on message sequence charts and petri nets. *Systems Analysis - Modelling - Simulation*, 34(3):335–367, 1999.
23. Ekkart Kindler, Axel Martens, and Wolfgang Reisig. Inter-operability of workflow applications: Local criteria for global soundness. In *Business Process Management, Models, Techniques, and Empirical Studies*, pages 235–253. Springer-Verlag, 2000.
24. Xiang Fu, Telfik Bultan, and Jianwen Su. Realizability of conversation protocols with message contents. In *Proceedings IEEE International Conference on Web Services (ICWS)*, pages 96–103. IEEE Computer Society, 2004.
25. W.M.P. van der Aalst. Inheritance of Interorganizational Workflows to Enable Business-to-Business E-commerce. *Electronic Commerce Research*, 2(3):195–231, 2002.
26. Andreas Wombacher, Peter Fankhauser, and Karl Aberer. Overview on decentralized establishment of consistent multi-lateral collaborations. In *Proc. of Intl. Conf. on e-Technology, e-Commerce and e-Service (EEE)*, pages 164–170, 2005.
27. Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, 1996.
28. Jacques Wainer, Paulo Barthelmess, and Akhil Kumar. W-RBAC - a workflow security model incorporating controlled overriding of constraints. *International Journal of Cooperative Infnformation Systems*, 12(4):455–485, 2003.
29. Shengli Wu, Amit Sheth, John Miller, and Zongwei Luo. Authorization and access control of application data in workflow systems. *J. Intell. Inf. Syst.*, 18(1):71–94, 2002.
30. D.F.C. Brewer and M.J. Nash. The chinese wall security policy. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 206–214, 1989.
31. Elisa Bertino, Elena Ferrari, and Vijay Atluri. The specification and enforcement of authorization constraints in workflow management systems. *ACM Trans. Inf. Syst. Secur.*, 2(1):65–104, 1999.
32. Andreas Schaad and Jonathan D. Moffett. A framework for organisational control principles. In *Proceedings of the 18th Annual Computer Security Applications Conference (ACSAC)*, page 229, Washington, DC, USA, 2002. IEEE Computer Society.

33. Andreas Schaad and Jonathan Moffett. Separation, review and supervision controls in the context of a credit application process: a case study of organisational control principles. In *Proceedings of the 2004 ACM symposium on Applied computing (SAC)*, pages 1380–1384, New York, NY, USA, 2004. ACM Press.
34. Andreas Wombacher. Decentralized decision making protocol for service composition. In *Proceedings IEEE International Conference on Web Services (ICWS)*, pages 203–210. IEEE Computer Society, 2005.
35. C. Gunter, S. Weeks, and A. Wright. Models and languages for digital rights. In *Proceedings of the 34th Hawaii International Conference on System Sciences (HICSS)*, page 9076. IEEE Computer Society, 2001.
36. Ross Anderson. *Security Engineering - A Guide to Building Dependable Distributed Systems*. Wiley Computer Publishing, 2001.
37. Karl Aberer, Philippe Cudre-Mauroux, and Manfred Hauswirth. Semantic gossiping: Fostering semantic interoperability in peer data management systems. *J. Semantic Web and Peer-to-Peer*, 2005. to be published.

Developing Semantic Web Services for Water Monitoring – A Case Study[1]

Aabhas V. Paliwal and Nabil Adam

CIMIC, RUTGERS University,
07102, Newark, NJ, USA
aabhas@cimic.rutgers.edu
adam@adam.rutgers.edu
http://cimic.rutgers.edu

Abstract. Semantics provide an expressive way of describing Web Services using ontologies. In this paper we present the application of semantics to web services implemented in the water monitoring domain. This augmentation provides a more efficient and flexible management capability for the process of water monitoring. The proposed approach, applying service oriented architecture, for a specific case study is generalizable, and at an appropriate level of abstraction is applicable to other domains. As a case study the paper characterizes part of an interdisciplinary research and presents a framework for the monitoring process by providing support for complex problem solving, knowledge modeling and reuse. It presents an approach, to ontologically motivated and semantically rich services for the water-monitoring domain, describes our experiences of designing the framework and developing the prototype, and discusses the merits of this integrated approach.

1 Introduction

The Internet is being transformed into a service-oriented medium. Web-accessible programs, databases, sensors, and a variety of other physical devices comprise these software and hardware oriented services - web services. Semantic markup exploits ontologies to facilitate information sharing, reuse, composition, and mapping [8]. Semantic markup enables listing of the properties and capabilities of Web services for knowledge-based indexing and retrieval of Web services by agent brokers and humans alike [4]. Ontology is a key component of this integration between the domains of semantic web and web services. The ontology, a formal agreed vocabulary whose terms are used in the construction of the semantics and provides a shared conceptual representation, is a pre requisite for this paradigm. It comprises of the classes of entities, relations between the entities, and the axioms related to the entities in the domain [2]. Domain-specific web service ontologies are subclasses of these general classes. They enable an individual service to inherit shared concepts, and vocabulary in a particular domain. Enactment of a process helps in evaluating the performance of the individual services and simulation is done to study the process in action, before en-

[1] This work is partially supported by a grant from the New Jersey Meadowlands Commission

actment. In the other direction, using web services for simulation, simulation models/components can be built out of web services [1]. Well-tested simulation models may be placed on the Web for others to use. Resources and tools used in simulation environments make excellent candidates for Web services.
In this paper we outline our approach for building semantically rich software services and illustrate its application in the water-monitoring domain. These services are based on the integration of ontologically motivated OWL-S based web services and the declaratively specified mediator service [3]. We describe the ontological terms used to provide a detailed semantic description of service parameters and return types and how this information is used to determine the web service calling sequence. Additionally, our approach initiates a development environment, within which new service based models can be built, based upon existing services or service templates, e.g., a service based modeling environment would build on existing models, such as RAMS[2] and ROMS[3] We describe our prototype demonstrator for an application based on a water monitoring alert scenario.
This paper is organized as follows; the next section describes a motivating example. Section 3 includes our proposed approach and we layout the framework based on the integration of the technologies discussed in section 1. This section also describes the application of our approach to the discussed scenario along with a discussion on the implemented prototype. Finally section 4 discusses the conclusion and the future work. cases.

2 A Motivating Example

Real-time water quality monitoring and modeling technology assists in the timely detection and effective response to natural, deliberate or accidental human-caused conditions that may threaten public health and the environment. The drinking waters may be subject to deliberate dumping of biological, radioactive or chemical contaminants, sewage treatment plant failures, oil/chemical spills, harmful algae blooms and pollutants and pathogens in runoff. Emerging sensor and information technologies are capable of providing the tools needed to continuously monitor the various water quality parameters, transmit them in real time and validate, display and interpret them, as well as predict the future state of these variables. Water body modeling and water distribution models are now available for the assimilation of real-time sensor data. To meet the current needs for the timely assessment of water safety and quality, the three components viz. sensor, modeling and information technologies need to be integrated and fully tested in an end-to-end real time monitoring and modeling system to be deployed in a real environment.

[2] Regional Atmospheric Modeling System (RAMS) - a state-of-the-art research model employed at the Center for Environmental Prediction (CEP) at Rutgers University.
[3] Regional Ocean Model System (ROMS) developed by the Institute of Marine and Coastal Sciences at Rutgers University.

Scientists at MERI[4] are currently carrying out various monitoring efforts in the New Jersey Meadowlands district. Quarterly and continuous water sampling is carried out in and around the Hackensack River; here the sensors are placed at various depths in the river stream. This monitoring effort is part of an overall effort to develop a decision support system for continuous monitoring of the water quality at the Hackensack River [9]. A network of sensors is monitoring various parameters including, oxygen level, temperature, salinity, turbidity, alkalinity, toxicity, or presence of heavy metals, etc. For the purpose of illustration, let us consider the following scenario.

Scenario: *A "High Chlorinated Hydrocarbons level" alert is raised on the monitoring control panel. The technician monitors the alert and routes the alert to the analyzing module service (AMS) for additional information. The automated AMS compares the current sensor data with archived information. AMS infers from the parameter taxonomy and suggests the monitoring of additional parameters, e.g., temperature, conductivity and solar radiation. AMS automatically stores the output of the analysis with a record of the parameters and monitoring conditions at the time of reporting and forwards it to the monitoring scientist who routinely inspects analysis results such as these. The scientist, using a sophisticated graphing service, graphs the results of the coverage over a certain time frame and decides that the "alert" needs further investigation. He requests to reserve the use of a Mass Selective Detector (5973 MSD) with 6890GC and system configurations for previous runs on similar alerts. The MSD is placed for recording the data and this process is monitored by the technician by live video, along with live data from the equipment. The scientist is sent the results. Later the scientist looks at the results and, intrigued, decides to replay the analyzer run, navigating the video and associated data. The AMS summarizes previous related analyses reported internally and externally, and recommends other scientists who have experience in this area. The scientist finds that these results appear to be deviant and checks for salinity and saturation levels based on the parameters. The scientist completes the situation report to be accessed by his colleagues using a notification service and asks for the results to be calibrated against a river dynamics model service with river flow and sediment flux functionalities. The scientist logs the analysis and associated metadata along with background information. The availability of the new data prompts other automatic processing and a number of databases to be updated; some processing of this new data occurs.*

This scenario draws out a number of underlying assumptions and raises a number of issues that are broadly applicable to these applications;

- Capability to resolve the domain semantics and infer facts to assist decision making, e.g., the parameters of temperature and conductivity are associated with increased levels of chlorinated hydrocarbons. Specific sensor ontology within the framework represents sensor configuration attributes used to monitor the event based on existing domain environment conditions. The parameter ontology also infers the correlated parameters of saturation and salinity. These parameter values are determined on the basis of the existing monitored parameter values and help the scientist in his assessment and decision making.

[4] Meadowlands Environmental Research Institute is collaboration between the New Jersey Meadowlands Commission and RUTGERS-CIMIC.

- Develop application over service oriented architecture and expose part or all of the functionalities as web services, e.g., some of the key tasks associated with the analyzing module, one of the principal components of the water monitoring process, are adjusting sensor configuration, near real time mining of sensor data, inferring of determined parameters. The analyzing module is designed as a composite web service wherein specific functionalities are exposed as individual web services. Alternatively, the data mining functionality may be outsourced and a third part web service forms a part of the composite AMS. This data mining web service is selected as per existing domain requirements and discovered at near run time.
- To support the process enactment and automation, the system needs semantic descriptions of processes, e.g., the scientist may require specialized graphs or he may want to assess the experiment based on different representation graphs. To achieve this he has to select the most appropriate service based on the input, output and service pre conditions. The semantic description of the graphing service, represented in OWL-S, help him select the best available service to suit the overall process. Similarly the best matching data mining service, as part of composite AMS, is discovered based on its semantic description for accurate analysis.
- Different stakeholders need to be able to retain ownership of their own content and processing capabilities, but there is also a need to allow access to others under the appropriate terms and conditions, e.g., the scientist accesses the river dynamics model service that forms a part of ROMS under the agreed upon conditions of collaboration.

3 Proposed Approach

Extensive research in the multi-agent area dealt with the issues related to distributed architectures e.g. discovery, matchmaking and composition. A major limitation of this technology was the inherent heterogeneity and lack of interoperability [6]. That is the problem web services technology may resolve. Web services are pieces of software available on the Web that people can access through a standard protocol and execute remotely. When used together, web services can deliver a complex functionality. Web services thus tackle the problem of heterogeneous sources and make them interoperable. The current Web services technology solves only part of the problem. As more Web services become available, information overload will make finding the needed service difficult. To overcome these problems, we must first recognize that current web services technology is basically a syntactical solution and that the semantic part is still evolving. To exploit their potential, web services must be able to orchestrate themselves into more complex services. Thus, we need ways to combine individual Web services into a distributed, higher-level service. We thus need an automation that semantically describes services, such that software agents can locate, identify, and combine the services. The Semantic Web semantically describes content available on the Web such that software agents can understand and process related information. In this manner, Web services form a part of the Semantic Web and vice

versa. This is the main objective of our approach in augmenting the web services with semantic web technology; to semantically express Web services capabilities and developing intelligent services to form the framework.
Below is a description of the major steps of our approach. This is followed by the detailed application of the approach to water monitoring domain.
1. *Identification of the components of the system and expressing the semantic relationships among the system components.*
This involves the identification of various objects (tangible and intangible) that form the various components of the system. In the above motivating example components that fall within the application domain include the sensors, the scientists, technicians, security agencies and water companies. The non-tangible entities that form a part of the system could be the information exchanged among the system entities. Next we establish the relationships between the system entities. These relationships are based on RDF triples and have semantics associated with them. In the above example we could establish a relationship between a sensor and a technician as “sensor isConfiguredby technician".
2. *Deriving the "roles" and the "services" associated with the components on the basis of the semantic relationships.*
The semantics of the relationships among the various system entities help establish the "roles" and the "services" associated with the various components. This determines the role of each component within the application domain and the functionalities associated with their services. In the motivating example, we represent the technician as a service and the functionalities associated would involve configuring the sensor and monitoring the analysis based on the readings transmitted by the sensor.
3. *Modeling the process between the services and the semantic web services fulfilling the human as well as the non-human tasks.*
The Process Modeling can be further broken down into:

a. Data Collection and Validation: This involves the aggregation of the data from the various system entities. This data may exist in different formats and describes the content or the information being exchanged or to be utilized by the various system entities. This would also involve data verification and validation in terms of checking for discrepancies or accounting for anomalies. Once the data is collected and validated it can then be analyzed.

b. Data Analysis and Knowledge Discovery: The validated data undergoes analysis. The data is then integrated from the various formats to form a multi-dimensional multi-format dataset. This data is then subjected to certain Knowledge Discovery Data (KDD) mining pre-processing as part of knowledge extraction. KDD helps in the knowledge discovery. This in turn helps in the data modeling and reasoning.

c. Reasoning and Inference: The KDD data when combined with the policies helps in the simulation of the results. In the motivating example the KDD data, when combined with the policies and the disaster estimates, helps in the scenario simulation. This also helps in ensuring an immediate initiation of a response.

d. Presentation: Finally the processed data has to be presented, filtered through varied levels of authentications and authorizations of the various system entities. This presentation could be in different formats based on the requirements and constraints of the end user.

