The 2nd European Young Researchers Workshop on Service Oriented Computing

http://www.cs.le.ac.uk/events/yrsoc2007

11-12 June 2007

University *of* Leicester

Stephen Gorton
University of Leicester
smg24@mcs.le.ac.uk

Monika Solanki
Imperial College
monika@doc.ic.ac.uk

Stephen Reiff-Marganiec
University of Leicester
srm13@le.ac.uk

Preface

Service Oriented Computing (SOC) is more than just ideas related to those services: in particular it is a chance of bringing together the business/user domain and the services domain. The word 'service' encompasses web services, semantic web services, grid services and e-services.

The 2nd European Young Researchers Workshop on Service Oriented Computing (YR-SOC 2007) is a 2-day workshop aimed at PhD students, young researchers working in the industry and researchers who have completed their studies in the last few years. It followed on from the highly successful inaugral event, hosted by De Montfort University in 2005. YR-SOC 2007 took place at the University of Leicester, UK, and was organised by Stephen Gorton, Monika Solanki and Stephan Reiff-Marganiec.

The aim of the workshop is to build a reputable and respectable forum for young researchers with inputs from industry practitioners. The core objectives are to exchange information regarding advancements in the state of the art and practice of SOC, as well as to identify the emerging research topics and define the future trends in this domain. Contributions cover aspects such as frameworks for building SOC applications, SOC composition, orchestration and choreography, SOC modelling and design, Semantic Web, ontologies, and SOC, and SOC discovery and selection (although the call was considering further areas).

The programme included invited talks from Mark Little (JBoss, UK), Steve Ross-Talbot (Pi4 Technologies, UK) and Martin Wirsing (Ludwig-Maximilians-Universität München, Germany), plus presentations by Luis Andrade (ATX Software, Portugal) and Nicolas Gold (SOSoRNet, UK). The technical programme included a rich variety of papers from young researchers across Europe, including Belgium, Germany, Italy and the UK.

The workshop received a total of 24 submissions, which were each reviewed by at least 3 people from a strong programme committee of international reputation. The committee decided to accept 16 papers. In addition, 7 poster submissions were received and the PC Chairs elected to accept them all for display at the workshop. These proceedings include all accepted submissions, which were updated in light of the reviews given. In addition, the abstracts of the poster submissions are included.

The workshop was organised with generous sponsorship from the following organisations:

University of Leicester
ATX Software
SENSORIA
SOSoRNet
De Montfort University (STRL)

June 2007 Stephen Gorton, Monika Solanki and Stephan Reiff-Marganiec

Programme Chairs

Stephen Gorton	(University of Leicester, UK)
Monika Solanki	(Imperial College, UK)
Stephan Reiff-Marganiec	(University of Leicester, UK)

Steering Committee

Barry Norton	(Open University, UK)
Stephan Reiff-Marganiec	(University of Leicester, UK)
Monika Solanki	(Imperial College, UK)

Programme Committee

We are extremely grateful to the committee for their reviews of the submissions:

Roberto Bruni	(University of Pisa, Italy)
Christoph Bussler	(CISCO Systems Inc., USA)
Antonio Cau	(De Montfort University, UK)
Schahram Dustdar	(Technical University of Vienna, Austria)
David Edmond	(Queensland University of Technology, Australia)
Dieter Fensel	(DERI Innsbruck, Austria)
Gianluigi Ferrari	(University of Pisa, Italy)
Reiko Heckel	(University of Leicester, UK)
Frank Leymann	(University of Stuttgart, Germany)
Aad van Moorsel	(University of Newcastle, UK)
Arun Mukhija	(University College London, UK)
Barry Norton	(Open University, UK)
Mike Papazoglou	(University of Tilburg, The Netherlands)
Stefan Tai	(IBM Research, USA)
Emilio Tuosto	(University of Leicester, UK)
Kenneth Turner	(University of Stirling, UK)
Mathias Weske	(University of Potsdam, Germany)
Hongji Yang	(De Montfort University, UK)
Jian Yang	(Macquarie University, Australia)
Gianluigi Zavattaro	(University of Bologna, Italy)

External Reviewers

In addition to the Programme Committee, we are grateful to the following people who also provided reviews:

Vasilio Andrikopoulos
Laura Bocchi
Pin Chen
Sara Corfini
Stefania Galizia
Roberto Guanciale
Andrew Hughes
Peter Kilpatrick
Jacek Kopecky
Benedikt Kratz
Hernan Melgratti
Bart Orriëns
Carlos Pedrinaci
François Scharffe
Monika Solanki
Daniele Strollo
Ioan Toma
Zhixian Yan

Table of Contents

Full Papers

Posters

Full Papers

Quality Estimation for Streamed VoIP Services

Mousa Al-Akhras and Hussein Zedan

STRL, De Montfort University, Leicester, UK
makhras@dmu.ac.uk, hzedan@dmu.ac.uk
http://www.cse.dmu.ac.uk/STRL/index.html

Abstract. Media services over IP networks are provided through Voice over IP (VoIP) Protocols. In this paper an architecture for Quality of Service (QoS) service within an enterprise business model is presented. In the proposed architecture an enterprise offers media streaming services to its clients and the client needs to pay for this service. The quality of the received stream will be measured and based on the measured quality the user's profile is updated with the required cost for using the service.

keywords Voice over IP, VoIP, Multimedia, Service Quality, Streaming.

1 Introduction

Service Oriented Computing (SOC) has become an active area of research. The research directions have concentrated mainly on areas such as compositionality, service description languages, orchestration, models, etc. However a fundamental research question to SOC is how these services provide functionalities within an acceptable level of quality. This is very much embedded within QoS area. Developers/Designers often relegate these issues to the service providers themselves. While this may be acceptable, we need to lift this responsibility to identifiable self-contained service that can measure QoS of a given service(s) within a given domain. This paper addresses the concept and the engineering of such a service. We also give a general architecture which is realisable within SOC. The considered domain of application is voice synthesis and streaming which has its use in for example the art world, video on demand and multimedia broadcasting. These QoS issues are packaged in a separate and unique service that could be used, composed, interacted with by other services.

Transmission of voice traffic over data networks such as packet switching Internet Protocol (IP) networks in what is coined as Voice over IP (VoIP) has revolutionised the way telecommunication services are being delivered. This integration of voice and data over one network offers several advantages over pure telephone networks and makes the development of new and innovative services possible. The IP network was originally designed to carry non real-time traffic such as email or file transfer. Carrying real-time traffic in addition to non real-time traffic over such networks is a challenge due to the possible degradations

due to packet loss, delay and jitter (difference in packet interarrival time). Even with such challenges, there are many great promises of such integration [1–3].

The advantages of the new technology are many, including: lower equipment cost than the case of the pure telephone network, lower bandwidth requirements, lower operating and management cost for one unified network, and integration of both voice and data into one network, which makes the creation of new and advanced services possible. Several applications are made possible by the integration of voice and data into one network. Some of the applications include: call-centre integration, directory service over the network, making international calls at the price of local calls, fax over IP, and broadcasting of voice and/or video traffic.

The latter service is quite important and it is the focus of this paper as we assume having a media server that streams voice/video to its customers, where the media stream is provided by one or more media organisation(s). The customers are charged for the media stream they receive based on the quality of the stream, which depends on the network characteristics at the time of streaming.

In SOC, the focus has thus far been on service composition, discovery, publishing and description. Little work has been done on QoS aspects of a given service. This paper provides an approach for measuring QoS for any multimedia service and in particular for streamed VoIP. The scenario for the above system is described in section 2. Several protocols have been developed to aid in VoIP signalling, and these protocols will be discussed in section 3. The ways the quality is measured in the system are discussed in section 4. Section 5 describes the QoS service architecture of the system. Section 6 describes the interface for the QoS service. Finally, section 7 concludes the paper.

2 Service Scenario

In our scenario depicted in Figure 1, an enterprise owns a media server that establishes sessions with the enterprise's customers. During a session different kinds of media can be streamed to the user over the IP network. The server gets its materials from several service providers including TV stations providing either live streaming of their available channels or some of their recorded programs. Other providers could include radio stations providing either live broadcasting of the radio stations or a set of recorded programs. Other sources of media could be a music database where different songs could be retrieved. The clients of the media server can reach these media sources through the media server. A client can connect to one or more media sources depending on his/her contract with the media server enterprise, and the media server can then stream the required materials over the IP network.

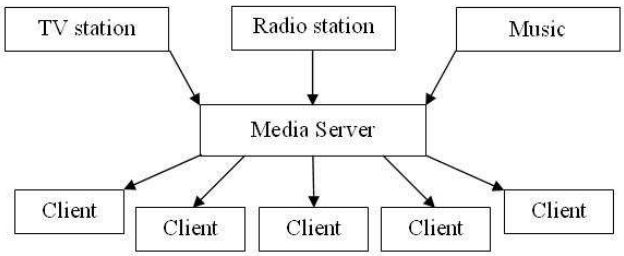


Fig. 1. Service Scenario

3 VoIP Protocols

The way one or more of these media sources is described, published, discovered and then streamed to the clients depends on the associated protocols. The basic protocol stack for transmitting VoIP packets is RTP/UDP/IP. Real-time Transport Protocol (RTP) is attached to provide information related to the speech packet such as a time-stamp to help in order playout of the packets. Also, such information can help identifying any loss of packets during transmission and the time stamp aids in determining the time taken for the packet to reach the receiver. User Datagram Protocol (UDP) is added in the transport layer, while Internet Protocol (IP) header is added in the network layer. There are also many protocols to help in session management. These include [1, 4–6]:

- Session Initiation Protocol (SIP): SIP is a flexible signalling protocol with many optional and user-defined fields that provide the required mechanism to set up and tear down media sessions and other signalling-related messages.
- Session Description Protocol (SDP): SIP does not care about the media type to be transported. It relies on SDP for exchanging media capabilities including the media format(s) to be used.
- Session Announcement Protocol (SAP): SAP is used for advertising multimedia sessions described using SDP and transported using SIP. The announcement is within the same scope as the session it is announcing.
- Real Time Streaming Protocol (RTSP): RTSP is the most important protocol in terms of its capabilities in streaming media sessions. RTSP allows clients to control media servers by instructing commands to record and playback multimedia sessions including functions such as *seek*, *fast forward*, *rewind*, and *pause*. A user can use SIP to invite a media server to a multimedia session, and then uses RTSP to control operations during the session.

4 Measurement of the quality

In our scenario described in section 2, it is essential to measure the quality for technical and commercial reasons as clients of the media server enterprise are expecting a certain level of quality for this service. If the quality does not meet

the user's expectations as described in his/her agreement with the enterprise, the payment should be cancelled/modified to reflect the degradation to the quality.

The quality in the system will be measured objectively on the user side using one of the International Telecommunication Union-Telecommunication Standardisation Sector (ITU-T) Recommendations called the E-Model [7]. In the E-Model the network and terminal impairments such as packet loss, delay and jitter are used to estimate the quality of the received stream and give estimation on a scale of 1 to 5 where the numbers means the following (5) Excellent, (4) Good, (3) Fair, (2) Poor, (1) Bad. Once the quality of the stream is estimated, the server is notified in order to update the user's profile with the session details including the duration and the quality.

5 QoS Service Architecture

The enterprise responsible for operating the media server must explicitly notify its customers that the quality of the stream is not guaranteed and they need to be aware of this while using the service. However, the required payment depends on the received quality.

Based on the company's policy there are two types of customers. Customers of the first type are aware of the variation in the quality and they are prepared to pay different rates for the received stream where the amount to be paid is proportional to the quality they receive. These customers may determine a lower bound for the quality below which the stream becomes useless and the stream is rejected. Customers of the second type are aware of the variation in the quality as well, but they are not prepared to pay for the service if the quality is below a certain level (good quality for example). As a kind of appreciation for its customers of the first type for accepting the stream even with a lower quality than expected, the enterprise offers 10% reduction in its streaming rates for these customers.

The company dynamically advertises its current streams with a guideline estimation of the expected quality derived from the current network conditions. The published quality is just a guideline and the user should be aware of a possible degradation and may be improvement could happen without prior notice. On the client side, media streams are played back to the client, which is facilitated by a media playback service. At the same time, the received media stream is captured by a QoS estimation service running on the client side. The QoS estimation service uses the media stream and the RTP/UDP/IP headers of the received packets in order to estimate the quality of the stream.

Once the quality is estimated on the client side, a record of the session details is built including a session unique identifier, duration and quality. This record is then reliably transmitted (in an XML format to guarantee platform indepen-

dence) to the media server or possibly to a separate server (to reduce the load on the media server). Once the session record arrives, the session details are extracted from the XML file. Then the cost is calculated and the user's profile is updated with the session details including the duration, quality and most importantly the cost for receiving that stream. Figure 2 illustrates the proposed architecture. Once the user's profile is updated with the session details, a bill could be produced possibly on a daily, weekly or monthly basis.

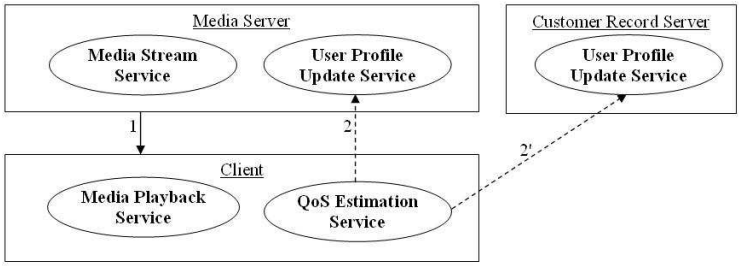


Fig. 2. Service Architecture

6 QoS Estimation Service: Details and Interface

As discussed earlier in section 5, the received media stream is played back and at the same time the stream is captured by a QoS estimation service running on the client side in order to estimate the quality. The QoS estimation service needs to be a non-intrusive based quality estimation service as it is running in real-time and because the original stream before being transmitted over the IP network is not available at the receiver side. The only material available at the receiver side is the degraded stream. This degradation is due to the nature of the IP network.

As described in section 4 the E-Model is used to estimate the speech quality non-intrusively. The E-Model uses some of the network and terminal impairments and utilizes the information contained in the RTP/UDP/IP in order to estimate the quality. The pseudocode for the E-Model is shown in Figure 3. From the RTP header information described in section 3, packet loss ratio and packet loss distribution for the stream can be measured. From such information, the E-Model estimates the degradation due to packet loss. Also, Degradation due to delay is estimated by the E-Model using the time stamp information contained in the RTP packet header. By combining degradation due to different impairments, the overall degradation can be computed as shown in Figure 3. This overall degradation can then be mapped into a quality estimation on a 1 to 5 scale as described in section 4.

```
%Sum the degradation due to all factors
DegradationSum=Degradation_Loss+Degradation_Delay+Degradation_Echo+ . . .
R0=ReferenceQualityValue
%Subtract Sum of Degradation from a reference value to calculate R-Rating factor
R=R0-DegradationSum
%Convert the R-Rating factor to a quality estimation on the range 1 to 5
QualityEstimation=mapR(R)
```

Fig. 3. Pseudocode for the E-Model QoS Estimation Service

The QoS estimation service can be called from within the media playback service. The media playback sends the session reference to the QoS estimation service, which could be done using a command button. The QoS estimation service then replies by sending its quality estimation to the media playback which could be shown to the user via a label within the media playback service. More details about this service and quality estimation issues can be found in [8].

7 Conclusions

In this paper a media service provisioning scenario is presented, where QoS is estimated through a specialised service. In this scenario an enterprise offers streamed media to its clients with a variable quality. Upon receiving the media stream, the quality of the stream is estimated and based on the quality of the received stream. The customer's profile is updated with the details of the stream, and the customer is billed accordingly.

References

1. Collins, D. Carrier grade voice over IP. *McGraw-Hill Companies,2nd edition*, 2003.
2. Low, C. The Internet telephony red herring. *Global Telecommunications Conference*, 1996.
3. Rosenberg, J. and Lennox, J. and Schulzrinne, H. Programming Internet telephony services. *IEEE Network* , 13(3):42–49, 1996.
4. Schulzrinne, H. and Wedlund, E. Application-layer mobility using SIP. In *IEEE Service Portability and Virtual Customer Environments*, pages 29–36, 2000.
5. Schulzrinne, H. and Rosenberg, J. Signaling for Internet telephony. In *Proceedings Sixth International Conference on Network Protocols*, pages 298–307, 1998.
6. Nachiappan, N. and Sjoqvist F. Survey of Voice over IP (VoIP). *tech. rep., Standford University* , 2004.
7. ITU-T. Recommendation G.107 - The E-model, a computational model for use in transmission planning. *International Telecommunication Union-Telecommunication Standardization Sector (ITU-T)*, 2005.
8. Mousa Al-Akhras. Packet Loss in Voice over Internet Protocol Networks. *Ph.D. Thesis (To be Submitted), School of Computing, Faculty of Computing Sciences and Engineering, De Montfort University, UK*, 2007.

A Novel Approach to Web Services Discovery

Marco Comerio

Universitá di Milano - Bicocca,
Via Bicocca degli Arcimboldi 8, 20126 Milano Italy
comerio@disco.unimib.it

Abstract. The discovery of a Semantic Web service (SWS) is the act of locating a machine-processable description of a SWS-related resource that may have been previously unknown and that meets certain functional criteria. The increasing availability of SWSs that offer similar functionalities requires the discovery process to be enhanced with a selection phase that considers non-functional properties (NFPs) of the SWSs. This paper proposes a technique to enrich SWS requests and descriptions with the specification of NFPs and a novel approach to a NFP-based SWSs discovery.

Key words: Web Services Discovery, Semantic Web Services, Non-functional properties, Semantic Matching.

1 Introduction

Service-Oriented Computing (SOC) is a computing paradigm that proposes services as the basic constructs for the development of rapid, low-cost and easy-to-compose distributed applications in heterogeneous environments [5].

Currently, Web services (WSs) are the technology enabling the implementation of the SOC paradigm to develop Web processes accessible within and across organizational boundaries. Nevertheless, there is a growing consensus that WSs alone are not sufficient to develop valuable and sophisticated Web processes due to the degree of heterogeneity, autonomy, and distribution of the Web [1].

In order to address this problem, an integrated technology for the next generation of the Web by combining Semantic Web technologies and Web services has been proposed. Semantic Web services (SWSs) allow for the (semi)-automatic development of Web processes representing complex interactions between organizations.

The discovery of a SWS is the act of locating a machine-processable description of a SWS-related resource that may have been previously unknown and that meets certain functional criteria. The increasing availability of SWSs that offer similar functionalities requires the discovery process to be enhanced with a selection phase that considers non-functional properties (NFPs) of the SWSs. NFPs can be considered to be constraints beyond the functionalities of a SWS. Therefore, NFPs might be quite relevant to match a SWS request with a SWS description. In fact, even if a SWS matches the requested functionalities, it can still be unacceptable in terms of NFPs (e.g., cost is too expensive).

The management of NFPs is not a simple task because: (i) NFPs are characterized by several properties; (ii) NFPs are of different kinds and (iii) NFPs are often inter-dependent. All these problems can be addressed by developing ontologies of NFPs (OntoNFPs) to formalize definitions, relations, dependencies, heterogeneous measurements and evaluation methods.

This scenario highlights two needs: (i) enrichment of SWS descriptions with a well-defined set of NFPs and (ii) enhancement of the matching functionality between SWS requests and descriptions. The paper is organized as follows. Section 2 presents the state of the art focusing on approaches available in the literature for describing NFPs and for matching SWS requests and descriptions. Section 3 proposes a technique for describing NFPs. Section 4 presents a novel approach for a NFP-based SWS discovery process. Finally, Section 5 draws conclusions and shows the most relevant open problems of my research activity.

2 Related Work

In the literature, several approaches for describing NFPs and for matching a service request with a service description are available. The importance of NFPs to support sophisticated service discovery, service selection, automated service negotiation and dynamic service substitution is the focus of [3] where a model to describe the domain independent NFPs of services is proposed. This work can be considered as a starting point towards a semantically enabled solution for NFP-based service discovery and selection.

Several approaches aim to create a model for describing various aspects related to SWSs. WSMO [9] is an example of these approaches. Regarding the description of NFPs, [6] discusses the current limitations of modelling NFPs with WSMO and makes a set of proposals towards a richer NFPs modelling support. One of the proposed approaches consists of describing NFPs according to the model for capabilities in WSMO. NFPs are defined by using logical expressions as pre/post-conditions, assumptions and effects are being defined in a capability. The new technique for modelling NFPs described in Section 3 can be used to enrich the one in [6] with the use of policies. A policy permits the specification of an offered service level agreement based on NFPs values.

An algorithm to measure the degree of matching between two concepts is proposed in [4]. The different values of matching degrees (namely exact, plug in, subsumes and fail) are determined by the minimal distance between concepts in a reference taxonomy tree. The proposed solution is based on DAML-S and it is performed by considering only the capabilities of a service in terms of inputs, outputs, preconditions and effects. NFPs are not considered as inputs of the matching algorithm.

In [7], NFPs are modelled with an extension of the *nonFunctionalProperties* class already defined in WSMO that supports a richer description of NFPs characteristics (e.g., metricName, valueType, MeasurementUnit). The semantic matching is performed between qualities of service (QoSs) requested by an user and offered by several providers. The proposed method consists in normaliz-

ing QoS values in the range [0..1], scaling the value ranges with the maximum and minimum values of each quality metric and evaluating the global degree of matching for each provider. This method deals only with QoSs that are a subset of NFPs. The heterogeneity of the full set of NFPs makes the proposed method difficult to apply. For example, the normalization of NFPs values is a complex activity because NFPs are strictly related to a reference domain.

3 NFP Description

NFPs are able to enrich the description of a service. Due to the wide range of possible NFPs, three different perspectives can be classified: (i) *Business*: NFPs related to service nature and described from a business perspective. An example is Payment Methods (i.e., acceptable methods to make a payment); (ii) *WS Provider*: NFPs related to service delivery and described from a technological perspective. An example is Performance (i.e., how fast a WS request can be completed); (iii) *Web service*: NFPs related to the implementation of the Web service. An example is the percentage of failure of a payment operation (i.e., the frequency of incorrect behavior of a payment operation).

Ontologies of NFPs (OntoNFPs) are used as formal references that fully describe properties and relations of each NFP. In this paper, I propose a technique for the description of two different types of NFPs: offered NFPs (offNFPs) to be included in SWS description and NFP constraints (conNFPs) to be included in SWS requests. OffNFPs are NFPs supported by the provider of the SWS. A SWS can expose different sets of offNFPs for the same functionalities. For example a SWS can offer a cost of 10 Euros and a response time of 10 seconds, or a cost of 20 Euros and a response time of 5 seconds. In order to express different sets of NFPs, the concept of *NFP-policy* as a joint offer of offNFPs is used. OffNFPs are described by:

- a *property name* (e.g., minimumCost), which represents a concept of OntoNFP;
- a *value* (e.g., 45), which is the value assumed by the property;
- a *measurement Unit* (e.g., USD), which represents the attribute of the relation pattern which are fillers of properties of an OntoNFP.

Otherwise, conNFPs are requirements that the requester of a SWS wants to have satisfied and they are collected in *NFP-Requests* that represent the requester-side counterparts of NFP-policies. ConNFPs are described by:

- *Property name* (e.g., Cost);
- *Constraint Expression*: can be composed by several items each one specified with:
  - *Constraint Operator* (e.g., =, between, exist);
  - *Constraint Value* (e.g., 50; [20..200], medium);
- *Measurement Unit* (e.g., USD);
- *Relevance*: defines the importance of a NFP constraint. It assumes a value in the range [0..1], where 0 means weak constraint and 1 means strong constraint (i.e., if the NFP is not exactly matched the service cannot be invoked).

4 Semantic Discovery Process

Fig.1 shows a novel approach to WS discovery that enriches the traditional UDDI-based process with the use of semantics and evaluation of NFPs. The proposed SWSs discovery process is composed of five different phases.

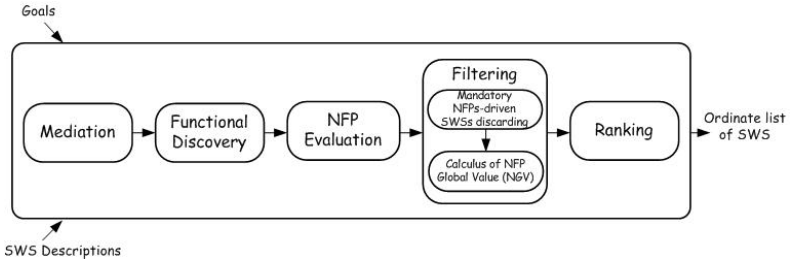


Fig. 1. Semantic Discovery Process

The first phase is *Mediation* which at setup time defines mediators to support matching functionalities. There are two main steps: (i) the definition of the mediators, that are matching rules for pairs of conNFPs and offNFPs; (ii) the definition of the meta-mediators, that identifies which matching rules need to be invoked for a NFP-Request.

After *Mediation* the discovery process proceeds at runtime with the *Functional SWS Discovery* that is the traditional SWSs discovery process. The activity has the goal of finding a list of SWSs that totally satisfy the functionalities required by an end-user [2].

The *NFP Evaluation* phase is in charge of exploiting the matching rules stored in mediators and meta-mediators to compute a set of numerical values in the range [0..1]. Each value expresses the degree of matching between a conNFP and an offNFP. The result of the *NFP Evaluation* is a set of values in the range [0..1] associated with each NFP-policy of each SWS identified in the *Functional SWS Discovery*.

This result is the input of the *Filtering* phase that consists in discarding SWSs with NFP-Policies that do not satisfy user minimum requirements and in evaluating a global degree of matching of each NFP-Policy. This activity is composed of two steps: (i) *Mandatory NFPs-driven SWSs discarding*: checks if the degree of matching is equal to 1 for all conNFPs with Relevance equal to 1. Otherwise, the related NFP-Policy is discharged and it is no longer considered during the selection process. (ii) *Calculus of NFP Global Value (NGV)*: for each NFP-Policy evaluates the NGV with the following formula:

$$NGV = \sum_{i=1}^{n} m_i * r_i \tag{1}$$

where m_i is the result of matchmaking the i-esim conNFP and r_i is the relevance of the i-esim conNFP. The result is a value in the range [0..n] that identifies the global degree of matching of a NFP-Policy.

Finally, the *Ranking* phase sorts the policies of the SWSs on the basis of their NGV values.

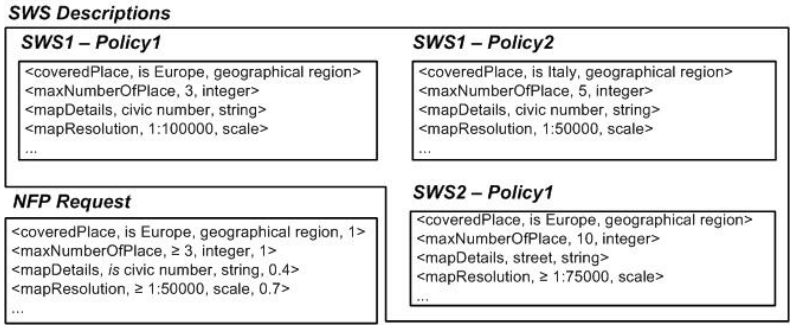


Fig. 2. Examples of NFP-Request and SWS Descriptions

In order to illustrate the proposed approach, let us consider the following scenario: a user is searching for a WS that is able to provide a map with the best route to cover a set of places. The NFP-Request (see Fig.2) is composed of a set of conNFPs expressed by <propertyName, constraintExpression, measurementUnit, relevance>. Let us suppose that mediators and meta-mediators have been already defined and *Functional SWS Discovery* has returned a set of similar SWSs, each associated with different NFP-policies characterized by different values of offNFPs. As shown in Fig.2, each offNFP is expressed by <propertyName, value, measurementUnit>.

NFP evaluation starts with the invocation of the meta-mediator associated to the NFP-Request. This component uses matching rules defined by mediators to evaluate the degree of matching between pairs of <offNFP, conNFP>. The results are the following: SWS1(Policy1)=<1,1,1,0.3>, SWS1(Policy2)=<0,1,1,1> and SWS2(Policy1)=<1,1,0.6,0.6>. The *Filtering* phase takes these results as inputs and concludes that: (i) SWS1(Policy2) is discharged. It is not considered during the selection process because it does not satisfy the strong constraint related to covered places; (ii) the NGVs are equal to 2.61 for SWS1(Policy1) and 2.66 for SWS2(Policy1). Finally, the Ranking phase orders SWSs and the related policies on the base of NGVs. SWS2 with Policy1 is shown to the user as the best SWS to invoke.

5 Conclusion

In this paper, I propose an ongoing work concerning the definition of a NFP-based SWS discovery process. The proposed approach leads to several activities

I am working on. The first one deals with the use of the WSMO environment to make experiments in order to validate the proposal. I am defining an extension of the Web Service Modeling Language (WSML)[8] of WSMO that permits the specification of NFP-Requests in SWS requests (Goal in WSMO) and NFP-Policies in SWS descriptions. Moreover, I am working on the design of mediators. I am currently investigating the possibility of designing them by means of F-logic rules stating when a conNFP of a NFP-Request is satisfied by an offNFP of a NFP-Policy.

Another activity is related to the definition of formulas to compute the degree of matching between two NFP values. The degree of matching of pairs <conNFP, offNFP> depends on the constraint operator o expressed in the NFP constraint and it is computed by formulas like degree=f_o(OffValue,ReqValue). I am working to define the formulas related to the most common constraint operators.

Acknowledgements

The work presented in this paper has been partially supported by the European IST project n. 27347 SEEMP - Single European Employment Market-Place and the Italian FIRB project RBNE05XYPW NeP4B - Networked Peers for Business.

References

1. J. Cardoso and A. P. Sheth. Introduction to semantic web services and web process composition. In *Proc. of First Intl Workshop on Semantic Web Services and Web Process Composition (SWSWPC'04)*, San Diego, CA, USA, 2004.
2. E. DellaValle and D. Cerizza. The mediators centric approach to automatic web service discovery of glue. In *Proc. of the 1st Intl Workshop on Mediation in Semantic Web Services (MEDIATE2005)*, pages 35–50, Amsterdam, The Netherlands, 2005.
3. J. O'Sullivan, D. Edmond, and A. ter Hofstede. Formal description of non-functional service properties. In *Technical report*, Queensland University of Technology, Brisbane. Available from http://www.servicedescription.com/, 2005.
4. M. Paolucci, T. Kawamura, T. R. Payne, and K. P. Sycara. Semantic matching of web services capabilities. In *Proc. of the First Intl Semantic Web Conference on The Semantic Web (ISWC '02)*, pages 333–347, London, UK, 2002.
5. M. Papazoglou, P. Traverso, S. Dustdar, F. Leymann, and B. Krämer. Service-oriented computing: A research roadmap. In *Service Oriented Computing (SOC)*, Dagstuhl Seminar Proceedings, 2006.
6. I. Toma, D. Foxvog, and M. C. Jaeger. Modeling qos characteristics in wsmo. In *Proc. of the 1st workshop on Middleware for Service Oriented Computing (MW4SOC 2006)*, pages 42–47, New York, NY, USA, 2006.
7. X. Wang, T. Vitvar, M. Kerrigan, and I. Toma. A qos-aware selection model for semantic web services. In *Proc. of the 4th Intl Conference on Service-Oriented Computing (ICSOC'06)*, pages 390–401, Chicago, IL, USA, 2006.
8. WSML. *The Web Service Modeling Language (WSML).* Available at: http://www.wsmo.org/TR/d16/d16.1/v0.21/20051005/, 2005.
9. WSMO. *The Web Service Modeling Ontology (WSMO). Final Draft.* Available at: http://www.wsmo.org/TR/d2/v1.2/20050413/, 2005.

Verification of WS-CDL choreographies

Flavio Corradini, Francesco De Angelis, Alberto Polzonetti

Department of Mathematics and Computer Science,
University of Camerino, Via Madonna delle Carceri 9,
62032, Camerino (MC), Italy
e-mail: {*name.surname*}@unicam.it

Abstract. In this paper we report an approach to formal verification of web service composition expressed using the WS-CDL language by means of model checking. We analyze the WS-CDL language source and isolate the necessary information to build a PROMELA model suitable for the verification using the SPIN model checker.

1 Introduction

In recent years, the need of distributed applications able to support information systems has driven the development of new technology for the vision of the Service Oriented Computing. Currently, this vision is implemented with the web service technology that rely on WSDL [1], SOAP [2], UDDI [3], WS-BPEL [4], WS-CDL [5] and many other WS-* standard.

In this context it is an important matter to deal with verification and validation, moreover, the needs of models specification is very important to show how different services can interact with each other by exchanging messages. Indeed, like any other software application composed by processes, or components, or objects, and so on, an application built from services needs a level of dependability that cannot be ensured with approaches that lack formal verification. Although, existing model checking [6] techniques can be extended or adapted to deal with new problems arising in the web service world.

In this paper we propose a technique to model check web services choreographies expressed in the WS-CDL language. WS-CDL [5] [7] (Web Service Choreography Description Language) is an XML-based language for specifying choreography models that describe collaborations between a collection of services that interact each other to achieve a goal. Each involved party wishes to remain autonomous and no party drive the interactions because there is not a centralization point of control. A choreography does not describe internal action of participants but concerns only with externally visible effects capturing them from a global perspective. We use WS-CDL as a model for the verification and we provide a translation of this model to a PROMELA model used by the model checker SPIN to verify properties [8].

SPIN is a state-of-the-art model checker developed by G.J Holzmann [8] based on automatons. It allows simulation and verification of models written in

PROMELA (Process Meta-Language) that are composed by a set of communicating processes that exchange messages along channels.

For its maturity an potentiality SPIN is a widely used model checker used in various application domain and several approaches for the verification of web services compositions, included the one presented here, are based on it.

2 WS-CDL to PROMELA Translation

We show the mapping from WS-CDL to PROMELA using a little example composed by four participants, each one is implemented as a web service. The participants are represented with four *roleType*: a *Buyer*, a *Seller*, a *CreditChecker* and a *Shipper*. The Buyer starts the interactions requesting a quote to the Seller. After that the seller has replied, the Buyer can accept the quote or can starts a bartering with the Seller using messages to update the proposed quote. When the Buyer accept the quote sends two messages to the seller: one carries the accepted quote and the other carries the channel that can be used for the communication with the Buyer itself. The Seller checks the credit card information to the CreditChecker and, if this operation is successful, forward the channel to the shipper with all the detail to complete the delivery. Now, the Shipper has a channel to directly communicate to the Buyer and can send it the right information for the delivery. Clearly, if the CheckCredit doesn't return a right credit information to the Seller the Buyer cannot complete the purchase.

The translation from WS-CDL to PROMELA includes the identification of the actors of the system modeled by web services and that can be mapped as processes in PROMELA. For each `participantType` in WS-CDL a process is built with the name used in the attribute `name`. Moreover a label `end_`*`processname`* is used to mark the initial state. This expedient is used to prevent a system deadlock if a process is never called. This can happen - as in the example - when a web service/process is not involved in the choreography because something prevent this. (In the example the `Shipper` isn't used if the `CreditChecker` returns a `CreditReject` message to the `Seller`). This is realistic because represents a web service that is waiting for a request, and it can be in that state also when the choreography will be complete and it had not participated.

Each channel variable defined in the choreography becomes a channel variable in PROMELA. In the mapping each channel can carry two `int`, the first represent the action invoked and the second is a constant that represents the type of the message exchanged by the processes. To do this, for each action and for each message type a constant is defined.

About channels, there is the necessity to make a distinction between channels used to send, and receive, data and channels used for channel passing. If a channel can be used for channel passing it can be translated into two channels in PROMELA. One for data ({`int, int`}) and one for sending other channels ({`int, chan`}). The distinction is made looking at the `channelType` that shows which channels are used to pass other channels and which is the type of the passed channel. Variables that has this latter `channelType` are not translated

into global channels because this will be local to the process that sends it for first.