3.1 Application of Proposed Approach to Water Monitoring

Below is the description of the application of our approach to the Water Monitoring domain. A summary of the overall framework is presented below followed by a detailed discussion of the integration of these technologies. The framework is based on the key steps identified in our approach and comprises of the following [11] [12].

3.1.1 System Components

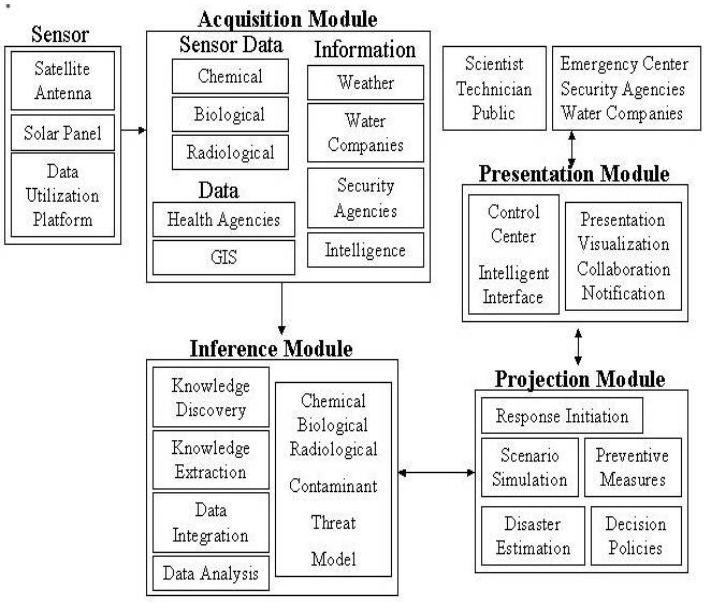


Fig. 1. Proposed Application Framework

We begin with identifying the key system entities based on the first step of our approach. An autonomous interacting service or a group of services would represent each of these components, viz. Water Companies, security agencies, emergency centers, public, scientists, technicians and sensors. Each component has specific attributes and descriptions. For example, the "sensor" entity. *Sensor*: is a physical component, which intermittently or continuously monitors various parameters e.g.; temperature, pH, dissolved oxygen, specific conductance, turbidity, oxidation, reduction potential, chlorophyll and record them. These effect automated sample collection if a specific sensor indicates chemical, biological or radiological contaminant presence. The sensor consists of a satellite antenna for the remote transmission of data/alerts and the related parameters. It has a data utilization platform for the collection and analysis of the sample data. As part of step 1 of our approach we proceed to identify the semantic relationships that exist among these system components. Due to lack of space a detailed discussion of this step is not presented in this paper.

3.1.2 Semantic Web Service

The integration of web services and the semantics take place when an ontology layer is placed over the service description language, WSDL. We achieve a similar integration in our system making use of OWL-S (previously DAML-S) [7] as part of step 2 of our approach. This enhances the ability to determine the types of services to be used in addition to the specification of the service parameters as OWL expressions. The OWL-S also provides the specification of the necessary pre-conditions and the after-effects of a service. The profile component of the OWL-S expression is used to express the service being provided or requested. This describes the parameters of the service in terms of the parameters of the process. We describe the process ontology with its specified parameters. In this part, as an example, we shall express the "Monitor" process. This would have its properties as a sub-property of input, which would be used to pass the parameters and a sub-property of output that would pass the return values. Monitor service would enable a technician to monitor parameters from a turbidity sensor. The constraint on the input parameter of this service is the alert from a sensor monitoring the turbidity. The process and the profile components of the service description are referenced together in a top-level service description. This also includes the reference to the protocol, which is to be used to access the service, known as service grounding. As part of our implementation we have chosen SOAP for service grounding.

At present we do not implement queries as part of our system. There is a lack of a particular standard format of expressing queries due to lack of standard query language for RDF and OWL. The exploration of semantic based query languages would form a major part of our future work with the maturing of the semantic web and the development and the standardization of query languages for the semantic web.

3.1.3 Domain Ontology

This part is based on step 2 of our approach. Interoperability between ontologies is an important issue, because the reuse of knowledge often implies that different existing ontologies are used together. This requires that the knowledge represented in the ontologies is not conflicting. Ontology interoperability is important for different existing ontologies as well as different versions of ontology. Given the scale and heterogeneity of our application domain we find the merging approach of creating a unified source is not scalable and is costly. Besides, an integrated information source would need to be updated as soon as any information in any individual source changes [10]. Furthermore, in certain cases a complete unification of a large number of widely disparate information sources into one monolithic information source is not feasible due to irresolvable inconsistencies between them that are irrelevant to the application. Due to the complexity of achieving and maintaining global semantic integration, the merging approach is not scalable. We have thus adopted a framework, of distributed ontologies, approach, which allows the sources to be updated and maintained independent of each other and enables composition of information via interoperation. In addition we follow the approach of fusing these ontologies with more general top-level ontologies, e.g., SUMO [5]. This allows the detection of possible implicit relations between concepts, which are not pre-defined, i.e. the detection of hierarchies, which were not initially obvious. We have also designed specific ontologies to describe the application domain in terms of the motivating scenario. Figure 2 depicts the ontology of the various parameters to be monitored.

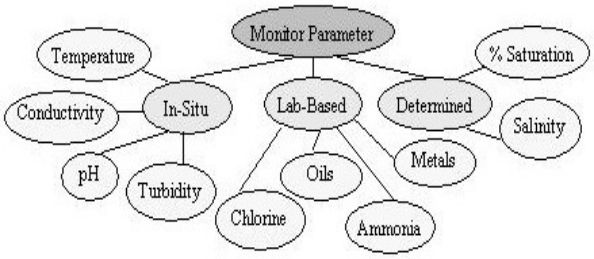


Figure 2. Parameter Classification

3.1.4 System Process Modeling

According to step 3 of our approach, we next present the main process modules that form our system. These modules effect the processes associated with Data Collection and Validation, Data Analysis & Knowledge Discovery, Reasoning & Inference and Presentation.

Acquisition Module: This module deals with the data collection from various sources. It records the data being transmitted by the sensors. It also collects health and hospital data from the health agencies. The GIS component within this module may deal with the geographical information associated with the various actors within the application

domain. In addition it collects data from various security agencies, the water companies and weather reporting agencies to keep an updated information base about the actual current situation.
Inference Module: This module serves as the analyzing core of the framework. It is made up of sub-modules that deal with the analysis of the data collected by the acquisition module. The data integration layer is based on top of the analyzing sub-module. This deals with the integration of the data that differs in format and originates from various sources. This data analysis and integration leads to knowledge extraction, which forms the base layer for knowledge discovery. All of these sub-modules combined, aid in the modeling of chemical, biological or radiological contaminant threats.
Projection Module: Disaster estimation along with the policies that support the decision making process forms the basis for simulation of various scenarios in light of the constructed models. This also helps in the formulation of effective preventive measures for damage control or averting a possible crisis. This is achieved with the help of the Response Initiation sub-module.

Presentation Module: This module comprises of the control center and the intelligent interface, which provides the information in different formats of presentations and visualizations. This helps the response team comprising of scientists, technicians, and the general public at the various security agencies, emergency centers and the water companies draw up an effective response to the raised alert.

3.2 Service Architecture

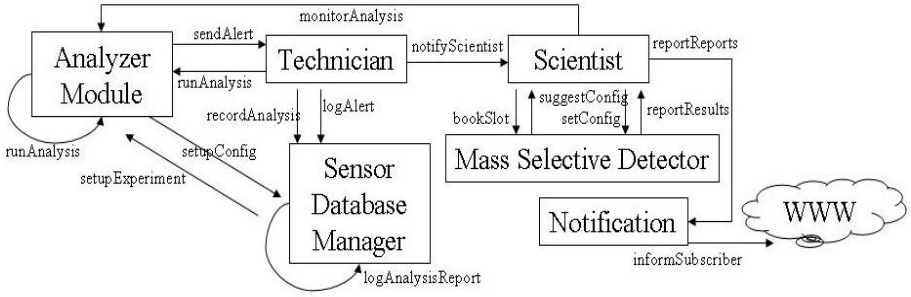


Figure 3. System Architecture with Web Services

In our prototype system the flow of information is shown in Figure 3. Here we have a *scientist service* (SS) and a *technician service* (TS) that are responsible for interacting with the scientist and the technician, respectively, and then for enacting their instructions within the framework. These services can be viewed as the computational proxies of the humans they represent, augmented with their personalized information about their owner's preferences and objectives. These personal services need to interact with other services in the domain in order to achieve their objectives. These other services include an *analyzer module service* (AMS), representing the various sensors; the s*ensor database manager service* (SDMS), which is responsible for managing

access to the database containing information about the sensor data, the graphing service (GS), and the device service for the m*ass selective detector* (MSDS). There is also *notification service* (NS), which is responsible for recording which scientists, are interested in which types of results and for notifying them when appropriate results are generated and an *archive retrieval service* (ARS) that can discover similar analyses of data or when similar experimental configurations have been used in the past. We shall now describe the motivating scenario in terms of the sequence of messages executed by the system components, the web services associated and the data exchanged.

The technician uses the *logAlert* to record data about the Alert when it arrives and the *setupExperiment* to set the appropriate parameters for the forthcoming experiment. At the appropriate time, the SDMS informs the AMS of the settings that it should adopt by the *setupConfiguration* service and that it should now run the experiment with the *runAnalysis* functionality. As a sequence for the *runAnalysis*, the AMS informs the SDMS of the results of the experiment and these are logged along with the appropriate experimental settings using the *logAnalysisReport*. Upon receipt of these results, the SDMS informs the NA of them. The NS then broadcasts the results by the *informSubscriber* to scientists who have registered an interested in results of that kind achieved with the help of the *seekSubscription*. When interesting results are received, the SS alerts the scientist by the *resultAlert*. The scientist then examines the results and decides that they are of interest and that further analysis is need. The scientist then instructs the SS to reserve a time slot for the *Mass Selective Detector (5973 MSD)* using the *reserveTimeSlot*. The researcher sets the appropriate configurations using the *setupConfiguration*. At the appropriate time, the experiment is started by the *executeExperiment* service. As a sequence for this service, the experiment is visually monitored with the help of the *visualMonitoring*, monitoring information is sent to the technician via the *monitorExperiment* and a report is prepared and sent to the SS using the *reportResults*. In preparing this report, the MSDS interacts with the ARS to discover if related experiments and results have already been undertaken. The SS using the *reportAlert* service alerts the scientist to the report. The scientist decides that the results should be discussed within the wider organization and the alert data is logged in the appropriate international database.

3.2.1 Prototype Implementation

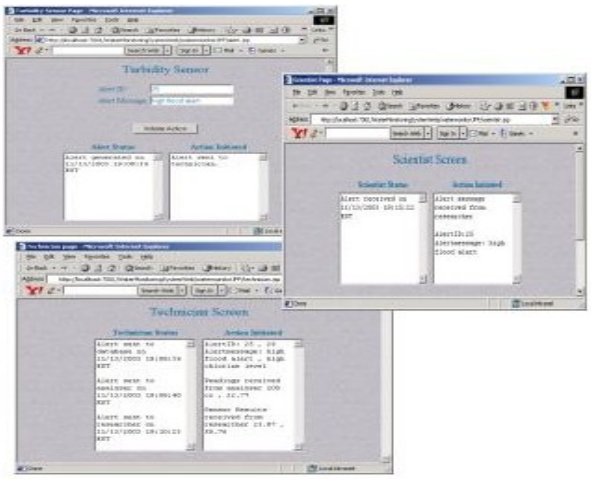


Figure 4. System Panels

The Water Monitoring System is designed as a web application and implemented using WebLogic application server platform. The design essentially consists of a representation of different services of the system such as technician service (TS), scientist service (SS) as java objects. These java objects as entities interact with each other through web services. The application allows interactions with these entities via user interfaces implemented in Java. The exchange of data between the entities is through SOAP/XML messages. Our prototype implementation simulates the alerts by varied sensors for the water monitoring system. The alerts and their triggers are predetermined however; the response of the system to these alerts is not pre-meditated. The application, as illustrated in Figure 4, consists of panels that represent the various components of the system. The panels shown in the diagram are illustrative of the motivating scenario described above. The panes in this panel show the action occurring at the specific service and the response initiated by the service.

4 Conclusion and Future Work

In this paper we outline our approach for building semantically rich software services, illustrate its application in the water-monitoring domain, and develop a prototype. This case study examines how our semantic web-services oriented framework was formed through the stages of the project, and its benefits to the water monitoring web application development. The main new characteristic of our approach with respect to previous similar systems is the integration of the web service technology, a service based approach and the semantic web in terms of an on-

tology for the representation of the domain. This integration improves the modeling of the information and the water monitoring process. We use the ontology to improve the communication among different services within the application domain, reducing ambiguities, in addition to the expressive description of the web services. This semantically rich service description offers an effective way of aggregating resources. This case study was conducted as our first step towards a semantic web services oriented stage, considering the functionality of web services. We plan to continue to enhance our web-services oriented architecture and our future work will be focused on pursuing a web service process flow on the basis of our architectural model, and constructing a system environment encompassing aspects such as reliability, application-level security, trust and orchestration.. The obvious next step would be to extend this system to incorporate complex processes. In addition we would also explore the semantic web query languages to augment the current system.