Each *activity* is mapped looking at several information that are present inside an `interaction`: (i) `relationshipType`, `fromRoleTypeRef` and `toRoleTypeRef`; (ii) `channelVariable`; (iii) `exchange name`, `informationType` and `action`. The `relationshipType` is used to identify the processes involved in the interaction. Using the `fromRoleTypeRef` value (resp. `toRoleTypeRef`), and looking at `participantType` definitions, it is possible the identification of the `name` of the `participantType` with the given role. This was used to name a process in PROMELA and it is the process that starts the interaction sending the message (resp. the process that receive the message). The message for the sending process is built as: `channelVariable! exchange name`, `informationType` while the message for the receiving process is the complementary one obtained using "?" instead of "!".

```
<interaction name="BuyerRequestsQuote"
  channelVariable="Buyer2SellerC"
  operation="requestForQuote"
  initiate="true">
  <participate
    relationshipType="BuyerSeller"
    fromRoleTypeRef="BuyerRoleType"
    toRoleTypeRef="SellerRoleType"/>
  <exchange name="request"
    informationType=
              "RequestForQuoteType"
    action="request">
    <send/>
    <receive/>
  </exchange>
  <exchange name="response"
    informationType="QuoteType"
    action="respond">
    <send/>
    <receive/>
  </exchange>
</interaction>
```

```
/* the Buyer process... */
BuyerToSellerC!REQUEST,
               RequestforQuoteType;
BuyerToSellerC?RESPONSE,
               QuoteType;

/* the Seller process... */
BuyerToSellerC?REQUEST,
               RequestforQuoteType;
BuyerToSellerC!RESPONSE,
               QuoteType;
```

Fig. 1. Activities

Continuing with the translation, in Figure 1 there is an example of a request-response interaction between the `Buyer` and the `Seller`. This is an interaction that is composed by a response after a request, both mapped using a pair of complementary statements in PROMELA. The `action` attribute shows if the message is a send message or a receive message from the perspective of the `fromRoleTypeRef` role. If `action` is *"request"* then the message starts from the `fromRoleTypeRef` while if it is *"respond"* the message is sent by the `ToRoleTypeRef`. This is necessary because in WS-CDL there are three kinds of interactions that respectively foresee only a *request*, only a *response* or - as in the example - a *request-response*. This seems misleading but the "`From`" and

"`To`" suffixes of `RoleTypeRef` refers to the `relationshipType` and not to the interaction itself.

The mapping of `sequence` is straightforward because relies on the sequence of statements in PROMELA. The `choice` and the *repetitions* (modeled with `workunit` in WS-CDL) are more complex because the global behavior of a participant must be projected locally to PROMELA processes. While this is simple for a single interaction activity that use complementary PROMELA statements, for a `choice` there is the necessity of an `if` statement in each PROMELA process involved in the choice. Same considerations apply to repetitions that require a `do` statement for each involved process. Then, the content of the `if` (resp. `do`) is constituted of complementary statements performed in the `choice` (resp. in the *repetition*).

For example, when the `Seller` performs the interaction that has the name `CheckCredit`, the `CreditChecker` respond with a `CheckOk` or with a `CheckFail` message. This imply a `choice` in WS-CDL and an `if` in PROMELA, both for the `Seller` and the `CreditCheker` as shown in Figure 2.

```
<choice> <sequence>
  <interaction name="CheckFails"
  channelVariable="Seller2CreditChkC"
  operation="creditCheck">
    <participate ... />
    <exchange name="checkCreditFails"
    informationType=
                "CreditRejectType"
    action="respond">
    <send/> <receive/>
    </exchange>
  </interaction>
  ...notify the Buyer...
</sequence> <sequence>
  <interaction name="CheckOk"
  channelVariable=
              "Seller2CreditChkC"
  operation="creditCheck">
    <participate  ... />
    <exchange name="checkCreditPasses"
    informationType=
                "CreditAcceptType"
    action="respond">
    <send/> <receive/>
    </exchange>
  </interaction>
  ... interaction to delivery...
</sequence> </choice>
```

```
/* CreditChecker */
if
    ::SellerToCreditChkC!
            CREDITCHECKFAILS,
            CreditRejectType->skip;
    ::SellerToCreditChkC!
            CREDITCHECKPASSES,
            CreditAcceptType;->skip;
    /*
     * In both cases the
     * execution ends
     */
fi;

/* Seller */
if
    ::SellerToCreditChkC?
            CREDITCHECKFAILS,
            CreditRejectType->
    /*
     * ...notify the Buyer
     * about the fail...
     */

    ::SellerToCreditChkC?
            CREDITCHECKPASSES,
            CreditAcceptType ->
    /* interaction to delivery */
fi;
```

Fig. 2. Choice

The `parallel` construct is a little bit different. When in the choreography there is a parallel construct the processes in the PROMELA model must instantiate one subprocess for each parallel activity.

Each process must handle its "part" of parallel activities. Supposing that a participant must handle two parallel activities, it must instantiate two subprocesses that synchronize on a specific channel created by the participant. While the complementary actions can be performed by other processes in the usually fashion. For example in Figure 3 is shown a synchronization between two processes, *subprocess1* and *subprocess2*, controlled using the local channel *syncro*. Although, the participant involved in the parallel interactions can be different, one participant can instantiate subprocesses that interact with other processes that are already in concurrent execution without requiring they to instantiate other subprocesses. This happens when the `fromRoleTypeRef` is the same for the parallel activities but the `toRoleTypeRef` is different for them.

```
proctype Process(){
        /* do something... */
        chan syncro = [2] of {int}
        /* for 2 activities */
        run subprocess1(syncro);
        run subprocess2(syncro);
        do
        ::full(syncro)->break;
        od;
        /* do something else... */
}
```

Fig. 3. Parallel

We summarize the translation in the Table 1 showing the relation between language constructs.

Table 1. Synoptic table of the translation

WS-CDL	**PROMELA**
`participantType`	a PROMELA process
`channelVariable`	one channel or two channels if the `channelVariable` is used for passing channels
`activity`	statement in the form: `channelvariable[!|?]exchange name, informationType`
`sequence`	sequence of statements
`choice`	`if` statement inside the involved processes
repetitions (formally `workunit`)	`do` statement inside the involved processes
`parallel`	instantiation of subprocesses

3 Conclusions

In this paper we report an approach to formal verification of web service composition expressed using the WS-CDL language by means of a translation to a PROMELA model suitable for the verification with the SPIN model checker.

At the moment, the developed mapping is only a proof of concepts to show how a choreography can be tested using existing model checking techniques. The mapping doesn't refer to the whole specification but only to the main concepts, namely, the main constructs that regulate compositions like interactions, sequence, choice and parallel.

Now, works in progress go in two directions. First of all, we would implement the mapping to realize a prototype able to handle WS-CDL choreography and to generate correct SPIN models. We would extend the mapping including more construct including the possibility to call other choreography, exception handling, finalizer, etc. Moreover, a particular importance must be given to counterexample presentation to guide the developer in the correction of errors. Finally, we want to investigate the formal basis of WS-CDL to theoretically prove the validity of the mapping and of the approach, regarding also the issue that not all choreographies are projectable as shown in [9] and [10].

References

1. Christensen, E., Curbera, F., Meredith, G., Weerawarana, S.: Web Services Description Language (WSDL) 1.1 (15 March 2001)
2. Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J., Nielsen, H.F.: Simple Object Access Protocol
3. Bellwood, T., Capell, S., Clement, L., Colgrave, J., Dovey, M.J., Feygin, D., Hately, A., Kochman, R., Macias, P., Novotny, M., Paolucci, M., von Riegen, C., Rogers, T., Sycara, K., Wenzel, P., Wu, Z.: UDDI spec technical committee draft (2004)
4. Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I., Weerawarana, S.: Business Process Execution Language for Web Services Version 1.1 (July 2002)
5. Kavantzas, N., Burdett, D., Ritzinger, G., Fletcher, T., Lafon, Y., Barreto, C.: Web Services Choreography Description Language Version 1.0 (9 November 2005)
6. Edmund M. Clarke, J., Grumberg, O., Peled, D.A.: Model checking. MIT Press, Cambridge, MA, USA (1999)
7. Barros, A., Dumas, M., Oaks, P.: A Critical Overview of the Web Services Choreography Description Language (WS-CDL) (March 2005)
8. Holzmann, G.J.: The SPIN Model Checker: primer and reference manual. Addison Wesley (2003)
9. Bravetti, M., Zavattaro, G.: Towards a Unifying Theory for Choreography Conformance and Contract Compliance. In: SC'07 Software Composition 2007. (2007)
10. Carbone, M., Honda, K., Yoshida, N., Milner, R., Brown, G., Ross-Talbot, S.: A Theoretical Basis of Communication-Centred Concurrent Programming. W3C-Working Note (October 2006)

Advanced Language Constructs for Developing Intra-organizational Service Architectures

Sven De Labey and Eric Steegmans
K.U.Leuven, Dept. Computer Science, 200A Celestijnenlaan
B-3000 Leuven, Belgium
{svendl, eric}@cs.kuleuven.be

Abstract. Jini technology delivers a promising environment for building intra-organizational, adaptive service architectures in Java. Currently, however, the level of abstraction is too low, forcing programmers to implement interactions that should have been hidden by the middleware. Moreover, Jini's limited expressiveness for fine-tuning service selection makes it hard to program complex client-service interactions. In this paper, we present ongoing work on *JASA*, the Java Advanced Services Architecture. JASA provides middleware abstractions for service computing and provides a Java extension, *ServiceJ*, with specialized concepts for implementing complex client-service interactions.

1 Introduction

Service Oriented Architecture (SOA) started off as a paradigm for enabling cross-organizational cooperation through the use of standardized, interoperable protocols such as XML-based Web Services. This same paradigm is now rising beyond its original domain, exploring new territories such as intra-organizational plug-and-play service architectures. Jini [1] provides an environment for building these architectures in Java. Current Jini technology supports *dynamic service discovery* and *Quality of Service (QoS) constrained service lookup*, at the same time subscribing to the standard Java programming model and as such allowing programmers to interact with services as if they were plain Java objects.

In this paper, we review Jini in the context of client-service interactions. We point out a number of shortcomings and introduce *JASA*, an advanced service architecture based on *ServiceJ* [2], our programming language with full-blown support for client-service interactions. This paper is structured as follows. Section 2 reviews Jini and defines a number of shortcomings. Section 3 gives an overview of JASA whereas Section 4 discusses ServiceJ. Section 5 explains how JASA manages services. Related work is presented in Section 6 and Section 7 concludes.

2 Jini as a Framework for Service-Oriented Computing

Jini technology focuses on building *scalable*, *adaptive* network systems [1] for *dynamic* environments such as SOAs. In this Section, we focus on how clients interact with Jini services (Sect. 2.1) and point out a number of shortcomings (Sect. 2.2) that underly the design goals of JASA.

2.1 Client-Service Interactions in Jini

Jini services are registered in registries that act as *brokers* between clients and providers. Providers use a service registry to register the services they provide, whereas clients use that registry to obtain references to services running on the SOA. In this paper, we focus on (1) Jini support for *service queries* and (2) the adaptability of the Jini service architecture.

Service Queries. Before a client is able to find services, a provider must first *register* these services, as shown in Fig. 1 (**S**). On registration, a provider includes *service templates* (**T**), which are objects that contain additional information (location, price, ...) about that service. On service lookup, clients may fine-tune their service queries by including some of those templates. The lookup service then matches these templates with those that were included by the provider and returns a service that fully matches the request of the client.

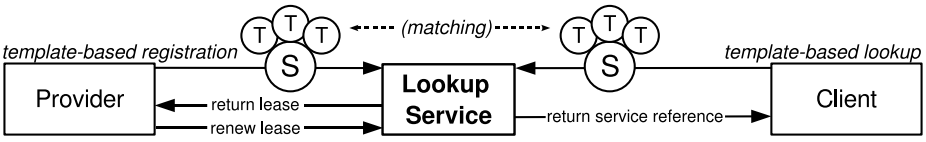


Fig. 1. Jini architecture overview

Adaptable Service Architecture. Jini relies on a *leasing* mechanism in order to guarantee that the lookup mechanism is synchronized with the actual state of the service architecture. On service registration, service providers are given a *lease* representing the time left until the service is automatically removed from the lookup service. This lease must be renewed periodically, otherwise the lookup service assumes that the provider has crashed, thus justifying service removal.

2.2 Shortcomings of Jini relative to Client-Service Interactions

Although basic support for service queries and service volatility is supported, the overall Jini architecture provides only limited support for *customized service interactions*. We identify three problems that lead to a list of goals for JASA:

1. **Low level of abstraction.** Programmers are responsible for interacting with the middleware when retrieving services (using templates) or when keeping services alive (using leases). These interactions obfuscate the business logic of an application, i.e. the actual interaction with the Jini service.
2. **Inflexible queries.** The expressiveness of templates is limited by two factors. First, only *exact matches* on *non-primitive types* are supported, thus disabling, for instance, searching for a printer that supports *more than* 20ppm. Second, templates are always combined *conjunctively*, thus disallowing searches for printers supporting *either* 600dpi *or* color printing.
3. **No Transparent Failover.** Leasing policies allow for an adaptive distributed system, but they still force the client to fail when a service becomes unreachable *during* a client-service session.

3 The Java Advanced Services Architecture

JASA combines concepts from Jini with *ServiceJ*, an extension of Java that introduces specialized language constructs for client-service interactions. This Section outlines JASA, whereas Section 4 introduces ServiceJ.

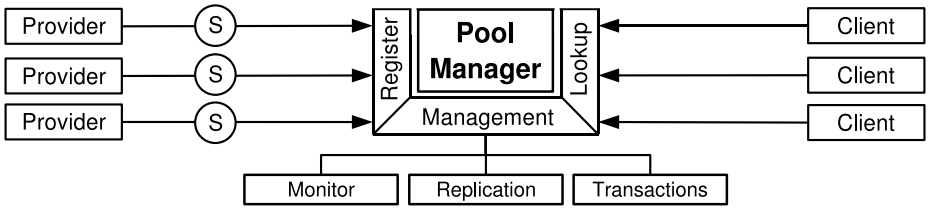


Fig. 2. Java Advanced Services Architecture

Service Pools. An overview of the JASA architecture is shown in Figure 2. The Jini lookup service is replaced by a more complex lookup mechanism, called the *Pool Manager*. This service manages *service pools*, which are sets of *interchangeable services* that conform to a *common service type*. For example, all services implementing the `Printer` interface are collected into a service pool of type `Printer`. The Pool Manager is accessed using one of the following services:

1. **Registration.** This service is used by *providers* to *advertise* and *unpublish* their services. Contrasting to Jini, providers no longer include *templates* and they no longer renew leases, as these issues are transparently dealt with by the middleware (see further). As such, service registration becomes trivial because providers only have to send a service reference to the pool manager.
2. **Retrieval.** This service is used by *clients* in order to retrieve a *pool* of interchangeable services, as explained in Section 4.
3. **Management.** This interface is used by services responsible for *monitoring* and for supporting cross-pool *consistency* and *transactions* (see Section 5).

4 ServiceJ – A Language for Client-Service Interactions

We propose a Java extension, called *ServiceJ*, with specialized constructs for client-service interactions. First, ServiceJ extends the type system with *type qualifiers* [3] to enable transparent service failover (Sect. 4.1). Second, ServiceJ provides developers with *declarative operations* for fine-tuning service selection (Sect. 4.2).

4.1 Type Qualifiers Specify Invocation Semantics

ServiceJ introduces a new kind of variables, called *pool variables*, that have special characteristics for interacting with remote services. Pool variables differ from normal variables in two ways:

– **Initialization.** Whereas Java variables are typically initialized by the programmer, pool variables are initialized by the JASA middleware. On method invocation, the middleware (1) obtains the service pool of type `T` from the Pool Manager, (2) selects a pool member, and (3) injects that reference into the pool variable. Then, the operation is invoked on that injected reference.

– **Fault-tolerant.** Pool variables transparently support *service failover*. If a method invocation fails, JASA transparently (1) selects another pool member, (2) injects it into the pool variable and (3) reinvokes the operation.

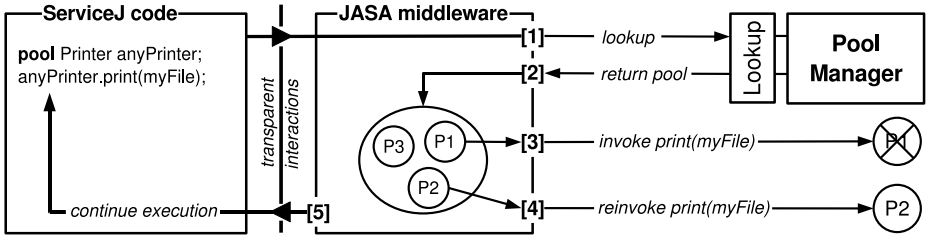


Fig. 3. Transparent service lookup and fault-tolerant client-service interactions

Both the JASA failover mechanism and the service pools are hidden for programmers, thus allowing them to concentrate on business interactions. The latter only have to decorate their pool variables with a *type qualifier*, **pool**, to indicate that the variable must be managed by the JASA middleware. The following code, for instance, prints a file (`myFile`, a normal variable) using a `Printer` service (`printer`, a pool variable):

```
pool Printer printer;     //create a pool of printers
printer.print(myFile);    //print 'myFile' on any available printer
```

Other qualifiers besides **pool** are defined for implementing different semantics such as *multicasting* method invocations. A detailed discussion on how type qualifiers influence application-middleware interactions is presented in [3].

4.2 Declarative Operations enable Fine-tuned Service Selection

Pool variables provide basic support for transparent failover, but we still need a mechanism for imposing *business requirements* and *QoS constraints* for fine-tuning service lookup. The *template-based query mechanism* of Jini is insufficient because templates cannot handle customized queries, thus forcing the programmer to write a considerable amount of boilerplate code to hardcode the query in the source code. We solve this problem by introducing *declarative operations*:

– **where.** This operation accepts a *boolean expression* representing QoS constraints. It triggers JASA to filter a service pool, retaining those pool members that comply with the constraint. For example, selecting those `Printer` services that support color printing is written as:

```
pool Printer p where p.supportsColorPrinting();
```

– **orderby.** This operation is used to specify user-defined preferences based on service-specific characteristics (e.g. price, location, ...). It triggers JASA to inject the service that best approximates the preferences of a user. For example, selecting the printer with the lowest printing cost is written as:

```
sequence Printer printer orderby printer.getCost();
```

Note that the **pool** qualifier is replaced by the **sequence** qualifier. This is a subqualifier of **pool** because it uses a more specific communication semantics: pools use *non-deterministic* selection whereas sequences use *deterministic* selection based on user preferences. See [3] for further information.

Runtime filtering. During compilation, *ServiceJ* is translated to normal Java code in order to support interoperability with existing Java applications. Arguments of **where** and **orderby** operations are transformed to separate *query objects* following the *Command pattern* [4]. Before an operation is invoked on a pool variable, these query objects are sent to the Pool Manager, where they are used to compose the desired service pool. This constrained pool is then returned to the client, where it is used by JASA to inject a pool member into the pool variable. Finally, the method is invoked on this injected pool member.

5 The JASA Management Architecture

Jini relies on a leasing mechanism to ensure that the lookup service has an up-to-date view on the service architecture. This works fine for volatile, mobile services, but it leads to excessive keepalive messaging in the case of *stable services* such as printers, calendars, etc. Therefore, we extend the JASA middleware with an *optimistic pooling* protocol for registering such stable services.

Optimistic pooling. This strategy no longer expects stable services to renew their leases temporarily. Instead, the Pool Monitor (see Fig. 2) assumes that stable services are *always available*. Failures are automatically detected after an unreachable service is injected into a pool variable and the transparent failover mechanism of JASA completely hides these problems for programmers. First, it notifies the Pool Manager about the failure, causing the failing reference to be removed from its service pool. Second, JASA injects another pool member and reinvokes the operation.

Other services related to pool management, such as the Transaction Manager and the Replication Manager cannot be discussed due to space restrictions.

6 Related Work

Jini underlies the development of JavaSpaces [5] and GigaSpaces [6], which both provide a distributed persistence and object exchange mechanism for code written in Java. Inspired by Linda [7], JavaSpaces and GigaSpaces use tuple matching algortihms for service lookup. Tuple matching, however, is implemented on top of Jini templates, leading to the disadvantages discussed in Sect. 2.2.

Another domain of related work is *grid computing* [8]. Similar to JASA services, *grid services* are registered and found through interactions with a *registry grid service* that ressembles our Pool Manager. The Java Commodity Grid Kit (CoG) [9], for instance, provides libraries for grid service lookup, but its query mechanism is weaker than Jini templates. It relies on string-based, exact matches and therefore fails to support expressive, type-checked, and customized service queries. Additionally, service management in JavaCoG requires a lot of boilerplate code, whereas JASA transparently manages these tasks.

A detailed comparison of JXTA [10], OGSI [11] and Jini support for Service Oriented Architecture was presented in [12] in the context of the Iceni framework [13]. Iceni is a dynamic management framework for service environments.

Layered on top of Jini, it provides two different views for each grid service: a *local* view through Jini interfaces and a *global* view based on OGSA [14] compliant web services. The main drawback is that the weaknesses of Jini propagate to the Iceni layer, such as the weak query mechanism and the obligation to refresh leases for stable services.

7 Conclusion and Future Work

Jini is currently the de facto standard for intra-organizational service architectures written in Java, but it provides no high-level concepts for complex client-service interactions. We have introduces JASA, an advanced middleware that hides the technical details of client-service interactions. JASA is based on ServiceJ, a language with specialized language constructs for *service queries* (**where** and **orderby**). Furthermore, JASA has additional protocols for managing services, such as optimistic pooling, replication management and transaction management.

Current work on JASA focuses on the ServiceJ-to-Java transformation. An interesting direction of future work is an evaluation on how frameworks layered on top of Jini, such as Iceni and JavaSpaces can be ported to work on top of JASA.

References

1. Jini Architecture Specification v2.1. (http://www.jini.org)
2. S. De Labey, M. van Dooren, and E. Steegmans: ServiceJ. A Java Extension for Programming Web Service Interactions. In: Proceedings of the 5th International Conference on Web Services, Salt Lake City, Utah (2007)
3. S. De Labey and E. Steegmans: A Type System Extension for Middleware Interactions. In: 1st Workshop on Middleware-Application Interaction. (2007)
4. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object Oriented Software. Addison-Wesley (1995)
5. Mamoud, Q.: JavaSpaces Technology: Beyond Conventional Distributed Programming Paradigms – www.javaspaces.org. (2005)
6. Gigaspaces Specification. (http://www.gigaspaces.com)
7. Carriero, N., Gelernter, D.: Linda in context. Commun. ACM **32**(4) (1989) 444–458
8. Lee, C., Talia, D.: Grid Programming Models: Current Tools, Issues and Directions. In: Grid Computing: Making the Global Infrastructure a Reality. (2003)
9. von Laszewski, G., Gawor, J., et al.: Features of the Java Commodity Grid Kit. In: Concurrency and Computation: Practice and Experience. (2001)
10. Gong, L.: JXTA: A Network Programming Environment. In: IEEE Internet Computing. (2001)
11. Sandholm, T., Seed, R., Gawor, J.: OGSI Technology Preview Core – A Grid Service Container Framework, www.globus.org/research/papers/. (2002)
12. Furmento, N., Hau, J., Lee, W., Newhouse, S., Darlington, J.: Implementation of a Service-Oriented Architecture on top of Jini, JXTA and OGSI. (2002)
13. Furmento, N., Lee, W., Mayer, A., Newhouse, S., Darlington, J.: ICENI: An Open Grid Service Architecture Implemented with Jini. In: ACM/IEEE SC 2002 Conference. (2002)
14. Foster, I., Kishimoto, H., Savva, A., Berry, D., et. al.: The Open Grid Services Architecture Specification, Version 1.0 – http://www.globus.org. (2005)

Service Referrals in BPEL-based Choreographies

Gero Decker[1], Oliver Kopp[2], Frank Puhlmann[1]

[1] Hasso Plattner Institut, University of Potsdam, Germany
{gero.decker,frank.puhlmann}@hpi.uni-potsdam.de
[2] Institute of Architecture of Application Systems, University of Stuttgart, Germany
oliver.kopp@iaas.uni-stuttgart.de

Abstract. Choreographies describe the interactions between two or more services from a global perspective and specify allowed service conversations. Choreographies typically do not rely on static binding, i.e. the participating services are not selected at design-time of the choreography. Some services might only be selected at runtime and this selection has to be propagated in the case of multi-lateral conversations. Hence, the notion of service referrals (also called link passing mobility) is recurrent in choreographies. In past work, we have proposed BPEL extensions for describing service choreographies, namely BPEL4Chor. This paper closely investigates the link passing mobility capabilities of BPEL4Chor and illustrates their semantics using π-calculus.

1 Introduction

Service-oriented architecture (SOA) is an architectural style for information systems that relies on message exchanges between loosely coupled services [4]. Web services are typically used for implementing an SOA. The first standards in the field of Web services such as XML, WSDL, and SOAP put simple request/response interactions between services into the center of attention. Further standards like BPEL [10] enable the implementation of services that engage in more complex interaction scenarios with its environment. This second generation of Web services supports long-running conversations in bilateral and multi-lateral settings.

As BPEL only considers conversations from the perspective of an individual service, a new viewpoint was proposed to capture conversations from a global point of view. These *choreographies* define allowed conversations and therefore serve as interaction agreement between different parties. In some choreographies it is already defined which concrete services are to participate in the conversations. Imagine e.g. a collaboration between two companies who defined their respective interaction behavior in the choreography. In other choreographies, a notion of roles or participant types can be found, leaving it open to select participating services just before starting a conversation or even after the conversation has already started. As typically more than two services participate in such conversations, it is important to pass on the reference to the concrete service during the conversation. The Service Interaction Patterns [1], a catalog of common patterns in interaction scenarios, highlight such *link passing mobility* as recurrent

phenomenon under the name of *Request with Referral*. We therefore conclude that support for link passing mobility is an essential feature of choreography description languages. As an alternative to existing languages and in order to enable more direct integration of service orchestrations and choreographies, we have introduced BPEL extensions for choreography modeling (BPEL4Chor) in [5]. In this paper we are going to closely investigate how link passing mobility is realized in BPEL4Chor. In order to provide unambiguous semantics we use π-calculus, a modern process algebra that inherently supports link passing mobility. An extended discussion on the advantages can be found in [12].

The remainder of this paper will be organized as follows. The next section discusses related work, before section 3 gives a short overview of BPEL4Chor. The main contribution will be found in section 4 where link passing mobility in BPEL4Chor is discussed. Section 5 concludes and points to future work.

2 Related Work

Since the formal semantics of BPEL4Chor is based on π-calculus, we refer to earlier work on the formal representation of process and interaction patterns [13,7]. Dynamic binding in π-calculus is introduced in [11]. In a nutshell, the π-calculus is based on a set of agent identifiers (denoted with uppercase letters) and another set of names (denoted with lowercase letters). Names are a unification of concepts known as pointers, links, channels, etc. The agents of the π-calculus can interact by sending names via names used as channels, denoted as $\overline{a}\langle b\rangle$, and receiving names via names used as channels, denoted as $a(x)$. The ordering of the send and receive operations can be sequential, denoted as $a(b).\overline{b}\langle x\rangle.\mathbf{0}$, parallel, denoted as $a(b).\mathbf{0} \mid \overline{x}\langle y\rangle.\mathbf{0}$, or exclusive, denoted as $a(b).\mathbf{0} + \overline{x}\langle y\rangle.\mathbf{0}$. Each execution path is terminated with $\mathbf{0}$. Furthermore, agents can create new, unique names during their execution, denoted as νx, where x is the new name. Due to space limitations, we refer to [9] for an extended introduction. Existing approaches for formalizing BPEL do not support dynamic binding and are hence improper for an extension to choreographies [8,3].

A strong competitor for BPEL4Chor is given by WS-CDL as a choreography language. While WS-CDL is able to support most of the service interaction patterns, it also introduces different realizations for the workflow patterns. Notable, these are difficult to map to BPEL [6]. Since BPEL is the state-of-the-art orchestration language for business processes, a mismatch between choreography and orchestration languages should be avoided. This paper focuses on an extension of BPEL to overcome these limitations. Other competitors are given by BPML [2] and BPSS [14]. However, both are outdated nowadays.

3 BPEL4Chor Overview

In contrast to other choreography languages such as BPSS and WS-CDL, BPEL4Chor does not have interactions as basic building blocks but rather communication actions, i.e. send and receive actions. Therefore, control flow depen-

Listing 1 Participant behavior description for a migration service

```
<process name="migrationservice" targetNamespace="urn:visa:ms"
  abstractProcessProfile="urn:HPI_IAAS:choreography:profile:2006/12">
  <sequence>
    <receive wsu:id="ReceiveEmployeeDetails" createInstance="yes" />
    <opaqueActivity name="PrepareVisaApplication" />
    <invoke wsu:id="SubmitVisaApplication" />
    <receive wsu:id="ReceiveConfirmation" />
  </sequence>
</process>
```

dencies are not defined between interactions but locally between communication actions. BPEL4Chor uses *participant behavior descriptions* (PBDs) for this purpose. For each participant type a PBD has to be provided. PBDs are a special kind of abstract BPEL processes. This enables to define control and data flow in choreographies as it is the case in BPEL.

In contrast to classic BPEL, where send and receive actions include information about who the respective interaction partner is (through the `partnerLink` and `operation` attributes), PBDs have to be glued together in a separate artifact, the *participant topology*. This document captures the structural aspects of the choreography and defines which two communication actions from the PBDs are connected through a message link. While the PBDs and the topology are free of web-service-specific configuration, *participant groundings* are introduced to provide the mapping of elements in the topology to WSDL specifications.

Listings 1 and 2 show two BPEL4Chor artifacts of a choreography description, where a visa is to be organized for a new employee. As the employing company has outsourced all migration related activities, it sends the employee's details to a migration service. This service prepares and submits a visa application to the government's immigration office. The immigration office sends a nomination approval to the employer which is needed for picking up the visa from the embassy. In addition, a confirmation is sent to the migration service.

4 Link Passing Mobility in BPEL4Chor

The example from the previous section illustrates the main concepts in BPEL4Chor. While merely control and data flow is defined in the participant behavior descriptions, the main structural setting can be found in the topology. Here, participant types and participant references are defined. It is possible that several references or even reference sets are used for one participant type. This indicates that different participants of the same type are involved in one conversation. Imagine e.g. a logistics scenarios where several shippers transport goods from a production site to a warehouse or imagine a bidding scenario where different bidders take part in one auction.

Listing 2 Participant topology for the visa application scenario

```
<topology name="visa" targetNamespace="urn:visa" xmlns:ms="urn:visa:ms">
  <participantTypes>
    <participantType name="MigrationService"
     participantBehaviorDescription="ms:migrationservice" />
    <participantType name="Employer" ... />
    <participantType name="ImmigrationOffice" ... />
  </participantTypes>
  <participants>
    <participant name="e" type="Employer" selects="ms" />
    <participant name="ms" type="MigrationService" />
    <participant name="io" type="ImmigrationOffice" />
  </participants>
  <messageLinks>
    <messageLink name="employeeDetailsLink" sender="e"
     sendActivity="SubmitEmployeeDetails" receiver="ms"
     receiveActivity="ReceiveEmployeeDetails"
     messageName="employeeDetails" />
    <messageLink name="visaApplicationLink" sender="ms"
     sendActivity="SubmitVisaApplication" receiver="io"
     receiveActivity="ReceiveVisaApplication"
     messageName="visaApplication" participantRefs="e" />
    <!-- ... -->
  </messageLinks>
</topology>
```

Although participant references are defined on a global level, not all participants necessarily know about all other participants involved. Through the receipt of messages or through explicit link passing the knowledge about participants is propagated. The immigration office knows which migration service is involved in the conversation as it receives a message from it. On the other hand, the office gets to know the employing company through the mechanism of link passing mobility. The migration service passes the reference to this company on to the immigration office as part of the visa application.

The notion of participant references cannot be directly found in π-calculus. On the other hand, send and receive activities are mapped to input and output actions on a π-channel, leading to the fact that message links from BPEL4Chor are represented by one or several π-channels. Several channels are needed in the case of several participants of the same type taking part in the conversation. We therefore introduce the term *message link instance* for corresponding to the actual connection between two participants in a conversation. The example from the previous section could be formalized as follows:

$$E \stackrel{def}{=} (\nu details, na)\ \overline{ed}\langle details, na\rangle.na(approval).\mathbf{0}$$

$$MS \stackrel{def}{=} (\nu c, application)\ ed(details, na).\overline{va}\langle application, c, na\rangle.c(conf).\mathbf{0}$$

$$IO \stackrel{def}{=} (\nu approval, conf)\ va(application, c, na).(\overline{na}\langle approval\rangle.\mathbf{0} \mid \overline{c}\langle conf\rangle.\mathbf{0})$$
$$SYS \stackrel{def}{=} (E \mid MS \mid IO)\ .$$

The message link `employeeDetailsLink` is represented by channel ed and `visaApplicationLink` by va. We see that ed and va are free names. This indicates that there is a static binding between the employer E and the migration service MS as well as between MS and the immigration office IO. In order to explicitly represent dynamic selection of the migration service by the employer (which is indicated by the `selects` attribute in the participant topology) a broker B could be introduced into the formalization:

$$SYS' \stackrel{def}{=} (E' \mid MS \mid IO \mid B) \text{ with } E' \stackrel{def}{=} lookup(ed).E \text{ and } B \stackrel{def}{=} \overline{lookup}\langle ed\rangle.B\ .$$

The propagation of knowledge about participants can be found in the formalization. We have mentioned that this propagation either takes place (i) through the receipt of a message or (ii) through passing on participant references. (i) can be found where MS sends c as attachment to the visa application. MS therefore passes a callback channel to IO for the confirmation. Hence, this is an example for indirectly representing participant references through message link instances. (ii) can be also found where MS sends the application to IO: na is the channel where the approval has to be sent to, again indirectly representing the participant reference for the employer. This formalizes the attribute `participantRefs` of the message link `visaApplicationLink` set to `e`. In both cases we see that the propagation of knowledge about participants corresponds to the notion of scope extrusion in π-calculus.

We can summarize that the information given in the PBDs is mainly encoded in control flow constructs in π-calculus, i.e. choice, parallelism and sequence. [13] shows how more complex control flow constructs are represented in π-calculus. The information given in the participant topology specifies the π-names used and defines which names have to be passed in interactions between the different π-processes. In addition to the attribute `participantRefs` indicating link passing mobility, message links can also have the attribute `copyParticipantRefsTo` set. As an effect the bound name in the receiving π-process is simply renamed.

All BPEL4Chor constructs can be translated to BPEL constructs. E.g. `participantRefs` indicates that a `copy from partnerLink` action takes place prior to a send activity and a `copy to partnerLink` after a receive activity. In the case of `copyParticipantRefsTo` set, the target partnerLink at the receiving side has a different name than the source partnerLink on the sending side.

5 Conclusion

This paper has investigated the link passing mobility capabilities of BPEL4Chor. For illustrating this, a sample choreography was partially given in BPEL4Chor and formally given in π-calculus. It was briefly discussed how constructs from BPEL4Chor map to those from π-calculus. As link passing mobility plays an

essential role in choreographies, any useful formalization of BPEL4Chor has to include this concept. Therefore, a complete mapping from BPEL4Chor to π-calculus is desirable. Since BPEL4Chor is heavily based on BPEL and since there is no complete π-formalization of BPEL so far, such a complete mapping goes beyond the scope of this paper and is therefore left to future work.