References

1. Chandrasekaran, S., Silver, G., Miller, J., Cardoso, J. and Sheth, A. Web Service technologies and their synergy with simulation. Winter Simulation Conference (WSC'02), (San Diego CA, December 2002)
2. Gómez-Pérez, A, Fernández-López M.; Corcho, O. “Ontological Engineering” Springer Verlag London Ltd. To be published in 2003.
3. Harris, S., Gibbins, N., Shadbol, N. Agent-based Semantic Web Services. World Wide Web Conference (WWW2003), Budapest, Hungary
4. Kulvatunyou, B., Ivezic, N. Semantic web for manufacturing web services. World Automation Congress Eight International Symposium on Manufacturing with Applications (Orlando FL, June 2002).
5. Kokla, M., Kavouras, M. Fusion of top-level and geographical domain ontologies based on context formation and complementarity. International Journal of GIS (7):2001.
6. Luigi, C., Ulises, C., and S`anchez-Marre`, M.OntoWEDSS: augmenting environmental decision-support systems with ontologies.Fujitsu Laboratories of America Technical Memorandum FLA-NARTM02-09.
7. McIlraith, S., Martin, D. Bringing semantics to web services. IEEE Intelligent Systems, Jan-Feb 2003.
8. McIlraith, S., Son, T., and Zeng, H. Semantic web services. IEEE Intelligent Systems (Mar/Apr 2001), Vol 16- 2.
9. Water Monitoring. MERI and the New Jersey Meadowlands Commission. http://cimic.rutgers.edu/hmdc_public/monitoring
10. Oliver, D., Shahar, Y., Shortliffe, E., and Musen, M. Representation of change on controlled medical terminologies. Proc. AMIA Conference (Oct. 1998).
11. Sheth, A., Bertram, C., Avant, D., Hammond, B., Kochut, K., Warke, Y. Managing semantic content for the web. IEEE Internet Computing (July/August 2002), (Vol. 6, No. 4).
12. Sollazzo, T., Handschuh, S., Staab, S., Frank, M., Stojanovic, N. Semantic web service architecture - Evolving web service standards toward the semantic web. FLAIRS 2002.

Binding- and Port-Agnostic Service Composition using a P2P SOA

Dominik Dahlem, David McKitterick, Lotte Nickel, Jim Dowling, Bartosz Biskupski, and René Meier

Distributed Systems Group, Trinity College Dublin
{dominik.dahlem, david.mckitterick, nickell, jim.dowling, biskupski, rmeier}@cs.tcd.ie

Abstract The assumption of the availability of port information at design time of service compositions in Service-Oriented Architectures (SOAs) is not valid for an increasing number of hosts on the Internet that do not have a public, static IP address. Existing workflow engines do not support services deployed on such hosts, as service invocations require the availability of port information defined either in concrete WSDL definitions or within a deployment descriptor of the BPEL workflow engine. This paper presents a workflow engine that supports runtime look-up of service endpoints based on a P2P middleware. Using a service identifier based on a DHT identifier, Service Proxy objects that encapsulate port information are downloaded over the structured P2P network from the host where the service is deployed. A Service Proxy delegates service invocations to an abstract protocol adaptor framework that uses dynamic invocation mechanisms to provide a protocol-independent execution of remote services, e.g., over GIOP/IIOP or SOAP. This allows us to specify binding- and port-agnostic service compositions in BPEL using abstract WSDL and our service identifiers. To validate our approach, we extended the ActiveBPEL workflow engine to support the discovery and consumption of services using our P2P middleware and the abstract protocol adaptor.

1 Introduction

One goal of Service-Oriented Architectures (SOAs) is to integrate and compose services that are deployed on heterogeneous middleware paradigms, e.g., CORBA, RMI, and J2EE. Web Services standards have been introduced to provide a language and platform-independent way of describing and invoking services on these platforms, and techniques for the composition of Web Services have emerged, such as the Business Process Execution Language (BPEL) [1,2]. Approaches have been developed to assist in the design and deployment of service compositions, such as semantic [3] or model-driven [4] techniques that help search the space of available services. However, they are based on middleware that requires the availability of binding and port information at deployment time.

In this paper, we present a strategy where service endpoints are looked up at runtime by an adapted workflow engine to support the binding to services

deployed on hosts that do not have a public, static IP address. Additionally, the presented client-side programming model is unaware of the actual interaction protocol and thus supports late binding using a protocol adapter framework configured at runtime. This allows us to support the deployment of services on hosts with DHCP-allocated IP addresses and, using Relay Peers, hosts that reside behind a Network Address Translation (NAT) Gateway.

The SOA and workflow engine described in this paper have been designed as part of the Digital Business Ecosystem (DBE) project[1], which aims at providing a SOA based on a P2P architecture for use by small-to-medium sized enterprises (SMEs). We identified the following requirements for the DBE SOA:

- Support for port-agnostic service compositions using service addressing and discovery mechanisms that do not require higher cost public IP addresses;
- Dynamic invocation mechanisms to promote loose coupling between consumers and the service invocation protocol;
- Enable service providers to advertise their services independent of the underlying paradigm (RMI, CORBA, J2EE, etc.) without wrapping the service into a Web Service container;

This paper focuses on service composition using BPEL which is built on top of Web Services Definition Language (WSDL) [5]. Abstract WSDL and BPEL together can enable service compositions to be defined independent of binding and port details. We implemented a Service Proxy framework that allows binding and protocol information for services to be encapsulated into objects that can be downloaded at runtime to access the service. Service Proxy objects use an Abstract Protocol Adaptor (APA) that enables the decision on which protocol to use to be delayed until runtime when the Service Proxy object is downloaded. Service Proxy objects are located using a service identifier that is based on a Distributed Hash Table (DHT) identifier in a structured P2P network. The P2P network is based on Bamboo [6] and all peers active in the system, including those without public, static IP addresses, can be located using the DHT identifier. Service identifiers can be discovered in a Service Registry. Finally, we extended an open source BPEL workflow engine, ActiveBPEL [7], to integrate it with our implementation of the P2P network, Service Proxy and APA frameworks.

2 Background

Service composition is a quite general term which is often used in relation to Web Service composition, because Web Services can be viewed as a service integration architecture covering many standards, languages and frameworks. BPEL [2] has become the accepted standard language for executable business processes defined by WSDL interfaces. It defines a process-centric model for the formal specification of the behaviour of business processes based on the interaction of the executable process and its partners [8]. A key aspect of BPEL is the notion of

[1] http://www.digital-ecosystem.org

partner links. Partner links describe the relationship between two services at the interface level, by providing both a link to services that are invoked by the process and also to clients which are invoking the process. The BPEL process itself knows only about partner links and is unaware of the underlying dependency on the concrete WSDL bindings and service endpoints. The BPEL specification does not require a static declaration of port-specific data, because an endpoint reference element allows for explicit dynamic assignment of service endpoints during the execution of a BPEL process [9]. Implementations of existing BPEL engines require concrete port and binding definitions to be available at deployment time, usually SOAP bindings and static endpoints. BPEL engines like BPWS4J [10] and ActiveBPEL [7] do not support the internal assignment or selection of service endpoints within their implementations when executing BPEL workflows so therefore this assignment must be defined by the BPEL developer.

Service discovery and selection have already been recognised and tackled by several research groups by introducing semantics into the process of designing and executing BPEL workflows. The METEOR-S project [11,12,13,14] provides tools to enable automatic service discovery and selection at deployment time based on constraints, which then uses the BPWS4J engine to execute static BPEL workflows. Service compositions are specified in BPEL but the required Web Services are described in service templates [11]. Service templates define a semantic description of a service [15] as well as constraints for service selection like user's preferences for service partners, Quality of Service (QoS) requirements, and service dependencies. An enhanced Universal Description, Discovery and Integration (UDDI) registry is queried with the service templates for matching service descriptions [16]. Matching service descriptions are further analysed if they meet the specified constraints [13]. Finally the best matching service is selected and with late-binding an executable BPEL process generated at deploytime. Another approach to overcome the static definition of services in a BPEL workflow comes from work done by the OWL-S research community [17]. They introduce a Semantic Discovery Service (SDS) that serves as a dynamic proxy between the BPWS4J engine and services [18]. Instead of invoking requests on statically bound services, the BPWS4J engine routes requests to the SDS. Services are described, advertised, and discovered via an OWL-S service profile. The SDS finds a matching service (or chain of services) and invokes the endpoint at runtime. Upon receiving a reply from the partner the SDS forwards the response back to the BPWS4J engine. Synthy [19] decouples a Web Service composition into logical and physical design stages. The logical stage establishes a relationship based on semantic annotations between candidate services for a composition while the physical stage is responsible for creating one or more concrete BPEL workflows based on QoS parameters relevant for this workflow. Additionally, Synthy adds a fault-tolerant level to workflow execution. Multiple workflows can be created at the physical design stage and deployed with the runtime engine. These executable workflows can be ranked according to QoS metrics. In case of a runtime failure in a workflow the runtime engine can select the next ranking workflow to be executed.

However, these projects do not address the look-up of endpoint information at runtime for a known service. There has been some existing work on the Web Services Invocation Framework (WSIF) [20,21] to enable delaying the decision on which binding to use until the service is executed, although it requires that multiple bindings are specified in advance with one of them chosen at runtime. Consequently, WSIF does not address the issue of services deployed on hosts that do not have a public IP address. The next section introduces our approach based on a structured P2P network and Service Proxies.

3 DBE SOA and the Workflow Engine

3.1 The DBE SOA

This section gives a brief overview of how service proxies, identified by a persistent and unique ServiceID, are discovered and downloaded using a structured P2P network. The DBE SOA provides two services to enable service registration/discovery and service binding, respectively. Firstly, a Service Registry is used to register and discover ServiceIDs along with their associated abstract WSDL. Secondly, a structured P2P network based on a Distributed Hash Table (DHT) architecture [6] is used to download a Service Proxy that encapsulates binding and port information for the service.

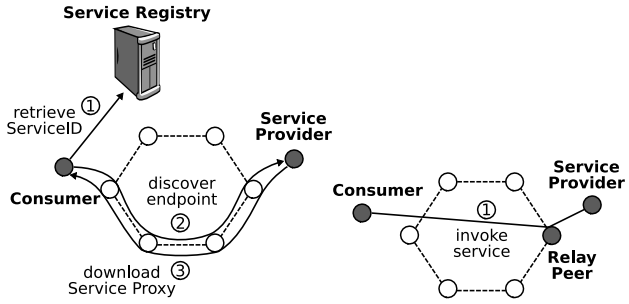
The Service Registry is currently a centralised component that provides service registration and look-up operations for both abstract WSDL and ServiceIDs for deployed service instances. BPEL developers can use the Service Registry to discover service instances to use in a service composition. We are in the process of developing a decentralised Service Registry based on an unstructured P2P network, described in [22].

The DHT architecture supports the deployment of services on hosts that do not not have a public IP address by providing a globally unique DHT identifier that is used to logically identify hosts in the DBE SOA, including those that reside behind NAT Gateways or have a DHCP allocated IP address. Peers that reside behind a NAT Gateway join the DHT by discovering a Relay Peer[2] that acts as a virtual server for their DHT identifier [24], relaying all messages to and from the NAT-restricted peer. Relay peers must have open static IP addresses.

The ServiceID contains two parts: the DHT identifier that identifies the peer that hosts the service and a local service identifier used by an application server on the peer. The ServiceID can then be used at runtime to discover the host for a service, enabling the Service Proxy object containing the actual binding information to be downloaded.

The Service Proxy Once a consumer has discovered a ServiceID in the Service Registry, it can be used to download a Service Proxy from the service provider's

[2] The discovery of Relay Peers is outside the scope this paper, but systems such as Skype use extensive caching of Relay Peer IP addresses and monitoring of performance [23].



(a) Service Discovery (b) Service Consumption

Figure 1. Discovering and downloading a Service Proxy using a Structured P2P Network

application server. The request for the Service Proxy is routed over the structured P2P network using the ServiceID to the peer hosting the service. If the peer hosting the service is NAT-restricted, the request is forwarded to its destination by a Relay Peer, see Figure 1 (a). The Service Proxy is deployed in the provider's local application server using a local, persistent unique identifier generated by the application server[3] and the endpoint information for the service, i.e., either the IP address of the peer itself or the IP address of its Relay Peer if it is a NAT-restricted peer. When the application server receives a request for a ServiceID, it returns a locally deployed Service Proxy for the service if found. A Service Proxy is a Java object that encapsulates the configuration information for services. Within this information the port-specific data, mainly the current endpoint address for a service, is found and can be used to interact with a service, see Figure 1 (b). The introduction of a Relay Peer into an invocation path introduces new failure modes to the transport protocol, meaning that not all invocation protocols are suitable for services deployed on NAT-restricted peers. For example, the use of SOAP RPC could result in lost replies if the Relay Peer unexpectedly fails.

The Service Proxy also specifies the invocation protocol and other binding related information which are supported by the service provider. A service provider can deploy a service with more than one protocol, i.e., more than one Service Proxy, enabling the consumer to choose the binding mechanism most appropriate for their environment. A Service Proxy Framework was implemented to provide a simple API for service consumers to download and handle proxies within the DBE environment.

[3] We use the Sun Servent application server, see http://swallow.sourceforge.net

3.2 Abstract Protocol Adaptor

Middleware often ties the deployed services and the interacting clients to their respective invocation protocols, such as CORBA's GIOP/IIOP or Java's RMI. The emergence of XML-based Web Services standards, such as WSDL and SOAP, help overcome cross-platform limitations. WSDL is not exclusively tied to the SOAP protocol. It enables the specification of multiple bindings, for example for EJBs, JMS, and IIOP.