References

1. A. Barros, M. Dumas, and A. ter Hofstede. Service interaction patterns. In *Business Process Management, volume 3649 of LNCS*, Nancy, France, September 2005. Springer.
2. BPMI.org. *Business Process Modeling Language*, 2002.
3. A. Brogi and R. Popescu. From BPEL Processes to YAWL Workflows. In *Web Services and Formal Methods, volume 4184 of LNCS*, pages 107–122, Berlin, 2006. Springer Verlag.
4. F. Curbera, F. Leymann, T. Storey, D. Ferguson, and S. Weerawarana. *Web Services Platform Architecture: Soap, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging and More*. Prentice Hall PTR, 2005.
5. G. Decker, O. Kopp, F. Leymann, and M. Weske. BPEL4Chor: Extending BPEL for Modeling Choreographies. *ICWS 2007*.
6. G. Decker, H. Overdick, and J. M. Zaha. On the Suitability of WS-CDL for Choregraphy Modeling. In *EMISA 2006, volume P-95 of LNI*, pages 7–19, Bonn, 2006. Gesellschaft für Informatik.
7. G. Decker, F. Puhlmann, and M. Weske. Formalizing Service Interactions. In *Business Process Management, volume 4102 of LNCS*, pages 414–419, Berlin, 2006. Springer Verlag.
8. S. Hinz, K. Schmidt, and C. Stahl. Transforming BPEL to Petri nets. In *Business Process Management, volume 3649 of LNCS*, pages 220–235, Berlin, 2005. Springer Verlag.
9. R. Milner, J. Parrow, and D. Walker. A Calculus of Mobile Processes, Part I/II. *Information and Computation*, 100:1–77, September 1992.
10. Web Services Business Process Execution Language Version 2.0 – OASIS Standard. Technical report, Organization for the Advancement of Structured Information Standards (OASIS), Mar 2007.
11. H. Overdick, F. Puhlmann, and M. Weske. Towards a Formal Model for Agile Service Discovery and Integration. In *Proceedings of the International Workshop on Dynamic Web Processes (DWP 2005)*, IBM technical report RC23822, Amsterdam, December 2005.
12. F. Puhlmann. On the Suitability of the Pi-Calculus for Business Process Management. In *Technologies for Business Information Systems*, pages 51–62. Springer Verlag, 2007.
13. F. Puhlmann and M. Weske. Using the π-Calculus for Formalizing Workflow Patterns. In *Business Process Management, volume 3649 of LNCS*, Nancy, France, September 2005. Springer.
14. UN/CEFACT and OASIS. ebXML Business Process Specification Schema (Version 1.01). `http://www.ebxml.org/specs/ebBPSS.pdf`, 2001.

Modelling Compensation with Timed Process Algebra

Simon D. Foster ⟨S.Foster@dcs.shef.ac.uk⟩

Department of Computer Science, University of Sheffield

Abstract. Compensation is a feature of Business Process Modelling which allows error correction based on non-atomic rollback. Several variant strategies exist for compensation in a similar way that many patterns exist for workflow. Existing formalisations model compensation and workflow in an ad-hoc fashion, mainly due to their high-level nature. In this paper we propose a low-level approach using *Timed Process Algebra*, which can be extended to model several strategies within a component-oriented framework with compositional semantics.

1 Preliminaries

Composition of Web services in a component-wise fashion is an increasingly desirable method of building distributed applications. A necessary feature of service composition is the handling of runtime exceptions, specifically with the purpose of error correction. When something goes wrong it is essential that prior work of the same overall goal or transaction can be reverted in an attempt to return the world to its previous state. However, unlike in the database world where such transactions can be kept atomic via locking, in the world of Web services things are not so simple due to distribution of resources. As a result, instead of performing atomic rollback, orchestrations need to perform *compensation* [1] – undoing work where possible and making "amends" where not. An example is shown in Fig. 1 where the shipment of goods must be cancelled if billing fails. RegisterShipment, upon completion, installs a compensation CancelShipment which will execute if and only if part of the transaction fails.

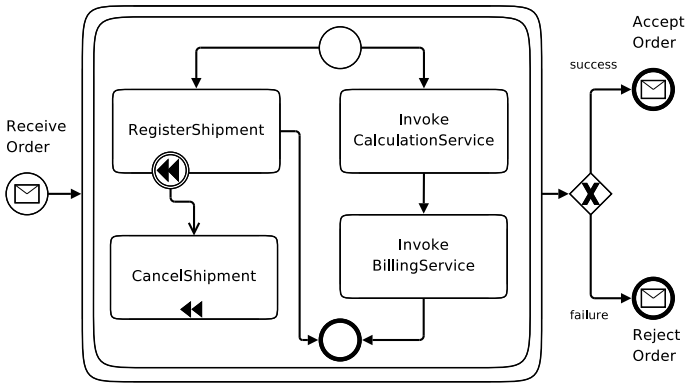


Fig. 1. Example compensable transaction

Existing formalisations of compensation, such as Sagas Calculi [2] and cCSP [3], extend process calculi with specialised compensation operators. Thus the expression $\mathsf{RegisterShipment} \div \mathsf{CancelShipment}$ represents the compensation pair in Fig. 1. This is united with a transaction block operator, which limits the scope of exceptions and the resulting compensations, and a specific strategy for compensation flow (e.g. distributed vs. centralised). Our intention is to improve on this approach by showing how compensation can be modelled using a set of more canonical operators. Thus, rather than hard-wiring a particular compensation strategy, we can give a semantics to variant strategies. We also take account of *dataflow* with its implications and take a more abstract, component oriented approach than the existing formalisations.

Timed Process Algebra is a field of Computer Science which looks at how temporal constraints can be placed on concurrent processes. As well as simply modelling the relative ordering of actions, as in CCS and CSP, time further constrains when an action can be performed based on its "speed" with respect to other events. In the context of CCS, time can be abstracted as a discrete multiparty event (σ, the "clock") limited by the presence of faster internal actions (τ) which are formed when something unobservable occurs (e.g. handshake synchronisation, hidden activity). This facilitates the *maximal progress* assumption – the maximum amount of internal activity must be performed within a clock's scope before it is allowed to tick, thus signalling that time has progressed. One of the most recent realisations of this concept is *CaSE* [4] (Calculus of Synchronisation and Encapsulation) – a timed extension of CCS with multiple clocks.

We have shown that this timed process algebra is particularly useful for modelling orchestration due to the elegant way in which block structured component systems may be represented [5]. Each block is associated with its own distinctly named clock which will only tick once the internal work is complete. Since these clocks may themselves be internalised and thus promoted (via clock hiding, which defines their scope), a hierarchical system results with lower blocks necessarily completing before higher blocks. This style of process algebra lends itself to compositional modelling of orchestration, where components are semantically substitutive, via the *bisimulation*-based equivalence theory. Bisimulation is a game-based equivalence which requires that two labelled transition systems be capable of matching each others moves in every possible evolution. CaSE adopts a weak form which abstracts from internal activity and thus identifies components which provide the same visible communications.

In the remainder of this paper we demonstrate how discrete time with maximal progress also allows the modelling of compensation without the need for specialised operators, with application to a suitable orchestration language.

2 A language for compensable orchestrations

Cashew-S is a language which represents orchestrations using block-type control flow constructs based around *workflow patterns* [6]. Workflow patterns attempt to draw out generic and flexible workflow constructs which have applicability

in many different problem domains, with examples including Sequence, Parallel Split and Exclusive Choice. Dataflow is driven by *isochronic broadcast* [7] which uses abstract time with maximal progress to determine when all consumers have been satisfied.

Initially we designed the language to act as a superset of the OWL-S service model [8], but we have since enhanced it with further features. In particular we have added a form of interruption and compensation based on cCSP, though simplified for the purposes of this paper (e.g. compensations cannot fail to complete[1]).

$$
\begin{aligned}
\mathit{Transaction} &::= \textbf{Perform } p \textbf{ Transaction } \mathit{CWorkflow} \\
\mathit{TransList} &::= \mathit{Transaction} \mid \mathit{TransList}; \mathit{Transaction} \\
\mathit{CWorkflow} &::= \textbf{Workflow } w \ (\mathit{Acceptors}) \ (\mathit{Offerors}) \ \mathit{CPattern} \\
\mathit{CPattern} &::= \textbf{Seq } (\mathit{CPerfList}) \mid \textbf{Par } (\mathit{CPerfList}) \\
&\mid \textbf{Inter } (\mathit{CPerfList}) \mid \textbf{Conc } z \ z \ z \ (\mathit{TransList}) \\
&\mid \textbf{Choice } (\mathit{CPerfList}) \mid \textbf{Skip} \mid \textbf{Throw} \mid \textbf{Yield} \\
\mathit{CPerformance} &::= \mathit{AtomicPerformance} \mid \mathit{Compensation} \\
&\mid \mathit{CWfPerformance} \mid \mathit{Transaction} \\
\mathit{CWfPerformance} &::= \textbf{Perform } p \ \mathit{CWorkflow} \\
\mathit{Compensation} &::= \mathit{CPerformance} \div \mathit{Performance} \\
\mathit{CPerfList} &::= \mathit{CPerformance} \\
&\mid \mathit{CPerfList}; \mathit{Connection} \\
&\mid \mathit{CPerfList}; \mathit{CPerformance} \\
z &::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid \cdots
\end{aligned}
$$

Fig. 2. Compensable fragment of Cashew-S language

We reproduce the compensable fragment of this language in Fig. 2 without dataflow (due to space constraints). *Performances* are named components: they are either primitives for calling Web services, communicating with the client and evaluating expressions (i.e. an *AtomicPerformance*) or encapsulated workflows. Performances within a transaction may be associated with another (non-compensable) performance to act as a compensation. Each workflow has associated with it a set of input and output handlers (resp. *Acceptors* and *Offerors*) and a pattern of composition. In this language and for the remainder of this paper w represents the name of a workflow and p the name of a performance.

Cashew-S contains five compensable workflow constructs for composing a list of performances. The **Seq** pattern models sequential composition. **Par** and **Conc** model two forms of concurrency, the former simply executing all performances with synchronisation at the beginning and end, and the latter allowing customisation of exactly how many performances need to be executed and resynchronised. **Conc** can thus model complex control flow patterns like the *discrim-*

[1] Or rather they can, but the failure must be in a separate transaction block so that there is no upwards propagation of errors into the containing block.

inator [6], which allows only a subset of concurrent flows to resynchronise after execution. **Inter** executes the performances in the order in which they become ready to execute (e.g. by satisfaction of dataflow). **Choice** picks the performance which becomes ready first, cancelling the others.

Compensation is triggered by a **Throw** pattern, which raises an exception in the block. The block then has to wait for the internals to halt (i.e. centralised compensation with interruption), which can be achieved using the **Yield** pattern or simply waiting for completion. Once all sub-threads have yielded, compensation proceeds in the reverse order of the pattern. A compensation for our example would be rendered as follows (minus internal details of the Web services, referred to pragmatically as a **Goal**):

$$\textbf{Perform}\ \mathsf{perfRegisterShipment}\ \textbf{Goal}\ \mathsf{registerShipment}\ (\cdots) \div \\ \textbf{Perform}\ \mathsf{perfCancelShipment}\ \textbf{Goal}\ \mathsf{cancelShipment}\ (\cdots)$$

The former performance (with goal registerShipment) is called the *forward* performance, while the latter is called the *compensation* performance.

3 Behavioural Semantics

The behavioural semantics is given using a conservative extension of CaSE [4] called *CaSiE* [9]. It allows both input-output synchronisation and deterministic multi-party synchronisation (restricted by maximal progress) over multiple distinct clocks. Processes are by default *patient* – permitting any clock transition, even if not directly causing them to mutate. Specific clocks may, nevertheless, be blocked if required (e.g. to prioritise a non-internal action) and we make extensive use of this in our agents below to inhibit clocks without an explicit outgoing transition. CaSiE also adds *interruption* actions which are of higher priority than both silent actions and clock ticks. These allow, for example, exceptions raised in a workflow to pre-empt remaining internal activity and perform compensation.

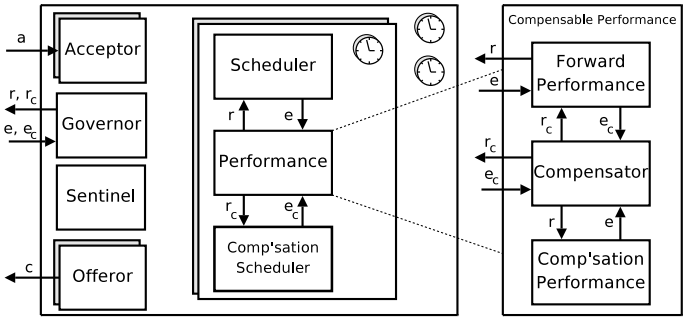


Fig. 3. Overview of compensation scheduling

The orchestration semantics architecture, illustrated in Fig. 3, is as follows. Each **Performance** in a workflow is composed with a Scheduler agent which defines the preconditions of execution particular to the pattern. Each workflow in turn has a Governor – a scheduler of schedulers, which communicates with the

environment and decides when (and if) the workflow as a whole can execute. Acceptor and Offeror agents manage workflow inputs and outputs, respectively.

A workflow's behaviour is divided into a finite number of distinct *phases*, each of which defines the current "macro-step" being performed (e.g. fulfilling preconditions, executing). Two main clock classes (indicated in the outer frame of Fig. 3) are used to distinguish the workflow phases, σ_n^w for normal activity and ρ_n^w for exception handling (where $n \in \mathbb{N}^*$ is the phase number). The Governor and Schedulers share these clocks to observe the current phase of the workflow, and normally do not otherwise communicate. Each performance also has a collection of clocks associated with it which the scheduler and compensation scheduler share to detect its current phase. Each workflow pattern uses schedulers supporting different execution strategies for their respective performances. For example, **Seq** schedulers engage in a token passing game – only the scheduler holding the token can execute its performance and once complete the token is passed onto the next scheduler (if one exists).

The workflow in Fig. 3 also has a Sentinel agent. This is only present if the workflow is directly within a transaction and manages compensation within the block. If a performance raises an exception, the Sentinel first waits for all sub-threads to cease. This process is mostly handled by maximal progress, but the Sentinel will also broadcast yield commands to any threads willing to receive them, causing them to halt. Once all activity has stopped the Sentinel will invoke the topmost compensation schedulers. Naturally the compensation schedulers follow the *reverse* strategy to regular schedulers where appropriate, for example in **Seq** the last performance to execute is the first to compensate and the token is passed back in the opposite direction. The actual compensations are managed by Compensator agents which are composed with compensable performances and install their respective compensations upon the forward performance's completion.

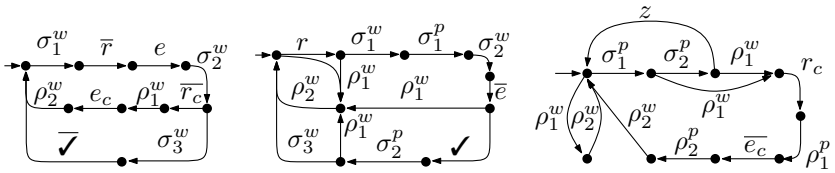


Fig. 4. Governor and Schedulers for **Par** pattern
(a is input, $\overline{a}$ is output, σ and ρ are clocks. Clocks mentioned in the graphs are blocked in all states unless they explicitly have outgoing transitions)

Fig. 4 shows the governor, scheduler and compensation scheduler, respectively, for **Par**. They use the following channels:

- r signals a component has fulfilled its preconditions and is **r**eady to execute;
- e is used to order a component to **e**1xecute;
- ✓ is used to signal that a component completed successfully;
- z is used to signal completion of the enclosing transaction.

Initially all schedulers wait for their associated performances to become ready by waiting for a signal on channel r (this is the condition of readiness for the

workflow). Once all schedulers have received this signal the workflow can enter the first phase via the ticking of σ_1^w. The governor then asks the environment for permission to execute via r, and receives it back via e. The execution phase then begins (σ_2^w) where the actual performances move into their execution phase via the first clock (σ_1^p) and permission to execute is likewise signalled via e.

Each performance can then signal success by outputting ✓, or failure by simply halting. Compensation is governed by the two ρ^w which drive the compensations schedulers. Alternatively, if execution was successful, the performance phase is advanced (σ_2^p) and then either the workflow as a whole can complete (indicated by σ_3^w) or it can fail indicated by ρ_1^w (if a sibling performance failed). Error handling completion is indicated by ρ_2^w. It should be noted that, since even after successful completion a workflow may need to compensate, the scheduler's initial state permits a ρ_1^w tick (otherwise compensation would be blocked).

The governor recognises the need to initiate compensation in the workflow by being able to send a request to compensate before the workflow signals success via σ_3^w (the environment will not allow this unless the Sentinel has initiated compensation from above). Thereafter it allows a ρ_1^w tick to initiate compensation. For a compensation scheduler to invoke compensation it must first detect completion of its performance via the performance clocks ($\sigma_{1,2}^p$) during workflow execution, otherwise it will just idle over the error handling phase (since no work has been done). If it does detect that the performance has run and thus needs compensation, it will follow basically the same pattern as the regular scheduler, using analogous r_c and e_c channels to invoke compensation in its performance.

With the above framework in place, implementing the actual compensations handled by Compensator agents is straightforward. Each Compensator (not shown) is composed with both a forward and compensation performance. Upon detecting successful completion of the associated forward performance via ✓, the compensation performance is "installed", meaning subsequent compensation commands on r_c and e_c will be forwarded to it. Prior to this such commands are simply forwarded on to the compensation schedulers in the forward performance (with the assumption that compensation is handled elsewhere).

If a transaction block successfully completes (i.e. the top-level workflow completes) the Sentinel broadcasts a signal on z to all compensation schedulers and Compensator agents, ordering them to reset or uninstall their compensations respectively.

4 Conclusion

We have outlined a method for modelling compensation using timed process algebra. Using general process algebraic operators we can model many variations of compensation and workflow, in a similar way to how Petri Nets have been applied to modelling different variations of the workflow patterns [6], though with the benefit of having compositionality as a core concept.

Our aim for the future is the expansion of the existing equivalence theory based on temporal weak bisimulation to the new calculus. Due to lack of space

many of technical details have been omitted, for example how interruption is handled, which requires an extension to the preemption scheme of CaSE [9]. A full operational semantics for Cashew-S will also be derived, with associated theory derived from the process algebra. Cashew-S should ultimately be sufficiently expressive to represent a useful class of WS-BPEL processes, specifically those with compensation and the single instance fragment of the workflow patterns. However questions remain, such as how should looping be modelled in the presence of compensation and what theory can be derived from the operational semantics. We also anticipate an elegant implementation of the semantics in the functional programming language *Haskell*, in which we already have a concise implementation of CaSE.

Acknowledgements

Many thanks to Mike Stannett, Barry Norton, Andrew Hughes and Ramsay Taylor for their valuable feedback on this paper. We would also like to thanks the anonymous reviewers for their feedback on this paper. Due to space constraints further expansion has been limited, though remaining suggestions will be dealt with in the forthcoming longer submissions.

This research is supported by EPSRC DTA EP/P501717/1.

References

1. Garcia-Molina, H., Salem, K.: Sagas. In: Proc. of the 1987 ACM SIGMOD Intl. Conference on Management of Data (SIGMOD '87), ACM Press (1987) 249–259
2. Bruni, R., Melgratti, H., Montanari, U.: Theoretical foundations for compensations in flow composition languages. In: Proc. 32nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'05), ACM Press (2005) 209–220
3. Butler, M., Ripon, S.: Executable semantics for compensating CSP. In: Proc. 2nd Intl. Workshop on Web Services and Formal Methods (WS-FM 2005). Volume 3670 of LNCS., Springer (September 2005) 243–256
4. Norton, B., Lüttgen, G., Mendler, M.: A compositional semantic theory for synchronous component-based design. In: 14th Intl. Conference on Concurrency Theory (CONCUR '03). Volume 2761 of LNCS., Springer (2003)
5. Norton, B., Foster, S., Hughes, A.: A compositional operational semantics for OWL-S. In: Proc. 2nd Intl. Workshop on Web Services and Formal Methods (WS-FM 2005). Volume 3670 of LNCS., Springer (September 2005) 303–317
6. van der Aalst, W., ter Hofstede, A., Kiepuszewski, B., Barros, A.: Workflow Patterns. Distributed and Parallel Databases **14**(1) (2003) 5–51
7. Norton, B., Fairtlough, M.: Reactive types for dataflow-oriented software architectures. In: Proc. 4th Working IEEE/IFIP Conference on Software Architecture (WICSA2004). Volume P2172., IEEE Computer Society Press (2004) 211–220
8. Martin, D., ed.: OWL-S : Semantic Markup for Web Services. OWL-S Coalition, `http://www.daml.org/services/owl-s/1.1/overview/` (2004)
9. Foster, S.: A timed process calculus with localised interruption. Technical Report CS-07-07, University of Sheffield (2007)

Service and Protection Level Agreements for Business Processes *

Ganna Frankova and Artsiom Yautsiukhin

Dept. of Information and Communication Technology - University of Trento
email: {frankova,evtiukhi}@dit.unitn.it

Abstract. The issue of business process design for a complex web service provision is gaining attention and has been addressed in a number of recent works. We argue that calculation of global quality of service and protection, which then is negotiated with a client as service and protection level agreements, for complex web services must be based on business process.
The quality of service in web service compositions plays a vital role and has opened a wide spectrum of challenges. Therefore, the orchestrator should design its business process aggregating web services in such a way as to make it more efficient from the quality of service and protection point of view. In this work we propose a methodology that identifies the concrete business process providing the highest quality of service and protection among all possible design alternatives.

1 Introduction

One of the most thought provoking issues in web services is that of building a business process to provide a complex added-value service. A Business Process (BP) in a web services context is a set of interrelated services and the flow of data between them that leads to the outcomes associated with a business activity. The research on business processes is well under way. There are many works focusing both on functional properties, primary goals, e.g., [7], and non-functional properties, quality of service, of web services and web services composition, e.g., [10]. However, the works do not take into account the concept of business process for web service composition. The works on security for web services are mainly restricted to protection of communication links [8] and access control [2]. There are only a few approaches covering security requirements for web services as a whole [3, 6].

The Quality of Service (QoS), e.g., execution time and availability, and Quality of Protection (QoP), e.g., time to recover after an attack and the percentage of successful virus attacks are of paramount importance for the success of web services. Web services should satisfy not only functional requirements but also be QoS/QoP driven. The problem becomes even more complex if we deal with composite web services.

To design a BP first we specify an abstract BP which fulfils the desired functional requirements at the high level. Various design alternatives that do

* This work has been partly supported by the IST-FP6-IP-SERENITY and IST-FP6-IP-SENSORIA projects.

not affect the functionality of abstract business process but allow configuring the workflow in different ways exist (e.g., one alternative provides faster processing, another one is more reliable). These design alternatives lead to several concrete BPs (refinements of the abstract BP) that precisely define all its steps at the low level and provide all the functionalities. Then, the most efficient concrete BP from the QoS and QoP point of view is selected. Furthermore, an orchestrator may trust one service provider more than another one on provision of a service that is a part of a concrete BP.

In this work, we propose a methodology that identifies the concrete business process providing the most fruitful qualities among all possible design alternatives for the given abstract business process. Moreover, the methodology takes into account the level of trust of service providers and adjusts the expected quality value correspondingly. The methodology is supported by a reasoning algorithm that allows easy recalculation of computed metrics if some changes in BP design occur, that is typical for highly dynamic environment such as web services.

2 Service and Protection Level Agreements: Background

The stake-holders involved in a BP are:

Definition 1 *Service Customer* (customer) is an entity that interacts with a complete, self-contained BP. *Service Orchestrator* (orchestrator) is an entity which manages a BP and agrees to satisfy the customer's requirements for the BP. *Service Provider* (provider) is an entity that has a task assignment, i.e., web service, that is a part of a higher-level BP, received from an orchestrator.

The involved partners should come to a formal agreement before the usage of web service. The agreement is defined as a contract between the provider and the orchestrator specifying the functionality of the outsourced service, quality and protection requirements. The requirements for complex web service the orchestrator manages are specified in a contract with a customer.

Here we assume that services provide desired functionality and we focus on non-functional requirements. We found it useful to divide the agreement into the following parts:

Definition 2 *Service Level Agreement* (SLA) is a contractual version of the Quality of Service (QoS) which specifies the performance criteria a provider promises to meet while delivering a service. *Protection Level Agreement* (PLA) is a contractual version of the Quality of Protection (QoP) which specifies security criteria, a provider promises to meet while delivering a service.

We argue that security requirements should be considered since client's data may be corrupted while under the control of the provider. On the one hand, there are plenty of works on QoS [9, 5], on the other hand, very little attention is devoted to QoP while identification and aggregation of security metrics useful for QoP is not a trivial task [6].

3 Business process hypergraph

The cornerstone of web services success lies in the ability to compose web services in order to build complex added-value services. Dealing with quality aware web service composition requires studying and finding the global SLA/PLA of complex web services according to the BP. In the proposed methodology, we use hypergraphs [1] to capture the structure of BP. We introduce the notion of business process hypergraphs as follows.

Definition 3 A **business process hypergraph** (BPH) $\mathcal{B}$ is a pair $\langle S, D\rangle$ where S is a set of service requirements and D is a set of hyperarcs. A *hyperarc* is an ordered pair $\langle N, t\rangle$ from an arbitrary nonempty set $N \subseteq S$ (source set) to a single node $t \in N$ (target node). Each hyperarc is associated with a weight $\omega_{\langle N,t\rangle}$ and a function $\varphi_{\langle N,t\rangle}$ which calculates value of a target node taking as arguments source nodes and the weight $\omega_{\langle N,t\rangle}$.

Since in our case each source node contributes differently to a target one, we use dummy nodes between source and target nodes, one for a hyperarc. The weights are assigned to the hyperarcs that connect source and dummy nodes and the weights for the hyperarcs between the dummy and target nodes are always 1. We do not depict the nodes to avoid unnecessary complexity.

We assume that there are more then one ways to implement an abstract BP. Each solution is given by the hyperpath defined as follows.

Definition 4 Let $\mathcal{B} = \langle S, D\rangle$ be a BPH, $X \subseteq S$ be a non-empty subset of services, and y be a service in S. A **hyperpath** $\mathcal{D}_{X,y}$ from X to y in $\mathcal{B}$ is a set of hyperarcs such that either $y \subseteq X$ or there exists a hyperarc $\langle Z, y\rangle \in D$ and there are hyperpaths from X to each service $z_i \in Z$.

As there are several ways to refine an abstract BP into the concrete BP, several hyperpaths exist. The global SLA/PLA of each hyperpath is different. Hence, the key issue is to determine the "minimal" hyperpath through a quantitative evaluation of BPH.

4 SLA and PLA for Business Processes: Methodology

The proposed methodology helps a service orchestrator to select the optimal concrete BP with the preferred QoS/QoP from several alternatives based on the abstract BP [1].

We made the following assumptions about a BP our methodology deals with: (i) BP is defined in a hierarchical way. A top level BP (BP_t) is based on services aggregated by one of structural activities such as a sequence, a loop, a choice or a parallel execution. Then, for each non-atomic service S_i a BP (BP_{S_i}) is determined and the decomposition continues until atomic, i.e, non-decomposing, services are reached. (ii) An orchestrator which does not trust a provider on

[1] For the notions of abstract business processes we refer to Web Services Business Process Execution Language Version 2.0, August 2006.

achievement of some requirements may use the level of trust to adjust corresponding metrics in such a way that after the modification the trust relation is established.

The methodology includes three phases, namely, (1) business process hypergraph construction, (2) aggregation functions design and (3) reasoning in business process hypergraph.

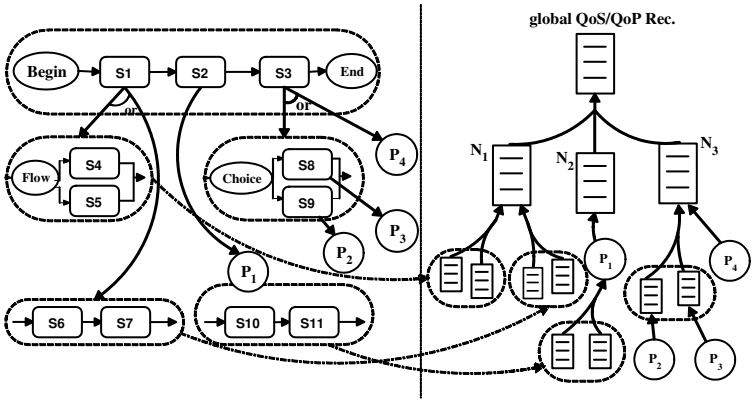


Fig. 1. Business process hypergraph construction.

Phase 1. Business Process Hypergraph Construction. The first phase of the methodology is devoted to BPH construction based on the various implementation of abstract BP designing by an orchestrator. The BPH construction process is presented as an algorithm in Figure 2.

Figure 1 shows the top level BP_t based on three services S_1, S_2, and S_3. Each service of the top level BP_t is associated with the node N_1, N_2, and N_3. The nodes are connected by one hyperarc with the top node, that means that the services all together contribute to satisfaction of the global requirements. Then, the services are decomposed. Since two alternative BPs exist for service S_1, two distinct hyperarcs and corresponding source nodes are added in the hypergraph. This means that each alternative business process contributes to the satisfaction of the target requirements separately. The services are executed by the same partner that runs BP_t. Service S_2 is delegated to provider P_1 (shown as a node in the BPH) and then decomposed. Service S_3 may be fulfilled by provider P_4 or may be decomposed into a business process whose activities are outsourced to P_2 and P_3.

Phase 2. Aggregation Functions Design. The second phase of the methodology is devoted to the aggregation functions. Each hyperarc is assigned with a set of weights that shows contribution of a source node to the target one. Weights of hyperarcs connecting providers with a target node denote the level of trust between the delegator and the delegatee. Each leaf node is assigned with a QoS/QoP value that *can* be achieved. The value corresponds to the QoS/QoP of atomic service.

Each hyperarc is assigned with an aggregation function $\varphi_{\langle N,t\rangle}$ (one for each structural activity) which calculates the value of a target node taking as argu-

ments the source nodes and the set of weights $\omega_{\langle N,t\rangle}$. Examples of the aggregation functions for QoS could be found in [5]. The authors provide aggregation functions for such numerical QoS metrics as cost, execution time, etc. In our work we take into account the level of trust of providers and propose an additional function to change the expected metrics according to the level of trust.

```
Algorithm Business Process Hypergraph Construction
begin
   specify the global QoS/QoP requirements for BP
   associate the global QoS/QoP requirements for BP with the top node of BPH
   while there are services with the corresponding decomposing BPs
      if service is outsourced then
         for each provider that has a task assignment
            add a node for provider in the BPH
            add a hyperarc from the added node to the target node
      else
         for each design alternative
            for each service of the alternative
               add a node in the BPH
            add a hyperarc from the added nodes to the target node
   end-while
end
```

Fig. 2. Business Process Hypergraph Construction

To the best of our knowledge there is no similar approach for security metrics and we propose the preliminary aggregation functions for number of attacks per month in the following table.

Structural activities	**Weight**	**Function**
parallel	probability of service execution[2]	$\varphi = \sum_{x_i} \omega_i * x_i$
sequence	probability of service execution[2]	$\varphi = \sum_{x_i} \omega_i * x_i$
choice	probability of service execution[2]	$\varphi = \sum_{x_i} \omega_i * x_i$
loop	$\omega = 1$	$\varphi = \varphi_1$
trust	level of trust	$\varphi = \omega_1 * \varphi_1$

Phase 3. Reasoning in Business Process Hypergraph. The third phase of the methodology is devoted to a reasoning algorithm that proceeds several concrete BPs calculating their QoS/QoP. We evaluate each requirement separately finding the "best" BP for each requirement rather than for all of them together.

The aggregation functions proposed by Jaeger [5] are superior. This allows us to apply one of already proposed algorithms for effective calculation of hyperpath in BPH [1, 4]. Quantitative evaluation of hyperpaths in BPH from leaf nodes to the top one determines the optimal concrete BP.

We note that QoP metrics are more complex than QoS ones and aggregation functions for QoP metrics design is not a trivial task. We developed and proved

[2] probability to find a source service to be executed if the target service is proceeding.

an algorithm for calculation of an optimal hyperpath that works with monotone aggregation functions. Due to lack of space we do not present it in this work.

5 Concluding Remarks

We have proposed a methodology that during design time helps a service orchestrator to determine the optimal concrete BP from several alternatives according to the preferred QoS/QoP. The methodology also allows the orchestrator to easily recalculate computed metrics if some changes in the BP occur (i.e., BP re-design). Furthermore, we are planning to adopt the approach for automatic composition of BP (i.e. automatic BP design) to support a business process planning tool in choosing the best concrete business process on the fly.

In the future, we will investigate the multi-requirement analysis, which requires the identification of a decision-making function that chooses the more preferable *set* of attributes (e.g. a weighted function). We also will define and validate the aggregation functions for more QoS/QoP requirements. Furthermore, to make the discussion more concrete, we will justify that the aggregation functions are appropriate using the e-business banking case study, a working scenario of the IST-FP6-IP-SERENITY project. In addition, we plan to elaborate the issue of trust to determine how it affects the expected qualities.

6 Acknowledgments

We thank Fabio Massacci for support in conducting this research. We acknowledge Marco Aiello for his constructive suggestions for improving the quality of the paper.

References

1. G. Ausiello, G. F. Italiano, and U. Nanni. Optimal Traversal of Directed Hypergraphs. Technical Report TR-92-073, International Computer Science Institute, Berkeley, CA, 1992.
2. E. Bertino, J. Crampton, and F. Paci. Access Control and Authorization Constraints for WS-BPEL. In *Proceedings of IEEE ICWS.*, 2006.
3. V. Casola, A. Mazzeo, N. Mazzocca, and M. Rak. A SLA Evaluation Methodology in Service Oriented Architectures. In *Proceedings of QoP Workshop.*, 2005.
4. G. Gallo, G. Longo, S. Pallottino, and S. Nguyen. Directed Hypergraphs and Applications. *Discrete Applied Mathematics*, 42(2-3):177–201, 1993.
5. M.C. Jaeger, G. Rojec-Goldmann and G. Mühl. QoS Aggregation in Web Service Compositions. In *Proceedings of IEEE EEE.*, 2005.
6. Y. Karabulut, F. Kerschbaum, P. Robinson, F. Massacci, and A. Yautsiukhin. Security and Trust in IT Business Outsourcing: a Manifesto. In *Proceedings of STM Workshop*, 2006.
7. N. Milanovic and M. Malek Current Solutions for Web Service Composition. *IEEE Internet Computing*, 8(6):51–59, November/December 2004.
8. OASIS Web Services Security: SOAP Message Security 1.1, February 2006.
9. T. Yu and K.-J. Lin. A Broker-Based Framework for QoS-Aware Web Service Composition. In *Proceedings of the IEEE EEE*, 2005.
10. L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and Q.Z. Sheng. Quality Driven Web Services Composition. In *Proceedings of WWW*, 2003.

A Model for exploring the Service-oriented Software Engineering (SOSE) challenges

Qing Gu and Patricia Lago

Department of Computer Science
VU University Amsterdam, The Netherlands

Abstract. Service-oriented software engineering (SOSE) aims to design, develop and maintain service oriented applications in a systematic approach. Due to the complexity that is introduced when software programs as services can be discovered, composed, instantiated, executed at runtime, SOSE poses new challenges in the way software is engineered as opposed to traditional software systems. In order for service-oriented systems to succeed, a well focused SOSE research agenda is needed. This paper proposes a model that can help researchers to identify the novelty of SOSE challenges and their research scope, and thus further helps to build the SOSE research agenda. To give an example of how to apply the proposed model, this paper explores the SOSE challenges that relate to business benefits by presenting a worked out instance of the model.

1 Introduction

Service-oriented architecture (SOA [1]) is a software architecture style that supports the communication of a collection of services to perform tasks. Recently, SOA draws enormous attention in the industry because systems that are built based on SOA promise to bring business benefits to the customers and technological benefits to the developers. In order for service-oriented systems to succeed, principles known from traditional software engineering (SE) need to be tailored to service-oriented development. Systematic approaches for designing, developing and maintaining service-oriented systems are needed. We generalize these approaches as Service-oriented Software Engineering (SOSE).

In a service-oriented system, a service is an independent unit of functionality delivered through a distributed network [2]. Additional complexity is introduced when SOSE has to ensure that services can be discovered, composed, instantiated, executed at runtime while complying with the requirements from all the stakeholders of the system. Engineering services requires different approaches as compared to traditional SE. Hence, SOSE poses more demanding and stimulating situations, which are called *challenges* in this paper, in the way services are engineered as opposed to traditional software systems.

Currently, the SOSE research challenges presented in the literature are very broad and cover a wide range of topics, from design principles to business models. We believe that a well focused agenda would better guide researches to define research topics and their scope. By understanding and prioritizing the

challenges from the current literature, researchers could come to a more focused SOSE research agenda. Furthermore, during our work, we observed that not all the challenges that have been proposed in the literature are really innovative. Since challenges with different novelty need different attentions, we suggest to categorize these challenges before creating the research agenda.

Due to the fact that the current proposed challenges cover a wide range of topics and the connections between the challenges are often ignored, a model that can well classify the challenges and can well indicate the inter-dependencies between the challenges seems useful. Based on this inspiration, we propose a model to assist researchers to explore the SOSE challenges because so far there is no commonly agreed model that formalizes these issues. This model guides researchers to focus on a subset of the challenges and helps to categorize these challenges based on their novelty by comparing with the ones in another SE discipline. In our experience, this categorization can further help researchers to determine the research priorities.

An instance of the model is presented to demonstrate useful applications of the model. This instance regards a subset of the challenges that are related to business benefits and chooses component based software engineering (CBSE), the ancestor of SOSE, as the benchmark. Through comparison, we categorize the challenges to a) false challenge, b) challenges that require tuning, and c) innovative challenges.

The paper is structured as follows. Section 2 presents the proposed model and an instance of the model is demonstrated in Section 3. Section 4 summarizes our contribution, conclusions and some directions for future work.

2 The proposed model for exploring SOSE challenges

A lot of work, such as research roadmaps [3], methodologies [4], and blueprints of service engineering [5], have been proposed by researchers to investigate SOSE challenges. In particular, [6] points out collaborative issues and problems caused by the dynamic nature of service-oriented system. An overview of the difference between traditional SE and SOSE is given in [7]. The research agenda proposed by NESSI [8] addresses eight high level research areas. In isolation, these works address part of the SOSE challenges. All together, these works propose a broad range of challenges. The lack of investigation on the relationship between the challenges hinders the researchers to decompose the research areas into specific research topics and impedes the cooperation between the researchers.

Moreover, we observed that not all the challenges that have been proposed in the literature are really innovative. Some of the proposed challenges have already been extensively investigated and validated in other SE disciplines (for instance, information hiding coming from object-oriented development). Since these challenges have been studied thoroughly, researchers need to understand better on how to apply the existing knowledge to SOSE. On the other hand, some challenges, such as dynamic composition, introduce different levels of innovation. These innovative challenges require more attention because they open

new research areas. These observations inspired us not to look at the challenges in isolation, but to compare them with other SE disciplines. In this way, by looking at the similarities and differences, we are able to determine whether one is intrinsically an innovative challenge.

In order to set research priorities, a model to categorize SOSE challenges would prove to be useful. This paper proposes such a model to explore SOSE challenges. It gives researcher a structured way of analyzing these challenges. By studying the inter-dependencies of the challenges and by comparing the challenges with the ones in other SE disciplines, researchers can use this model to conclude the novelty of the studied challenges.

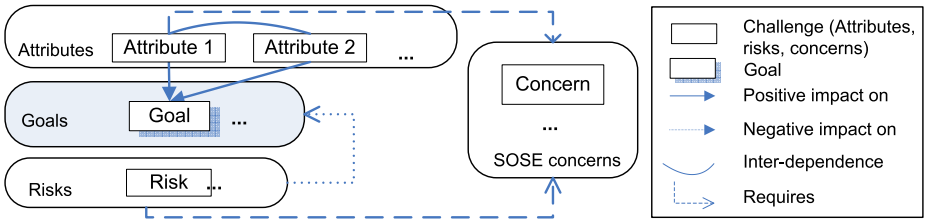


Fig. 1. A model for exploring SOSE challenges

The model presented in Figure 1 consists *Goals*, *Attributes*, *Risks* and *SOSE concerns*. Each of the tiers contains one or more elements. The tier *Goals* refers to one or more purposes that all the selected challenge should share. Examples of the goals can be increasing flexibility, business benefits, technological benefits, etc. The tier *Attributes* contains the challenges that have an impact on the attainment of the goals. The tier *Risks* refers to the challenges that do not have mature technical solutions in order to be directly applicable. This lack of maturity hinders the achievement of the goals, and even obstructs service-oriented systems to be fully industrially exploited. The tier *SOSE concerns* indicates some SOSE activities (such as requirements engineering, testing, etc) that might be influenced by the attributes or the risks or both.

We found it is neither effective to analyze a challenge individually nor possible to analyze all the challenges at the same time. By selecting the challenges that share the same common purposes, we can focus on a subset of the challenges. That is the reason why we first need to decide on the elements in the tier *Goals*. Intuitively, after the goals are determined, we are seeking the challenges that have positive impact or negative impact on the selected purposes. The challenges, such as design principles, engineering techniques, characteristics, that encourage the attainment of the purposes can be put in as the elements in the tier *Attributes*. At the same time, we might notice that some state-of-the-art challenges still do not have solutions. These challenges might cause problems when service-oriented systems will be widely applied. We can put them in the tier *Risks*. By looking at

all the selected challenges in the model and relating them to the SE activities, we might notice that some of the SE activities have to be adapted and some additional activities have to be added due to the changes that the challenges introduce in SOSE. We include these activities in the tier *SOSE concerns* and they are also regarded as SOSE challenges in the model.

After all the elements have been filled in the model, we can further study the inter-dependencies between the challenges. Our motivation to do so is, firstly, because we need these inter-dependencies to define the research scope. Secondly, it is because these inter-dependencies are one of the reasoning evidences for determining the novelty of the challenges. In the end, we can choose another SE discipline as the benchmark. By comparing the challenges and the inter-dependencies in the model with the ones in the other SE discipline, we can categorize these challenges based on their novelty.

3 An instance of the model

The conceptual model presented in Section 2 might at this point look very abstract. For demonstration purposes, we present an instance of the model by choosing 'business benefits' as the goals and CBSE as the benchmark.

The motivation for targeting on the 'business benefits' is that we believe service orientation will reach success only if a) companies adopt SOA as (part of) their IT portfolio and b) companies developing and maintaining services can gain the appropriate returns on their investments. Therefore, how much the customer and the vendor companies can benefit from SOSE is critical to whether it can be widely accepted in practice. Since business benefits comprise a large variety of sub-benefits, we need to further identify several aspects of the business benefits that we would like to focus on. In this example, we specifically analyze four, namely flexibility, quick time to market, scalability and low cost. These four are also the key business benefits that major vendors advocate [9]. They are represented by four elements in the central tier *Business benefits* of Figure 2.

In the next step, we can decide on elements in the tier *Attributes*, *Risks* and *SOSE concerns* based on the business benefits. These elements are the challenges that we are going to explore. In order to categorize them by their novelty, we choose CBSE as the benchmark being this often regarded as the ancestor of SOSE. Both paradigms share a number of common properties. By comparing the SOSE challenges and their inter-dependencies with the ones in CBSE, we can discriminate the novelty of these challenges. We define the challenges that do not exist in CBSE or challenges that are applied differently in CBSE and SOSE as *innovative* challenges, the challenges that are applied in the same way in CBSE and SOSE as *false* challenges, and the challenges that are applied in the similar way but with different inter-dependencies as challenges needing *tuning*.

In order to give a concrete example of how this instance works for categorization, we use the challenge dynamically discoverable & composable for explanation. The capability of discovery & composition is a challenge that requires turning due to the different focuses CBSE and SOSE addressed. For instance, services can

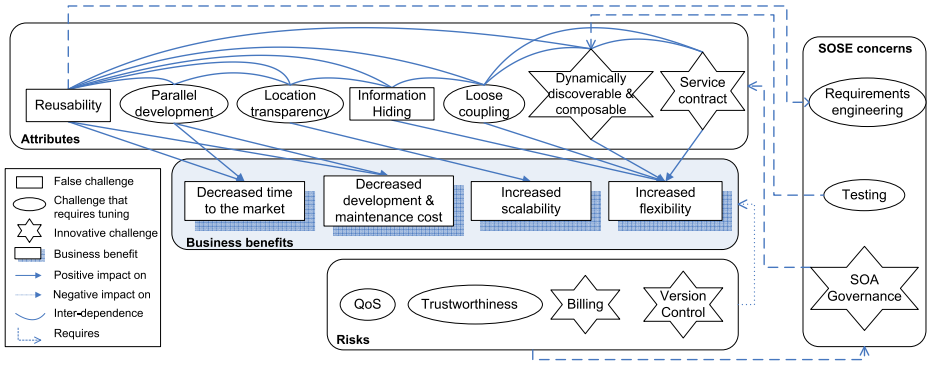


Fig. 2. Instance for Business benefits, with challenges wrt. CBSE

be selected at execution time while components are building blocks assembled during design time. However, adding dynamism to the capability of discovery & composition brings new challenges requiring additional research investigation, such as how to increase loose coupling, how to enrich the service contract and how to optimize testing procedures, etc. Therefore, we suggest regard Dynamically discoverable & composable as an innovative challenge.

Through this classification, we can further determine which research challenges require the highest priority. One could argue that the challenges that require tuning should require the highest priority because 1) mature research results could be achieved effectively thanks to existing experience in CBSE; 2) mature research results from SOSE can also spark new developments in CBSE. One could also argue that innovative challenges require the highest priority because it is novel and solutions are needed urgently. The purpose of the model is to help researchers to identify the novelty of the challenges. We leave researchers free to determine the research priorities according to their research context.

By creating the instance, we are able to analyze the causes and effects among the selected challenges and their inter-dependencies. This instance not only can help us determine the novelty of the challenges but also can facilitate us to scope our research area. For instance, let us suppose we decide to study dynamically discoverable & composable. We should keep in mind that the relevant challenges that have inter-dependency relationship with, such as service contract and loose coupling presented in the created instance, should also be studied accordingly.

4 Conclusions and future work

As a new development paradigm, SOSE promises to bring a number of benefits to IT companies and their clients. Due to the complexity of the service-oriented landscape, a well focused SOSE research agenda is required.

This paper proposes a model that can be used to analyze the inter-dependencies of SOSE challenges and to identify the novelty of these challenges. For demon-

stration purposes, we also present an instance of the model which focuses on the SOSE challenges related to the business benefits.

Through establishing the model and creating its instance, we have learned that the advantages of the model are threefold. Firstly, we can better rationalize which are the relevant challenges to serve predefined purposes (such as business benefits). Secondly, by using another SE discipline (such as CBSE) as the key for comparison, we can further determine the novelty (such as innovative, false or requiring tuning) of SOSE research challenges. Such a classification enables us to realize to what extent the challenges that are currently claimed in the literature require more scientific explanation. Thirdly, by tuning the results coming from other disciplines to solve SE challenges, SOSE can also spark new developments in these other disciplines.

As future work, we aim to mature the instance by refining and completing the inter-dependencies between the challenges that are identified so far. For instance, we would not only consider the positive impact that the SOA attributes have on the business benefits, but also their side effects. Thus, trade-off relationships need to be studied. After that, we will create more instances of the model by investigating other benefits and comparing the challenges to other SE disciplines. In this way, we are able to rationalize the results by providing more scientific evidence. Once the instances are complete and mature, we envision using these instances as tools for professionals too, to decide on SOSE investments and to measure their impact.

References

1. Erl, T.: Service-Oriented Architecture: Concepts, Technology, and Design. Prentice Hall (2005)
2. Michael N. Huhns, M.P.S.: Service-oriented computing: Key concepts and principles. IEEE Internet Computing **9**(1) (2005) 75–81
3. Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F.: Service-oriented computing research roadmap (2006)
4. Papazoglou, M.P., Yang, J.: Design Methodology for Web Services and Business Processes. In: Proceedings of the Third International Workshop on Technologies for E-Services (TES 2002). Volume LNCS; Vol. 2444., Springer-Verlag (2002) 5464
5. Zirpins, C., Baier, T., Lamersdorf, W.: A Blueprint of Service Engineering. In: First European Workshop on Object Orientation and Web Service (EOOWS), Darmstadt,Germany (2003)
6. Tsai, W.T.: Service-oriented system engineering: A new paradigm. In: Service-Oriented System Engineering, 2005. SOSE 2005. IEEE International Workshop, Beijing, China (2005) 3– 6
7. Tsai, W.T., Wei, X., Paul, R., Chung, J.Y., Huang, Q., Chen, Y.: Service-oriented system engineering (sose) and its applications to embedded system development. Service Oriented Computing and Applications (2007) 3–17
8. NESSI: Nessi strategic research agenda (vol. 3.fp7-1). (2006)
9. Stevens, M.: The benefits of a service-oriented architecture. Online at www.developer.com (2002)

Methodology for a Precise Development Process of Service Oriented Applications *

László Gönczy

Department of Measurement and Information Systems
Budapest University of Technology and Economics
Budapest, Magyar tudósok krt. 2. H-1117, Budapest- Hungary
gonczy@mit.bme.hu

Abstract. This paper aims at presenting an engineering process which bridges precisely defined service configurations and standard deployment techniques. Service models are described in a well-founded manner using the SRML language, developed in the SENSORIA FP6 project. These models are then basis of i)an ontology-based validation and ii)deployment of services in the emerging industrial service configuration specification SCA. Reasoning systems allow the developer to validate the consistency of model instances. In the SOA context, e.g. architectural compatibility of the model and the specification can be validated and possible infeasibilities of the metamodel can be identified. The design of an Eclipse-based framework is presented in which most of the MDA-conform development tasks are executed in a (semi-)automated way.
Keywords: Model-Driven Deployment, SCA, Ontology-based Validation, SRML, Development Process

1 Introduction

As there is an increasing number of technical implementations of Service Oriented Configurations available, common model-based analysis, development and deployment mechanisms are needed. Therefore, one of the current research trends is to use standard visual modeling languages and derive formal analysis models. This paper introduces a method for analyzing service configuration descriptions and develop configurations with the possibility of a precise analysis. Service Component Architectures (SCA) [15] is used as an industrial modeling and deployment technology. Sensoria Reference Markup Language [7] aims at building a formal framework for service description.

Service Component Architectures (which is now getting on the way of becoming an OASIS standard) is a specification for defining services by their interface description. The power of this specification is given by its technological heterogeneity; it allows the usage of Java, BPEL, PHP and standard Web services technologies (WSDL), with some other technical options being under development. However, current SCA-compliant tools do not support validation of SCA

* This work was partially supported by the *SENSORIA* European FP6 project (IST-3-016004).

models. Therefore, in the Sensoria project the aim is to build a chain to process different types of SRML/UML/SCA/BPEL models and perform formal analysis and generate SCA-conform configuration descriptors from high level models.

2 SOA modeling techniques

This section briefly introduces SCA (an industrial specification for service assembly) and SRML, which is developed in a research project for support a formal framework for service development.

SCA - an Industrial Specification for SOA Description Service Component Architecture is an industrial specification to define service oriented applications above the middleware layer, therefore making the design independent from technical details. The main goals of SCA are (based on [15]) simplifying development, assembly and deployment of services and separating the core business logic from middleware-related decisions.

Main elements of SCA are *components* representing individual business services. Components which are deployed together are called *modules* which can be organized to *subsystems*. Modules have *entry points* and can rely on *external services*. Technical details of accessing these are described by *bindings*. Interfaces can be connected by *wires*.

From the technical point of view, modules can be implemented in Java, BPEL, PHP or as Web services. Each technology has its own *implementation model*. Abstract *intents* can be attached to components to specify policies, such as interaction mechanisms, messaging requirements, security or transactionality attributes of a given service. SCA modules and interfaces are described in specific XML schema files. During my work, I use IBM WebSphere Integration Developer ([12]) which supports parts of the SCA specification.

Precise Definition of Services Using SRML SRML-P (referred here as SRML) was developed in the Sensoria EU project at the University of Leicester and University of Lisbon [7, 6]. The aim of this language is to describe the *definition* of services and their compositions in a middleware-independent way. The main advantage of using SRML is that it can be considered as a bridge from high-level service specifications to formal methods. Further languages of the SRML family, such as SRML-F [3] will extend the service definitions with configuration management description, so that the model can be run on a service execution engine which supports negotiation, SLA management, etc. The formal foundation of such an engine is given by process algebraic semantics. Herein I concentrate on the description of process modules, but in the near future I also plan to integrate SRML extensions (or further language constructs) with industrial standards (such as WS-Security, WS-Transactions, WS-RM, WS-Reliability, etc.).

3 A Development Framework

This section proposes a framework to support the development of service oriented applications.

3.1 General Approach

As scientific results can be communicated by using some industrial technologies to present their benefits, the framework should concentrate on widely known modeling and deployment techniques. This approach is shown in Fig. 1.

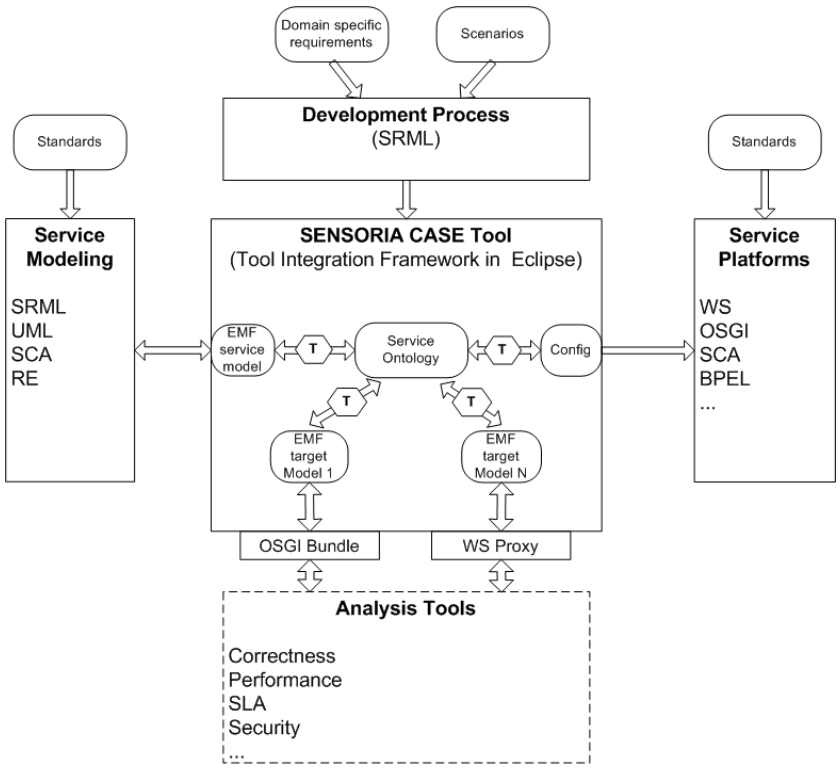


Fig. 1. Model-based development framework for services

As stated above (and also in [7]), SRML has some common goals with SCA. However, currently there is no available runtime support for applications written in SRML. For design time support, in [11] an initial mapping from WSDL and (parts of) BPEL is described. An EMF-based SRML editor will also be developed, but creating complex systems directly in SRML may require additional skills from developers. While SRML is a powerful as the basis of a formal analysis,

service configurations should be generated for an industrial language. Therefore, using SCA as an implementation platform (or, more precisely, implementations of SCA specification) was an obvious decision.

Therefore, we aim at BUTE at generating

- SRML models from high level descriptions
- SCA configurations from SRML models and
- deployment descriptors to standard platforms from SRML-like models.

According to the service-driven approach of Sensoria, we considered development and analysis tools as services. Of course, currently only a subset of model transformations (denoted by a T in a hexagon) are fully implemented. SRML and SCA have corresponding metamodels and basic connections (reference model and transformation) in VIATRA; the basis for their definitions were [7], [15] and [16]. As SRML contains some elements which cannot be directly connected to SCA-compliant solutions (e.g., the *Types* of *Interactions*), the step of generating SCA elements from SRML obviously causes loss of information, the level of which is determined by the target implementation platform.

Analysis of service configurations can be executed by ontology-based validation using the RACER [10] tool. This allows the developer to find infeasabilities of the model and answer questions about (i) the consistency of the model and (ii) domain specific requirements against service configurations. Examples include questions like *"Are all protocol definitions compatible?"* and *"Do security specifications of all composite services meet user requirements?"*. Automatic derivation of such questions for the reasoning tool is part of future work.

3.2 Model Transformations in VIATRA2

VIATRA2 is a modular, Eclipse-based model transformation framework, developed at BUTE [19]. It is a general purpose model transformation tool, with built-in support for parsing high level models such as UML and BPM. Parsers for additional languages can be written as Eclipse plugins (e.g., there is a BPEL parser implemented). In the framework, models are stored as typed, directed graphs. Alternatively, models can be created by the textual editor of the tool, however, in an engineering process this serves only testing purposes. The role of model transformations during the development is to generate models of different description levels and to derive analysis models of the system. Previously, the tool was successfully used in design and analysis of embedded systems, workflow descriptions, etc.

The formalism behind model transformations is graph pattern matching with negative application conditions, described for instance in [4]. To ease the development of complex transformations, in addition to the declarative pattern matching mechanism, VIATRA allows the user to define a control structure of transformation steps using Abstract State Machine instructions.

3.3 Contributions

The main contribution of the current work will be that it bridges an industrial standard modeling technique to a formally defined language (using SCA2SRML transformation), and then descriptors to an industrial deployment platform from formal models (by transformation SRML2SCA), therefore it extends the engineering cycle with precise formal analysis possibilities.

4 Related Work

There are several frameworks in the literature for model-based development of web service systems, just a few examples: [13] also uses high level modeling of web services and their connections as a starting point for deployment.

A framework for automated generation of WSDL files from UML models is described in [18]. A similar problem is addressed in [8]. A previous work at BUTE [9] aims at defining a framework for model-driven development of services taking the model of [2] as a basis for system description. A lot of work deal with extension of existing Web services technologies in order to provide some additional functionality (e.g., [1]), but these works usually do not have a precise metamodel in the background, moreover, they require modifications of existing proxies, clients, etc.

Finally, Service Availability Forum is an industrial specification for combining services to increase the availability of the overall system [14]. SAF as a deployment platform is also targeted by the research at BUTE.

5 Conclusions and Future Work

In this paper a framework for combining the power of a formal description language and industrial standards/specifications was described. The main benefits and novelties of this framework are that on the one hand it uses industrial technologies for runtime management of services and on the other hand combines service modeling with the formal analysis power of SRML.

Nevertheless, there are many future research directions from which the derivation of non(or extra)-functional attributes form SRML languages and inclusion of additional development platforms (such as BPEL) will be the next steps. As the full support of EMF based models in VIATRA is planned for early 2007, SRML models will be directly read from the visual editor (which is also to be completed in a few months) and their VIATRA representations will be synchronized with the design model.

Development processes for domain specific services (which are described in details in Sensoria Case Studies) can also be described in SRML, where services will correspond to development and analysis tools and the "composite service" will represent the development of a concrete application service, which itself is a composite service. The coordination of analysis and deployment steps in such an application can be described in SMRL-P as well. Such an SRML process corresponds to the development of (classes of) domain-oriented applications.

Acknowledgements I would like to say thanks to Daniel Varro for his valuable advice, to Zsolt Deri for feedback about the specifications and to all "Sensorians" for discussions.

References

1. Alwagait, E., and Ghandeharizadeh, S.:DeW: A Dependable Web Services Framework. In Journal RIDE, IEEE Computer Society, 2004. pp. 111-118.
2. Baresi, L., R. Heckel, S. Thöne and D. Varró: Style-Based Modeling and Refinement of Service-Oriented Architectures. Journal of Software and Systems Modelling, Vol. 5(2), pp. 187–207, June 2006.
3. Bocchi, L., J.L. Fiadeiro and A. Lopes: The SENSORIA Reference Modelling Language, Primitives for Configuration Description(SRML-F,v.1.0). SENSORIA Research Report, 2007.
4. Ehrig, H., R. Heckel, M. Korff, M. Lowe, L. Ribeiro, A. Wagner, and A. Corradini:Algebraic approaches to graph transformation. In Handbook of Graph Grammars and Computing by Graph Transformation, volume I: Foundations, pp. 247–312. World Scientific, 1997.
5. Eclipse Modeling Framework, URL: http://www.eclipse.org/modeling/
6. Fiadeiro, J.L., A. Lopes and L. Bocchi:A Formal Approach to Service Component Architecture. In Proc. of WS-FM 2006, LNCS 4184, pp. 193 213, 2006.
7. Fiadeiro,J.L., A. Lopes and L. Bocchi: The SENSORIA Reference Modelling Language, Primitives for Service Description (v.1.3). SENSORIA Research Report, 2006.
8. Gronmo, R., D. Skogan, I. Solheim and J. Oldevik: Model-driven Web Services Development. Presented at the 2004 IEEE Intl. Conf. on e-Technology, e-Commerce and e-Service (EEE-04), Taipei, Taiwan.
9. Gönczy, L., J. Ávéd and D. Varró: Model-based deployment of web services to standards-compliant middleware. In Proc. of ICWI2006, Iadis Press, 2006.
10. Haarslev, V., R. Moller and M. Wessel: RACER User's Guide and Reference Manual Version 1.7.19.
11. Hong, Y.:WSDL and BPEL to SRML-P language transformation. MSc Thesis, University of Leicester, 2006.
12. IBM WebSphere Integration Developer. http://www-306.ibm.com/software/integration/wid/
13. Manolescu, I., S. Ceri, S. Comai and P. Fraternali: Model-driven design and deployment of service-enabled web applications. ACM Trans. Inter. Tech., Vol. 5 , N.3 (August 2005), pp. 439 - 479
14. SA Forum: Application Interface Specification. `http://www.saforum.org`.
15. SCA Consortium. Building Systems using a Service Oriented Architecture. Version 0.9, 2005.
16. SCA Consortium. Service Component Architecture – Assembly Model Specification, Version 0.9, 2005.
17. Software Engineering for Service-Oriented Overlay Computers (SENSORIA) European project IST-3-016004. `http://sensoria.fast.de`
18. Vara, J.M., V. de Castro and E. Marcos: 2005. WSDL Automatic Generation from UML Models in a MDA Framework. In International Journal of Web Services Practices, Vol.1, No.1-2, pp. 1-12.
19. VIATRA2 Framework, An Eclipse GMT Subproject, URL: http://www.eclipse.org/gmt/.

Policy-Based Service Selection

Helge Janicke[1] and Monika Solanki[2]

[1] heljanic@dmu.ac.uk
Software Technology Research Laboratory,
Gateway House, De Montfort University,
Leicester LE1 9BH
[2] monika@doc.ic.ac.uk
Department of Computing
Huxley Building, Imperial College,
London SW7 2AZ

Abstract. Conventional means of selecting a service are predominantly based on the results returned from service discovery. Typically most approaches employ a matchmaker which works in tandem with a service registry and user preferences based on the functionality of services. Non-functional attributes of services, such as reliability, timeliness and other quality of service criteria are typically not addressed or only in the static form of a contract that is made between the provider and the consumer of a service. In this paper we propose an extended architecture that allows consumers to specify additional policies for service selection based on their preferences for the non-functional service requirements. Policies are enforced by a policy-engine and are based on past requests made by the consumer.

1 Introduction

With the proliferation of Web Services as a business solution to enterprise application integration, the *Quality-of-Service* (QoS) offered by Web Services is becoming the utmost priority for service providers and consumers. This means constant availability, connectivity, and high responsiveness are key to keeping a business competitive and viable. Due to the dynamic and unpredictable nature of networks over which services are deployed, providing the acceptable QoS is a challenging task. It is important to accurately estimate the levels of service from the perspective of both clients and web service providers. QoS metrics for a service include several dimensions such as time, cost, performance, reliability, accessibility, throughput, availability, interoperability and security. In most existing approaches for the management of QoS, either quality is described as ad-hoc properties or a generic (agnostic) approach to quality description is taken, with quality description based on external definitions. In this paper we propose to augment the standard service oriented architecture with the notion of an observer and a policy engine. The former enabling the observation of dynamic QoS attributes, the latter to provide a policy-based approach to address the problem of QoS-based service selection from a set of already discovered services while

maintaining the integrity of their respective QoS properties and the QoS of any service composition.

During Web service compositions, the providers and consumers define binding service level agreements (SLAs) also referred to as a policy or contract that specifies various dimensions of the service such as delivery deadlines, quality of the product being traded, production standards to be adhered to and cost of the delivered product. The SLA is a legally binding document and specifies the contractual obligations of the service provider with respect to the guaranteed level of service and the penalties associated with failure to comply with the contract. Such an agreement collectively defines the "quality-of-service"(QoS) that is expected to be delivered by the service provider. Whether offered within an organisation or as a part of a paid service across organisational boundaries, QoS aspects of services are important in a service-oriented computing environment. While managing QoS in distributed systems is not a novel problem, a number of additional issues arise in the context of a service-oriented computing environment.

- In traditional distributed systems, the network and system resources are within the control of one organisation and its partners. In the domain of Web services, the consumers and providers can rapidly change, making the nature of networks unpredictable.
- In the case of composite services, the QoS properties of individual services in the composition contribute to the overall QoS. Composition of composite services may often yield aggregated QoS properties that may not be directly derived from those of the individual services. An important question that arises here, is how can the provider of a composite service ensure adherence to the overall QoS requirements.

The monitoring and control of QoS properties for a service has several advantages such as ensuring correct behaviour of the service, maintaining or surpassing QoS levels expected by consumers, identifying and addressing problems and timely response to change. Publication of QoS properties for a service, along with its capabilities can help service consumers in selection between provider Web services with the same or similar functionality.

2 Architecture

In current service based systems, services descriptions are published in the registry, where all service specifications are stored. These specifications can be discovered by other agents that require the execution of a specific service. Alternatively to leaving the task of service discovery to the application level, the infrastructure can provide dedicated match-making services to lift the burden of the application programmer. However, match-making does typically only take into account the service specification as advertised in the registry and the service properties that have been requested by the consumer application. The result is a set of matching services, from which the application must choose according to

some preference. Typically these preferences are dependent on the actual QoS, that in most cases is not directly observable by the application itself.

Our proposed architecture, depicted in Figure 1, assists service consumers in this selection process to find the most suitable provider. This choice can depend on various factors such as the actual QoS that is provided or the provider's interactions within the system. We propose that a consumer registers a policy with a *policy-engine* that is part of the infrastructure. The policy represents its preferences with respect to service selection. As every consumer can submit its own policy, the selection is highly customised and tailored to the individual needs. From the policy we can determine which information is relevant [1] to obtain preferences and deploy or adjust mechanisms that observe the interaction between providers and consumers. We term as *observers* dedicated components that observe the interaction between services at the message passing level or by means of invasive or non-invasive instrumentation.

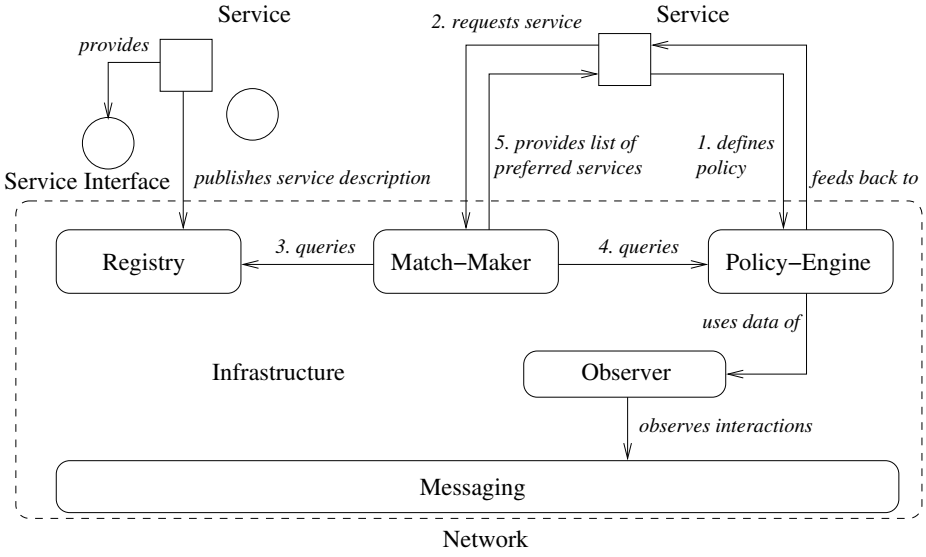


Fig. 1. Service Architecture

The consumer initially registers a preference policy with the policy engine, before requesting a service from the matchmaker. The match-maker then queries the registry to discover suitable services and subsequently selects according to dynamic QoS attribute, by ordering them according to preferences provided by the policy engine. The policy engine constantly evaluates the policies against the information that is provided by the observers and adjusts the individual preferences of providers accordingly. In case the service is making a request, the matchmaker orders the discovered services according to the preferences of the consumer. Policies can also define obligations, viz. actions that the policy engine

must perform under specific conditions. For example, if priorities for service provision change then the policy engine must notify the service of this change. This potentially leads to a re-binding to a more preferable service. In case the change of preference occurs during a transaction it is at the discretion of the service to either immediately terminate the transaction or to rebind after the transaction completed. This choice is highly domain-dependent.

The advantage of this approach is that dynamic QoS attributes that are otherwise not observable by the application itself can be taken into account. One example is the provider's interactions with others. Also the burden of an explicit preference management is lifted from the application programmer as this is delegated to the policy engine in form of a preference policy. Of course preference policies are domain dependent and policy engine may not be able to evaluate all policies, for example if the policy references observations which the infrastructure is incapable of providing. However this can be analysed and fed back to the service.

3 Preference Policies

The preferences that underlie the provider selection are defined in the form of preference policies. These policies are written from the service consumer's perspective and are enforced by the policy engine. Preference policies are expressed in the form of condition-action rules. They are based on a policy model that has been introduced in previous work e.g. [2]. The condition of a rule represents a set of behaviours that triggers the action. The model has a sound formal semantics in ITL [3], and therefore lends itself for applications that require a high level of assurance. Due to space limitations we do not introduce the model here and refer the reader to our previous work e.g. [2]. Rules are of the form:

when b [**increase** | **decrease**] **preference in** s [**little** | **medium** | **strong**]

where b is the specification of a set of behaviours and s is the name of a service. Here the increase or decrease in preference is modelled as an atomic action that assigns a new preference value. The concrete effect of these actions is dependent on the underlying preference model. For the purposes of this paper we assume a simple model where preference is modelled as a single integer. Higher values denote a higher preference. We assume that initially there is no established preference, viz. all preference values are 0. In this setting we define the meaning of little, medium and strong to mean 1, 2 and 3.

3.1 A Stock-Quote service

We take the example of a stock-quote service, where the stakeholders are a customer (C) and three different stock exchange services (SE_1, SE_2 and SE_3) that are discovered by the matchmaker. Initially the customer is undecided about the preferences in these services. As none of the services provides guaranteed

response times as part of their service specification, the customer defines a policy that states that services that has always provided a response-time of less than 2 sec over the last 10 interactions are preferred to those that at some point exceeded 5 sec. Any service where the response-time exceeds 10 sec is strongly not preferable. This is expressed using the following three policy rules.

scope (SE_1,SE_2,SE_3) :
when 10: always responsetime(s) <= 2 sec **increase preference in** s **medium**
when 10: sometime responsetime(s) >= 5 sec **decrease preference in** s **little**
when 0: responsetime(s) >= 10 sec **decrease preference in** s **strong**

Here the preceding t : in the premises of the rules state that the behaviour of s observed in the last t interactions is considered. Given these rules, the policy engine determines the information that is needed for its enforcement. In this case it is the response-time of services SE_1, SE_2 and SE_3 over the last 10 interactions. The observer will then start to record the response-time for these three services. With every service invocation the policy engine will check the premise defined by the rules to determine whether the rule fires. For example if response-times for the following 12 service requests for SE_1 were as depicted in the table below then the enforcement of the preference policy would be as follows.

request:	1	2	3	4	5	6	7	8	9	10	11	12
response-time:	11sec	2sec	2sec	1sec	1sec	1sec	1sec	2sec	2sec	2sec	2sec	5sec

After the first request the last policy rule fires, as the response-time is longer than 11 sec. This leads to a reduction of service SE_1's preference from 0 to -3 (**decrease ... strong**). Following this none of the policy rules fire as the interactions have a good response-time. Reaching request 11 the first policy rule fires, as over the last 10 requests the response-time was indeed less or equal to 2 seconds. This increases the preference value from -3 to -1. With the last request the response-time degrades again and fires the second rule decreasing the preference to -2. Depending on interactions that have been observed with the other potential providers a preference ordering can be established and C is able to select the best provider given its stated preference policy.