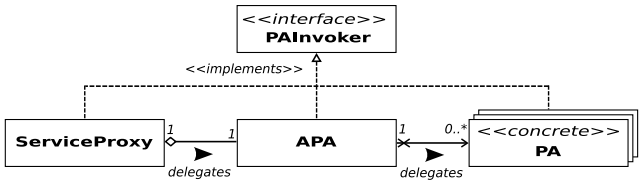


Figure 2. The APA Class Diagram

The Abstract Protocol Adaptor (APA) is a client-side programming model that provides a dynamic invocation interface, similar to that provided by WSIF, to make calls on a generic RPC invocation interface called `PAInvoker` (depicted in Figure 2). The APA framework can be integrated into any component, which requires a protocol-independent communication mechanism, such as a workflow engine. The `APA` promotes loose coupling between clients and the protocol used to access a services, as the protocol is declared or discovered at runtime rather than at compile time. The `APA` is configured at invocation time by a Service Proxy object with the necessary endpoint, the required invocation protocol, and possibly additional configuration. The Service Proxy exposes the same interface as the `APA` , and delegates invocations to the `APA` with all configuration values. The `APA` is ignorant of protocols and simply delegates the invocation to the respective concrete protocol adaptor `PA` at runtime. If a service is implemented to support a single invocation protocol, e.g., SOAP, then the client application will need a compliant SOAP protocol adaptor to create the SOAP calls.

The `APA` can be extended to implement any number of protocol adaptors that can create the necessary setup for a dynamic invocation of a remote service. So far, the following three protocol adaptors have been implemented, SOAP, kSOAP and object serialisation over HTTP. The SOAP `PA` supports the invocation of remote Web Services which expose SOAP endpoints. The kSOAP `PA` supports Web Service communication from mobile clients to kSOAP compliant services.

3.3 Adapting the ActiveBPEL Workflow Engine

The DBE SOA, Service Proxy, and APA frameworks have been integrated into an open source BPEL workflow engine, called ActiveBPEL. The ActiveBPEL organisation [7] has implemented a pluggable workflow architecture in Java that

is fully compliant with the BPEL specification, version 1.1 [1]. The engine runs on a servlet container and uses the Axis Web Services container for communicating with other Web Services and clients. It supports SOAP bindings for service invocations. Taking advantage of the pluggable architecture, our work mainly involved integrating this engine with the structured P2P network and APA framework by substituting the default invocation handler with our customised one.

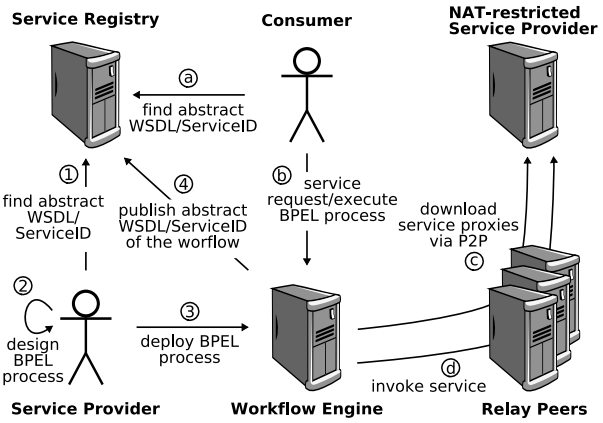


Figure 3. Design, Deployment, and Consumption of a BPEL Process

The deployment process for our adapted workflow engine requires a set of abstract WSDL definitions for all the services aggregated within the BPEL process, see step 1 in Figure 3. These abstract WSDL definitions effectively provide general representations of the technical interface of each service without any information about deployment platforms, bindings to that platform, or the endpoint location of the service. Within the architecture of the DBE a service provider publishes their abstract WSDL definition to a Service Registry with a ServiceID attached. During the design stage of a BPEL process, abstract WSDL definitions are selected to represent a service within the process, see step 2 in Figure 3. The partner relationship between the BPEL process and the selected service interfaces are defined in the `EndpointReferences` (EPRs), specified by WS-Addressing [25], as part of the `PartnerLink` in the Process Deployment Descriptor (PDD) of the ActiveBPEL engine. The ActiveBPEL engine supports the dynamic binding to ports with the `dynamic` and `invoker` endpoint reference strategies. However this requires the BPEL developer to specifically include the assignments and related activities within the BPEL process and possibly implement some additional service to provide the updated endpoint information. In our approach, we avoid using explicit assignments within the BPEL process by declaring the ServiceID of the associated published service as a dynamic URI reference to the service in the `Address` element within the EPR of the

`PartnerLink` . This ServiceID is used to discover Service Proxy objects that contain the binding and port information of a service instance at runtime. The complete BPEL process, containing a reference to the abstract WSDL definitions and their ServiceIDs, can be deployed to our extended workflow engine, see step 3 in Figure 3. The deployment process also publishes the service's abstract WSDL and ServiceID to the Service Registry, see step 4. Consumers can then discover composed services in the Service Registry, see step *a*, and invoke them, potentially via a Relay Peer, see steps *b* to *d*.

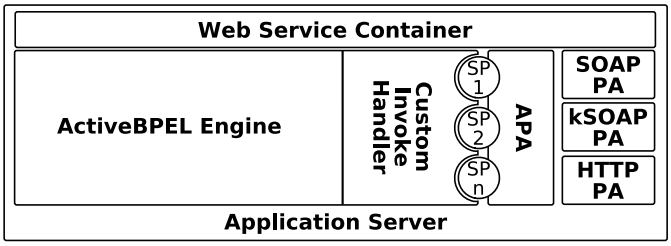


Figure 4. Modified ActiveBPEL Engine

The runtime process for our adapted workflow engine, illustrated in Figure 4, differs only in the behaviour of `Invoke` activities. The `Invoke` activity attempts to invoke an external service based on the abstract WSDL definition and the partner relationship. Our extension to the engine extracts the declared ServiceID for a service from the EPR instead of extracting an endpoint URL to bind to the service with SOAP, see workflow engine in Figure 3. Using this ServiceID, the Service Proxy is downloaded over the structured P2P network. Once a Service Proxy is located and downloaded, it is used to delegate the invocation of an operation to the protocol adaptor framework. The engine is unaware of the binding and port information for this dynamic invocation, as this information is encapsulated by the configuration capacity of the Service Proxy. This enables transparent consumption of services, independent of location, transport, protocol, and even data model.

4 Discussion

Our work has been evaluated by comparison with related work in the area of the execution of BPEL-defined service composition. We examine the performance of our approach for binding- and port-independent composition in relation to other approaches using the following criterion: support for abstract WSDL, interaction paradigm, endpoint discovery and binding mechanisms. The use of the abstract WSDL results in a reduced level of coupling between a service definition and service instance. The interaction paradigm, endpoint discovery, and binding mechanism enable a dynamic invocation approach and hide the underlying

middleware paradigm. The results presented in Table 1 are based on our analysis of these approaches (see Section 2). Due to the lack of concrete information regarding the binding mechanisms used in Meteor-S, the assumption was made that the communication protocol is SOAP.

	DBE Engine	**ActiveBPEL**	**Meteor-S**	**SDS**
Abstract WSDL	yes	no	no[a]	no
Interaction Paradigm	Web Services	Web Services	Web Services	Web Services
Endpoint Discovery	Runtime P2P N/W	Design-time[b]	Deployment-time	Runtime
Binding Mechanism	Dynamic Proxy	SOAP	N/A	Dynamic Proxy

[a] Uses Extended WSDL

[b] Also allows for in-model dynamic assignment of endpoints at runtime as stated in the BPEL specification.

Table 1. Comparison of Workflow Engines

All workflow engines compared in this paper use the BPEL specification to define a service composition. Two workflow engines are used in the different approaches, ActiveBPEL and BPWS4J. BPWS4J provides the execution environment for both Meteor-S and SDS, while ActiveBPEL was extended to support dynamic endpoint look-up and binding.

The interaction paradigm for all approaches is based on Web Services standards and requires the services to be deployed in a Web Service container. With our approach a service provider can expose a service using any supported middleware paradigm by the protocol adapter framework and abstract WSDL to describe the interface of this service.

In terms of endpoint discovery, the ActiveBPEL engine offers the least flexibility. The aggregated Web Services of the service composition needs to be discovered at design time including the binding and port information. The Meteor-S project focuses on developer tools that allow binding of Web Services to an abstract process based on constraints and generate an executable process at deployment time [11]. Consequently, the resulting BPEL process is static with respect to the endpoints of the chosen Web Services. While SDS does allow for the runtime discovery of service endpoints, these endpoints are restricted to concrete WSDL port definitions that can only be used by a host with a public IP address. Our approach supports the discovery of endpoints at runtime using a Service Proxy and a service identifier that logically identifies a host in the structured P2P network.

The ActiveBPEL engine and the Meteor-S approach use the standard SOAP binding mechanism for the invocation of services. The SDS is itself a locally

bound Web Service that acts as a single dynamic proxy between the workflow engine and the Web Services to be discovered. In our approach, the binding mechanism is not restricted to SOAP. Each service is associated with at least one Service Proxy that encapsulates a binding model for the service.

A feature of our approach to endpoint discovery and service binding is that services deployed on hosts with a non-static IP address can republish their Service Proxies on events such as a change in the host's Relay Peer or DHCP-allocated address. This helps increase robustness of the SOA.

5 Conclusions and Future Work

This paper described a P2P SOA architecture with explicit support for hosts without public, static IP addresses and a workflow engine based on our SOA that supports binding- and port-agnostic service composition. Service interfaces are described using abstract WSDL, and service consumption uses a Service Proxy and Abstract Protocol Adaptor framework that can be extended to any particular interaction paradigm supporting our dynamic invocation interface. Service compositions are defined using BPEL, and service endpoints are looked up at runtime using a service identifier, based on a DHT identifier, to download a Service Proxy over the structured P2P network. This approach supports hosts that reside behind a NAT Gateway by using a Relay Peer that acts as a virtual server for the NAT-restricted peer. We extended the open-source ActiveBPEL workflow engine in order to use the P2P SOA and the dynamic invocation framework, and compared this extension with other composition approaches based on BPEL. Our approach was shown to be more flexible for service providers to support hosts that do not have a public IP address and to be independent from the underlying middleware paradigm.

Issues that remain open for future research include a more flexible and generic execution of service compositions. Instead of downloading Service Proxies, a XML-based configuration file can be used to configure a middleware component that is responsible for the invocation. We are also working on a Relay Peer election algorithm and invocation protocols that better support the different failure modes introduced by Relay Peers in a service invocation path. In order to increase the robustness of our DBE SOA the next version will be based on a decentralised Service Registry available over an unstructured P2P network and consequently avoiding single points of failure. Finally, we are extending the workflow engine to support the intelligent selection of services at runtime for specific domains in the DBE.

Acknowledgements

The work described in this paper was partly supported by the "Information Society Technology" Programme of the Commission of the European Union under research contract IST-507953 (DBE).

References

1. BEA Systems, IBM, Microsoft, SAP AG, Siebel Systems: Business process execution language for web services. Version 1.1 (2003)
2. Wohed, P., van der Aalst, W.M.P., Dumas, M., ter Hofstede, A.H.M.: Analysis of web services composition languages: The case of BPEL4WS. In: ER. Volume 2813 of LNCS. (2003) 200–215
3. Cardoso, J., Sheth, A.: Semantic e-workflow composition. Journal of Intelligent Information Systems **21**(3) (2003) 191 – 225
4. Orriëns, B., Yang, J., Papazoglou, M.: Model driven service composition. In Orlowska, M.E., Weerawarana, S., Michael P. Papazoglou, e.a., eds.: ICSOC. Volume 2910 of LNCS. (2003) 75 – 90
5. van der Aalst, W.: Don't go with the flow: Web services composition standards exposed. IEEE Intelligent Systems **18**(1) (2003) 72–76
6. Rhea, S., Geels, D., Roscoe, T., Kubiatowicz, J.: Handling Churn in a DHT. Proceedings of the 2004 USENIX Technical Conference, Boston, MA, USA (2004) 127–140
7. ActiveBPEL LLC: ActiveBPEL engine, open source BPEL server (2004-2005)
8. Srivastava, B., Koehler, J.: Web service composition - current solutions and open problems. In: Proceedings of Workshop on Planning for Web Services (ICAPS). (2003)
9. Cubera, F., Khalaf, R., Mukhi, N., Tai, S., Weerawarana, S.: The next step in web services. Communications of the ACM **Vol. 46**(10) (2003) 29–34
10. Curbera, F., Duftler, M.J., Khalaf, R., Mukhi, N., Nagy, W.A., Weerawarana, S.: BPWS4J. Published online by IBM at http://www.alphaworks.ibm.com/tech/bpws4j (2002)
11. Aggarwal, R., Verma, K., Miller, J., Milnor, W.: Constraint driven web service composition in meteor-s. In: Services Computing, 2004 IEEE International Conference on (SCC'04), IEEE Computer Society (2004) 23–30
12. LSDIS Lab, University of Georgia: (Meteor-s: Semantic web services and processes)
13. Oldham, N., Thomas, C., Sheth, A.P., Verma, K.: METEOR-S web service annotation framework with machine learning classification. In: SWSWPC. Volume 3387 of LNCS. (2004) 137–146
14. Verma, K., Sivashanmugam, K., Sheth, A., Patil, A., Oundhakar, S., Miller, J.: METEOR-S WSDI: A scalable P2P infrastructure of registries for semantic publication and discovery of web services. Information Technology and Management **6**(1) (2005) 17–39
15. Rajasekaran, P., Miller, J., Verma, K., Sheth, A.: Enhancing web services description and discovery to facilitate orchestration. In: First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC). (2004)
16. Srinivasan, N., Paolucci, M., Sycara, K.P.: An efficient algorithm for OWL-S based semantic search in UDDI. In: First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC). Volume 3387 of LNCS. (2004) 96–110
17. McIlraith, S.A., Son, T.C., Zeng, H.: Semantic web services. IEEE Intelligent Systems **16**(2) (2001) 46–53
18. Mandell, D.J., McIlraith, S.A.: A bottom-up approach to automating web service discovery, customization and semantic translation. In: International Semantic Web Conference. Volume 2870 of LNCS. (2003) 227–241