4 Related Work

In the domain of Web services, several industrial standardisation efforts such as HP's Web Service Management Framework (WSMF) [4], IBM's Web Service Level Agreement (WSLA) [5,6], the Web Service Offering language (WSOL) [7,8] and approaches based on Web service policies (WS-Policy) [9] define some efforts to formalise the QoS dimensions. Ontologies for semantically described Web services, such as OWL-S [10] and WSMO [11] consider very generic aspects of QoS. Web services are described by functional and non-functional properties. Cost, time, and reliability are the only dimensions analysed in the METEOR-S Project [12] that proposes a framework for the annotation of Web Services.

Several approaches have been studied within the service-oriented computing domain that monitor behaviour of services, however they do not address policies defined by individual parties. Li et al. [13] presents a framework for runtime monitoring of interaction behaviour of services against pre-defined interaction constraints specified using pattern and scope operators from the Property Specification Pattern System. In the framework proposed by Baresi et al. [14] assertions are specified on service compositions modelled as BPEL processes. Assertions can be specified in CLIX as well as directly as code snippets within a C# implementation in the .NET environment. A major limitation here is that only pre and post conditions can be specified as assertions. Lazovik et al. [15] propose an approach based on interleaving planning and execution where assertions are specified in an XML based language. In [16] WS-Policy has been used as a language for specifying non-functional constraints (i.e. security) to be monitored. Constraints are specified using the Web service Constraint language (WS-Col) which are embedded in a WS-Policy file for the service. Mahbub et al. [17] proposes a framework for runtime monitoring of services composed in BPEL4WS where monitoring requirements are specified in Event calculus.

5 Conclusion and Future Work

In this paper we extended the conventional service-oriented infrastructure with the notion of a policy engine and the concept of observers. The policy engine accepts preference policies and evaluates them on behalf of a service in the light of available observations made of the behaviour of other services. This is used to establish a preference ordering between potential services that have been discovered by the matchmaker. The advantage of this approach is that the selection of services can depend on dynamic attributes of these services and can lead to a dynamic reconfiguration in case that the preference ordering of potential service providers change over time. Furthermore by expressing preferences as policies the burden of implementing the selection process at the application level is reduced, as policies express these requirements at a much higher level of abstraction. Enforcing policies centrally allows also for an extended set of observations that can be made, e.g. taking into account the interactions of others with these potential providers.

This of course also raises security concerns, as to what degree information about the behaviour of other services can be deduced from preference orderings. We aim to address this issue by allowing services to specify access control policies that clearly state which other services can or cannot observe measures on interactions, enabling the users of such an infrastructure to control what traces their interactions are observable by others. We also plan to address the issue of contract validation within this framework. The policy engine for example could enforce contractual agreements between services, for example policies that enforce a penalty payment if a certain agreed QoS level is not maintained.

References

1. Janicke, H., Cau, A., Siewe, F., Zedan, H.: Deriving Enforcement Mechanisms from Policies. In: to appear in Procceedings of POLICY2007, IEEE (2007)
2. Janicke, H., Cau, A., Siewe, F., Zedan, H., Jones, K.: A Compositional Event & Time-based Policy Model. In: Procceedings of POLICY2006, London, Ontario, Canada, IEEE (2006)
3. Cau, A., Moszkowski, B., Zedan, H.: The ITL homepage. online (2005)
4. Catania, N., Kumar, P., Murray, B., Pourhedari, H., Vambenepe, W., Wurster, K.: Web service management framework, version 2. Technical report, HP Labs (2003)
5. Ludwig, H., Keller, A., Dan, A., King, R.: A service level agreement language for dynamic electronic services. In: WECWIS '02: Proceedings of the Fourth IEEE International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems (WECWIS'02), Washington, DC, USA, IEEE Computer Society (2002) 25
6. Ludwig, H., Keller, A., Dan, A., King, R., Franck, R.: Web service level agreement (wsla) language specification, version 1.0. Technical report, IBM Corporation (2003)
7. Tosic, V., Pagurek, B., Patel, K.: Wsol - a language for the formal specification of classes of service for web services. In: ICWS. (2003) 375–381
8. Tosic, V., Patel, K., Pagurek, B.: Wsol - web service offerings language. In: CAiSE '02/ WES '02: Revised Papers from the International Workshop on Web Services, E-Business, and the Semantic Web, London, UK, Springer-Verlag (2002) 57–67
9. Box, D., Curbera, F., Hondo, M., Kaler, C., Langworthy, D., Nadalin, A., Nagaratnam, N., Nottingham, M., von Riegen, C., Shewchuk, J.: Web services policy framework (ws-policy), version 1.1. (2003)
10. The OWL-S Coalition: OWL-S 1.1 Release. (2004) http://www.daml.org/services/owl-s/1.0/.
11. ESSI WSMO working group: Web Service Modelling Ontology (2004) http://www.wsmo.org.
12. Patil, A.A., Oundhakar, S.A., Sheth, A.P., Verma, K.: Meteor-s web service annotation framework. In: WWW '04: Proceedings of the 13th international conference on World Wide Web, New York, NY, USA, ACM Press (2004) 553–562
13. Li, Z., Jin, Y., Han, J.: A runtime monitoring and validation framework for web service interactions. In: ASWEC '06: Proceedings of the Australian Software Engineering Conference (ASWEC'06), Washington, DC, USA, IEEE Computer Society (2006) 70–79
14. Baresi, L., Ghezzi, C., Guinea, S.: Smart monitors for composed services. In: ICSOC '04: Proceedings of the 2nd international conference on Service oriented computing, New York, NY, USA, ACM Press (2004) 193–202
15. Lazovik, A., Aiello, M., Papazoglou, M.: Associating assertions with business processes and monitoring their execution. In: ICSOC '04: Proceedings of the 2nd international conference on Service oriented computing, New York, NY, USA, ACM Press (2004) 94–104
16. Baresi, L., Guinea, S., Plebani, P.: Ws-policy for service monitoring. In: TES. (2005) 72–83
17. Mahbub, K.; Spanoudakis, G.: Run-time monitoring of requirements for systems composed of web-services: initial implementation and evaluation experience. In: IEEE International Conference on Web Services. Volume 1., IEEE (2005) 257–265

Course Generation as a Web-Service (CGWS) for E-Learning Systems *

Tianxiang Lu, Carsten Ullrich, and Babara Grabowski

German Research Center for Artificial Intelligence
Stuhlsatzenhausweg 3, D-66123 Saarbrücken, Germany
{tianxiang.lu,carsten.ullrich}@dfki.de

Abstract. A course generator provides personalized learning experiences by assembling structured sequences of learning objects that help learners to achieve their learning goals. They can be stored in a multitude of repositories and are selected by the course generator based on a set of pedagogical methods that take into account the learners' goals and individual properties such as mastery. This pedagogical knowledge needs to be elicited from pedagogical experts and hence is rather expensive to assess. Therefore, once a course generator has been developed, it makes sense to make course generation available as a web-service for web-based e-learning environments in order to reuse the pedagogical knowledge. In this paper, we present the web-service we developed for the course generator of the LeActiveMath system. We describe the service oriented architecture during the design and the suitability via the application.

1 Introduction

Traditional web-based e-learning environments like Moodle offer access to learning resources using courses authored by humans. These courses are mostly static, hence each learner is presented with the same sequence of learning object, regardless of his individual needs. In contrast, course generation [2] allows the automatic generation of a structured sequence of learning objects that is adapted to the learners' competencies, individual variables, and learning goals. A course generator implements human expert knowledge, which is hard and expensive to assess. The contribution of this paper is to use web-service standards for providing a *Course Generator as Web Service* (CGWS). In contract to very broad approaches that define complete frameworks for educational services (e.g., [7]), this work focuses on a specific service, namely how to access the course generation knowledge in an (mostly) automatic matter. This papers presents the implementation of the CGWS (the master thesis of the first author); the underlying research is presented in [5, 6, 4]. In order to assess the potential interest and requirements of third-parties in a CGWS, we performed a survey containing several questions about potential use cases, e.g., general interest in course

* This publication was generated in the LeActiveMath project, funded under the 6th Framework Programme of the European Community (Contract Nr. IST-2003-507826). The authors are solely responsible for its content.

generation, but also technical questions, e.g, the learner models used, etc. The results of the survey served as the basis of the requirements analysis of the CGWS. Based on the requirements, we designed a loosely coupled architecture for CGWS, whose basic components we will describe in the next section. Following a top-down approach, in § 2 we define the interfaces between the server (CGWS) and the client (external web-based e-learning environments). We then illustrate the interactions between them. In the final section, we present examples of applications of the CGWS, encountered difficulties and lessons learned.

2 Design

The basic components needed for course generator are *content repositories* and *learner models*. The content repositories store the learning objects that are used by the course generator to assemble courses. However, most repositories use their own metadata schema and storage technology. Integrating data from different repositories is a well-known problem and is typically tackled by a *mediator*, which provides a uniform interface for accessing multiple heterogeneous data stores (e.g. file systems and different databases) [8]. In case of the web-service world, it is necessary that repositories can be added and removed without human intervention from the mediator side: a service should be able to register its repositories automatically. A solution to this problem is described in § 2.1.

A *learner model* [1] stores information about the learner (e.g. properties, preferences and the degree to which he has mastered the content) and hence provides information that is necessary for the personalized generation of courses. A CGWS requires that client can register its learner model, so that the CGWS can access it during course generation. The challenge for such an interface is that it has to cope with various types of learner models, but also the fact that some web-based e-learning environments do not even have a learner model.

A *course generator* uses the information provided by the above components and assembles learning objects from one or many repositories into a personalized course with respect to a pedagogical learning goal (a task, [6]). Tasks are fundamental concepts in course generation. A pedagogical task represents the learning goal of a user and consists of a *pedagogical objective* and of a *set of learning objects* that specifies the course's target concepts. An example is the task *(discover (def_slope))*, which represents the learning goal of a learner who wants to discover and understand the mathematical definition of the "slope". A different learning goal is, say, *(train (def_slope))*. A course supporting the user in reaching the former learning should contain all learning objects required by this individual user to understand the goal concept.

2.1 Interfaces of the CGWS

The CGWS provides two main kinds of interfaces: the core interface and the repository integration.

The core interface consists of the following methods: *getTaskDefinitions()* is used to retrieve the pedagogical tasks which the course generator can process (described in XML format); *generateCourse(objective, learning object _Ids, learnerId)* starts the course generation on the given educational objective, identifiers of learning objects and the identifier of learner (or LearnerKnowledgeMap, see § 2.3). The result of the course generator is a structured sequence of learning objects. The format of the result complies to IMS-CP, an established standard for the exchange of courses between different web-based e-learning environments.

The interface for repository registration consists of the following methods: *getMetadataOntology()* provides the client with a description of the metadata structure used in server. The description is provided as an ontology that describes the pedagogical types and relationships of learning objects [5].

The method *registerRepository(name, rep_url, ontology_url, ontology_map_url, testLO)* registers the repository that the web-based e-learning environments client wants the course generator to use. Since different repositories often use different metadata for describing the learning objects they store, the client needs to provide information to the CGWS that helps to "understand" the metadata. Therefore, the clients specifies the URL of the ontology describing the metadata structure used in the repository and in addition an URL of an ontology mapping between the two ontologies. This way, the mediator can translate between the metadata used in the server and in the client.

The method *unregisterRepository(rep_url)* unregisters the given repository.

2.2 Client Interfaces

A web-based e-learning environments client needs to provide the following interfaces: the *ContentAPI* needs to be able to answer queries coming from mediator that ask for (a) the type of the given learning object; (b) all the learning objects that are connected to a given learning object by a given relation; (c) all properties the given learning object has [4].

The *LearnerPropertyAPI* makes the learners' properties accessible to the CGWS in case the client contains a learner model and wants the course generator to use it. In the current implementation, this interface is not yet implemented. It would require a mediator architecture similar to the one used for repository integration.

2.3 Interaction between Client and Server

In this section we describe the communication between client and server. We will focus on the most relevant interactions.

Repository Registration The registration phrase happens as follows: In a first step, the client retrieves the metadata ontology used in the CGWS. The ontology is then used to generate a mapping between the metadata used on the server and the one on the client. In the current system, the existing mappings were manually authored. Then, the repository is registered using the

method *registerRepository()*. The repository is added to the list of available repositories and made known to the mediator. Subsequently, the mediator fetches the ontology mapping from the client and generates a wrapper for querying the *contentAPI* of the client.

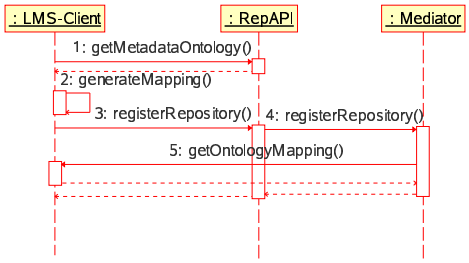


Fig. 1. web-based e-learning environments client registers its repository into CGWS

The most important step in the registration and the contribution of this work is the automated generating of a wrapper (adapter or proxy), which takes charge of binding and querying the *contentAPI* during the course generation. The *contentAPI* should be provided as a web-service, so that the communication can be performed automatically. In this way, a repository can be registered automatically during run-time, without human implementation work required for the wrapper.

Generating a course A client starts the course generation using the service method *generateCourse()*. During the generation, the course generator queries the mediator for the learning objects needed for the course but also the learner model for information about the learner. After the course was generated, the course generator will return an internal data structure that represents the structure of the course. This structure is transformed to an IMS manifest, packaged in a SOAP message and send to the client.

Often, web-based e-learning environments do not have a learner model, but still want to generate adaptive courses for their users. For these cases, the course generator allows to define a temporary learner model, which consists of property-value pairs that represent the current knowledge of the user. This model is called *LearnerKnowledgeMap*. For instance, a first year university student who masters Differential Calculus can have the following *LearnerKnowledgeMap*: (("'Definition of Differential Calculus, good"'), ("'Educational level, University, 1. Semester"')). A client can use a *LearnerKnowledgeMap* by calling the service method *generateCourseWithLearnerKnowledgeMap()*. If the course generator requires the value of a property not stored in the map, a default value is used.

3 Applications

The design was implemented using Java and Axis2. The logic part on the server side is implemented as Java packages, which call the course generator directly using Java-API or using XML-RPC if the server is distributed. Axis2 is responsible to facilitate the web-service details, which includes the WSDL generation and the hot deployment of web-service. SOAP is used for message exchange.

The course generator described in this paper was developed in the FP6 project LeActiveMath, in a joint cooperation between AI and pedagogical experts. Figure 2 contains a screenshot of a course generated in LeActiveMath.

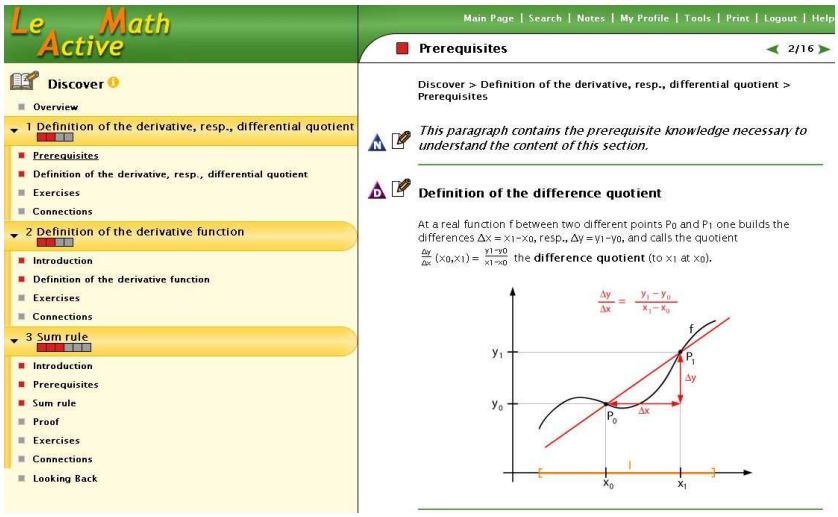


Fig. 2. A generated book in ActiveMath

An external client of course generator was developed at the University of Applied Sciences Saarland. The MathCoach system is a web-based e-learning environments tool especially designed for exercises and experiments within a mathematical context [3]. Using the mechanisms described above (e.g., an ontology mapping), it was possible to make the functionalities of the course generator available to MathCoach.

A third use case of remote access to the course generator was done in the teal project, which targets e-learning in office environments. This illustrates the general applicability of our work. Currently we are working on an integration of the CGWS into the web-based e-learning environments of the distant university of Shanghai Jiao Tong University.

During the application we encountered several difficulties: (a) while the mediator architecture allows an easy technical integration, it is still a challenge

to design the ontology mapping, especially in the often occurring case of poor metadata; (b) for technically unexperienced clients the usage of Web services is still challenging despite being a standardized technique today.

4 Conclusion and outlook

In this paper we described a service oriented architecture for course generator implemented as a web-service. This allows external clients to access the services provided by the course generator, a component that substantially relies on pedagogical knowledge and therefore is expensive to design. This work is the first approach to provide such a *course generator as web-service* (CGWS) to external web-based e-learning environments, which improves the remote communication between systems or machines and enables a cooperated environment of developing web-based e-learning environments. Additional research needs to be invested regarding the generic integration of learner models. In the near future, we will provide an official web-service access point for CGWS so that other web-based e-learning environments can reuse the course generator.

References

1. Peter Brusilovsky and Eva Millán. User models for adaptive hypermedia and adaptive educational systems. *The Adaptive Web: Methods and Strategies of Web Personalization*, 2007.
2. Peter Brusilovsky and Julita Vassileva. Course sequencing techniques for large-scale webbased education. *International Journal of Continuing Engineering Education and Lifelong Learning*, 13(1/2):75–94, 2003.
3. Barbara Grabowski, Susanne Gäng, Jörg Herter, and Thomas Köppen. MathCoach and LaplaceScript: Advanced Exercise Programming for Mathematics with Dynamic Help Generation. In *International Conference on Interactive Computer Aided Learning ICL*, Vilach, Austria, 2005.
4. Philipp Kärger, Carsten Ullrich, and Erica Melis. Integrating learning object repositories using a mediator architecture. In *Proceedings of the First European Conference on Technology Enhanced Learning*, pages 185–197, Heraklion, Greece, oct 2006. Springer.
5. C. Ullrich. Description of an Instructional Ontology and its Application in Web Services for Education. In *Proceedings of Workshop on Applications of Semantic Web Technologies for E-learning, SW-EL'04*, pages 17–23, Hiroshima, Japan, 2004.
6. C. Ullrich. Course generation based on HTN Planning. In A. Jedlitschka and B. Brandherm, editors, *Proceedings of 13th Annual Workshop of the SIG Adaptivity and User Modeling in Interactive Systems*, pages 74–79, 2005.
7. Peter Westerkamp. *Flexible Elearning Platforms: A Service-Oriented Approach.* Logos Verlag, Berlin, 2005.
8. G. Wiederhold. Mediators in the Architecture of Future Information Systems. *The IEEE Computer Magazine*, 1992.

Automated Web Service Composition in Practice: from Composition Requirements Specification to Process Run.

Annapaola Marconi, Marco Pistore, and Paolo Traverso

ITC-irst,
via Sommarive 18, Trento, Italy
{marconi,pistore,traverso}@itc.it

Abstract. Several works address the problem of the automated composition of stateful services, e.g., specified in WS-BPEL. However, the key problem of their practical applicability in real composition scenarios is still open. Addressing this problem requires to provide an easy and affordable way to specify behaviours of component services and complex composition requirements, as well as composition techniques that are powerful enough to scale to scenarios of realistic size. In this paper we present our approach for the specification of composition requirements, we briefly explain how it can be integrated within an existing automated composition framework and show the feasibility and efficiency of the automated composition task on a real scenario that entails a high level of complexity: the Amazon E-Commerce Services and an e-payment service offered by an important Italian bank.

1 Introduction

Service composition is one of the fundamental ideas underlying service-oriented applications: in order to achieve, a given business goal, complex services are obtained by combine existing services published on the Web. The ability to automatically compose web services is an essential step to substantially decrease time and costs in the development, integration, and maintenance of web services. To be used in practice, automated composition should handle the complexity of real world component web services and being able to generate an executable, ready to run and possibly easy refining new web process.

Several works address the problem of the composition of stateful services, see, e.g., [8, 4, 5, 12, 11, 10]. However, the key problem of the practical applicability of these approaches in real composition scenarios is still open. Addressing this problem requires to answer questions such as how to specify the stateful behavior of the component services, how to specify the business requirements that define the goal of the composition, and whether the composition techniques are powerful enough to scale to scenarios of realistic size.

In the paper we show that the features offered by our composition framework in particular the possibility to define complex, structured business requirements for the composition and the very efficient underlying composition techniques are adequate to handle complex industrial composition scenario. The considered scenario requires the composition of two real services, namely the Amazon E-Commerce Services [1] and the e-payment service offered by an important Italian bank. The goal of the composition is to generate an e-Bookstore application that allows to order books and buy them via a secure credit card payment transaction. This composition scenario is particularly challenging since all component services export complex interaction protocols and handle structured data in messages. As a consequence, developing by hand the composite service that orchestrates the components, e.g., in terms of a WS-BPEL process, is a complex, time consuming and error prone task.

We show that the approach reduces dramatically the effort for the composition task by automatically generating a complex executable WS-BPEL process in few minutes starting from composition requirements that can be easily specified and that have an intuitive graphical notation. This result provides a first positive answer to the question of the practical applicability of automated composition techniques.

The rest of the paper is structured as follows. In Section 2 we present our e-Bookstore composition scenario. In Section 3 we describe how the composition requirements for this scenario can be expressed in our framework, while in Section 4 we briefly present our automated composition approach and show how it works on the e-Bookstore scenario. Finally, Section 5 concludes the paper with some concluding remarks and a discussion on future work.

2 The e-Bookstore Composition Domain

In this section we briefly introduce the e-Bookstore (eBS) case study. The idea is to automatically synthesize a composite process that allows potential customers to search for books, add them to a

virtual cart, checkout the order and monitor the credit card payment process. To accomplish its task, the eBS interacts with three separate, independent, and existing services: a service that allows to search books on Amazon.com catalog, a service that handles virtual carts, and a credit card payment service. We suppose that the behavior of each component service is specified through its WSDL [6] and abstract WS-BPEL [3] descriptions.

Amazon E-Commerce Service (ECS) exposes Amazon's product data and e-commerce functionality: from retrieving information about products in the Amazon.com catalog, to handling customer shopping cart, to inspecting content from customers and vendors. ECS publishes a WSDL document [2] that defines all the available ECS operations, their messages, and the data structure of each message. Together with the WSDL description, Amazon provides several documents (see e.g. [1]) describing in details how to submit requests to ECS and the data that is returned by the service, as well as how to handle errors. As one can see from its WSDL description, all ECS operations are synchronous atomic (request-response) web service invocations. However, in order to actually work, these operations must be invoked in a precise sequence of steps. In practice, they belong to specific business workflows. In the Amazon ECS Developer Guide [1] these workflows are described informally. In order to make the workflow explicit and formally defined, we modeled the abstract WS-BPEL specification of the Amazon Book-Search (ABS) and the Amazon Virtual-Cart (AVC) services starting from the descriptions in [1].

In the following example we show the obtained abstract WS-BPEL specifications of the AVC and ABS services.

Example 1 (The Amazon Book Search and Virtual Cart interfaces). Figure 1 contains the compact representation of the abstract WS-BPEL specification of the AVC and ABS services[1]. When the AVC receives a request to create a new cart and the operation is successful, the client can start to add items and eventually checkout its shopping cart. If the checkout is successful, the client can either clear the cart or keep its content for future use. In all these interactions if something goes wrong the AVC sends an error message describing the reason of the fault. It is important to notice that each message part (e.g. part body of message cartCreate in Figure 1) has a complex structure. The precise definition of each complex data type can be found in the ECS WSDL specification (see [2]).

The ABS protocol is pretty simple: the client sends its identification information through the login request and, if the authentication is successful, he can repeatedly send search requests or logout from the service. In the ABS, as well as for the AVC, the data type of each message part is a complex XML Schema type and its definition can be found in [2].

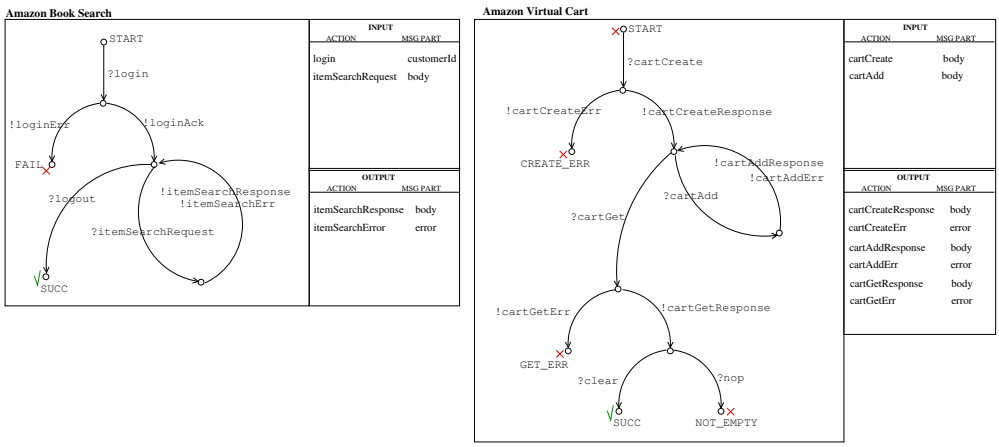


Fig. 1. The Amazon Book Search and Virtual Cart services

The Virtual Point of Sale service (VPOS) models a real on-line payment procedure offered by an Italian bank. The next example describes the interaction protocol that the VPOS expects online shops to follow when using the service.

Example 2 (The Virtual POS interface). When the VPOS receives the request to start a new payment procedure, it checks the correctness of the request (identity of the online shop) and either sends an

[1] Labels of input transitions start with a "?", labels of output transitions start with a "!". Moreover the final states of the protocols are marked either as successful (symbol ✓) or as failing (symbol ×) states. These minimal "semantic" annotations are necessary to distinguish those executions that lead to a successful completion of the interaction from those that are failed.

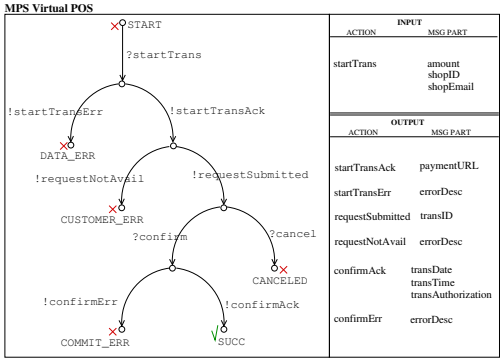


Fig. 2. The Virtual POS service

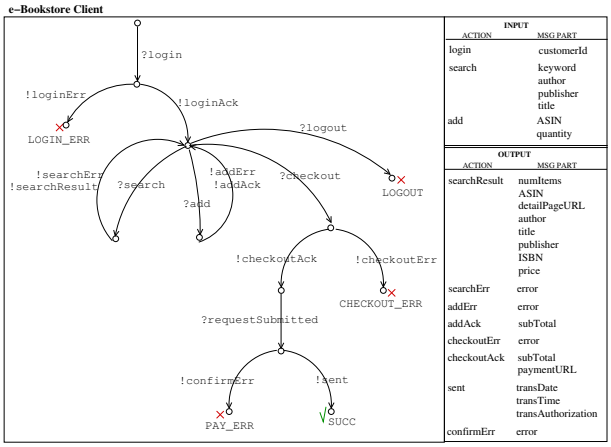


Fig. 3. The customer interface

error message or a message carrying the information about the URL (part `paymentURL` of message `startTransactionAck`) that the shop must communicate to its customer. This information will be used by the customer to communicate its payment data (identi cation information and credit card number) directly to the bank. Notice that this protocol is such that the online shop never has direct information on the customer sensitive data. Once the interaction between the bank and the customer has occurred, the `VPOS` noti es the online shop the outcome. At this point, the online shop can either `confirm` or `cancel` the payment transaction. If con rmed, the transaction is executed by the bank and the outcome (carrying all transaction details if successful) is sent to the online shop.

In addition to the descriptions of the two Amazon services and of the MPS service, we need to de ne as input to the composition problem the interaction protocol that the `eBS` exposes to its customers.

Example 3 (The e-Bookstore customer interface). As a rst step (see Figure 3), the customer is required to `login` using its unique identi cation code. Once his identity has been veri ed, he can start interacting with the `eBS` searching for book offers and adding them to his virtual cart. When the customer checkouts its cart to conclude the order and attests the payment by sending the transaction information (`transID`), the e-Bookstore can either send a con rmation of the order, carrying all the details of the payment transaction, or an error message.

3 Composition Requirements

Given the description (i.e. the WSDL and abstract WS-BPEL) of the component services (`ABS`, `AVC`, and `VPOS`) and of the customer interface (`eBS`), the next step is the formal speci cation of the composition requirements. As we will see in the rest of this section we propose a simple way for the developer to express requirements that de ne complex conditions, both for what concerns the control ow and for the data exchanged with the component services.

3.1 Control Flow Requirements

The `eBS` service main goal is to sell books . This means we want the `eBS` to reach a situation where the customer has lled his virtual cart, con rmed the order and payed through the online payment procedure. However, it may be the case that there are no available books satisfying the customer search, or that the customer doesn't conclude the order, or that the payment transaction fails. We cannot avoid these situations, therefore we cannot ask the composite service to guarantee this requirement. In case this requirement cannot be satis ed, we do not want the `eBS` to con rm order of books without being sure that our customer accepted the offer and that the payment procedure was successful, as well as we do not want displeased customers that have payed books that are not available. Thus, our global termination requirement must take into account the transactionality of each component service within the overall composition. The control- ow requirement should be something like: do whatever is possible to sell books and if something goes wrong guarantee that there are no single commitments .

The following example describes the control- ow requirements speci cation for the `eBS` composition problem.

Example 4 (e-Bookstore control- ow requirements). In the speci cation of each service interaction protocol (see Figures 1, 2, and 3) some states are marked as successful (symbol ✓) and others as failing

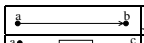
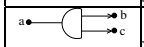
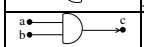
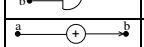
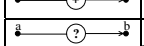
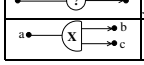
	*forwarder*: simply forwards data received on the input node to the output node
	*function*: upon receiving data on all input nodes, computes the function result and forwards it to the output node
	*fork*: forwards data received on the input node to all the output nodes
	*merge*: forwards data received on some input node to the output node, preserving temporal order
	*cloner*: forwards, one or more times, data received from the input node to the output node
	*filter*: receives data on the input node and either forwards it to the output node or discards it
	*xor*: forwards data received on the input node to (exactly) one of the output nodes

Table 1. Basic elements of the data-flow requirements specification language.

(symbol $\times$). These annotations are used to specify the transactional requirements of the composition problem. In particular, if we consider the eBS scenario, the speci cation is the following:

	eBS	ABS	AVC	VPOS
Primary	✓	✓	✓	✓
Secondary	×	✓/×	×	×

The speci cation distinguishes two different requirements: a primary and a secondary one. The primary requirement is to reach a situation where all the component services are in a successfull state (in our case it models the condition sell books). The secondary requirement (modeling the condition no single commitments) is to reach a situation where all the component services are in a failing state. Notice that in the secondary requirement the ABS can either be in a successful or failing state, this depends on the fact that such a service, unlike credit card payment or cart handling, doesn't need transactionality: we do not care whether the search is successfull in case of failure of the other services.

Our approach thus provides the developer with the ability to specify with a simple tabular notation control- ow requirements that are then translated into a formal internal notation that is hidden to the developer and allows for the automation of the composition task (see Section 4).

3.2 Data Flow Requirements

The termination requirement presented in the previous section is only a partial speci cation of the constraints that the composition should satisfy. Indeed, we need to specify also requirements on the data ow. In order to provide consistent information, the eBS service needs to exchange data with the components and its customer in an appropriate way (e.g. every time the customer sends a search request, the eBS must use the book search information to prepare the itemSearchRequest message for the ABS service).

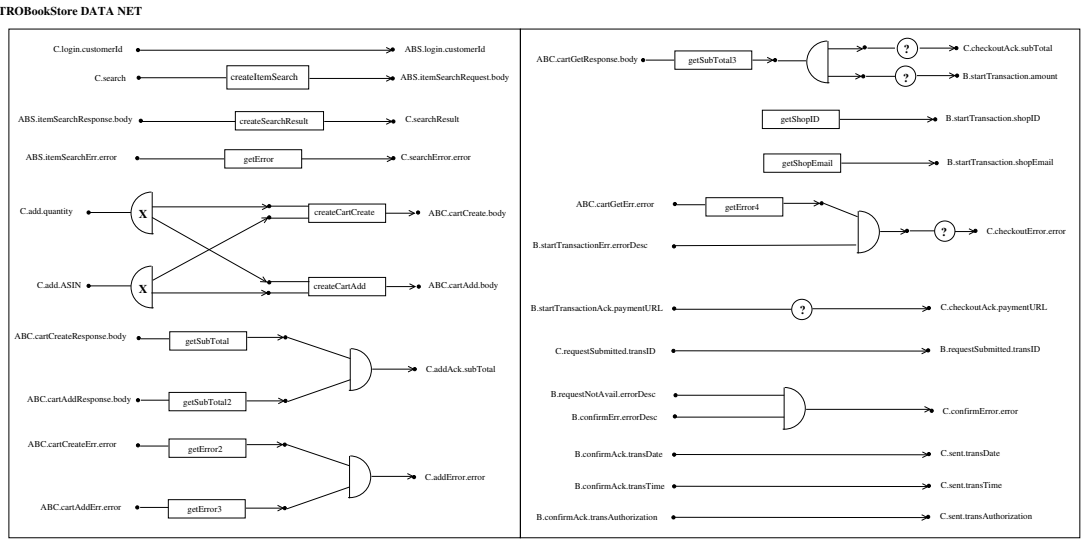


Fig. 4. The data flow requirements for the e-Bookstore composition problem

The aim of the data ow modeling language presented in [9] is to allow the speci cation of complex requirements concerning data manipulation and exchange by means of an intuitive and easy-to-de ne graphical notation. In particular, data ow requirements specify explicitly how output messages

(messages sent to component services) must be obtained from input messages (messages received from component services). All these requirements are collected in a diagram called *data net* (see Figure 4), whose nodes are sources of input messages, consumers of output messages, or internal variables for temporary storage of data, and whose arcs represent ow or manipulation of data. Data nets are able to represent a variety of constraints on the ow of data, including whether an input message can be used several times or just once, how several input messages must be combined to obtain an output message, whether all messages received must be processed and sent, and so on. In Table 1 we brie y describe the basic elements of a data net. Refer to [9] for a complete and formal description of the language.

The speci cation of the data net for the e-Bookstore example is presented in Figure 4, which we will (partially) explain in the following example.

Example 5 (e-Bookstore data- ow requirements). When the `eBS` receives a `login` request from the customer, it must forward the `customerId` information to the `ABS` service. To obtain the `body` to be sent in the `itemSearchRequest` to the `ABS`, the `eBS` must apply its internal function `createItemSearch` to manipulate the data received in the `search` message from the customer. The `eBS` must obtain the `error` information that it sends in the `searchError` message to the customer by computing its internal function `getError` on the `body` of the `itemSearchError` message received from the `ABS`. The `quantity` and `ASIN` information received in the `add` message from the customer can either be used to prepare the `cartCreate` message (through the `createCartCreate` operation) or to prepare the `cartAdd` message (through the `createCartAdd` operation) to be sent to the `AVC`. The `eBS` exploits the internal functions `getShopId` to obtain the `shopId` information in the `startTransaction` message to be sent to the `VPOS` (and similarly for the `shopEmail` data). And so on.