19. Agarwal, V., Chafle, G., Dasgupta, K., Karnik, N., Kumar, A., Mittal, S., Srivastava, B.: Synthy: A system for end to end composition of web services. Journal of Web Semantics **3**(4) (2005)
20. Duftler, M.J., Mukhi, N.K., Slominski, A., Weerawarana, S.: Web services invocation framework (WSIF). In: Workshop on Object-Oriented Web Services (OOPSLA). (2001)
21. Mukhi, N., Khalaf, R., Fremantle, P.: Multi-protocol web services for enterprises and the grid. In Hopgood, P.B., Matthews, D.B., Wilson, P.M., eds.: EuroWeb. Workshops in Computing, BCS (2002)
22. Sacha, J., Dowling, J.: A self-organising topology for master-slave replication in peer-to-peer environments. Workshop on Databases, Information Systems and Peer-to-Peer Computing (DBISP2P) (2005)
23. Baset, S., Schulzrinne, H.: An analysis of the skype peer-to-peer internet telephony protocol. Technical report, CUCS-039-04, Department of Computer Science, Columbia University (2005)
24. Dabek, F., Kaashoek, M.F., Karger, D., Morris, R., Stoica, I.: Wide-area cooperative storage with cfs. In: SOSP '01: Proceedings of the eighteenth ACM symposium on Operating systems principles, New York, NY, USA, ACM Press (2001) 202–215
25. W3C: Web Services Addressing (WS-Addressing). http://www.w3.org/Submission/ws-addressing/ (2004)

SwinDeW-B: A P2P Based Composite Service Execution System with BPEL

Jun Shen[1, 2], Yun Yang[2], Quang Huy Vu[2]
1-Knowledge and Software Engineering Lab
Advanced Computing Research Centre
School of Computer and Information Science
University of South Australia
Mawson Lakes, Adelaide, Australia, 5095

2-Centre for Internet Computing and E-Commerce
Faculty of Information and Communication Technologies
Swinburne University of Technology
Hawthorn, Melbourne, Australia 3122

Abstract. The problems of poor performance, poor scalability and data block associated with centralised engines for workflow management systems have been recognised in many researches and projects for years. In the meantime, BPEL4WS (BPEL for short), a new flow based Web service execution language, is used widely in commercial applications. This paper presents peer-to-peer (p2p) based decentralised Web service engines for BPEL-based applications, which are aimed to solve the problems of the centralised counterparts. The architecture, mechanisms, protocols and algorithms to partition business processes into basic activities and launch the activities on a p2p network will be addressed. The paper also presents a solution for the current weaknesses of registry and index services for Web service publishing and discovery. Our prototype, SwinDeW-B, has been developed to demonstrate the feasibility and advantages of the proposed approaches.

1. Introduction

In Service-Oriented Architecture (SOA), services are software agents that are self-contained with well-defined interfaces. Service consumers use their client software agents to request services from the providers' software agents' interfaces. Web services are independent software components designed to support machine-to-machine interaction over a network. Web services based on HTTP or SMTP and a set of XML-based open standards such as WSDL [4], SOAP and UDDI that help them satisfy the requirements to realise SOA [3]. The most important features of Web services are the interoperability and the portability.

IBM, Microsoft and BEA have tried to define BPEL4WS (Business Process Execution Language for Web Service, BPEL in short) [2]. For a BPEL process all the parties involved are called partners, and for each partner it has a Web service interface to communicate with. IBM develops a centralised engine, BPWS4J, which runs on a single host to perform the job of coordinating Web services into a business process. In recent investigations and theoretical researches, researchers have observed that the centralised architecture of a Workflow Management System (WfMS) is a cause of the system weaknesses including poor performance, poor scalability, and so on [1, 13]. A research prototype named SwinDeW (*Swin*burne *De*centralised *W*orkflow), a distributed WfMS built on a peer-to-peer (p2p) network, was developed to overcome those weaknesses [14]. Later, SwinDeW has been adapted and extended with the support of Web services which is named as SwinDeW-S [11].

The purpose of this paper is to extend SwinDeW-S with a process definition language that is specific for composite Web services. BPEL was chosen for this purpose because it is stable in the current form for several years and has been used widely in commercial systems. The first objective of our research is to specify the architecture, mechanisms, protocols and algorithms for a distributed p2p based system to deploy composite Web service described in BPEL. The second objective is to enhance the system with rich semantics for Web service automatic publishing and discovery. The prototyped SwinDeW-B (SwinDeW with BPEL) system supports processes described in BPEL but executed in a p2p mode. In addition, as part of the system's mechanisms of deployment, the need to register Web services with a central server like in the case of UDDI is avoided. Instead, Web service owners only have to run an instance of the system, register their Web service with it and their Web service can be automatically discovered and involved to participate in a business process when it satisfies certain criteria within that specific application.

This paper is organised as follows. Section 2 gives an overview of related work on decentralised WfMS, composite Web services, BPEL as well as the JXTA p2p network and SwinDeW-S. Section 3 analyses the problems in existing systems and requirements for improvements. Sections 4 and 5 discuss the SwinDeW-B system architecture and design in details, followed by Section 6 where the prototype is presented and compared with BPWS4J. Section 7 concludes the paper finally.

2. Related work and background

Decentralising workflow management systems have attracted a wide interest from organisations and researchers in recent years. DynaFlow [8] of University of Florida concerns more about dynamics involved in inter-organisational workflows. Exotica/FMQM [1] is developed at IBM Almaden Research Centre to analyse the

effect of decentralised architecture on workflow management systems, aiming at scalability and fault tolerance. Endeavors [6] is a distributed workflow management system developed at University of California to provide a platform for process execution over the Internet. The METEOR-S project [9] proposes a framework for the annotation of Web services and analyses the dimensions of cost, time and reliability. Furthermore, the p2p concepts have been adopted in WfMS recently. For example, the Matrix project (www.npaci.edu/DICE/SRB/matrix) developed at San Diego Supercomputer Centre, delivers Grid workflow protocols and workflow language descriptions necessary to build a p2p infrastructure for Grid workflow management system.

BPEL [2] provides a language to specify the business process behaviour based on Web services. In other words, it is a high level language for creating composite Web services. BPEL deems a process as an active entity that involves its partners in structured activities. Therefore, a BPEL process model includes atomic activities as well as structural activities to constrain and link the atomic activities. Each atomic activity may have incoming and outgoing control links. Control links are used to synchronise the execution of activities in processes. Together with the structural activities, they help specify the control flows in the processes. Each outgoing control link is presented by the element <source> while each incoming control link is presented by the element <target>.

Currently, the dominant information distribution paradigm on the Web is still based on the client-server model. However, the problems include poor performance, poor scalability and data block, i.e., communication bottleneck due to the inefficient deployment of resources. A p2p network is one of the most innovative information distribution architecture emerging in recent years [10]. In this paper a p2p network is regarded as the network of nodes where every node is equal in being responsible to share resources and overhead of running the network.

JXTA [5, 12] is a framework developed by Sun Microsystems for developing p2p applications. SwinDeW [13] is a JXTA-based decentralised workflow management system developed to overcome the problems like poor performance, poor scalability, vulnerability, inflexibility, unsatisfactory system openness, and lack of support for incomplete process. SwinDeW-S [11] extends SwinDeW with WSDL-based Web service interfaces. Web service consumers not only need to know what a Web service can do, but also what quality, availability and reliability among many other attributes the Web service can offer. Once there is a way to describe those characteristics of a Web service and what a Web service consumer exactly wants, Web services can be effectively discovered and used in the right place. SwinDeW-S can be enhanced by replacing the current simple descriptions with rich-semantics Web service descriptions using OWL-S [7].

3. Problem analysis

3.1 Weaknesses of centralised Web service orchestration systems

Most of the engines for orchestrating composite Web services today are centralised. BPWS4J is a centralised engine for composing and deploying composite Web services. The engine runs on a single Java Virtual Machine (JVM). For each process, the engine takes in its BPEL document, the WSDL description of the interface that the process will present to clients, and the WSDL descriptions of Web services that the process may invoke during its execution.

We realise some shortages of this implementation. First, one JVM has to take charge of the whole BPEL process. It by itself has to do all the tasks of invoking Web services, organising data, managing the sequence of the activities and creating multiple threads for concurrent activity flows. When the BPEL process is complex, the burdens put on the JVM becomes very high and would degrade the performance. When there is more than one client requesting services from the process, there would be many instances of the process running on the single JVM. With the limitation of the CPU and RAM capacity, when the number of clients increases to a certain point, the system may collapse down. This is the scalability and performance issue. Secondly, the engine with centralised architecture is the core of data transmission which is very likely to cause the data block and communication bottleneck problem. In conclusion, the centralised architecture is the cause of problems including poor performance, poor scalability and data block.

3.2 Limitations of the registry and index methods in service publishing and discovery

Currently, the most common approach for Web services is using registry service, for example, UDDI. In a registry service, service providers register their Web service information with the registry. To do so, they must be authorised by the registry owner and must follow the information format specified by the owner. The disadvantage of this approach is that Web service providers have no flexibility in describing their Web services for publishing and the services can hardly be effectively discovered without rich semantics descriptions. Another limitation is the whole system depends on a single registry owner and this can be very vulnerable.

Another way is to use the index approach which is somewhat in contrast to the registry approach. In the index approach, Web service owners do not have to register their Web services but just expose the description of the services. Any party who is interested can collect the information and create an index about the Web services without the knowledge of service providers. There can be many indexes providing

different quality of information. The parties who need to find Web services can refer to the indexes. Which index is preferred will be determined by the selection from the pool of public service providers. The problem with this method is that the information of an index is very likely to be out-of-date.

The common problem of the above two approaches for Web service publishing and discovery is that they are not appropriate in a dynamic environment like the case of composing distributed Web services. Some alternative mechanisms have been proposed, for example, WSIL (ibm.com/developerworks/webservices/standards/). However, while service providers are not required to publish their WSDL, it is essential for some active agents, like peers, to bridge the gap through other approaches.

4. System architecture and design

4.1 Overall architecture

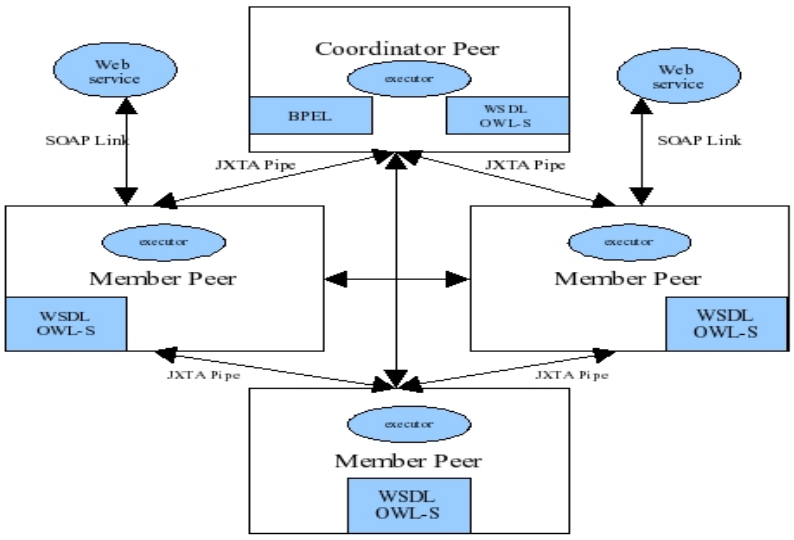


Fig. 1. System architecture of SwinDeW-B

Figure 1 depicts the overall architecture of the SwinDeW-B's composite Web service engines. The engines are composed of a p2p system built on the JXTA framework. BPEL is chosen as the language to orchestrate composite Web services. A composite Web service is described by a BPEL file and a set of WSDL files. By processing the files the Coordinator peer obtains the knowledge of which activities to be performed and the temporal order of performing them to complete a composite Web service. The Coordinator peer then converts the knowledge into the form that can be distributed into the p2p network without losing any information about the structure of the

process. The member peers hosted elsewhere on the p2p network are chosen to allocate parts of the process based on their capabilities. One capability is the ability whether a peer can invoke a required Web service. Another capability is the performance of the peer. Different kind of capability criteria can be used to select peers, which are owned and hosted by different individuals or organisations. A peer can invoke a Web service when its owner plugs a specific Web service invocation component in it.

4.2 Peer publishing and discovery protocols

The purpose of the discovery protocol is to maximise the discovery speed while minimising data traffic raised by the discovery process. At the deployment time, the Coordinator peer discovers peers in order to assign them the activities in BPEL processes. The essential principles of the discovery are that the Coordinator peer has a description about the peers it needs; the description is propagated on the p2p network; each peer on the network that receives the description will interpret it and see if it matches the description; the matching peers will send the response message back to the Coordinator peer. Besides acknowledging the matching of the description, the response messages also carry information about the peer status so that the Coordinator peer can select the best peer from the discovered matching peers. By this mechanism, the system is open for adding rich semantics using OWL-S in the discovery description, thus enhancing the system efficiency. Unlike traditional service publishing mechanisms, where service providers advertise their capabilities in a specific registry by a 'push' mode, peers in SwinDeW-B discover each other pro-actively in a 'pull' mode. Moreover, by grouping peers and checking availability, more suitable peers are assigned activities rather than solely relying on what service a peer has published.

4.3 Messaging protocol

In SwinDeW-B all kinds of content to communicate over the p2p network are wrapped in instances of class *DWFMessage*, which will be converted to XML format and put into an instance of *Message*, a JXTA class. Every *DWFMessage* message has two parts: the header and the content. Compared to messages in SwinDeW, there are some features added to the header part in SwinDeW-B in order to support the Web service discovery protocol presented in Section 4.2. To avoid messages from being propagated forever on the network, the attribute *count* is assigned to each message. When a message is generated, its *count* attribute is set to 0. Each time the message hops to a node on the network, *count* is increased by 1. If the *count* reaches a certain specified limit the message will be discarded from the network. Each message is also

assigned a time stamp put in the element <start>. This attribute is used to create a timing window for response messages. For example, when a peer sends a request message, it sets the start attribute with the current logic time of the message generation, sends the message and waits for the response. When a response message arrives, the peer compares the current time with the start of the response. If the difference is within a certain specified *timeout* the response will be processed, otherwise it will be discarded. The name of the peer that generates a request message is also attached in the header. This prevents the receiving peers from doing duplicate work. To support for this detection each peer has to maintain a history of incoming messages. The attribute *visitedPeers* helps each peer to immediately detect and discard messages that has come to and been processed by them before. By checking if it is included in the list of the visited peers of the message, the peer can decide whether it is an old message looped back again.

4.4 Control and data links

The control link is used to synchronise the execution of two activities. The target activity is never executed before the source activity finishes its execution. In SwinDeW-B, because the two activities are distributed on the p2p network, they cannot share the same link. Therefore the two activities must communicate via the p2p network to implement the link. The source link of the source activity also contains the contact information with the peer which hosts the target activity in addition to the traditional information. After the source activity finishes its execution, it evaluates the source link transition condition and sets the status for the source link. Then the source link with its status and the target activity name are wrapped in a message and sent to the peer indicated in the contact information. Upon receiving the message, the peer gives the message to the target activity with the name that matches with the target activity name information in the message. Hence, through this communication mechanism the two activities separately hosted on two peers are synchronised.

In the execution of a BPEL process, the activities are connected together through sharing data which are stored in variables. In the case of the decentralised system SwinDeW-B, variables are cloned and distributed on the p2p network along with the activities that need them. The copies of the same variable in peers on the p2p network form a data link between the peers. The link is stored in terms of the routing data and each copy of the variable maintains to synchronise the variable content with the other copies. Compared to the centralised architecture, it should be more cautious for designers to control and constrain concurrent modification of the same variable in SwinDeW-B.

4.5 Setting up routing data

The routing data are necessary for keeping BPEL processes' structure and establishing control links and data links when the processes are distributed on a p2p network. To construct the structure of a process defined in a BPEL process in a decentralised environment, each atomic activity of the process maintains a list of its successors, i.e. the basic activities which are constrained by BPEL structure activities to be executed after it. To maintain the control links, each source link of each activity maintains a reference to the target activity which holds the matched target link. And to maintain the data links, each process variable keeps a list of its target activities – the activities which will use it in their execution. In SwinDeW-B, the routing information is transferred by the JXTA pipe advertisements.

5. Operation of SwinDeW-B

A BPEL process' structure is expressed by BPEL structural activities. However when the process is deployed on a p2p network, only its atomic activities are distributed on the network. Therefore, the Coordinator peer has to partition the process into a graph of atomic activities with process structure information.

In SwinDeW-B, we developed a conversion algorithm named *Conv* whose input is a node of a tree *subRoot*. If *subRoot* is a <process> node, *Conv* will recursively apply the same conversion on the child node. If *subRoot* is a <flow> node, *Conv* will add all the predecessors and all the successors of *subRoot* to the predecessor list and successor list respectively of each of *subRoot*'s child nodes. After *subRoot* is processed, the same conversion is recursively applied to each of its child nodes. Therefore the conversion processes all the nodes of the tree until all the leaves of the tree are processed. After the conversion completes, the atomic nodes in the tree are extracted and put into a list. The list is the graph version of the tree containing sufficient process structure information insides each of the atomic nodes. The algorithm can be described by the following pseudo code:

```
Function Conv(subRoot As Node)
 Begin
    if(subRoot = <process>) then
           for each child N of subRoot
                  Begin
                         Conv(N) //recursive
                  End
    else if(subRoot = <flow>) then
           for each child N of subRoot
                  Begin
                         subRoot.addAllPredecessorsOn(N)
                         subRoot.addAllSuccessorsOn(N)
                  End
           for each child N of subRoot
                  Begin
                         Conv(N) //recursive
                  End
    else if(subRoot = <sequence>) then
```

```
            N := subRoot.getFirstChild()
            subRoot.addAllPredecessorsOn(N)
            for each child M of subRoot and M is not N
                  Begin
                        L := M.getLeftNode()
                        RD := L.getRightMostDescendant()
                        M.appendPredecessor(RD)
                  End
            N := subRoot.getLastChild()
            subRoot.addAllSuccessorsOn(N)
            for each child M of subRoot and M is not N
                  Begin
                        R := M.getRightNode()
                        LD := R.getLeftMostDescendant()
                        M.appendSuccessor(LD)
                  End
            for each child N of subRoot
                  Begin
                        Conv(N) //recursive
                  End
      end if
End
```

Linking activities is to match the source links with the target links and let the source links know the activities which match their target links. This knowledge is used to build the routing data discussed in Section 4.5. The matching can be achieved by simply iterating through the source links of each activity and for each source link, iterating through the target links of all the rest activities to find the target link with the same link name. If the target link is found, the activity that owns the target link is set as the target activity of the source link. Target activities of a variable are the activities that share the variable for their execution. The variable needs to know all the activities that share it so that data links can be maintained when the process is distributed on a p2p network as discussed in Section 4.5.

6. Prototype

The BPEL and WSDL files that define a BPEL process need to be parsed in order for the defined process to be put into the machine memory for later processing. A SwinDeW-B peer that parses the files becomes a transient Coordinator peer of the defined process to initiate the deployment and execution of the process. At the execution stage, the peers who have been allocated specific process activities will execute services autonomously in a fully decentralised manner. In order to cooperate with other peers on the p2p network, each peer must join the network. After the peer joins the p2p network, the peer's administrator can join multiple virtual communities. After a BPEL process has been loaded into the memory, the Coordinator peer needs to launch it on the p2p network before it can be executed. First, the Coordinator peer partitions the BPEL process into a graph of atomic activities using the algorithm described in Section 5. Then the control links between the atomic activities are set up. After that, the Coordinator peer discovers the peers for the activities through the peer discovery protocol as discussed in Section 4.2. After all the activities have found their

best peers to migrate in, the Coordinator peer distributes the activities to their peers. If the peers are allocated successfully, the process is ready to be executed.

The Coordinator peer will send the 'execution' message to all the peers that are hosting one or more activities of the process. Then the Coordinator peer also requests its *Executor* instance to begin executing the activities that reside locally on the Coordinator peer itself. Upon receiving the message, each peer requests its *Executor* instance to begin execution. In the execution mode, *Executor* launches a processing loop. In each loop cycle, it first checks whether there is any unfinished activity. If all the activities have finished, *Executor* will terminate the execution mode, otherwise it iterates through the unfinished activities and checks whether all the conditions for each activity are satisfied in order to execute the activity.

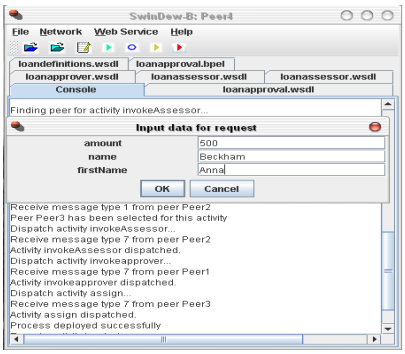


Fig. 2. Submit input data to process

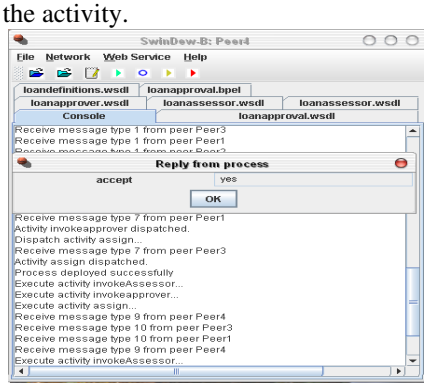


Fig. 3. Receive result from process

Figures 2 and 3 demonstrates the interfaces of the <receive> and <reply> activities in the SwinDew-B prototype. For commercial BPEL engines, the interfaces of the these activities would be implemented by Web service interfaces.

Features	SwinDeW-B	BPWS4J
Performance	By involving many peers in executing processes, the performance can be ensured. When the number of processes deployed and the complexity of the processes increase, more peers can be added to the network to keep the same level of performance.	Difficult to flexibly guarantee the performance of the system. When the number of processes deployed and the complexity of the processes increase, performance will deteriorate.
Scalability	When more clients request services, more peers can be added to ensure the same level of service quality. It can cope with the expansion of organisations much easily as workload can be evenly distributed among existing and new peers.	The more clients request services, the worse the service quality will be offered by a centralised engine.
Data block	Data are distributed almost equally on the p2p network, therefore the risk of data block is lower.	The server is the central point of data transfer. The risk of data block is higher.
Web service publishing and discovery	Web services can be published and dynamically discovered inside the system. The system is open for more advanced Web service descriptions with rich semantics to be integrated in.	Web services must be published and discovered which are bound manually by process owners before the process can be deployed and executed.

To illustrate the contribution of our research, the comparison between SwinDew-B and BPWS4J has been summarised in the table above. These criteria demonstrate the advantages of p2p based BPEL engines over traditional centralised engine, as we concerned in the problem analysis in Section 3.

7. Conclusions and future work

This research aims at developing a decentralised system for orchestrating composite Web services. Two main problems have been identified. One problem is the shortcomings caused by traditional centralised composite Web service engines, such as poor performance, poor scalability and data block related communication bottleneck. The second problem is the limitation of the current approaches for Web service publishing and discovery. Accordingly, the overall requirements of the research are identified. The first requirement is to develop a p2p-based decentralised system for composing Web services. The second requirement is to provide a mechanism for publishing and discovery of Web services with an extensible Web service description.

In this paper, the overall architecture and design of SwinDew-B, a decentralised system for composite Web service orchestration, is presented. Having demonstrated the feasibility and soundness of the prototype, the research has been successful in developing the architecture, mechanisms, protocols and algorithms for a decentralised composite Web service platform which is open for advanced semantic Web service publishing and discovery services to be integrated in. Nevertheless, as the proposed mechanisms introduced some communication overheads, how to optimise the paying off is a future work for further experimentation with comparing with existing benchmarks.

Acknowledgement

This work is partly supported by Australian Research Council under Discovery Grant DP0210654 and Swinburne Vice Chancellor's Strategic Research Initiative Grant 2002-2004. Dr Jun Yan and Mr Jonathan Derham had implemented the earlier versions of SwinDeW and SwinDeW-S.

References

[1] G. Alonso, C. Mohan, R. Guenthoer, D. Agrawal, A. El Abbadi and M. Kamath, "Exotica/FMQM: A Persistent Message-Based Architecture for Distributed Workflow Management", in Proc. of IFIP WG8.1 Working Conference on Information Systems for Decentralised Organisations, August 1995.

[2] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weerawarana, "Business Process Execution

Language for Web Services Version 1.1", BEA, IBM and Microsoft, http://www-106.ibm.com/developerworks/library/ws-bpel/, May 2003.

[3] D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, D. Orchard, "Web Services Architecture", http://dev.w3.org/cvsweb/~checkout~/2002/ws/arch/wsa/wd-wsa-arch-review2.html, 2004.

[4] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana, "Web Service Description Language (WSDL) 1.1", http://www.w3.org/TR/wsdl, 2001.

[5] J.D. Gradecki, "Mastering JXTA – Building Java Peer-to-Peer Applications", Wiley Publishing, Indianapolis, Indiana, 2001.

[6] A. S. Hitomi, G. A. Bolter and R. N. Taylor, "Endeavors: A Process System Infrastructure", in Proc. of the 19th International Conference on Software Engineering (ICSE'97), pp.598-599, 1997.

[7] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, K. Sycara, "OWL-S: Semantic Markup for Web Services", http://www.daml.org/ services/owl-s/1.1/, 2005.

[8] J. Meng, S.Y.W. Su, H. Lam, A. Helal, J. Xian, X. Liu, S. Yang, "DynaFlow: A Dynamic Inter-Organisational Workflow Management System", Technical Report, University of Florida, http://www.cise.ufl.edu/tech_reports/tr02/tr02-004.pdf, 2002.

[9] METEOR-S Project, http://lsdis.cs.uga.edu/Projects/METEOR-S/, 2005.

[10] O'Reilly P2P Directory, http://www.openp2p.com/pub/q/p2p_category, 2005.

[11] J. Shen, Y. Yang, J. Yan, "Adapting P2P based Decentralised Workflow System SwinDeW-S with Web Service Profile Support", in Proc. of the 9th International Conference on Computer Supported Cooperative Work in Design (CSCWD'05), pp. 535-540, 2005.

[12] Sun Microsystems, Project JXTA v2.0: Java™ Programmer's Guide, May 2003.

[13] J. Yan, "New Process Coordination Technologies for Peer-to-peer based Decentralised Workflow Systems", PhD thesis, Swinburne University of Technology, Melbourne, Australia, 2004.

[14] J. Yan, Y. Yang and G. K. Raikundalia, "SwinDeW - A Peer-to-peer based Decentralised Workflow Management System", to appear on IEEE Transactions on Systems, Man and Cybernetics, Part A, http://www.it.swin.edu.au/personal/yyang/papers/SwinDeW-TSMC.pdf.