Each operation arc in the datanet in Figure 4 refers to an internal function that the new composite service uses to manipulate data. Since our aim is to automatically generate the executable WS-BPEL code implementing the composite service, we require that the speci cation of each internal function is given as a user de ned XML Path Language (XPath) expression, which is the standard language used in WS-BPEL assignments.

4 The Automated Composition Approach

In this section we discuss how, given the (WSDL and abstract WS-BPEL) description of the component services and the speci cation of the composition requirements, we can automatically obtain the executable WS-BPEL de nition of the composite service, namely of the `eBS` service.

The approach we apply is described in detail in [12, 9]. In this approach, the WS-BPEL processes de ning the component services are modeled as state transition systems (STS) [2]. Similarly, the data net is encoded as a STS constraining the operations that the composite service can perform to manipulate messages (refer to [9] for all the encoding details). Together, the STSs of the component services and that of the data net de ne the *composition domain*, i.e., they encode the constraints on the behavior of the composite service imposed by component processes and by the additional constraints on the management of data represented in the data net. The formal speci cation of the composition control- ow requirements (as shown in 3.1) is used as *control goal*. The automated generation of the composite service consists in generating a *controller*, i.e., an STS that interacts with the services encoded in the composition domain by exchanging messages with them, guaranteeing the satisfaction of the control goal. The work in [11] shows how to generate the controller using the Planning as Model Checking approach, which is able to deal with large domains and with complex control ow requirements. This approach exploit powerful BDD-based techniques developed for Symbolic Model Checking to ef ciently explore domains of large size.

Once the STS codifying the controller has been generated, it is translated into executable WS-BPEL to obtain the new process which implements the required composition. The translation is conceptually simple; intuitively, input/output actions in the controller model an interaction of the composite service with one of the component services, while synchronizations with actions in the STS modeling the data net correspond to manipulations of data by means of XPath expressions and assignments.

The presented approach has been implemented within the ASTRO toolset (`http://astroproject.org`) that has been designed as an extension of ActiveBPEL Designer [7], a commercial software for designing and developing WS-BPEL processes which is based on the Eclipse platform. The ASTRO toolset supports all the phases of web service automated composition: from the speci cation of control- ow and data- ow requirements (by means of graphical tools

[2] The translation is restricted to all WS-BPEL basic and structured activities. We do not support yet fault, exception and compensation handlers.

for drawing data net diagrams and specifying control- ow requirements) to the deployment and execution of the new composite service on the Active BPEL engine.

The following table shows the results of the eBookstore automated composition problem [3].

	Time (sec.)		BPEL
	model construction	composition & emission	complex activities
E-BOOKSTORE	2.7	605.2	177

We distinguish between model construction time and composition time. The former is the time required to to translate the WS-BPEL component services into STS and to encode the composition goal. The latter is the time required to synthesize the controller and to emit the corresponding eBS executable WS-BPEL process.

We have asked one of our experienced programmers to develop manually the WS-BPEL program for the e-Bookstore case. The task of manually encoding and testing the composition required more or less 20 hours. While, assuming to have the abstract WS-BPEL speci cation of the component services, the speci cation of the composition requirements took no more than one hour. The complexity of the composition problem derives from several aspects. First of all this scenario requires a high degree of interleaving between components. Moreover, when developing the composite service, the developer must take into account both the control- ow requirements and the requirements on data manipulation and exchange, which in this case are really elaborate. Thus, another advantage of our approach is a clear separation between the data- ow and control- ow aspects. The complexity of the composition task can also be deduced from the size of the new composite WS-BPEL process, which in the e-Bookstore scenario consists of 177 activities.

Another important aspect is the quality of the generated WS-BPEL processes. To evaluate this aspect, we compared the automatically generated and the hand-written solutions. As a result, we discovered that the two solutions implement the same strategy and have a similar structure.

5 Conclusions and Future Works

The work presented in this paper is a rst step towards addressing the key problem of the practical applicability of automated composition techniques in real composition scenarios. Future work will include the possibility to automatically derive (part of) the data net speci cation starting from the semantic of the data used in the component services, and to detect failures or changes in the component services, check the realizability according to the composition requirements, and re-con gure the composite process.

References

1. Amazon Services. Amazon E-Commerce Service - Developer Guide. http://developer.amazonwebservices.com/.
2. Amazon Services. AWSECommerceService WSDL Specification. http://aws.amazon.com/.
3. T. Andrews, F. Curbera, H. Dolakia, J. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weeravarana. Bpel4ws (version 1.1), 2003.
4. D. Berardi, D. Calvanese, G. De Giacomo, and M. Mecella. Composition of Services with Nondeterministic Observable Behaviour. In *Proc. ICSOC'05*, 2005.
5. A. Brogi and R. Popescu. Towards Semi-automated Workflow-Based Aggregation of Web Services. In *Proc. ICSOC'05*, 2005.
6. E. Christensen, F. Curbera, G Meredith, and S. Weerawarana. Web Services Description Language (WSDL) 1.1, 2001. W3C.
7. ActiveBPEL Designer. The Active Endpoints BPEL Designer - http://www.active-endpoints.com.
8. R. Hull, M. Benedikt, V. Christophides, and J. Su. E-Services: A Look Behind the Curtain. In *Proc. PODS'03*, 2003.
9. A. Marconi, M. Pistore, and P. Traverso. Specifying Data-Flow Requirements for the Automated Composition of Web Services. In *Proc. SEFM'06*, 2006.
10. M. Pistore, A. Marconi, P. Traverso, and P. Bertoli. Automated Composition of Web Services by Planning at the Knowledge Level. In *Proc. IJCAI'05*, 2005.
11. M. Pistore, P. Traverso, and P. Bertoli. Automated Composition of Web Services by Planning in Asynchronous Domains. In *Proc. ICAPS'05*, 2005.
12. M. Pistore, P. Traverso, P. Bertoli, and A.Marconi. Automated Synthesis of Composite BPEL4WS Web Services. In *Proc. ICWS'05*, 2005.

[3] The composition times have been obtained on a Pentium Centrino 1.6 GHz with 512 Mb RAM of memory running Linux

A Survey of Service Oriented Development Methodologies

Ervin Ramollari[1], Dimitris Dranidis[1], and Anthony J. H. Simons[2]

[1] South East European Research Centre (SEERC)
17 Mitropoleos Str., 54624 Thessaloniki, Greece
{erramollari, ddranidis}@seerc.org
[2] Department of Computer Science, University of Sheffield
Regent Court, 211 Portobello Street,
Sheffield, S1 4DP, UK
a.simons@dcs.shef.ac.uk

Abstract. Service orientation is a new software engineering paradigm that introduces opportunities as well as challenges. Although existing processes and practices can be reused for service oriented development, novel techniques are required to address unique SOA requirements. Work in this area is quite active and only recently is producing some initial results. The aim of this paper is to present a state-of-the-art survey on current service oriented development approaches. The characteristics that distinguish between these approaches are discussed and a number of actual methodologies that have emerged or are still emerging are described and compared.

Key words: SOA, service oriented software engineering, methodologies, survey

1 Introduction

Service-Oriented Computing represents a paradigm shift in software engineering, where the key abstraction is that of *services*, utilized to support rapid and low-cost application development through service composition. While technology and standards, such as Web services, are important to achieve SOA, it has been widely recognized that they are not sufficient on their own. Just by applying a Web service layer on top of legacy applications or components does not guarantee true SOA properties, such as business alignment, flexibility, loose coupling, and reusability. Instead, a systematic and comprehensive approach is of critical importance, taking into account the business requirements and following recommended practices. As Gartner [1] had predicted, "by 2007, 70 percent of SOA and Web services engagements will require a cohesive, end-to-end service delivery methodology and tool set".

A number of preliminary methodologies have emerged to address the huge demand for process guidance and proven best practices in SOA projects. However, a survey on these methodologies and an analysis of their properties is currently

lacking. Related work mainly treats service-oriented methodologies from a general point of view without referring to specific proposed ones. Arsanjani from IBM [2] broadly classifies SOA approaches under six categories: *business process driven*, *tool-based MDA*, *wrap legacy*, *componentize legacy*, *data-driven*, and *message-driven*. Papazoglou et al [3] provide a research roadmap, where among other things, they briefly explore the state of the art and some grand challenges in service oriented engineering. Zimmermann et al [4] discuss about analysis and design techniques for service-oriented development and integration, with IBM SOMA method as an example.

This paper goes into more depth by surveying actual approaches and methodologies. The characteristics and criteria that are used for comparison are discussed first, and then the actual methodologies are presented and compared using these characteristics.

2 Characteristics of SOA Development Methodologies

Below we present the criteria that we use to evaluate and compare SOA development approaches:

Delivery strategy: There exist three common strategies in delivering a SOA, depending on the amount of front-end analysis of the business domain and the treatment of existing legacy systems [12]. The *top-down* strategy is closely tied to an organizations' existing business logic, from which required services are derived. The *bottom-up* strategy is the opposite in that it focuses on legacy systems, and Web services are built on an as-needed basis. The *meet-in-the-middle* (agile) strategy finds a balance between incorporating service-oriented design principles into business analysis environments without having to wait before integrating Web services technologies into technical environments [12].

Lifecycle coverage: Some proposed approaches aim to support the full SOA lifecycle, including planning, analysis and design, construction, testing, deployment, and governance activities, while others limit their scope to a subset of these phases, such as analysis and design.

Degree of prescription: SOA methodologies range from the most prescriptive ones that specify phases, disciplines, tasks, and deliverables for each of them, while others provide less detail, by purpose or not, leaving room for more flexibility and tailoring of the approach depending on the project context.

Availability: A number of methodologies proposed by industry players such as IBM, Sun, Microsoft, and others, are proprietary and the detailed specifications are not openly available. In contrast to open methodologies whose documentation is available to the interested public, for the proprietary methodologies it is difficult to fully analyze their capabilities and to make comparisons.

Process agility: A number of methodologies suggest an agile approach to Service Oriented development in order to address risks and add flexibility to change. Yet, some others follow a more rigid approach in the process lifecycle, or do not address the issue of agility at all.

Adoption of existing processes/techniques/notation: A large number of SOA methodologies propose reusing proven existing processes like XP and RUP, and techniques like OOAD, CBD, and BPM, seeing service-oriented development as an evolutionary rather than revolutionary step in software engineering. Also standardized notations, such as UML and BPMN, are being adopted to visually model various artefacts.

Industrial application: It is important that a methodology be validated in proof-of-concept case studies to show that it has practical applicability and to refine it based on feedback from the case studies. Unfortunately, most of the existing SOA methodologies are at an early stage and have not been applied yet in industrial projects.

Supported role(s): A service-oriented methodology may support the provider view, the consumer view, or both the provider and consumer views in an integrated framework. In the consumer's view, development is declarative and business process oriented through service composition, while in the provider's view it is programmatic and component oriented.

3 Analysis of Existing Methodologies

IBM Service-Oriented Analysis and Design (SOAD) [5]: SOAD proposes elements that should be part of a service-oriented analysis and design methodology, hence it is an abstract famework rather than a holistic methodology. SOAD builds upon existing, proven techniques, such as OOAD, CBD, and BPM. It also introduces SOA-specific techniques, such as service conceptualisation, service categorization and aggregation, policies and aspects, meet-in-the-middle process, semantic brokering, and service harvesting.

IBM Service Oriented Modeling and Architecture (SOMA) [6] SOMA is a full-blown modeling methodology by IBM consisting of three steps: *identification*, *specification*, and *realization* of services, flows (business processes), and components realizing services. The process is highly iterative and incremental. However, because SOMA is proprietary to IBM, its full specification is not available. It has been recently announced that the Rational Unified Process has been combined with SOMA to result in what is called *IBM RUP for SOMA* [15].

SOA Repeatable Quality (RQ) [7]: SOA RQ is a proprietary methodology by Sun Microsystems that is based on a RUP-like iterative and incremental process consisting of five phases: inception, elaboration, construction, transition, and conception. UML compliant artefacts are used for documenting various deliverables of these phases.

CBDI-SAE Process [8]: The CBDI Forum is currently developing a SOA methodology as part of its CBDI-SAE SOA Reference Framework (RF). The four key discipline areas of the process are: consume, provide, manage, and enable. Each area groups similar disciplines that are further broken down to process units and then to tasks. This methodology aims business-IT integration through top-down analysis of business requirements as well as bottom-up legacy sys-

tem integration. The CBDI-SAE process aims to cover the whole SOA lifecycle, including deployment, monitoring, and governance activities.

Service Oriented Architecture Framework (SOAF) [9]: SOAF consists of five main phases: information elicitation, service identification, service definition, service realization, and roadmap and planning. It is concurrently based on two types of modeling activities: "To-be" modeling, which is the top-down business oriented approach describing the required business processes, and "As-is" modeling, which is the bottom-up approach describing current business processes as they are shaped by the existing applications.

Service Oriented Unified Process (SOUP) [10]: As the name suggests, this approach by K. Mittal is primarily based on the Rational Unified Process. Its lifecycle consists of six phases: incept, define, design, construct, deploy, and support. However, SOUP lacks detailed documentation and leaves room for adaptation. It is used in two slightly different variations: one adopting RUP for initial SOA projects and the other adopting a mix of RUP and XP for the maintenance of existing SOA rollouts.

Methodology by [11]: In their paper, Papazoglou et al examine a service development methodology from the point of view of both providers and consumers, which attempts to cover the full SOA lifecycle. It is partly based on well-established development models, such as the RUP, CBD, and BPM. The methodology utilizes an iterative and incremental process that comprises one preparatory and eight distinct main phases.

Thomas Erl's [12]: The service oriented analysis and design methodology documented in Thomas Erl's book [16] is considered the first vendor-agnostic one to be published . This methodology is a step by step guide through the two main phases: analysis and design. The activities in the analysis phase take a top-down business view where service candidates are identified. These serve as input for the next phase, service oriented design, where the service candidates are specified in detail and later realized as Web services.

BPMN to BPEL [13]: In this approach the business process is expressed in an abstract model (Business Process Modeling Notation or BPMN) and according to transformation rules it is automatically mapped to an execution language (Business Process Execution Language or BPEL) that can be executed by a process engine. The authors in [13] coined the term *business process oriented programming* to refer to an evolutionary step in software engineering where programming power is given to the business analyst.

Steve Jones' Service Architectures [14]: The scope of this top-down methodology consists of the first steps in a project necessary to ensure that true SOA properties are satisfied in the final delivery. It is technology agnostic and takes a top-down business view reaching up to the point of service candidate discovery (i.e. identification). The methodology adopts a broadly four-step process (What, Who, Why, and How), of which the first three are covered in preparation for the fourth step.

Comparison of the listed methodologies according to the identified characteristics is summarized in the table below.

	IBM SOAD	IBM SOMA	SOA RQ	CBDI-SAE	SOAF
Delivery strategy	M	M	M	M	M
Lifecycle coverage	A&D	A&D	complete	complete	A&D and planning next phases
Prescriptive	1	4	3	4	3
Proprietary	yes	yes	yes	no	no
Agile	n/a	3	4	2	2
Existing process	no	RUP (recently)	RUP	?	no
Existing techniques	OOAD, BPM	?	?	?	no
UML	yes	?	yes	?	?
Applied in industry	yes	extensively	extensively	not yet	a case study
Consumer view	yes	yes	yes	yes	yes
Provider view	yes	yes	yes	yes	yes

	SOUP	Papaz.	Erl's	BPMN to BPEL	Jones' SA
Delivery strategy	M	M	T	T	T
Lifecycle coverage	complete	complete	A&D	A& D and Impl.	Initial planning
Prescriptive	1	2	4	2	1
Proprietary	no	no	no	no	no
Agile	5	3	1	n/a	n/a
Existing process	RUP, XP	RUP	no	no	no
Existing techniques	no	CBD, BPM	BPM	BPM	no
UML	no	no	no	no	no
Applied in industry	not yet	not yet	not yet	not yet	not yet
Consumer view	yes	yes	yes	yes	yes
Provider view	yes	yes	yes	no	no

Table 1. Comparison of SOA development methodologies. (A relative quantitative scale 1-5 is used for some criteria. Also, M = Meet-in-the-Middle, T = Top-Down, B = Bottom-Up, and ? = No Data)

4 Conclusions

In this paper we presented a state of the art survey of the current service oriented engineering approaches and methodologies. One interesting point is that current SOA methodologies build upon existing, proven techniques, such as OOAD, EA, and BPM. Also, agile processes like XP and RUP are being employed successfully in SOA projects. However, the service paradigm introduces unique requirements that should be addressed by innovative techniques. Another interesting point is that most of the surveyed SOA methodologies propose the meet-in-the middle strategy, where both business requirements and existing legacy applications are taken into account to derive services. Although top-down analysis of the business domain produces services of high quality and long-term value, reality constraints require existing investment on IT infrastructure to be incorporated as well.

Generally, the service oriented development methodologies that have emerged are quite new and do not yet offer the required level of maturity. It is too early to determine whether any one of these methodologies is more appropriate than the others, or even to consider unifying some of them into a widely acceptable standard, as has been the case with the Rational Unified Process and the UML for object orientation. Therefore, we could say that this is a time of "methods war" for service oriented engineering that will eventually result in well-established and standardized methodologies.

References

1. M. Cantara, "Common features of external service providers' SOA frameworks and offerings", Gartner, September 2005.
2. A. Arsanjani, "Toward a pattern language for Service-Oriented Architecture and Integration, Part 1: Build a service eco-system", IBM Corporation, available from http://www-128.ibm.com/developerworks/webservices/library/ws-soa-soi/, July 2005.
3. M. P. Papazoglou et al, "Roadmap of Service Oriented Computing", March 2006, available from http://infolab.uvt.nl/pub/papazogloump-2006-96.pdf.
4. O. Zimmermann et al, "Analysis and design techniques for Service-Oriented Development and Integration", available from http://www.perspectivesonwebservices.de/download/INF05-ServiceModelingv11.pdf.
5. O. Zimmermann et al, "Elements of Service-Oriented Analysis and Design", IBM Corporation, available from http://www-128.ibm.com/developerworks/library/wssoad1/, June 2004.
6. A. Arsanjani, "Service-oriented modeling and architecture", IBM Corporation, available from http://www-128.ibm.com/developerworks/webservices/library/ws-soa-design1/, November 2004.
7. SUN Microsystems, "SOA RQ methodology - A pragmatic approach", available from http://www.sun.com/products/soa/soa_methodology.pdf.
8. P. Allen, "The service oriented process", in *CBDi Journal*, February 2007, http://www.cbdiforum.com/report_summary.php3?page=/secure/interact/2007-02/service_oriented_process.php&area=silver.
9. A. Erradi et al, "SOAF: An architectural framework for service definition and realization", in *Proceedings of the IEEE International Conference on Services Computing*, pp 151-158, Chicago, USA, September 2006.
10. K. Mittal, "Service Oriented Unified Process (SOUP)", available from http://www.kunalmittal.com/html/soup.shtml, 2006.
11. M. P. Papazoglou and W. J. van den Heuvel, "Service-oriented design and development methodology", *International Journal of Web Engineering and Technology (IJWET)*, 2006.
12. T. Erl, *Service-Oriented Architecture: Concepts, Technology, and Design*, Upper Saddle River: Prentice Hall PTR, 2005.
13. C. Emig et al, "Development of SOA-based software systems - and evolutionary programming approach", in *Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services*, p 182, Guadeloupe, French Caribbean, February 2006.
14. S. Jones, "A Methodology for Service Architectures", Capgemini UK plc, available from http://www.oasis-open.org/committees/download.php/15071/A%20methodology%20for%20Service%20Architectures%201%202%204%20-%20OASIS%20Contribution.pdf, August 2005.
15. Ali Arsanjani, "IBM's SOA method: SOMA, Service-Oriented Modeling and Architecture", IBM Corporation, available from http://www-03.ibm.com/developerworks/blogs/page/AliArsanjani?entry=soma_service_oriented_modeling_and&ca=drs-bl, December 2006.
16. "SOA-Glossary", Cambridge Technology Enterprises, available from http://www.ctepl.com/soaterms.shtml.

Verifying Business Process Compatibility

Peter Y. H. Wong and Jeremy Gibbons

Computing Laboratory, University of Oxford, United Kingdom
{peter.wong,jeremy.gibbons}@comlab.ox.ac.uk

Abstract. Business Process Modelling Notation (BPMN), developed by the Business Process Management Initiative (BPMI), intends to bridge the gap between business process design and implementation. In this paper we describe a process-algebraic approach to verifying process interactions for business collaboration described in BPMN. We first describe briefly a process semantics for BPMN in Communicating Sequential Processes; we then use a simple example of business collaboration to demonstrate how our semantic model may be used to verify compatibility between business participants in a collaboration.

1 Introduction

Modelling of business processes and workflows is an important area in software engineering. Business Process Modelling Notation (BPMN) allows developers to take a process-oriented approach to modelling of systems. There are currently around forty implementations of the notation, but the notation specification adopted by the Object Management Group (OMG) [9] does not have a formal behavioural semantics, which we believe is crucial in behavioural specification and verification activities.

In our previous work [12] we have given a process semantics to a subset of BPMN in the language of CSP [10]. In this paper we show how this semantics allows formal reasoning about business-to-business collaboration where there are multiple business processes under consideration. Consider, for instance, the simple example of an airline ticket reservation shown in Figure 1. The figure depicts the message flows between two participants, the traveller and the travel agent, which are independent business processes and may be assumed to have constructed separately during the development process. Clearly a necessary behavioural property for a successful collaboration is *compatibility* between the participants: the assumptions each makes about their interaction are mutually consistent. For example, from the traveller participant's perspective the behaviour of interest is the ability to cancel an itinerary by sending a message to the travel agent participant prior making her ticket reservation, while from the travel agent participant's perspective such sequence of tasks might not be allowed. By applying our semantic model, this property can be verified or disproved automatically.

In the remaining sections we describe briefly our semantic model and revisit the example to show how the semantics may be used to verify compatibility between collaboration participants and how such a property can also be specified

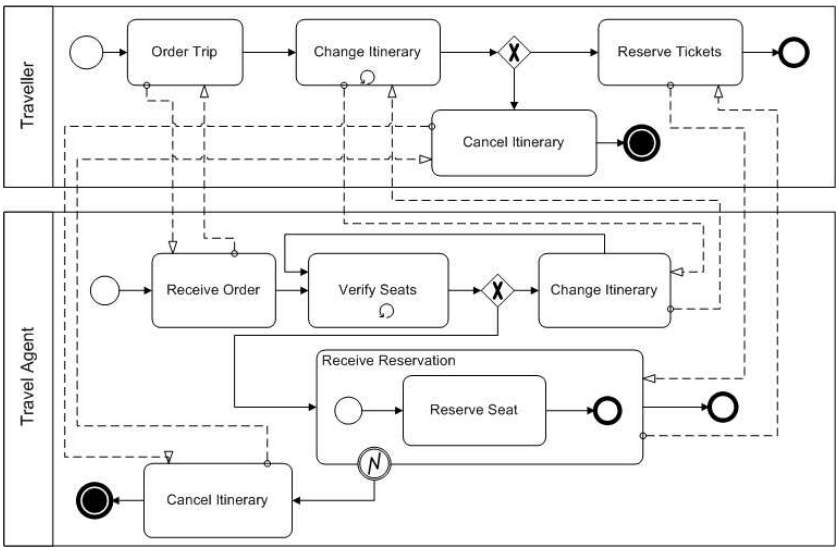


Fig. 1. A business collaboration of an airline ticket reservation.

in BPMN. We conclude this paper with a comparison with related work. A more detail description of the example can be found in our longer paper [13]. The formal definition of our semantic model may be found in our earlier papers [12].

2 Syntax and Semantics of BPMN

States in our subset of BPMN can either be pools, tasks, subprocesses, multiple instances or control gateways; they are linked by sequence, exception or message flows; sequence flows can be either incoming to or outgoing from a state and have associated guards; an exception flow from a state represents an occurrence of error within the state. Message flows represent unidirectional communication between states of different pools where each pool forms a container for some business processes.

We use Z [14] to define the BPMN's abstract syntax and semantic function, following Bolton and Davies' work on UML activity graphs [2]. In the remaining section we summarize our syntactic description and semantic model of BPMN, full details can be found in our earlier paper [12]. Our semantics permits automatic translation, requiring no user interaction. According to the specification [3], each BPMN state type has associated attributes describing its properties and our syntactic definition has included some of these attributes. For example, we define each type of state with the free type *Type*, a partial definition of which is given by :

$$Type ::= agate \mid start \mid end\langle\langle \mathbb{N} \rangle\rangle \mid task\langle\langle Task \rangle\rangle \mid miseq\langle\langle Task \times \mathbb{N} \rangle\rangle$$

Note the number of loops of a sequence multiple instance state type is recorded by the integer in the constructor function *miseq*. We describe each state by recording its content type and its sets of transitions and message flows.

Each BPMN diagram encapsulated by a *pool* represents an individual business participant in a collaboration, built up from a well-configured finite set of well-formed states [12]. While we associate each BPMN diagram with its diagram name, a business collaboration hence is built up from a finite set of names, each associate with its BPMN diagram and we associate each business collaboration with its collaboration name.

We define a semantic function which takes a syntactic description of a BPMN collaboration diagram, identified by its collaboration name and returns the CSP process that models the behaviour of that diagram. That is, the function takes one or more *pool* states, each encapsulating a separate BPMN diagram representing an individual participant within a business collaboration, identified by its collaboration and returns a parallel composition of processes each corresponding to an individual participant. We may apply our semantic function to transform the syntax of the diagram shown in Figure 1 mechanically into a parallel composition of processes, each corresponding to a business participant. We denote the processes corresponding to the traveller and the travel agent participants by the names *Tr* and *Ag* respectively. We define set I to index the processes corresponding to the states of the traveller participant.

$$I = \{\, start, order, change, xs, cancel, reserve, end, abort \,\}$$

We use channels $init.a$ to denote transitions to states of participant a and $starts.a$ to denote initiation of its tasks or subprocesses. We write $msg.t.x$ to denote communication of message x during task or subprocess t. We use compound events $fin.i$ and $abt.i$ where i ranges over $\mathbb{N}$ to denote successful termination and abortion of the business process. The process *Tr* mechanically obtained by the translation we have described above is as follows:

$$\begin{aligned} Tr = \mathsf{let}\ X &= \Box\, i : (\alpha Y \setminus \{fin.1, abt.1\}) \bullet (i \rightarrow X \,\Box\, fin.1 \rightarrow Skip \,\Box\, abt.1 \rightarrow Stop) \\ Y &= (\ \|\ i : I \bullet \alpha P(i) \circ P(i)) \\ \mathsf{within}\ &(Y \,[\![\, \alpha Y \,]\!]\, X) \setminus \{\!|init|\!\} \end{aligned}$$

where for each i in I, the process $P(i)$ is as defined below. We use N, ranging over $\mathbb{N}$, to denote the number of instances of the task *change*, as specified by the second argument to constructor function *miseq*. We write αQ to denote the set of possible events performed by process Q.

$$\begin{aligned} P(start) &= init.tr.order \rightarrow Skip \fatsemi fin.1 \rightarrow Skip \\ P(order) &= (init.tr.order \rightarrow Skip \fatsemi \\ &\quad starts.tr.order \rightarrow Skip \fatsemi msg.order!x : \{\, in, last \,\} \rightarrow Skip \fatsemi \\ &\quad msg.order.out \rightarrow Skip \fatsemi \\ &\quad init.tr.mchange \rightarrow Skip \fatsemi P(order)) \,\Box\, fin.1 \rightarrow Skip \\ P(xs) &= (init.tr.xs \rightarrow Skip \fatsemi (init.tr.cancel \rightarrow Skip \,\Box\, init.tr.reserve \rightarrow Skip) \fatsemi \end{aligned}$$

$$
\begin{aligned}
& P(xs.3)) \,\Box\, fin.1 \rightarrow Skip \\
P(cancel) = {} & (init.tr.cancel \rightarrow Skip \,\fatsemi \\
& starts.tr.cancel \rightarrow msg.cancel!x : \{\, in, last \,\} \rightarrow Skip \,\fatsemi \\
& msg.cancel.out \rightarrow Skip \,\fatsemi\, init.tr.abort \rightarrow Skip \,\fatsemi \\
& P(cancelit)) \,\Box\, fin.1 \rightarrow Skip \\
P(abort) = {} & (init.tr.abort \rightarrow Skip \,\fatsemi\, abt.tr.1 \rightarrow Stop) \,\Box\, fin.1 \rightarrow Skip \\
P(reserve) = {} & (init.tr.reserve \rightarrow Skip \,\fatsemi\, starts.tr.reserve \rightarrow Skip \,\fatsemi \\
& msg.reserve!x : \{\, in, last \,\} \rightarrow Skip \,\fatsemi\, msg.reserve.out \rightarrow Skip \,\fatsemi \\
& init.tr.end \rightarrow Skip \,\fatsemi\, P(reserve)) \,\Box\, fin.1 \rightarrow Skip \\
P(end) = {} & init.tr.end \rightarrow Skip \,\fatsemi\, fin.1 \rightarrow Skip
\end{aligned}
$$

$$
\begin{aligned}
& P(change) = \\
& \mathsf{let}\ \ A(i) = i > 0\ \&\ \\
& \qquad (init.tr.change \rightarrow Skip \,\fatsemi \\
& \qquad starts.tr.change \rightarrow Skip \,\fatsemi\, msg.change!x : \{\, in, last \,\} \rightarrow Skip \,\fatsemi \\
& \qquad msg.change.out \rightarrow Skip \,\fatsemi \\
& \qquad init.tr.xs1 \rightarrow Skip \,\fatsemi\, A(i-1)) \,\Box\, init.tr.xs \rightarrow Skip \\
& \quad X(n) = (init.tr.mchange \rightarrow Skip \,\Box\, init.tr.xs1 \rightarrow Skip) \,\fatsemi \\
& \qquad (n > 0\ \&\ (init.tr.change \rightarrow (msg.change.in \rightarrow X(n-1) \\
& \qquad\qquad \Box\ msg.change.last \rightarrow init.tr.xs1 \rightarrow init.tr.xs \rightarrow Skip)) \\
& \qquad \Box\ n = 1\ \&\ (init.tr.change \rightarrow msg.change.last \rightarrow \\
& \qquad\qquad init.tr.xs1 \rightarrow init.tr.xs \rightarrow Skip) \\
& \qquad \Box\ n = N\ \&\ msg.change.end \rightarrow init.tr.xs \rightarrow Skip) \\
& \mathsf{within} \\
& ((A(N) \,[\![\, \{\, msg.change.in, msg.change.last, init.tr.xs \,\} \,]\!]\, X(N)) \,\fatsemi \\
& \ P(change)) \,\Box\, fin.tr.1 \rightarrow Skip
\end{aligned}
$$

The process Ag can be defined similarly using the semantic function. Their collaboration hence is the parallel composition of processes Tr and Ag.

$$Collab = (Tr \,[\![\, \alpha Tr \,||\, \alpha Ag \,]\!]\, Ag) \setminus \{\!|msg|\!\}$$

3 Verifying Compatibility

The CSP behaviour models traces ($\mathcal{T}$), stable failures ($\mathcal{F}$) and failures-divergences ($\mathcal{N}$) admit refinement orderings based upon reverse containment [10]. This means that a CSP process can be a specification as well as a model of an implementation; hence it is possible to design and compare specifications using BPMN. To check if the traveller participant is compatible with the travel agent participant we use the stable failures refinement to compare *Spec* with *Collab* where *Spec* is the CSP process corresponding to the traveller participant without message flows, which can be derived mechanically:

$$Spec \sqsubseteq_{\mathcal{F}} (Collab \setminus (\alpha Collab \setminus \alpha Spec)) \tag{1}$$

We have chosen $\mathcal{F}$ to reason about our semantics of BPMN since $\mathcal{T}$ allows the deadlock process *Stop* as a refinement all specifications, which is inadequate, while divergences can be avoided via syntactic restriction. This refinement checks whether the collaboration behaves as specified by the traveller participant; in order for this to happen the travel agent needs to be *compatible* with the traveller participant. We have specifically defined our semantics to allow refinement assertions such as this one to be automatically checked by a model checker like FDR [5]. In this particular example we find that refinement assertion (1) does not hold; this means the participants in the collaboration described in Figure 1 are not compatible. When we ask FDR to verify assertion (1), the following counterexample in the form of a failure [10] is given, where Σ denotes the set of all event names.

$$(\langle starts.tr.order, starts.tr.cancel \rangle, \Sigma)$$

This counterexample tells us that a deadlock has occurred while the traveller is cancelling her itinerary. There are two ways to correct this collaboration, either by changing the traveller's or the travel agent's internal process description. We have chosen the latter; a compatible collaboration with an abstracted view of the traveller and a modified travel agent participant is shown in Figure 2. Note the change in the travel agent participant allowing the task state *Cancel Itinerary* to be triggered before the subprocess state *Receive Reservation*.

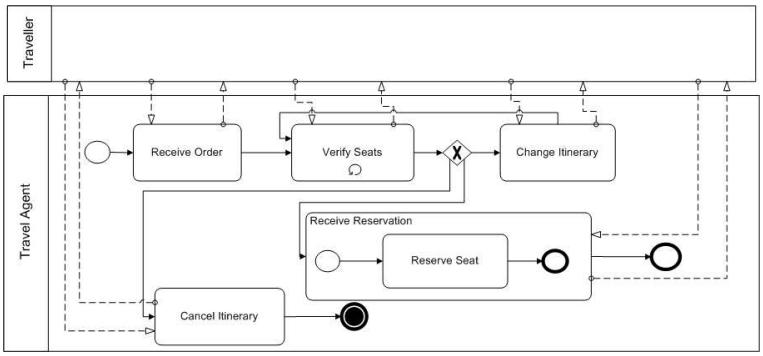


Fig. 2. A BPMN diagram describing a compatible business collaboration of an airline ticket reservation.

By applying our semantic function to the syntax of this diagram we obtain the following parallel composition of processes, each corresponding to a participant.

$$Collab2 = (Tr \,[\![\, \alpha Tr \,||\, \alpha Ag2 \,]\!]\, Ag2) \setminus Msg$$

To check for compatibility we ask FDR to verify the refinement assertion (2). This time, the verification is successful.

$$Spec \sqsubseteq_{\mathcal{F}} (Collab2 \setminus (\alpha_{Collab2} \setminus \alpha_{Spec})) \tag{2}$$

4 Related Work

To the best of our knowledge, no work has been done towards the compatibility verification of business collaborations described in a graphical modelling notation like BPMN; other approaches have mainly focused on the compatibility problem of web services choreographies described by XML-based languages such as WS-BPEL [1, 6], WSCI [11, 4] and WSCDL [8, 7]. While they focus on compatibility at the implementation level, we have moved it forward to the design level, allowing collaboration to be verified and agreed upon before implementation. While our earlier work [12] described the application of a process semantics and refinement in verifying consistency between BPMN diagrams each with a different level of abstraction, our modelling approach in compatibility verification described in this paper utilizes the extended semantics described in the extended version of our earlier paper to verify collaboration compatibility between BPMN diagrams participating in a business collaboration.