On Simulation Based Quality of Service and Performance Analysis of Web Services

Dhananjay Kulkarni[1] and Chinya V. Ravishankar[1]

[1] Department of Computer Science and Engineering,
University of California – Riverside,
Riverside, CA 92521
{kulkarni, ravi}@cs.ucr.edu

Abstract. In this paper, we try to simulate and analyze the performance of a generic web service. We abstract the protocol and platform-specific details in order to study a typical client server environment, which is representative of most services we use over the Internet. Our simulation framework is based on a theoretical model of web server. In this simulation framework, the web server, Internet, and the clients accessing the service are modeled as an open queuing network. Our primary contribution is to simulate this model using CSIM, which is a process-oriented, general-purpose simulation toolkit with C/C++ interfaces. Our simulations show that there are a number of parameters that affect the response time e.g. file size, server speed, the available network bandwidth and the buffer size. We study the effect of these parameters on the *serving capacity* of the web server, which is defined as the boundary beyond which the response time of the system increases drastically and results in poor quality of service. As an extension to our simulation framework, we explore alternatives to improve the system performance. The central thesis of our experiments is that, as the load on the web server increases, the time required to serve a file increases gradually up to point and thereafter increases asymptotically towards infinity. Although, we present a very generic simulation model, our results can be used to identify bottlenecks in typical web services or information dissemination environments. Moreover, important lessons learnt about the serving capacity can be used as guideline to improve the performance of highly loaded web services – by either deploying faster servers or increasing the network bandwidth.

1 Introduction

Over the years, the Internet and its usage has grown very rapidly. For example, large amounts of information is being stored on web severs or file servers, instead of books. At the click of a button, almost any kind of information can be searched, downloaded or viewed online. In general, such client-server environments abound the Internet, wherein users connect to a server and access information available at the server. Typical information dissemination sites such as news agencies [1, 2, 3], digital libraries [4, 5], and data set repositories [6] are a small set of examples of services available over the Internet. As reported in [14], the Internet usage is growing at an alarming rate and the web services are becoming more dynamic. Unfortunately, poor planning regarding the expected load or performance can prove disastrous for the web service provider.

Hence, there is a need to simulate and study the web server performance, or any information disseminating client server environment. In this document, web server means all the servers that are accessed by clients over the Internet, which includes FTP servers, HTTP servers, directory servers, mail servers or just about anything on the Internet that represent a client-server system.

In this project, we simulate the performance of a typical web service. The basic idea is to model the web server, Internet and the clients accessing the web service as an open queuing network. The theoretical model is based on the work presented in [7]. Our contribution lies in using a readily available simulation toolkit and in developing an extensible framework to simulate and study the model. We have developed the simulation framework using CSIM [8], which is a process-oriented, general-purpose simulation toolkit that can be integrated into C/C++ programs.

We define response time of a job as the sum of the time required to service it and the time for which it remains queued. The primary goal of our simulation is to determine the *serving capacity* (or maximum capacity) of the web server. The s*erving capacity* is defined as the boundary, beyond which the response time of the system increases drastically. As we will see through the experiments, there are a number of parameters that affect the response time, including file size, server speed, the available network bandwidth or the buffer size. Our results imply that as the load on the web server increases, the time required to serve a file increases very gradually up to a point, and thereafter increases asymptotically towards infinity. This defines the serving capacity of the web server.

Roadmap: In section 2, we present the basic model for a web service. In section 3 we describe the simulation model. In section 4, we analyze the results of our experiments. In section 5, we simulate the alternative models and compare the results with the generic model. We conclude the paper by presenting our important results and directions for future work.

2 Model

The web server along with the Internet and browsers is nothing but a client-server system. A queuing model [9] can be used to model and analyze the performance of such environments. Typically, there are thousands of clients simultaneously accessing a web service. These clients may be using variety of platforms and web browsers to access the Internet. This prompts us to model the web service as an open queuing system. The web server thus handles a number of jobs simultaneously, but at the lowest level only one job may use the resource at a time. Other jobs have to wait in the queue before they are processed (or serviced).

2.1 Open Queuing System

By abstracting the hardware and software details of a web server, we can model a web service as an open queuing system i.e. new jobs arrive from outside and eventually depart from the system. While considering the basic model of the system, we have

ignored the issues related to the interface between the client and the web server. e.g. TCP/IP and HTTP protocols running at both ends and CGI scripts running on the web server. Packet loss and retransmission issues are also not accounted, because the model becomes too specific to the protocol implementation. The server is only concerned with servicing a number of HTTP or FTP requests (or number of hits) generated by the client browsers. Hence, only 4 nodes (single server queues) are needed to model the basic system. The inter-arrival times, and the service time distribution is assumed to be exponential. Moreover, this system forms a product form network [9] and hence each queue can be analyzed independently.

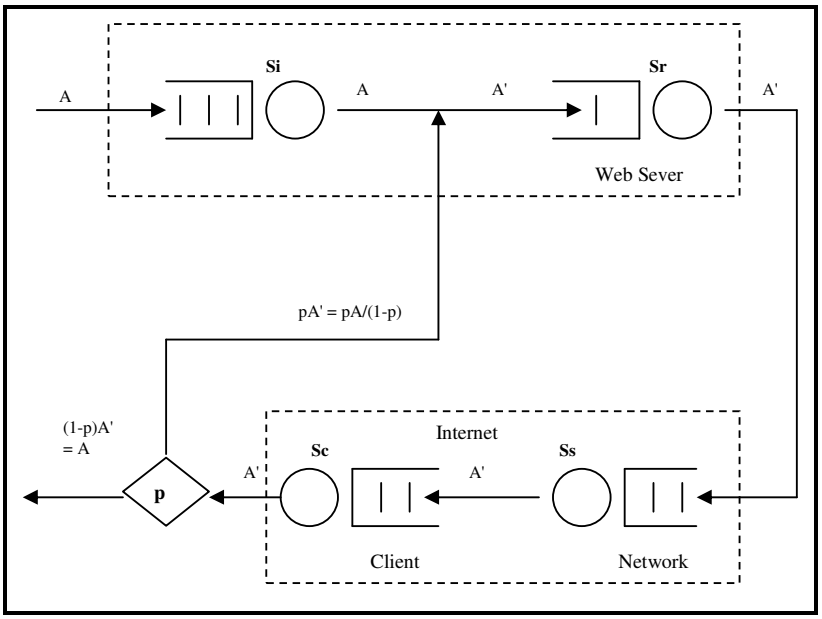


Fig. 1. Web Service Modeled as an Open Queuing System

The queuing model is shown in figure 1. The web server consists of nodes S_i and S_r. Both are modeled as M/M/1 queues with infinite buffer space. When the request arrives, all the one-time initialization takes place at node S_i and then the request proceeds to node S_r. At node S_r, a single buffer worth of data is read from the memory, processed and then passed onto the network. Node S_s is again M/M/1 queue and forms the network interface at the server side. At S_s the data is transmitted to the Internet at server's transfer rate. This data travels through the network and is received by the client browser S_c, which is modeled as a delay center (i.e. M/M/∞ queue). Note that if the request (file transfer) is not fully complete, the job branches and returns back to node S_r for further processing. The probability that the requested file is fully transmitted is the ratio of the buffer size used at the web server to the average file size. Moreover, job flow balance states that the rate of jobs leaving any stable node must equal its arrival rate. The arrival rate at each queue is shown in the figure itself.

2.2 Model Parameters

Network Arrival Rate (A): The network arrival rate is the average number of requests (or hits) that are received by the web server per second. It is important to note that this is not the instantaneous value, but the mean number of requests that arrive at the server. We assume the inter-arrival time of the requests is exponentially distributed. A more convenient way to imagine this value is 'hits per day', which is the (60×60×24) times the network arrival rate. This rate depends on the number of clients that are simultaneously trying to access the web server. In this document we have assumed it to be 20 hits per sec, unless otherwise stated.

Average File Size (F): Whenever a client tries to access the web service, a certain amount of data (including control and requested data) is transferred from the server to the client. The size of the requested files can vary widely - depending on what the client wishes to access and how the sever stores this information. In our model, we assume the average file size to be 5275 bytes. In some experiments we vary this value to examine its effect on the response time.

Buffer Size (B): Whenever the web server tries to service the request (e.g. for a file transfer), it does so by sending a single 'chunk' of data at a time. So a larger file might require more number of chunks, than a smaller file that is to be transferred from the same server. The buffer size usually corresponds to the disk block size on the server machine (so that a single chunk of data can be read in one read operation). A representative value of 2000 bytes is used is our analysis. This parameter does not have a significant impact on the overall performance of the system.

Initialization Time (I): When a request arrives at a server, a small amount of time is required to do the one time initialization. E.g. recording the information about the client's request, suffix mapping. The service rate S_i is 1/I. A reasonable value for initialization time is 25 milliseconds.

Static Server Time: (Y): The static server time represents the time that is spent in processing the buffer, irrespective of the size of the buffer. This might include context switch, setting up the buffer space in the memory etc. In this model the value of Y is assumed to be 7 milliseconds.

Dynamic Server Rate (R): This value represents the rate at which the server actually processes the buffer. The processing includes reading the requested data from the disk into the buffer and sending it to the node S_s. The dynamic server rate depends on the type of processor being used. For today's computers we approximate the dynamic server rate to be 2.5 Mbytes per second. The service rate of the node S_r is calculated as $1 / [Y + (B / R)]$.

Server Network Bandwidth (S): The rate at which the server can send the data over the Internet is known as the server network bandwidth. We assume S to be 1.875 Mbytes per second.

Client Network Bandwidth (C): There is buffering done at the client side as well. The client network bandwidth represents the rate at which the client software (browser) can receive a buffer. We assume that the browsers have average rates of 883.75 Kbytes per second.

3 Simulation

We have simulated the system using CSIM, which is a process-oriented, general-purpose simulation toolkit. We have used the C interfaces from CSIM on a Windows NT 4.0 platform. The appendix provides the CSIM simulation code. In the CSIM terminology - the client browser, the web server and the network are modeled as a separate *facility* and the client requests are modeled as *processes*. Parameters that are kept constant during the simulation are assumed to have default values as shown in table 1.

Table 1. Default Values of the Simulation Parameters

Parameter	Value
Network Arrival Rate (A)	20 hit/sec
Average File Size (F)	5275 bytes
Buffer Size (B)	2000 bytes
Initialization Time (I)	25 milli secs
Static Server Time (Y)	7 milli secs
Dynamic Server Time (R)	12.5 MB/sec
Server Network Bandwidth (S)	1.875 MB/sec
Client Network Bandwidth (C)	883.75 KB/sec

4 Analysis

We will first present the analysis of the basic model. Our goal is to find out the maximum upper bound on the 'number hits per day' that can be serviced. This is defined as the maximum capacity (M) of the web server. Next, we will see how the network bandwidth and average file size influence the response time. We also observe the effect of average file size on the maximum capacity of the web server.

4.1 Finding Maximum Capacity

Figure 2 shows the effect of arrival rate on the response time of the server. We can clearly identify the interval beyond which the performance of the server becomes very poor. Thus, the max capacity for the simulated system is approximately 39 hits per second. We notice that the response time (T) is mere fraction of a second for arrival rates (A) less than 39 (approximately). As the server approaches full utilization, T increases towards infinity. For this model of web server 39 hits per sec (i.e. 3369600 hits per day!) seems to be the maximum capacity (M). This implies that if the load is increased on the web server (when it is operating near maximum capacity), the response times experienced by clients will be extremely poor. This situation resembles a kind of deadlock (or thrashing), where the web server attempts to serve more and

more files at such slow speeds that no files are completely transferred. Operating beyond this maximum capacity is likely to provide a poor quality of service to the users.

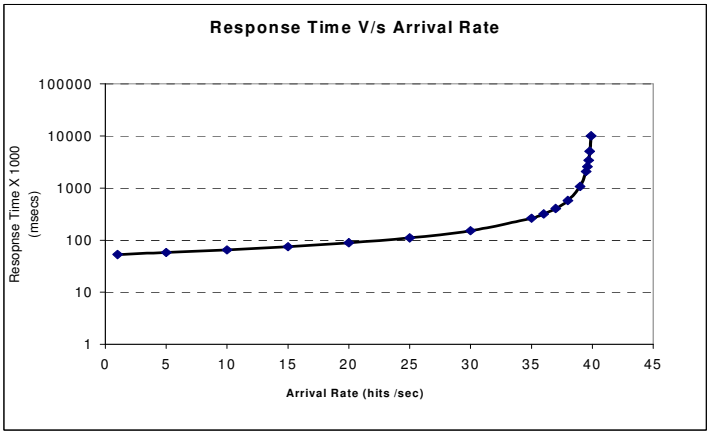


Fig. 2. Maximum Capacity of the Basic Model

4.2 Influence of Network Bandwidth

The effect of S on the response time depends on whether the bottleneck is the web server or the network bandwidth. As seen in figure 3, the response time of the system decrease initially with increase in the bandwidth, after that it remains almost steady.

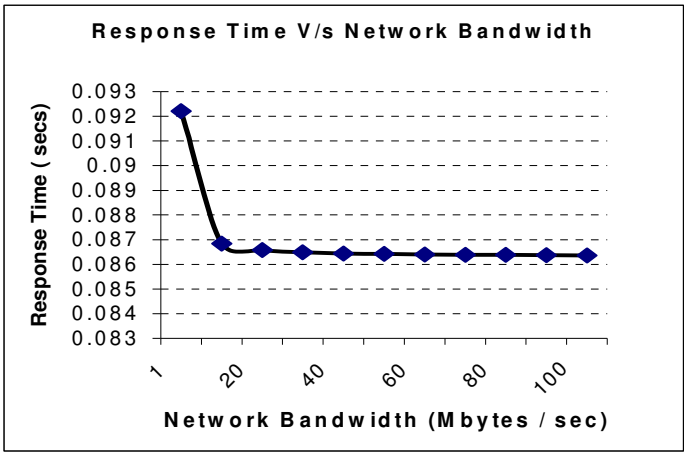


Fig. 3. Effect of Network Bandwidth on Response Time

4.3 Influence of Average File Size

As shown in figure 4, the file size heavily influences the response time. In fact, beyond the ridge, the values obtained are meaningless and the web server makes hardly any progress.