References

1. 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*, May 2003.
2. C. Bolton and J. Davies. Activity Graphs and Processes. In *IFM '00: Proceedings of the Second International Conference on Integrated Formal Methods*, volume 1945 of *LNCS*, pages 77–96, 2000.
3. *Business Process Modeling Notation Specification*, Feb. 2006. `www.bpmn.org`.
4. A. Brogi, C. Canal, E. Pimentel, and A. Vallecillo. Formalizing Web Services Choreographies. In *Electronic Notes in Theoretical Computer Science 105*, pages 73–94, 2004.
5. Formal Systems (Europe) Ltd. *FDR2 User Manual*, 1998. `www.fsel.com`.
6. H. Foster, S. Uchitel, J. Magee, and J. Kramer. Compatibility verification for web service choreography. In *IEEE International Conference on Web Services*, 2004.
7. H. Foster, S. Uchitel, J. Magee, and J. Kramer. Model-Based Analysis of Obligations in Web Service Choreography. In *IEEE International Conference on Internet and Web Applications and Services*, 2006.
8. N. Kavantzas, D. Burdett, G. Ritzinger, T. Fletcher, and Y. Lafon. *Web Services Choreography Description Language 1.0*, 2005. W3C Candidate Recommendation.
9. Object Management Group. `http://www.omg.org`.
10. A. W. Roscoe. *The Theory and Practice of Concurrency*. Prentice Hall, 1998.
11. W3C. *Web Service Choreography Interface 1.0*, Nov. 2002. `www.w3.org/TR/wsci`.
12. P. Y. H. Wong and J. Gibbons. A Process Semantics for BPMN, 2007. Submitted for publication. Extended version available at `http://web.comlab.ox.ac.uk/oucl/work/peter.wong/pub/bpmn-extended.pdf`.
13. P. Y. H. Wong and J. Gibbons. Verifying Business Process Compatibility, 2007. Submitted for publication.
14. J. C. P. Woodcock and J. Davies. *Using Z: Specification, Proof and Refinement*. Prentice Hall International Series in Computer Science, 1996.

A modified Logic Scoring Preference method for dynamic Web services evaluation and selection

Hong Qing Yu[1] and Hernán Molina[2]
[1] Department of Computer Science, University of Leicester, UK
hqy1@mcs.le.ac.uk
[2] Department of Informatics, School of Engineering, University of La Pampa, Argentina
hmolina@ing.unlpam.edu.ar

Abstract. The Logic Scoring Preference (LSP) method extends existing scoring techniques and provides a means for the development of complex criterion functions using continuous preference logic. It has been successfully used to evaluate hardware system, software system and websites. However, the current LSP method is not sufficient to help evaluating and selecting web services in the context of dynamic service composition. This area depends not only on static service functional and non-functional requirements (which are traditionally considered in LSP applications), but also on run-time context such as business policies and pre/succeeding service instances in workflow patterns. Thus, we propose a modified LSP method to target the problem of dynamic web service evaluation and selection.

1 Introduction

Web services technology is widely used by industry today. With increasing number of web services available, there is in general more than one service to fulfill a given task. Consequently, the biggest challenge is to choose the most appropriate service which satisfies not only functional requirements but also non-functional, context related requirements, such as policies, the business' target market, QoS (Quality of Service) and existing system workflow patterns.

In this paper, we propose a modified Logic Scoring Preference (LSP) [1] method to achieve dynamic web service evaluation and selection. The remaining content is structured as follows. We conclude the introduction with a motivating example. The simple introduction of LSP method is described in section 2. Our modified LSP method is introduced in section 3. A worked evaluation example and related work comparing are shown in section 4. Section 5 concludes the paper and highlights further work.

1.1 Motivating Example

Considering a scenario (shown in Fig. 1), where a business organization needs a payment service to complete an online product selling business process, some context aspects should be considered. Firstly, the market aspect, the target customers might be at home or traveling. In the case that they are traveling, the customers could make use

of several devices, such as a laptop, PDA, landline, desktop or mobile phone. Secondly, the workflow aspect requires that the protocols of the payment service are suitable for integrating to existing online selling service. Thirdly, the QoS aspects, such as security must be high; performance rate should be reasonable and privacy should be respected. Fourthly, the policies include that customers are supposed to understand one language out of English, Spanish and French. Moreover, the service provider must be located in Europe and a lower transaction fee is better. Finally, being able to accept more types of bank card such as Visa, MasterCard, Solo and Switch is preferable. There are four services available which can functionally fulfill the payment task shown in Fig. 1. Now the question is which one is the most suitable for use.

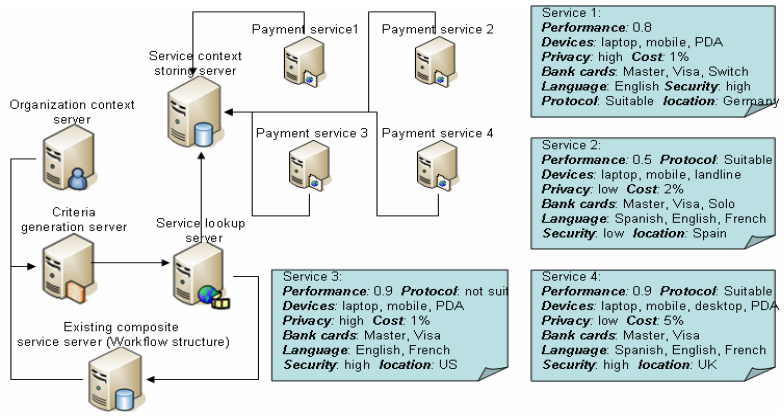


Fig. 1: Motivating Example

2 The LSP method

LSP is a quantitative method based on scoring techniques and a continuous preference logic [2]. Basically, the method allows establishing an evaluation criterion by specifying the expected properties of a system. To each one of these properties a criterion function is assigned. These functions transform specific domain values to a normalized scale indicating the degree of satisfaction of the corresponding preference. Then, all preference values can be properly grouped using a stepwise aggregation structure to yield a global preference. This can be achieved by means of a preference aggregation function, called *generalized conjunction/disjunction* or *andor*, combining weighted power means to obtain the global preference e_0 as in:

$$e_0 = \left(W_1 e_1^r + \ldots + W_k e_k^r\right)^{1/r}, W_1 + \ldots + W_k = 1 \quad (1)$$

Where the power r can be suitably selected to obtain desired logical properties (see [3, 2] for further details). The LSP decision method has been applied in the evaluation and selection of hardware and software systems [3, 1], as well as in the evaluation and comparison of web applications [4]. The strong drawback of these applications is that they require the participation of human expert interactions, which is not suitable for working in a dynamic environment as we discussed before.

3 The modified LSP method (MLSPM)

3.1 Context model

The first step of our method is to model the dynamic context environment and gather useful information such as preference elements and their values from a well defined model (Context model). Defining a context model actually is an analysis process that needs to understand the purposes, scales, background information and the activity rules, etc. It is very difficult to build a unified model which includes all aspects and their attributes. In this work, we use a context model which is just an expression model for showing how our dynamic method works for our web service evaluation and selection purpose (see Fig. A.1 in appendix A).

In this paper, we are not going to illustrate the detailed techniques of context reasoning and mining addressed in the "inContext" project [5]. However, we assume the model is implemented by XML technology and a well defined mediator can be used to resolve mismatches between different terminologies (data level).

3.2 Specification of Type-based Unified evaluation methods

We can design different evaluation metrics based on the given context model types. For example device elements may be described as value="computer, PDA", type="string set overlap". In our current work, we have identified four different evaluation functions to capture the variety of types. The four functions are: "*exact match*" (equation 3), "*set overlap*" (equation 4), "*level match*" (equation 5) and "*specific value*" (equation 6).

Typical usage is linked to the data type of the context aspect: if the context aspect can be expressed by a Boolean, then exact match would be used; considering sets of information, set overlap is useful; level match is useful for ordered discrete values (such as low, medium and high); and finally specific value allows for complex functions that calculate a numerical value (e.g. for distance functions). Because of the link to the data type, it can be automatically determined which function should be used. In addition, the weight $\omega < 0$ expresses the lower value is desired or $\omega \geq 0$ means higher value is respect. Thus the global preferences evaluation function is changed from equation (1) to (2).

$$e_0 = (|\omega_1|E_1^r + |\omega_2|E_2^r + \ldots + |\omega_n|E_n^r)^{1/r} \text{ with } 0 \leq E \leq 1, \sum_{i=1}^{n} |\omega_i| = 1 \tag{2}$$

The respective formulas to compute values for these functions (E1 to En) are as follows:

$$E = \begin{cases} 1 & \textit{if criterion is met} \\ 0 & \textit{otherwise} \end{cases} \tag{3}$$

$$E = (e_1 + e_2 + \ldots + e_n)/n \text{ with } e_i \text{ being a score for each element of the set} \tag{4}$$

$$E = i_c / i \text{ where } i \text{ is the number of levels and } i_c \text{ is the current level match} \tag{5}$$

$$E = \begin{cases} 1 - \left(\dfrac{v_{\max} - v}{v_{\max} - v_{\min}} \right) & iff\ \omega \geq 0 \\ \left(\dfrac{v_{\max} - v}{v_{\max} - v_{\min}} \right) & otherwise \end{cases} \quad (6)$$

$v_{\min}$ being the minimum value for all services, $v_{\max}$ the maximum value and v the value for the current service in (6). For the motivating example, we match evaluation functions to each preference as in Table A.1 (see Appendix A).

3.3 Global aggregation

We need a global preference aggregation function $L(g_1(a_1),...,g_n(a_n))$ to calculate all aspects of preferences. The function itself must reflect specific requirements and logic conditions, such as simultaneity, replaceability and others [2]. The function g is one of the individual evaluation methods which were discussed in the previous section. We use the conjunctive partial absorption function as global aggregation structure (see Fig. 2) [3]. We separate preferences into a critical group ($EP^C{}_i$) and a desired group ($EP^D{}_j$). The critical group presents all mandatory requirements; the desired group takes all the other preferences.

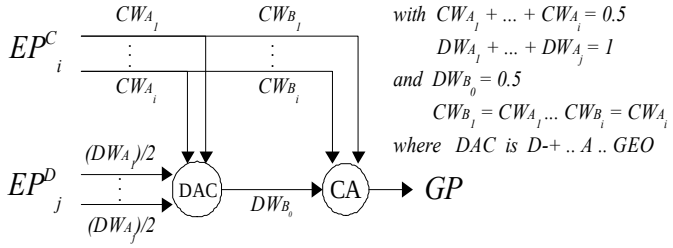


Fig. 2: The structure of the *conjunctive partial absorption* aggregation function

Behaviors of the conjunctive partial absorption function are such that the global preference value (denoted by *GP*) will be 0[1] when any of the critical preferences are not satisfied, in which case the service will be discarded. On the other hand, a web service that satisfies all critical preferences will be valuated to a non-zero value, from which the degree of satisfaction of the desired preferences can raise or reduce the final global preference. The first function (denoted by a circle named DAC) is dynamically selected by an automated calculate method (ACM) to reflect dynamic preferences. The second function (denoted by a circle named CA) is always there to achieve the desired absorption behaviors.

We design an automated calculate method (ACM) to find a logic correctness GCD (Generalized Conjunction/disjunction) function based on Continuous Logic [6] for the desired preferences without considering critical preferences. Thus, Fig. 2 shows that

[1] In $f(x) = x^r$, When $r < 0$ and $x = 0$, then an error occurs, we use 0 to express it.

all weights of desired preference sum up to 1. The logic meaning can be reflected by meaningful weights. We consider the value of the weight ω_i belongs to a set $_{A,\, A \in (0,1)}$. Then we have an ordered set W = $(\omega_{1,\ldots}\, \omega_n)$, and $\omega_1 \geq \ldots \geq \omega_n$. Based on the meaning of or-ness in OWA decision making method [7], we can get the following function:

$$\lambda_{orness} = \frac{1}{n-1}\sum_{i=1}^{n-1}(n-i)|\omega_i|, \omega_i \text{ is the } i_{th} \text{ place in set V} \qquad (7)$$

Where V is reordered set of W, and the reorder algorithm firstly is to find the same weights $\omega_{1+1,\ldots,}\omega_{1+i}$ value to ω_1 . Put $\omega_{1+1,\ldots,}\omega_{1+i}$ to the tail of the set. Secondly, taking $\omega_{2+1}\ldots\omega_{2+i}$ which have the same value of ω_2 in the new set in front of the $\omega_{n-i}\ldots\ \omega_n$. The rest of the refreshed set repeats second step until the last element that has not been reordered before. For example, if W = {0.2, 0.2, 0.15, 0.15, 0.1, 0.1, 0.1}, then, V = {0.2, 0.15, 0.1, 0.15, 0.1, 0.1, 0.2} the value λ presents the degree of the "or-ness" by equation (7) calculated. The mapping table between the value λ and the GCD operators is given in Table A.2 of Appendix A.

4 Worked example and related work

Applying our method to the motivating example, we can consider that the devices preference (0.25), performance rate (0.5), privacy (0.01), cost (-0.04) and bank cards (0.2) requirements are desired preferences. Communication protocol (0.1), security (0.2), location (0.1) and language (0.1) are critical preferences. First we calculate the "or-ness" position based on equation (7).

$$\lambda = \frac{1}{4}(0.5\times 4 + 0.25\times 3 + 0.20\times 2 + 0.04) = 0.7985$$

Comparing to the appendix A, the result shows D-+ is a suitable GCD function to be selected (power r = 3). According to equation function (2), we got following values for the four competitive services:

e_1 = 0.201 (S1), e_2 = 0.126 (S2), e_3 = 0.181 (S3) and e_4 = 0.276 (S4)

The next stage is to use the conjunctive partial absorption aggregation function to determine which service is best to use. We take all critical evaluation values and output values from last step of each service into the CA operator (power r = -0.72), and then we get the final evaluation values by using function (2) again: e_1 = 0.333 , e_2 = 0 , e_3 = 0 and e_4 = 0.493 the returned ranked list is {S4,S1}.

Comparing to related work [8] and [9], our method has 5 outstanding advantages. (1) MLSPM combines evaluation and selection activities together contrast to [8] and [9] which only address selection issues. (2) MLSPM can deal with all types of preferences such as *String* and *Boolean*. However, the other works only focused on numbering type. (3) Our method is more dynamic and easier. (4) The context model we defined covers more evaluation aspects, such as policy rules, workflow requirements beyond QoS. (5) The key feature of LSP method is considering the logic relations between preferences rather than just simply using weight mechanism.

5 Conclusion and future work

In this paper we outlined a modified LSP method that allows us to dynamically evaluate and select the most suitable web services considering large number of available services, changing contexts and run-time service requests in a timely and efficient manner. In addition, we gave an example scenario in which our LSP method was well applied.

Future work will cover the definition of the meanings of weights used in this paper from the perspective of user preferences, context mining and reasoning techniques since their outcomes will be the inputs for web service evaluation and selection. Additionally, implementation issues of the modified LSP method, as well as related mechanisms will be addressed.

6 Acknowledgment

We thank Dr. Stephan Reiff-Marganiec and Dr. Luis Olsina for their comments and suggestions. This work is also supported by inContext **(**Interaction and Context Based Technologies for Collaborative Teams) project: *IST IST-2006-034718* and WEE-NET (Web Engineering Network of Excellence) project: *ALFA II-0359-FA.*

References:

1. S. Y. W. Su, J. Dujmovic, D. S. Batory, S. B. Navathe, R. Elnicki. *A Cost-Benefit Decision Model: Analysis, Comparison, and Selection of Data Management Systems*. ACM Transactions on Database Systems, Vol. 12, No. 3, September 1987, pp. 472-520.
2. Dujmovic, J.J., *Continuous Preference Logic for System Evaluation*. In Proceedings of Eurofuse 2005, edited by B. De Baets, J. Fodor, and D. Radojevic, ISBN 86-7172-022-5, Institute "MihajloPupin", Belgrade, 2005, pp. 56-80.
3. Dujmovic, J.J., *A Method for Evaluation and Selection of Complex Hardware and Software Systems.* The 22nd International Conference for the Resource Management and Performance Evaluation of Enterprise Computing Systems. CMG 96 Proceedings, Vol. 1, 1996, pp. 368-378.
4. Olsina L., Rossi G.: *Measuring Web Application Quality with WebQEM*. IEEE Multimedia, Vol. 9(4), 2002, pp. 20-29.
5. "inContext project", http://www.in-context.eu/.
6. Levin V. I., *Generalizations of the Continuous Logic*. Automation and Remote Control, Vol. 62, No. 10, 2001, pp. 1743-1755.
7. Robert F., *OWA operators in Decision Making*. In C. Carlsson ed., Exploring the limits of Support Systems, TUCS General Publications No. 3, Turku Centre for Computer Science, 1996, pp. 85-104.
8. Liu Y., Ngu A.H.H. and Zeng L., *QoS Computation and Policing in Dynamic Web Service Selection*, WWW2004, May 17-22, 2004, ACM 1-58113-912-8/04/05.
9. Wang X., Vitvar T., Kerrigan M. and Toma I. *A QoS-aware Selection Model for Semantic Web Services*, Digital Enterprise Research Institute, ICSOC 2006.

Appendix A

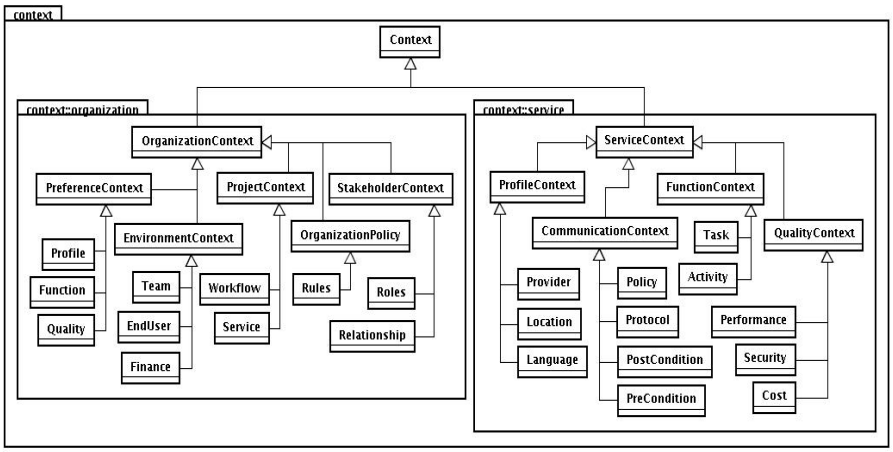


Fig. A.1 context model

Desired preferences	Evaluation methods	Critical preferences	Evaluation methods
Performance rate	(6)	Protocol	(3)
Devices	(4)	Security	(3)
Privacy	(5)	Location	(3)
Cost	(6)	Language	(4)
Bank cards	(4)		

Table A.1: Assigned evaluation methods to example

Value of y	GCD Operator symbols	Operation
$y < 0.3333$	GEO	Geometric mean
$0.3333 \le y < 0.3750$	C-	Weak QC
$0.3750 \le y < 0.4375$	C--	Weak QC (-)
$0.4375 \le y < 0.5000$	A	Arithmetic mean
$0.5000 \le y < 0.5625$	D--	Weak QD (-)
$0.5625 \le y < 0.6232$	SQU	Square mean
$0.6232 \le y < 0.6250$	D-	Weak QD
$0.6250 \le y$	D-+	Weak QD (+)

Table A.2: The range of selection GCD operators [3]

Modelling and Analysing an Identity Federation Protocol: Federated Network Providers Scenario*

Maurice H. ter Beek[1], Corrado Moiso[2], and Marinella Petrocchi[3]

[1] ISTI–CNR, Via G. Moruzzi 1, 56124 Pisa, Italy
[2] Telecom Italia, Via Reiss Romoli 274, 10148 Torino, Italy
[3] IIT–CNR, Via G. Moruzzi 1, 56124 Pisa, Italy

Abstract. We continue our work on modelling and analysing security issues of an identity federation protocol for convergent networks. This protocol was proposed by Telecom Italia as a solution to allow end users access to services on the web through different access networks, without explicitly providing any credentials, while the service providers can trust the user's identity information provided by the access networks and access some user data. As an intermediate step towards a full-blown formal security analysis of this protocol, we specify one specific user scenario in the process algebra Crypto-CCS and verify its vulnerability w.r.t. a man-in-the-middle attack with the model checker PaMoChSA.

1 Introduction

We continue work initiated in [1] by formally specifying one specific user scenario of the identity federation protocol for convergent networks introduced in [2] and by analysing the possibility of a man-in-the-middle attack. We use Crypto-CCS, a CCS-like process algebra with cryptographic primitives [3, 4], in combination with the Partial Model Checking Security Analyser PaMoChSA [5] developed by the Security group of IIT–CNR.

The protocol proposed in [2] permits end users to access services through different access networks (*e.g.* mobile and fixed ones), without explicitly providing any credentials, while the application level can trust the identity and authentication information provided by the access networks. As a result, service providers (SPs) identify a user by using the authentication procedure performed by the network provider. After identity federation, a single sign-on suffices for a user to access all services belonging to the same "circle of trust" of SPs while keeping personal data private. This is an advantage over introducing (and remembering) one's credentials time and again. The protocol may thus grant users anonymous access to services and at the same time allow the application level to limit access to authorised users. Moreover, without knowing who the user is, SPs can still obtain her/his location or age or charge her/his account.

In Section 2 we sketch our scenario. We then describe our analysis approach in Section 3, after which we formally specify our user scenario in Section 4. A small security analysis is presented in Section 5. Section 6 contains our conclusions.

* This work has been partially funded by the EU project Sensoria (IST-2005-016004).

2 The Case of Federated Network Providers

The protocol of [2] uses a token injector mechanism to translate the identity and authentication information provided by a secure access network to the Internet application level (of a lower security level). The token injector thus plays the identity provider role described in the Liberty Alliance specifications [6].

We consider a user scenario with two network operators, a Mobile Operator MO and a Fixed Operator FO, both of which have implemented the token injector mechanism and have established a trusted relation with a SP, for instance a travel site. The user is a customer of both operators, has an active registration on the travel site and has already federated her/his account on the travel site with her/his profile on each of the operators (*i.e.* two federations have been activated: one between the SP and the MO accounts and one between the SP and the FO accounts). During the process of federation, the token injector generates an opaque-id (or pseudonym) for the user and sends it to the SP. This opaque-id is then stored on the repositories associated to the token injector and to the SP and it is exchanged in all communication between the operators and the SP, so as to identify the user in a secure way. Consequently, the user can access the travel site by mobile phone (GPRS) or by PC (ADSL) without introducing credentials. Instead, the network authentication is forwarded to the travel site, which identifies the user. Even though the SP does not know the user's mobile phone number, the opaque-id enables it to use the operators service (as SMS gateway) to nevertheless exchange or request information about the user. The architecture of this federated network providers scenario is sketched in Figure 1.

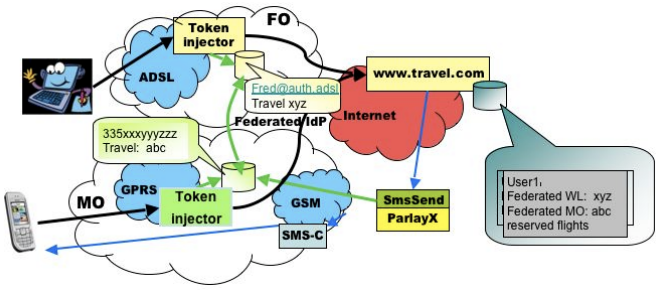


Fig. 1. Federated network providers scenario in convergent networks.

3 Modelling and Analysis Approach

We adopt the approach of [4] based on the observation that a protocol under analysis can be described as an open system in which some component has an unspecified behaviour (not fixed in advance). One then assumes that, regardless of this behaviour, the system works properly (*i.e.* satisfies a certain property). In our scenario, one can imagine a hostile adversary trying to achieve some kind of advantage w.r.t. the honest participants. Such an adversary is added to the specification of the protocol in the next section as a component with a behaviour defined only implicitly by the semantics of the specification language.

We assume the adversary to act in Dolev-Yao fashion [7] by using a set of message manipulating rules that model cryptographic functions. A message encrypted with the public key of one of the participants cannot be decrypted by anyone but the person who knows the corresponding private key. As is common, we adopt a black-box view of cryptography by assuming all cryptographic primitives involved in the network protocol to be perfect. To analyse whether a system works properly, at a certain point the adversary's knowledge is checked against a security property. If the intruder has come to know information it was not supposed to know, then the analysis has thus revealed an attack w.r.t. that property, *i.e.* a sequence of actions of the adversary that invalidates the property.

First, we fix some notation related to the security primitives we deal with. Sending message *msg* from sender A to receiver B over the ith communication channel c_i is denoted by $c_i\ A \mapsto B : msg$. We use the following security primitives:

$pk_i,\ pk_i^{-1}$	*public and private key of agent i*
$\{\_\}_{pk_i^{-1}}$	*message signed by agent i*
$\{\_\}_{pk_i}$	*message encrypted by public key of agent i*
$\{\_\}_{KEY}$	*message encrypted by symmetric key KEY*
n^i_j	*nonce related to j generated by i*

A *nonce* is a parameter that varies with time, *e.g.* a special marker intended to prevent the unauthorised replay or reproduction of a message.

A model defined in Crypto-CCS consists of a set of sequential agents able to communicate by exchanging messages. Inference systems model the possible operations on messages and therefore consist of a set of rules of the form $r = \frac{m_1 \cdots m_n}{m_0}$, with premises $m_1, \ldots, m_n$ and conclusion m_0. An instance of the application of rule r to closed messages $m_1, \ldots, m_n$ is denoted $m_1 \cdots m_n \vdash_r m_0$.

The control part of the language consists of compound systems and its terms are generated by the grammar (only constructs used below are presented):

$S := S_1 \parallel S_2 \mid A$	*compound systems*
$A := \mathbf{0} \mid p.A \mid [m_1 \cdots m_n \vdash_r x]A; A_1$	*sequential agents*
$p := c!m \mid c?x$	*prefix constructs*

where $m_1, \ldots, m_n, m$ are closed messages or variables, x is a variable and c is a channel. Informally, the Crypto-CCS semantics used below are:

- $c!m$: a message m sent over channel c;
- $c?x$: a message m received over channel c which replaces variable x;
- $\mathbf{0}$: a process that does nothing;
- $p.A$: a process that can perform an action p and then behave as A;
- $[m_1 \cdots m_n \vdash_r x]A; A_1$ (the inference construct): if (by applying an instance of rule r with premises $m_1, \ldots, m_n$) a message m can be inferred, then the process behaves as A (where m replaces x), otherwise it behaves as A_1.
- $S_1 \parallel S_2$: the parallel composition of S_1 and S_2, *i.e.* $S_1 \parallel S_2$ performs an action if either S_1 or S_2 does. It may perform a synchronisation or internal action, denoted by τ, whenever S_1 and S_2 can perform two complementary send and receive actions over the same channel.

The language is completely parameteric w.r.t. the inference system used. The inference system that is used below to model our scenario is shown in Figure 2.

$$\frac{x \quad y}{Pair(x,y)}\ (pair) \qquad \frac{x \quad pk_y^{-1}}{\{x\}_{pk_y^{-1}}}\ (sign) \qquad \frac{x \quad KEY}{\{x\}_{KEY}}\ (enc)$$

$$\frac{Pair(x,y)}{x}\ (1st) \qquad \frac{\{x\}_{pk_y^{-1}} \quad pk_y}{x}(ver) \qquad \frac{\{x\}_{KEY} \quad KEY}{x}\ (dec)$$

$$\frac{Pair(x,y)}{y}\ (2nd) \qquad \frac{x}{x}\ (check)$$

Fig. 2. Inference system for the federated network providers scenario.

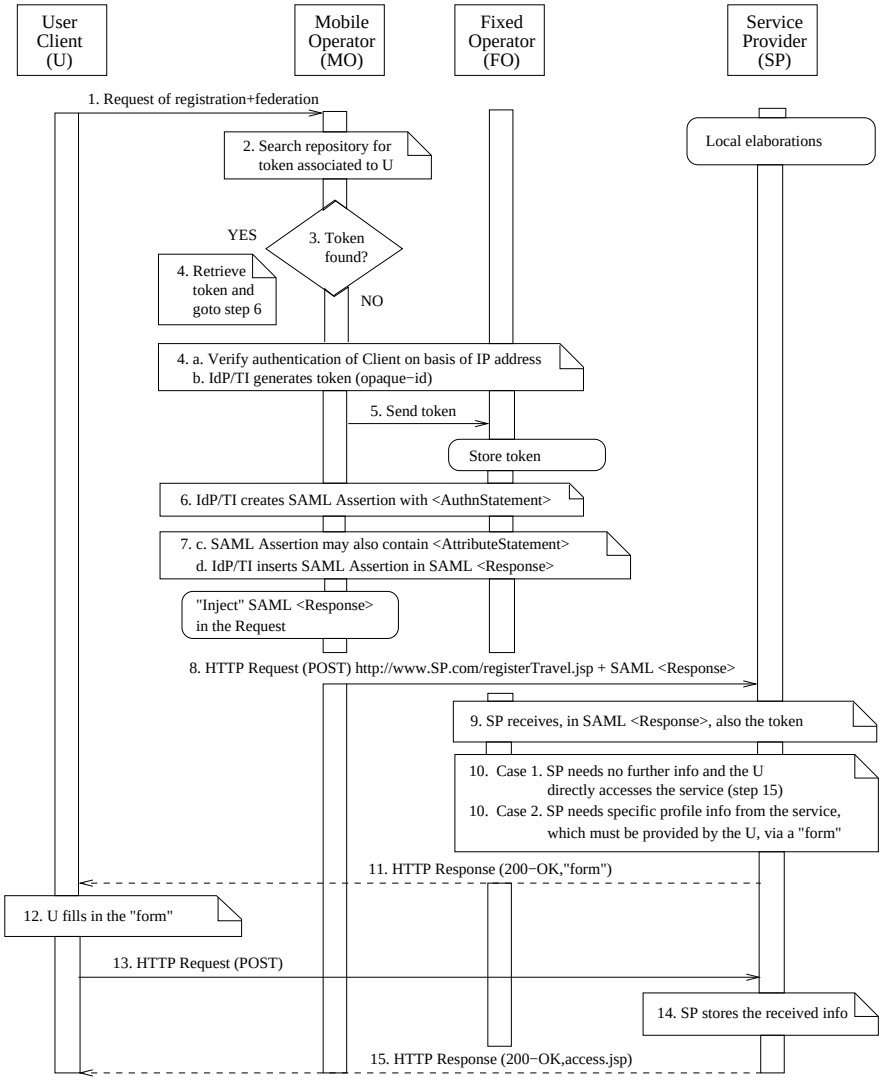


Fig. 3. Message sequence chart of federated network providers.

Rule (*pair*) builds the pair of two messages x and y. Rules (*1st*) and (*2nd*) return the components of a pair. Rule (*sign*) digitally signs a message x by applying the secret key pk_y^{-1} of agent y. Rule (*ver*) verifies a digital signature $\{x\}_{pk_y^{-1}}$ by applying the public key pk_y of signer y. Rule (*enc*) encrypts a message x by applying the symmetric key KEY. Rule (*dec*) decrypts a message $\{x\}_{KEY}$ by applying the symmetric key KEY. Finally, rule (*check*) performs checks on the correctness of authentication statements and on the freshness of nonces.

4 Specification of Federated Network Providers Scenario

The federated network providers scenario that we formalise involves a user U, a service provider SP and two federated network providers: a fixed operator FO and a mobile operator MO. Our starting point is when U initiates the process of federated registration with SP through MO. From [2] we inherit the assumption that all communication between FO and MO is secure: we consider them to share a secret key KEY_{FM}. The process is lined out in detail in Figure 3.

SAML [8] is an XML standard to exchange information on authentication and authorisation data between security domains intended to implement mechanisms for single sign-ons. A *SAML assertion* declares a subject authenticated by a particular means at a particular time. For our purposes, it contains a field *Subj* with the token id_U identifying U, a field $Auth\,Stat$ with an authentication statement asserting that U was authenticated (and the mechanism by which s/he was), and a field $Attr\,Stat = \langle attr\,list, n_U^{IdP} \rangle$ with a list of attributes of U related to her/his service accesses and a nonce to avoid replay attacks.

To avoid dealing twice with the same process of token generation, we propose a formalisation that slightly enriches the procedure presented in [2, 6]: as soon as one of the two network providers receives a request from U, it searches its repository for a token already associated to U. If this token is found, then it is retrieved and the procedure continues as in the federated registration scenario. Otherwise it is generated and immediately sent to the other federated network provider, where it is stored for subsequent interactions between U and SP:

$$\begin{array}{lllll} c_0 & U \mapsto MO & : & r \\ c_{MF} & MO \mapsto FO & : & \{id_U, U\}_{KEY_{FM}} \\ c_1 & MO \mapsto SP & : & \{r, SAML\ assertion\}_{K_{MO}^{-1}} \\ c_2 & SP \mapsto U & : & \{ok/ko\}_{K_{SP}^{-1}} \end{array}$$

Next we specify this federated network providers scenario in Crypto-CCS. This specification is more expressive than the one in standard notation given above, because all operations and security checks on the various messages are explicitly modelled. Each process is parameterised by the terms or variables it has in its knowledge (from the beginning or because it received them earlier).

$U_0(r) \doteq c_0!r.c_2?x_{sign}.$ *send request, receive signature,*
$\quad [x_{sign} \quad K_{SP} \vdash_{ver} x_{acc}].\mathbf{0}$ *verify signature and stop*
$SP_0(0) \doteq c_1?x_m.SP_1(x_m)$ *receive SAML assertion + request and goto next state*

$SP_1(x_m) \doteq [x_m \quad k_{IdP} \vdash_{ver} x_p]$ *verify signature,*
$[x_p \vdash_{2nd} x_{enc}]$ *extract encryption,*
$[x_{enc} \quad KEY \vdash_{dec} x_{dec}]$ *decrypt,*
$[x_{dec} \vdash_{1st} x_{pair}]$ *extract pair: token + Auth Stat,*
$[x_{dec} \vdash_{2nd} x_{n_U^{IdP}}]$ *extract nonce,*
$[x_{pair} \vdash_{1st} x_{id_U}]$ *extract token,*
$[x_{pair} \vdash_{2nd} x_{auth}]$ *extract Auth Stat,*
$[x_{auth} \vdash_{check} x_{auth}]$ *test correctness Auth Stat,*
$[x_{n_U^{IdP}} \vdash_{check} x_{n_U^{IdP}}]$ *test freshness nonce,*
$[x_{id_U} \quad x_{n_U^{IdP}} \vdash_{pair} (x_{id_U}, x_{n_U^{IdP}})]$ *build pair to store,*
$c_S!(x_{id_U}, x_{n_U^{IdP}})$ *store token + nonce pair,*
$[access \quad k_{SP}^{-1} \vdash_{sign} x_{sign}]$ *prepare signature to*
$c_2!x_{sign}.\mathbf{0}$ *grant access and stop*
$MO_0(0, n_U^{MO}, id_U, KEY_{FM}) \doteq c_0?x_r.$ *receive request and*
$MO_1(x_r, n_U^{MO}, id_U, KEY_{FM})$ *goto next state*
$MO_1(x_r, n_U^{MO}, id_U, KEY_{FM}) \doteq [id_U \quad U \vdash_{pair} (id_U, U)]$ *create pair,*
$[(id_U, U) \quad KEY_{FM} \vdash_{enc} \{(id_U, U)\}_{KEY_{FM}}]$ *encrypt pair,*
$c_{MF}!\{(id_U, U)\}_{KEY_{FM}}.$ *send token to FO,*
$[id_U \quad auth \vdash_{pair} (id_U, auth)]$ *create pair,*
$[(id_U, auth) \quad n_U^{MO} \vdash_{pair} ((id_U, auth), n_U^{MO})]$ *create pair,*
$[((id_U, auth), n_U^{MO}) \quad KEY \vdash_{enc} \{((id_U, auth), n_U^{MO})\}_{KEY}]$ *encrypt pair,*
$[x_r \ \{((id_U, auth), n_U^{MO})\}_{KEY} \vdash_{pair} (x_r, \{((id_U, auth), n_U^{MO})\}_{KEY})]$ *create pair,*
$[(x_r, \{((id_U, auth), n_U^{MO})\}_{KEY}) \quad k_{MO}^{-1} \vdash_{sign} x_{sign}]$ *sign pair,*
$c_1!x_{sign}.\mathbf{0}$ *send SAML assertion + request and stop*
$FO_0(KEY_{FM}) \doteq c_{MF}?x_{enc}$ *receive encryption,*
$[x_{enc} \quad KEY_{FM} \vdash_{dec} x_{dec}]$ *retrieve decryption,*
$c_S!x_{dec}.\mathbf{0}$ *store token + identity pair and stop*

For the sake of readability, we did not fully spell out the digital signatures (*i.e.* we just applied the private key to the message to be signed). Moreover, we assumed a direct communication channel between *SP* and *U* in order to grant/deny access, while in reality all communication passes through *IdP*. The whole process is described by $U_0(r) \parallel MO_0(0, n_U^{MO}, id_U, KEY_{FM}) \parallel FO_0(KEY_{FM}) \parallel SP_0(0)$.