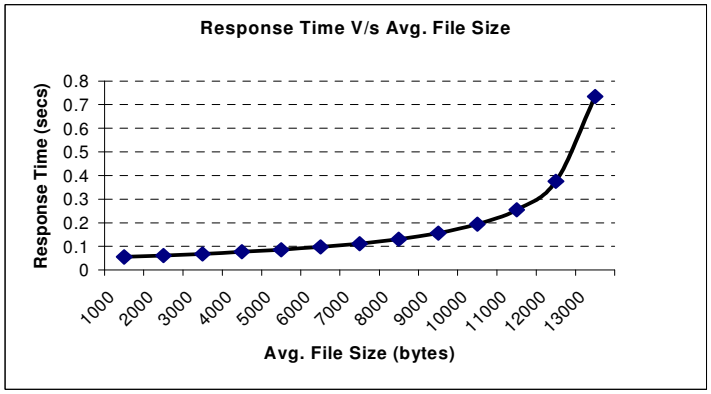


Fig. 4. Effect of Network Bandwidth on Response Time

4.4 Effect of Average File Size on Maximum Capacity

Maximum capacity is greatly affected by the file size. Figure 5 shows that M decreases exponentially with respect to F. When F is relatively small, a small change in F has a significant effect on M. However, for higher values of F, the max capacity is already low and hence changes in F have little effect.

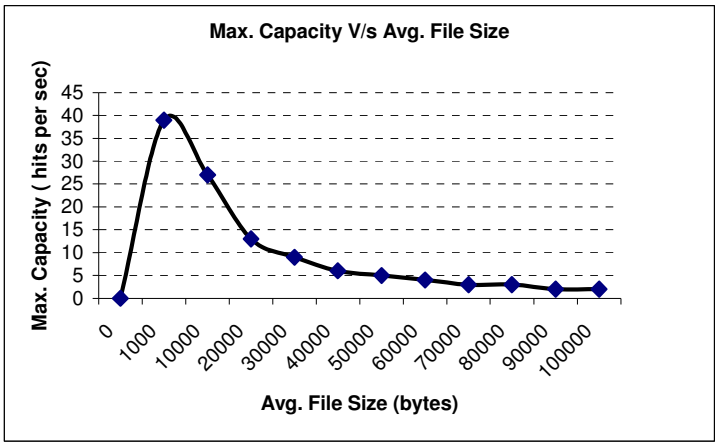


Fig. 5. Effect of Average File Size on Response Time

5 Multi-server Alternatives

The performance of the web service starts degrading when the server tries to work beyond its maximum capacity. It is essential to explore alternatives to improve the performance of the system, before the response time becomes unbearable (adding to the frustration of the clients accessing the web service). We have considered the following alternatives and drawn comparisons between their relative merits and demerits.

1. Using multiple servers
2. Replacing the existing server with a faster web server
3. Increasing the network bandwidth
4. Adding a slower server

5.1 Multi-server Model

Sometimes it is not cost effective to replace the existing web server with a faster one. In such situations, we can get better performance by adding more servers. Usually, the web site content is mirrored on numerous machines, thus creating a Redundant Array of Inexpensive Devices (RAID). This helps in sharing the load among two or more servers. For the multi-server model, the basic model is modified as shown in Figure 6. Most of the components in the multi-server model are same as in the basic web service model, except the probabilistic branch that is introduced to share the load among two servers. If every q fraction of the requests are serviced by S_{r1} then the remaining (1-q) fraction of requests are serviced by node S_{r2}. During simulation, the requests are equally divided among the two servers i.e. q = 0.5. Thus, the mean arrival rate at each server is half of the total arrival rate A'.

Table 2 shows the response times for the alternative models. We can draw the following inferences about overall performance of the web server. It is observed that, adding servers having comparable (or same) service rate is more beneficial than using multiple-server systems (with mismatched loads). In the latter case, mismatched loads lead to situations where the slower server becomes overloaded and the faster server remains idle, thus defeating the purpose of using multiple servers. The best alternative is to replace the server itself with a faster server. For moderate to high arrival rates, increasing the network bandwidth can give better performance. For very high arrival rates it is advisable to add a redundant identical server. The worst scheme is to add a slower server to the RAID. It is interesting to note that this scheme actually causes degradation of performance. Notice that all configurations seem to have a maximum capacity when the network arrival rate is around 40 hits per second.

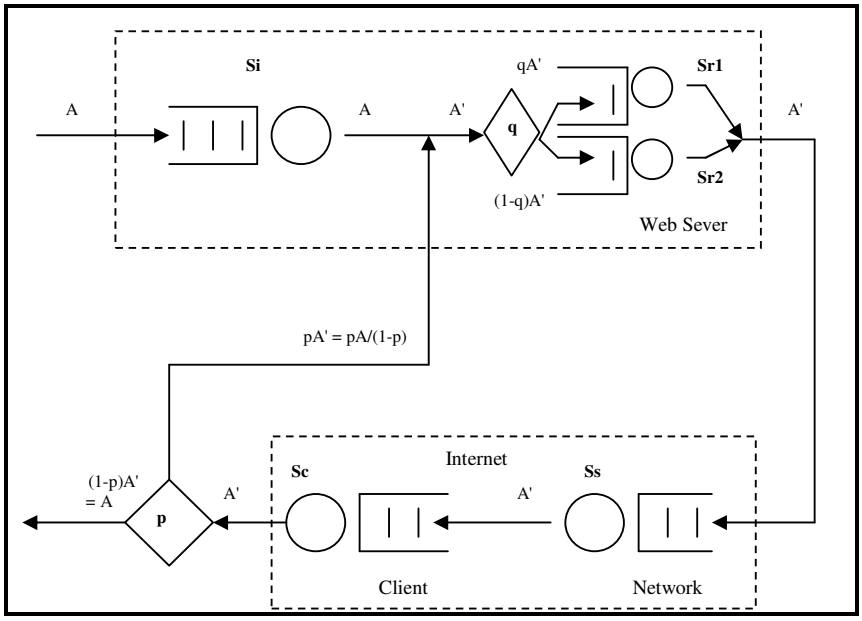


Fig. 6. Web Multi-server Modeled as an Open Queuing System

Table 2. Response Times for Alternative Models

Network Arrival Rate (A) Hits per sec	**Response Time (secs)**				
	Single Server	**Multiple-servers (2 matched servers)**	**10 times Faster Server**	**10 times Increased Band-width**	**Adding a 5 times slower server**
10	0.06540	0.06315	0.06490	0.06286	0.06408
20	0.08929	0.08223	0.08832	0.08659	0.08354
30	0.15260	0.13539	0.15063	0.14980	0.13709
35	0.26470	0.23729	0.26161	0.26190	0.23926
36	0.31808	0.28770	0.31453	0.31523	0.28970
37	0.40512	0.37146	0.40112	0.40226	0.37356
38	0.57599	0.53857	0.57146	0.57312	0.54073
39	1.08079	1.039021	1.07561	1.07790	1.04125
39.5	2.08346	2.03925	2.07790	2.0805	2.04151
39.6	2.5840	2.53929	2.57838	2.58114	2.54157
39.7	3.41790	3.37260	3.41220	3.41503	3.37496
39.8	5.08510	5.03939	5.07936	5.08227	5.04168
39.9	10.0851	10.03943	10.0798	10.0828	10.0417
40.0	Infinity	Infinity	Infinity	Infinity	Infinity

6 Conclusion

Using a theoretical model proposed in earlier literature, we have simulated the Internet, the web server and the clients accessing the web service as an open queuing network. Some very useful inferences can be drawn using this simple simulation model. The most important one being that, there is some upper limit to the number of requests that a web server can handle. As the load on the server increases, the response time increases gradually till it comes close to the maximum capacity. Hereafter, it increases asymptotically and finally reaches a stage where the server behaves as if it is serving a large number of requests, but in reality, not much progress is being made. We also observe that average file size has a significant influence on the system performance. Finally, we have explored the alternatives to improve the performance of the system. Our results should be helpful in providing meaningful insights into a typical web server performance and identifying bottlenecks in the multi-server environments.

For future work, it should be interesting to compare our simulation results with real-world measurements. One direction could be to use a commercially available web server monitoring tool, such as [10] and create a test-bed that closely follows our parameter settings. Another extension to our work could be to model the lower-level details of a network environment. For example, simulating information servers that are connected through a proxy [11] with different system parameters or which connected in a particular topology. Lastly, our model can easily be extended to use a real-time dataset, such as webpage 'clicks' experienced by popular web portals. This would give further insights into modeling bursty or peak traffic performance for popular online shopping [12] or auction sites [13] that abound the Internet.

Acknowledgments. This work was supported in part by grants from Tata Consultancy Services, Inc. and the Digital Media Innovations program of University of California.

References

1. BBC News Homepage: http://www.bbc.co.uk, 2005
2. CNN New Homepage: http://www.bbc.co.uk, 2005
3. NASA News Highlights: http://www.nasa.gov/news/highlights/, 2005
4. ACM Digital Library: http://portal.acm.org/dl.cfm, 2005
5. IEEE Xplore, Release 2.1: http://ieeexplore.ieee.org/Xplore/guesthome.jsp, 2005
6. Arizona State University, Data Hub: www.public.asu.edu/~huanliu/DHub/project.html, 2005
7. Louis P. Slothouber: A Model of Web Server Performance, Proceedings of 5th Conference on WWW, 1996
8. Mesquite Software Homepage, CSIM 19: http://www.mesquite.com/, 2005
9. R. Jain: The Art of Computer Systems Performance Analysis, John Wiley and Sons, 1991
10. HP Labs, HTTPERF Homepage: http://www.hpl.hp.com/research/linux/httperf/, 2005
11. Y. Fujita and M. Murata and H. Miyahara: Performance Modeling and Evaluation of Web Systems, Proceedings of Performance Modeling and Evaluation of Web Systems, 1998
12. Amazon.com Website: http://www.amazon.com, 2005
13. EBay Inc. Website: http://www.ebay.com/, 2005
14. Internet Usage Statistics, Internet World Stats Portal: http://www.internetworldstats.com

Appendix: CSIM C Simulation of the Web Server Model

```
#include <stdio.h>
#define NARS   5000          // Number of arrivals to be simulated
EVENT done;                  // Event to signal done
TABLE tbl_Sys, tbl_SI, tbl_SR, tbl_SS, tbl_SC;
QTABLE qtbl_Sys, qtbl_SI, qtbl_SR, qtbl_SS, qtbl_SC;
FACILITY f_SI, f_SR, f_SS, f_SC;
double u_SI, lambda_SI, u_SR, lambda_SR, u_SS, lambda_SS, u_SC,
lambda_SC;
int cnt;                     // Number of active tasks
FILE *fp;
double Network_Arrival_Rate, Average_File_Size, Buffer_Size, Init_Time,
Static_Server_Time, Dynamic_Server_Rate, Server_Network_BW,
Client_Network_BW;
void sim()     {                          // Main process
       int i;
       fp = fopen("csim.out", "w");
       set_output_file(fp);
       set_model_name("Web Server Model");
       create("sim");
       Init();                            // Default values
       cnt = NARS;
       for(i = 1; i <= NARS; i++)    {
              hold(expntl( 1.0 / lambda_SI));// hold inter-arrival
              cust_SI(i);
       }
       wait(done);                        // Wait until all done
       report_tables(); report_qtables(); mdlstat();
}
void sim_SR(double n) {
       create("sim_SR");
       hold(expntl( 1.0 / lambda_SR));      // hold inter-arrival
       cust_SR(n);
}
void sim_SS(double n) {
       create("sim_SS");
       hold(expntl( 1.0 / lambda_SS));      // hold inter-arrival
       cust_SS(n);
}
void sim_SC(double n) {
       create("sim_SC");
       hold(expntl( 1.0 / lambda_SC));      // hold inter-arrival
       cust_SC(n);
}
void cust_SI(double n) {
       TIME t1, t2;
       create("cust_SI");
       note_entry(qtbl_SI);
       t1 = clock;
       sim_SR(t1);                        // Initiate process cust_SR
       reserve(f_SI);
              hold(expntl(1.0 / u_SI));     // Use facility
       release(f_SI);
       t2 = clock;
       record(t2-t1, tbl_SI);
       note_exit(qtbl_SI);
}
void cust_SR(double t){
       TIME t1, t2;
       create("cust_SR");
       sim_SS(t);                         // Initiate process cust_SS
```

```
        note_entry(qtbl_SR);
        t1 = clock;
        reserve(f_SR);
                hold(expntl(1.0 / u_SR));     // Use facility
        release(f_SR);
        t2 = clock;
        record(t2-t1, tbl_SR);
        note_exit(qtbl_SR);
}
void cust_SS(double t) {
        TIME t1, t2;
        create("cust_SS");
        sim_SC(t);                            // Initiate process cust_SS
        note_entry(qtbl_SS);
        t1 = clock;
        reserve(f_SS);                        // Use facility
                hold(expntl( 1.0 / u_SS));
        release(f_SS);
        t2 = clock;
        record(t2-t1, tbl_SS);
        note_exit(qtbl_SS);
}
void cust_SC(double t) {
        TIME t1, t2;
        create("cust_SC");                    // Initiate process cust_SC
        note_entry(qtbl_SC);
        t1 = clock;
        reserve(f_SC);
                hold(expntl(1.0 / u_SC));     // Use facility
        release(f_SC);
        t2 = clock;
        record(t2-t1, tbl_SC);
        record(t2-t1, tbl_Sys);
        note_exit(qtbl_SC);
        cnt--;
        if(cnt == 0)
                set(done);
}
```