5 Verification of a Security Property

As an intermediate step towards a full-blown security analysis of the protocol, we verified the vulnerability of our federated network providers scenario w.r.t. a man-in-the-middle attack. Such an attack is an adversary's attempt to intercept and modify messages between two trusted participants, in such a way that neither participant is able to find out that their communication channel has been compromised. We used model checking to verify whether the specification of the insecure channel between *MO* and *SP* can withstand such an attack. In [1] we already verified that the insecure channel between *IdP* and *SP* can.

The analysis boils down to verifying the following property: whenever *SP* concludes the protocol apparently with *MO*, it was indeed *MO* that executed it. To

this aim, we introduced two special actions in the specification: *commit*(*SP,MO*) and *run*(*MO,SP*). The former represents the fact that *SP* has indeed terminated the protocol with *MO*, while the latter represents the fact that *MO* indeed started communicating with *SP*. We then translated the property into requiring *run*(*MO,SP*) to always precede *commit*(*SP,MO*). We did the same to test for possible misbehaviour or interceptions of communications between *MO* and *FO*, by introducing the actions *commit*(*FO,MO*) and *run*(*MO,FO*) in the specification.

We used the model checker PaMoChSA v1.0 [5] to verify this. We considered an adversary X and set its initial knowledge to the set of public messages that it knows at the start of the protocol, *i.e.* the public keys of *MO*, *FO* and *SP* and its own public and private key denoted by pk_X and pk_X^{-1}. With as input the specification, the logic formula (*commit*(*SP,MO*) AND (NOT *run*(*MO,SP*))) OR (*commit*(*FO,MO*) AND (NOT *run*(*MO,FO*))) and the intruder's initial knowledge $\{pk_X, pk_X^{-1}, pk_{MO}, pk_{FO}, pk_{SP}\}$, the result of the analysis was **No attack found**.

To verify the logic formula specified above, the tool set out to find a run of the protocol with the following characteristic: At the end of the run, the adversary either knows message *commit*(*SP,MO*), but it does not know message *run*(*MO,SP*) (*i.e. SP* is convinced to have finished talking with *MO*, while in reality *MO* has never started talking with *SP*), or it knows message *commit*(*FO,MO*), but it does not know message *run*(*MO,FO*) (*i.e. FO* is convinced to have finished talking with *MO*, while in reality *MO* has never started talking with *FO*). Since the tool did not find such a run we conclude that, at the conceptual level, the network protocol is correct w.r.t. the analysed security property.

6 Conclusion

The result of the security analysis presented above, together with the one presented in [1], strengthens our confidence in the formal specifications of the scenarios presented in these two papers. In particular, it leads us to believe that we correctly inserted digital signatures, encryption and nonces into the protocol.

References

1. ter Beek, M.H., Moiso, C., Petrocchi, M.: Towards Security Analyses of an Identity Federation Protocol for Web Services in Convergent Networks. In: Proc. AICT'07, IEEE Computer Society (2007)
2. Bonifati, M., De Lutiis, P., Moiso, C., Morello, E., Sarchi, L.: Identity Federation for Services in Convergent Networks. In: Proc. ICIN'06. (2006) 109–114
3. Focardi, R., Martinelli, F.: A uniform approach for the definition of security properties. In: Proc. FM'99. Volume 1708 of LNCS., Springer (1999) 794–813
4. Martinelli, F.: Analysis of security protocols as open systems. Theoretical Computer Science **290**(1) (2003) 1057–1106
5. IIT–CNR: Partial Model Checking Security Analyzer PaMoChSA v1.0 (2007)
6. T. Wason et al.: Liberty ID-FF Architecture Overview v1.2 (2005)
7. Dolev, D., Yao, A.: On the Security of Public Key Protocols. IEEE Transactions on Information Theory **29**(2) (1983) 198–208
8. OASIS Security Services: Security Assertion Markup Language SAML v2.0 (2005)

Posters

Aspect Oriented Web Service Composition and Choreography Analysis

Connie Haoying Bao and Nicolas Gold

King's College London, CREST,
Strand, London, WC2R 2LS.
{Haoying.Bao, Nicolas.Gold}@kcl.ac.uk

Abstract. In this paper we discuss a novel Aspect Oriented approach to the analysis of Web Services Composition and Choreography. Recent development on Web Services Business Process Execution Language (WS-BPEL) and Web Service Choreography Description Language (WS-CDL) defined standards for service orchestration and global integration. However, the standards have their limitations, e.g. the lack of modularity and difficult to accommodate changes tat run-time. The root of these problems is that the current standards do not specify at process level. The contribution of this paper is to facilitate the composition and maintenance of aggregated services by applying Aspect Orientated (A.O.) techniques. In particular Query Based (Q.B) analysis that supports modularization and service decomposition. Applying Q.B. techniques to service composition problem is novel; this work explores and demonstrates its effectiveness and how it addresses modularisation and decomposition at process-level.

Keywords: Web services, composition, choreography, Aspect Oriented.

1 Introduction

Recent development on Business Process Execution Language for Web Services (WS-BPEL), Web Services Description Language (WSDL) and Web Service Choreography Description Language (WS-CDL) defined *service orchestration and choreography* standards that would integrate services into composite processes and processes into choreographed workflows. However, the standards have their limitations; among them is the lack of modularity support, also known as separation of concerns, where specifications and non-functional requirements are tangled [3].

The interactions in a business process are state-based interactions, however interactions defined by WS-BPEL and WS-CDL are stateless message exchanges. WS-BPEL defines such business processes using its formal descriptions of the process' message exchange protocols. It uses XML constructs to describe basic logic, structural and concurrent activities, this description could be used to analyze the process, and its interactions.
Consistency of object-oriented behavioural models, such as scenarios and state

machines, has already been extensively studied; for a WS-BPEL interaction four types of incompatibilities were identified in [9] that may arise during a WS-BPEL interoperation, among them are *Structural incompatibilities* that are mismatches in XML types between sender and receiver; and *Value incompatibilities* feature matching XML types, but an unexpected value within the message.

Our work attempt to first address the structural and value incompatibilities using A.O. techniques, which performs static and dynamic analysis on BPEL and WS-CDL descriptions, identifies and mediates over the incompatibilities. This approach also forms an extra layer in the component based architecture that opens up opportunities for A.O. to introduce an extra dimension in service composition. This work presents some of concepts we are exploring, including A.O. approach to hot-fixes on long running processes, performance monitoring and compliance monitoring in financial services.

2 Applying AOP to WS Protocols

Several research projects recognized that services composition needs to be flexible and dynamic. Among works on Adaptive Composition [4,5,6] AO4BPEL implementation [3] using AOP techniques to increase flexibility and modularity, is the most similar approach to the proposed work. AO4BPEL uses *Aspect-aware engine* approach, implemented an A.O. BPEL engine; original BPEL process is preserved, the engine checks for Aspects before and after each activity and execute them. The engine implements the popular Joint Point & Point cut mechanism, supporting dynamic composition of workflows.

Our work on applying Aspect Oriented method other than the Joint Point & Point cut mechanism is a novel one. We propose to use the Process Transformation approach [2] merging aspects and workflow process before deployment. We use the A.O. mechanism demonstrated by JQuery [7], a light weight query browser implemented using declarative configuration language. The prolog-like query language and interface provides maximum flexibility and modularization power in analysing a workflow. Despite the popularity and wide adaptation of AspectJ and other Joint Point and Advice implementations, the paradigm suffers badly from serious performance overhead [7,8], minor change could results in a major 256% increase in compile time.

Subject to experimental results we would like to demonstrate with our approach with the Query Based A.O. paradigm would prove to be more efficient performance wise.

References

1. Khalaf, R., Leymann, F.: E Role-based Decomposition of Business Processes using BPEL. ICWS, 2006, 770-780.
2. Eder, J. Gruber, W., Pichler, H.: Transforming Workflow Graphs. In Proceedings of the 1st International Conference on Interoperability of Enterprise Software and Applications (INTEROP-ESA), February 2005.
3. Charfi, A. and Mezini, M. "Hybrid Web Service Composition: Business Processes Meet Business Rules", In Proceedings of the 2nd International Conference on Service Oriented Computing (ICSOC 2004). 2004.
4. Dynamic and adaptive composition of e-services, *Inf. Syst.*, *Elsevier Science Ltd.*, **2001**, *26*, 143-163
5. Jing-Fan Tang; Bo Zhou; Zhi-Jun He; Pompe, U. Adaptive workflow-oriented services composition and allocation in distributed environment, *Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on*, **26-29 Aug. 2004**, *Volume 1*,, Page(s): 599 - 603 vol.1
6. Q. Z. Sheng, Z. M. M. D. & Ngu, A. H. Enabling Personalized Composition and Adaptive Provisioning of Web Services . *In Proc. of the 16th International Conference on Advanced Information Systems Engineering (CAiSE),*, **June . 2004 .**, *Springer Verlag . Riga, Latvia.*
7. Janzen, D. & Volder, K. D. Navigating and querying code without getting lost AOSD '03: Proceedings of the 2nd international conference on Aspect-oriented software development, ACM Press, 2003, 178-187
8. Kiczales, G., Hilsdale, E., Hugunin, J., Kersten, M., Palm, J., Griswold, W. *An Overview of AspectJ*. In Proc. Of ECOOP '01, LNCS 2072, pp. 327-353, Springer, 2001
9. Shankar R. Ponnekanti and Armando Fox. Interoperability among independently evolving web services. ACM/Usenix/IFIP Middleware '04, Toronto, Canada, October 2004.

WS-Engineer: Tool Support for Engineering Web Service Compositions and Choreography

Howard Foster

Department of Computing
Imperial College London
180 Queen's Gate, London SW7 2BZ, UK

1 Introduction

This poster presents an engineering approach to designing, implementing and maintaining web service behaviour specifications, covering some emerging standards in the current web service standards stack. We present a rigorous approach to formally specifying, modelling, verifying and validating the behaviour of web service compositions with the goal of simplifying the task of designing coordinated distributed services and their interaction requirements. Through the use of our Eclipse plug-in tool, known simply as WS-Engineer, Web Service standards are used as an implementation example, with models generated from the XML of BPEL4WS and WS-CDL, that represent web service orchestrations and choreography respectively. Through the use of formal model verification techniques, the behaviour specified in these XML documents can be checked for service interaction violations between design and implementation, between processes (as service conversations) and by the role of the processes. Obligations analysis considers how each service either fulfills or falls short of providing sufficient interactions in service choreography.

2 The Approach

Our approach is undertaken as follows. A designer, given a set of web service requirements, specifies a series of Message Sequence Charts (MSCs) to describe how the services will be used and to model how each service interacts (i.e. service conversations) in a given service scenario. The resulting set of scenarios is composed and synthesized to generate a design behavioral model, in the form of the Finite State Process (FSP) algebra and then compiled in to a Labelled Transition System (LTS). Equally, WS-CDL specifications can also be constructed to describe a multi-partner collaboration policy, and a similiar translation to FSP model can be undertaken. The service implementation is undertaken by a BPEL4WS engineer, who builds BPEL4WS processes directly from either specification or requirements. The BPEL4WS implementations are used to generate further behavioral models by a process of abstracting interactions from the

BPEL4WS process to yield a model of interaction based upon specified semantics applied to BPEL4WS through the FSP algebra. Verification and validation consists of comparing and observing traces of these transition systems. The approach can assist in determining whether the implementation contains all the specified scenarios of the design or policy and whether any additional scenarios exhibited by the implementation are acceptable to the end-user. In addition, checks can be made on the models with respect to desirable general global properties such as absence of deadlock and liveness (using model-checking). Feedback to the users is in the form of MSCs. Our approach currently supports analysing BPEL4WS, WS-CDL and WSIF (as Invocation Logs), yet as other standards emerge these can be incorporated in to the approach. Additionally we are working on generating templates of standards based implementations for both choreography and service implementation.

3 The Tool

The tool, written as a plug-in to the Eclipse integrated development environment, provides the service engineer with features to generate, enhance and analyse documents implementing the service process and the service policy. Building upon a suite of analysis tools, the WS-Engineer plug-in extends an LTSA Eclipse plug-in with support for translating BPEL4WS, WS-CDL and WSIF specifications to the Finite State Process (FSP) notation. As these are translated, they can be compiled in to Labelled Transition Systems (LTS) for analysis. The tool provides different views for graphically viewing models (as graphical LTS models), animating models and verifying models against a variety of predefined properties. For example, a "Service Compatibility View" allows the engineer to compare orchestration, choreography and design models. One particularly interesting feature is to perform an "Obligations Analysis", where particular roles and their interactions are compared with that of implementation (as BPEL4WS processes) or policy (in the form of WS-CDL). The tool is available at: http://www.doc.ic.ac.uk/ltsa/eclipse.

4 Future Work

The main contribution in our work is to provide an approach, which when implemented within the tool, provides a mechanical verification of properties of interest to both designers and implementers of web service compositions. The use of a formal, well defined, process algebra (in this case FSP) provided a semantic mapping between the composition implementation (in the BPEL4WS specification for web service compositions and WS-CDL for service choreography policies). The approach provides an extendable framework in which further specifications and properties can be defined and implemented to assist in an efficient, mechanical service testing and analysis tool set.

Reengineering Systems for Multi Channel Access – Systematic Literature Review Protocol

Clive Jefferies and Pearl Brereton

School of Computing and Mathematics,
Keele University, Keele, Staffs, United Kingdom,
red29@epsam.keele.ac.uk

Abstract

Many companies and institutions could benefit from adapting their software systems to support multi channel access. This means that the software can be accessed by heterogeneous devices such as mobile phones, PCs or PDAs. To completely replace these systems may be expensive and time consuming [4]. Another way is to reengineer the system as a service. There are two types of re-engineering that can be used for service enablement: *black box* – where no knowledge of the program code is needed or *white box* – where the program code itself is changed. Black box techniques can be difficult if Business Logic is tied up with Presentation Logic [2]. Black box approaches are also seen as a temporary solution rather than a permanent one and best used as an interim solution [1].

A Systematic Literature Review (SLR) [3] enables a researcher or group of researchers to gather information on a problem or research topic in a rigorous and non biased way. I will be using a SLR to search for methodologies and the issues involved in service enablement. Before conducting a SLR a Review Protocol must be created. This protocol states what will be done during the review and who will do it.

The starting point of the SLR Protocol is the draft research questions. These form the basis of the way in which the review will be formed and the purpose of the review. An example research question from this review is: Have there been any proposed solutions and frameworks to address reengineering systems as services?

Once the questions have been decided, these guide the generation of the search terms. The terms must be decided along with any synonyms, abbreviations and alternative spellings so that a comprehensive yet concise search string can be generated. Once the search terms have been decided the resources used for the review are then selected and documented. The resources that will be used in this study are ACM or the IEEExplore digital libraries and Google search engine.

The next sections of the SLR Protocol are the inclusion and exclusion criteria. For the inclusion criteria you are looking for aspects that would mean that the paper should be included in the literature review. The exclusion criteria are used to say what explicitly will not be included. These help filter the potentially large number of papers found in the search. The next step in the SLR Protocol is to outline the data to be extracted. For

this procedure the researcher must create documentation which states exactly what data is to be extracted from each paper during the review.

For the SLR Protocol the researcher must then say how the extracted data will be synthesised for subsequent processing. In this study the synthesised data will be used to choose a reengineering method for service enablement or to suggest a new method. This method will then be applied to IBHIS, an existing service-based system [5] and finally evaluated. The IBHIS system is a service-based broker that is used to query multiple heterogeneous data sources. Although this system uses web services to communicate between the broker and the data sources, the system itself is not exposed as a service so therefore will be an ideal candidate for reengineering.

References

1. Anand S., Chatterjee A. M., Kumar V., Raut V. & Singh V. (2005). Towards Legacy Enablement Using SOA and Web Services: Leverage legacy systems with SOA. <http://webservices.sys-con.com/read/164558.htm> Accessed on 09.05.2007

2. Kousoukos G., Andrade L., Gouveia J. & El-Ramy M. (2006). Service Extraction. <http://www.pst.ifi.lmu.de/projekte/Sensoria/del_12/D6.2.a.pdf> Accessed on 08.05.2007.

3. Kitchenham, B. (2004). Procedures for performing systematic reviews. Technical report Software Engineering Group, Department of Computer Science, Keele University.

4. Newcomer E., Lomow G. (2005). *Understanding SOA with Web Services.* Addison Wesley. ISBN 0-321-18086-0.

5. Turner M., Zhu, F., Kotsiopoulos, I., Russell, M., Budgen, D., Bennett. K., Brereton, P., Keane, J., Layzell, P. & Rigby, M. (2004). Using Web Services Technologies to create an Information Broker: An Experience Report. Presented at 26th International Conference on Software Engineering (ICSE 2004), Edinburgh, Scotland, 2004.

Using Enhanced Causal Paths based on Passive Tracing in Determining a Web Service Topology

Marian Mohr and Nicolas Gold

King's College London, CREST,
Department of Computer Science, London, United Kingdom
marian.mohr@kcl.ac.uk, nicolas.gold@kcl.ac.uk
http://www.dcs.kcl.ac.uk/pg/mohrmari,
http://www.dcs.kcl.ac.uk/staff/nicolas

Abstract. This paper describes an approach to understanding a system's behaviour and provides suggestions for improving currently available techniques with regards to causal paths.

1 Position

As distributed systems become more complex by the day, so does the crucial task of understanding the actual works that underlie their operations. Vague answers to questions such as: "what is the effect of X on Y?" – X oftentimes being a system under development and Y being a set of production systems – are commonplace. In [1], Moe and Sandahl describe a variety of issues that can be addressed, simply by way of understanding of the effects of a particular system, subsystem or even individual message on a system, including amongst others, the discovery of hidden problems, bottlenecks and sources of exceptions. While the overall research focus so far has been on general message flows, covering topics such as SLA monitoring [2] [3] and execution tracing [4], and mature technologies such as CORBA, much less has been written with regards to tracing web services, particularly the effects of service calls beyond the first point of contact.

We propose to explore this matter by reviewing core challenges in understanding overall system behaviour, illustrating, comparing and contrasting current approaches and ultimately proposing new approaches to enhancing "casual reasoning" by means of exploiting properties unique to service environments. Challenges that we consider critical in our review include the selection and gathering of data that forms the base of any analysis, considering model-based, intrusive and non-intrusive techniques as well as related performance implications, the "location" of the analysis and other practical concerns such as a lack of a global clock [5]. To support the review, we suggest a classification of approaches into six classes:

1. *program comprehension* studies source code to derive a global interaction model.

2. *intrusive tracing* is a technique commonly found in the form of logging, which requires active support by developers, who need to inform a certain entity of the beginning and end of any communication that should be traceable. A popular example is ARM [6]. Variants of this class, such as application annotations as in the case of Pip [7] and proxies [8] are included in this class.
3. *analysis of server-logs* is a less intrusive approach, based on the availability and correctness of logs from all servers and the post-execution analysis of these logs to determine associations [9], however requires standardization due to a large variety of application servers.
4. *message identification* suggests modifying each message by attaching a unique identifier by means of which correlated messages can be identified [10]. Currently implementations are entirely proprietary.
5. *model analysis* makes use of high-level system descriptions, most commonly found in the form of a design artefacts, such as BPEL orchestrations [11].
6. *causal paths* represents entirely passive, non-intrusive observation of network traffic, for example by means of port mirroring, and inference of paths using algorithms based on message characteristics or environment variables such as time. Examples for this are nesting, convolution [12] and linking [4]. In contrast to other approaches, it can only provide probabilistic results, however it is entirely independent of other systems.

We place particular emphasis on "causal paths", as it is the least intrusive approach and the quality of its results is subject exclusively to the quality of the algorithms deployed. By way of focusing on web service interactions, it is possible to exploit its key characteristics in improving these algorithms. We believe that improvements can be achieved in the following:

1. *data collection* – demonstrating that by means of using filters that can be applied to well-formed messages (e.g. SOAP) at data collection points, it would be possible to reduce the overall overhead of collecting and routing message information. This technique would not only reduce resource utilization and correspondingly network congestion, but would also enable us to highlight rare and therefore possibly more important messages.
2. *focused data collection* – suggesting the use of a technique known as message replay based on semantic evaluation of method signatures. A technique well-known in the area of wireless communication security could be applied, since web services allow for thorough analysis thanks to its XML message format.
3. *message pattern identification* – making use of standards such as WS-Transaction to facilitate the process of associating individual messages with a particular message flow. Instead of attaching unique identifiers to messages, one could use knowledge of messaging standards to improve the analysis process, possibly resulting in near-identifier algorithm performance.
4. *design pattern identification* – testing for common service patterns such as RPC and publish/subscribe to identify message flows, the latter being especially important, as literature rarely discusses this scenario. Since certain design patterns recur frequently, knowledge about them could be used in improving analysis performance.

This paper outlined an approach to the understanding of a system's behaviour and has sketched out specific suggestions for improving currently available techniques with a focus on causal paths.

References

1. J. Moe and K. Sandahl. Using execution trace data to improve distributed systems. In *Proceedings of the International Conference on Software Maintenance*, pages 640–648, 3-6 Oct. 2002.
2. Akhil Sahai, Vijay Machiraju, Mehmet Sayal, Aad P. A. van Moorsel, and Fabio Casati. Automated SLA Monitoring for Web Services. In *Proceedings of the 13th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management: Management Technologies for E-Commerce and E-Business Applications*, volume 2506 of *Lecture Notes In Computer Science*, pages 28–41, London, UK, 2002. Springer-Verlag.
3. Alexander Keller and Heiko Ludwig. The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services. *Journal of Network and Systems Management*, 11(1):57–81, March 2003.
4. Patrick Reynolds, Janet L. Wiener, Jeffrey C. Mogul, Marcos K. Aguilera, and Amin Vahdat. WAP5: black-box performance debugging for wide-area systems. In *WWW '06: Proceedings of the 15th International Conference on World Wide Web*, pages 347–356, New York, NY, USA, 2006. ACM Press.
5. Zoltán Ádám Mann. Tracing System-Level Communication in Object-Oriented Distributed Systems. In *Proceedings of the OMG Information Day*, March 2001. Budapest (Hungary).
6. Mark W. Johnson. Monitoring and Diagnosing Applications with ARM 4.0. White paper, IBM Corporation, December 2004. Final.
7. Patrick Reynolds, Janet L. Wiener, Jeffrey C. Mogul, Mehul A. Shah, Charles Killian, and Amin Vahdat. Pip: Detecting the Unexpected in Distributed Systems. In *Proceedings of the 3rd ACM/USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 115–128, San Jose, CA, May 2006.
8. G. Carls, Ying Lu, and G. Aschemann. Validating interaction patterns of CORBA based network management systems. In *Network Operations and Management Symposium, 2000. NOMS 2000. 2000 IEEE/IFIP*, pages 31–44, 10-14 April 2000.
9. Wim De Pauw, Sophia Krasikov, and John F. Morar. Execution patterns for visualizing web services. In *SoftVis '06: Proceedings of the 2006 ACM symposium on Software visualization*, pages 37–45, New York, NY, USA, 2006. ACM Press.
10. Mike Y. Chen, Anthony Accardi, Emre Kiciman, Dave Patterson, Armando Fox, and Eric Brewer. Path-Based Failure and Evolution Management. In *Proceedings of the 1st ACM/USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 309–322, San Francisco, CA, March 2004.
11. M. Pistore, F. Barbon, P. Bertoli, D. Shaparau, and P. Traverso. Planning and Monitoring Web Service Composition. In *Artificial Intelligence: Methodology, Systems, and Applications*, volume 3192 of *Lecture Notes in Computer Science*, pages 106–115. Springer Berlin / Heidelberg, 2004.
12. Marcos K. Aguilera, Jeffrey C. Mogul, Janet L. Wiener, Patrick Reynolds, and Athicha Muthitacharoen. Performance debugging for distributed systems of black boxes. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 74–89, New York, NY, USA, 2003. ACM Press.

Typed Abstractions for Client-Service Interactions in OSGi

Sven De Labey and Eric Steegmans
K.U.Leuven, Dept. Computer Science, 200A Celestijnenlaan, B-3000 Leuven, Belgium
{svendl, eric}@cs.kuleuven.be

1 Introduction

OSGi technology provides a service-oriented, component-based environment and offers standardized ways to manage the software lifecycle. It subscribes to the Service-Oriented Programming paradigm by providing a *central service registry* which is used by providers to publish their services. These published services can then be retrieved by clients using an LDAP-based query mechanism.

In this text, we briefly evaluate the OSGi service registration mechanism based on three *desired properties* for service registries: *static service properties*, *dynamic service properties*, and *derived properties*.

2 Client-Service Interactions in OSGi

During service registration in OSGi, the service provider may attach a *dictionary* of service properties to the service being registered. Clients can assemble properties from this dictionary into LDAP-like queries (Lightweight Directory Access Protocol) such as `(&(objectClass=Printer)(ppm=20))` so as to fine-tune their search. A search based on a key/value pair such as `(ppm=20)` is only valid when the service provider has added the key `ppm`.

Evaluation Given our three desirable properties for service registries, we can evaluate the OSGi registration mechanism:

- **Static Service Properties.** These properties never change during the lifetime of a service, so it is possible to include them as properties in a static dictionary. Therefore, they can be handled well by the OSGi service registration protocol.
- **Dynamic Service Properties.** Properties such as the length of a queue are dynamic. This information cannot be added when the service is registered, so the OSGi registration mechanism cannot allow clients to conduct a search based on dynamic service properties. This is a major limitation because services are highly dynamic entities with frequently changing characteristics.
- **Derived Properties.** A search based on derived properties (e.g. calculating the cost for printing a file), is not supported by the LDAP-based query mechanism of OSGi. Therefore, the result of such derived properties can only be computed when the client has obtained a reference to a service. Thus, in OSGi, such derived properties can only be taken into account after the client has retrieved all candidate services and after it has calculated the required information by contacting each of these services. This approach is inefficient and reduces the readability of the client code.

Another drawback of OSGI is that, by relying on LDAP as a query language, it bypasses compile-time checks on service queries because these queries are passed as *strings*. This leads to potential `InvalidSyntaxExceptions` at runtime. Summarizing, OSGI uses a *dynamically* typed query language for describing *static service properties* and what we need, is exactly the opposite.

3 Typed Abstractions for OSGi

We are using ServiceJ, our Java extension for service interactions, to augment the semantics of interactions between clients and the OSGi registration mechanism. The idea underlying this extension is that service properties are modeled by *operations exported by the service API*. In stead of defining a *dictionary* with a key `ppm`, for example, we use *methods* such as `getPagesPerMinute()` provided by the `PrinterService` interface. ServiceJ introduces special variables, called *pool variables*, that can be used to program client-service interactions. When the client declares a variable such as "`pool Printer printer;`", the ServiceJ middleware automatically contacts the OSGi service registry and retrieves an appropriate service, which is then injected into the pool variable. This injection mechanism can be fine-tuned using *declarative operations*. ServiceJ aims at realizing our goals for service registries as follows:

Static and Dynamic Service Properties. ServiceJ introduces a `where` clause that accepts a *boolean expression* as an argument. This boolean expression can be used to query for both static and dynamic service properties because the boolean expression is evaluated *just-in-time* (i.e. when an operation is invoked on the pool variable). For example, retrieving a `PrinterService` with a queue containing at most 3 other jobs is written as:

```
pool PrinterService ps where ps.getQueue().getLength()<=3;
```

Derived Properties. Because we use regular instance methods in stead of a dictionary, it becomes possible to take into account derived properties. Information is provided by the client using the *actual parameters* of a method invocation. This information is used at the Service Registry to compute the resulting set of services. For example, getting a printer that charges the user less than a predefined `maxCost` for some `fileInfo` is written as:

```
pool PrinterService ps where ps.getCostFor(fileInfo)<maxCost;
```

Typed Abstractions. In contrast with OSGi, which uses a dynamically typed query language for static service properties, ServiceJ enables us to use a statically typed language that takes into account both *dynamic* and *derived* service properties. Being a statically typed language, ServiceJ does not force service clients to catch `InvalidSyntaxExceptions` each time they interact with the service registry because erroneous queries are signaled at compile time.

4 Status and Future Work

Currently, the extension of the OSGi registration mechanism has been implemented and a case study is being developed. Future work includes the introduction of a transaction mechanism based on ServiceJ *session blocks*, and the development of a *bridge* between Jini and OSGi based on ServiceJ.

A Mapping BPEL4WS Processes into CSP

Tuvshintur Tserendorj

Institute AIFB
University of Karlsruhe (TH), Germany
tuvshintur.tserendorj@aifb.uni-karlsruhe.de
http://www.aifb.uni-karlsruhe.de

Abstract. We study the applicability of process algebraic language such as CSP (Communicating Sequential Processes) and its verification technique for the emerging paradigm of Web Services. Furthermore, we present an approach for faithfully translating BPEL4WS [1] processes to CSP. The translator takes as input a BPEL4WS specification, automatically generates a CSP specification and enables the desired formal analysis such as deadlock, livelock and refinement checking for business processes with the model checker FDR2 [1].

Key words: Bpel4ws, ws-bpel, csp, model checker FDR2, verification

1 Introduction

Web services are rapidly emerging as a new communication paradigm for the execution of business processes across and between organizations. If these processes are unreliable, a failure in them can cause high economic losses. By using formal techniques it is possible to find errors in specifications at design time, thus increasing their reliability. In recent years various industrial standard specifications towards specifying business processes and web services have been proposed. A widely-accepted standard is the BPEL4WS specification. It provides a language for the formal specification of business processes and business interaction protocols. However, the language lacks formal semantics, and this hinders the formal analysis[2]. Although some efforts report on using such as formal languages CCS [2], π-calculus [3] and Petri nets [4] for defining formal semantics for BPEL4WS, no interesting relation with them has really been proved for a practical usage. We believe that this gap needs to be filled. We want to use the popular model checker FDR2 for verifying business processes and web services. CSP is a mature mathematical theory for specifying and reasoning about concurrent systems. It models a system as a collection of processes which run concurrently, communicate over unbuffered channels and synchronize on particular events. The reason to choose CSP is to profit not only from its well-defined specification language, but also from the analysis tools such as FDR2, etc. In addition, we believe that the refinement checking of CSP can be an extremely valuable tool for matchmaking of BPEL4WS processes and web services [2].

[1] Business Process Execution Language for Web Services
[2] deadlock, livelock and non-determinism

2 The Mapping

We have defined the formal semantics of BPEL4WS as a semantic translation function $\Psi : \mathit{BPEL4WS} \rightarrow \mathit{CSP}$. This is done inductively over the syntax of BPEL4WS processes in such a way that the primitive activities such as *receive*, *reply* and *invoke* have been handled as communications over channels. Furthermore, we have encoded the constructors *sequence*, *flow*, *switch* and *while* by sequential, alphabetized parallel, guarded processes and so-called μ-recursion, respectively. The fault and compensate handlers are described by individual CSP processes whereas the *scope* is encoded as nested processes. Although the constructors such as *sequence* and *flow* are straightforward to translate, we defined further optimizations for generating CSP specifications. For instance, one can define a BPEL4WS parallel process using the parallel constructor *flow*. If the activities occurring in the *flow* process were totally ordered via *links* it wouldn't be optimal to generate a corresponding CSP parallel process. We take such cases into account and generate optimized CSP processes so that the state space exploration problem can be avoided as far as possible. In particular for the abovementioned case we generate a sequential process, instead of a parallel process. The feature *links* is encoded as synchonized-events. Our tool is implemented as an Eclipse Plug-in that can be utilized in the BPEL visual environment of the Eclipse Technology Project.

3 Open issues and future works

An approach for faithfully translating BPEL4WS processes to CSP is presented, and this in turn has enabled the desired formal analysis such as deadlock, livelock and refinement checking for business processes with the model checker FDR2. Our current implementation is relatively robust. There are some issues we haven't completely handled yet, but we are still working on it, namely the compilation of the return value of the *invoke* and *assign* activities. We will extend our implementation in the way so as to make a smooth integration with FDR2. This extension should allow the BPEL designer to directly use the model checker FDR2 in the BPEL visual environment. Furthermore, we study the possibility of adopting a functional language as an in-built language for better handling of the activities such as *assign*.

References

1. A. W. Roscoe *The Theory and Practice of Concurrency*, Prentice Hall Series in Computer Science 1998.
2. Biplav Srivastava, Jana Koehler *Web Service Composition - Current Solutions and Open Problems*
3. Manuel Mazzara and Roberto Lucchi *A pi-calculus based semantics for WS-BPEL*, Journal of Logic and Algebraic Programming 2006
4. Karsten Schmidt and Christian Stahl. *A Petri net semantic for BPEL4WS - validation and application*, AWPN'04

Inference Security Threats in Service-Based Systems

Philip Woodall and Pearl Brereton

School of Computing & Mathematics, Keele University,
Keele, Staffordshire, ST5 5BG
p.m.woodall@cs.keele.ac.uk

Abstract

Existing access control models are not sufficient to prevent service-based information brokers releasing sensitive information from autonomous, heterogeneous and distributed data sources. Under certain conditions, an information broker can gather information from multiple sources, which is non-sensitive in isolation, but is sensitive when presented as a whole. Tools such as IBHIS, a service-based information broker based on the healthcare domain, can collate information from data sources that expose their data as a service [1]. Each data source manages its own separate security policy, which is upheld by IBHIS using the S-DAC access control model thus preventing the direct release of sensitive information [2]. However, consider the case where anonymised healthcare data from one data source is linked with identifying data, such as age, NHS Trust, or date of diagnosis from another data source. It may be possible for users of IBHIS to infer the identity of a patient and obtain sensitive information about a patient indirectly.

An analysis of the inference problems in service-based information brokers has been conducted using the Systematic Literature Review (SLR) methodology. A SLR is a structured, rigorous and auditable method used to obtain, extract and synthesise information relevant to a particular research question [3]. The SLR methodology—recently adopted by the Software Engineering community—was used to identify a set of inference strategies from the existing literature. The term ‘inference strategy’ refers to the method (series of processes) by which a user can obtain sensitive information from a data source. From the results of the SLR, a taxonomy of inference strategies has been developed by analysing and grouping the inference strategies according to their processes (the stages required to complete the inference strategy).

Using the taxonomy, the processes for each group of inference strategy were compared to the processes in a service-based information broker. The comparison showed that the general processes performed by service-based information brokers, when obtaining disparate information, are very similar to the majority of the processes required to perform certain inference strategies. It is, therefore, not only important to prevent users from attempting to infer sensitive information, but imperative to implement measures to prevent service-based data integration tools, such as IBHIS, from releasing sensitive information inadvertently to its users. Without these measures, a system with the brokerage capabilities of IBHIS would pose a

considerable threat to the confidentiality and privacy requirements of its underlying data sources.

In order to address this problem we propose the Service-Based Inference Control (SBIC) framework, which is suitable for detecting and preventing the inadvertent release of sensitive information by information brokers such as IBHIS. The SBIC framework shows how the existing detection and prevention techniques can be combined with other components to avert this release of sensitive information.

References

1. Budgen, D., Turner, M., Kotsiopoulos, I., Zhu, F., Bennett, K., Brereton, P., Keane, J., Layzell, P., Russel, M., Rigby, M.: Managing Healthcare Information: The Role of the Broker. From Grid to Healthgrid: Proceedings of HealthGrid 2005, IOS Press, (2005) 3-16
2. Turner, M., Brereton, P., Budgen, D.: Service-enabled Access Control for Distributed Data. IEE Proceedings-Software, Vol. 153. No. 1 (2006) 24-30
3. Kitchenham, B.: Procedures for Undertaking Systematic Reviews. Technical Report, TR/SE-0401, School of Computing and Mathematics, Keele University, Keele. (2004)

YR-SOC 2007 is sponsored by: