

Supporting Context-Awareness in Adaptive Service Composition

Javier Cubo, Carlos Canal, and Ernesto Pimentel

Department of Computer Science, University of Málaga
Campus de Teatinos, 29071, Málaga, Spain
{cubo, canal, ernesto}@lcc.uma.es

Abstract. Context-awareness features involve that the adaptive service composition has to take dependencies of contexts into account. In any other case, such composition may reach inconsistent states. Therefore, we need to manage not only the explicit contexts given by the user as parameters, but also the implicit context information. We obtain such implicit contexts from the message of the user, making more independent our system, since the user does not have to provide such information. We propose to extend the STS model including those context-awareness features. In addition, we need to verify that the adaptive service composition is free of inconsistency problems by controlling the context changes depending on the conditions, user preferences, services capabilities or type of device used by the user.

Keywords: Context-Awareness, Adaptive Service, Components, Composition, Model-Based Methodologies, Pervasive Computing.

1 Introduction

Adaptive service composition is emerging as a technology that allows the integration between pre-existing software entities (components and Web services; in the sequel we use services to refer both components and Web services). These services are equipped with external interfaces giving information about their functionality, due to their black-box nature. However, it is frequent to find out interaction problems when we compose pre-existing services, so is required a process of adaptation in order to avoid these mismatching behaviours. They are due to that the interfaces of the constituent components of a system do not always fit one another.

Context-aware computing focus on building adaptive systems sensitive to their context (location, identity, time and activity) [10].

The increasing using of mobile and portable devices is giving rise to a new market of mobile and pervasive applications. Context-aware computing is incorporated into these systems, with the inclusion of certain context-awareness features. So they are different from traditional distributed applications, since a mobile system is able to change location allowing the communication via mobile devices with some limitations which need to be resolved (connectivity, bandwidth, local resources, etc.). Due to the contextual nature of these systems, it is important the improvement from the user experience, as well as to provide connectivity and services all the time, by being capable of adapting the composition with respect to changes in context. In such a way, the possible inconsistency situations generated in these systems could be avoided.

Our work aims to develop software adaptors capable of acting as a third-party component between the services to interoperate taking their contexts into account. We make use of a model-based methodology [35] to avoid mismatching behaviours, and we plan use validation techniques to solve inconsistencies or faults. Therefore, this paper focuses on extend the

concept of software adaptors [5, 7] by modeling explicitly context dependencies of already existing services, and using them in the component composition.

In case we reuse services without taking the contexts into account, then some inconsistent states will be achieved due to the contextual nature of the mobile and pervasive systems, and the service composition will not be correct. However, our model pretend to formalise the context-awareness features in these systems, which will allow us to detect and avoid mismatching behaviours, as well as faults or inconsistencies.

An overview of our proposal is presented in Figure 1. Firstly, different kinds of user, using different devices, request one or more services. We get (implicit) context information from the message. This info can be such as user language, type of device, user preferences, and so on. We should have this information without asking the user for it. In such a way, for instance, the user could travel with his/her own laptop and request services by obtaining the result in his/her language, currency, and so on, independently his/her location. Even the way to show the result could change depending on if the user is using a laptop or a cell-phone. A discovery process finds the pre-existing services (in our research work we use services implemented using WF or BPEL; see Section 2), and the service descriptions (AWF/ABPEL) are abstracted to obtain the behavioural service model (CA-STs is an extension, presented in Section 4.1, of Symbolic Transition Systems, STSs [18]). Next, being given a set of CA-STs, mismatch detection is computed to check whether the services involved need adaptation or not. If a mismatch exists, we apply adaptation techniques that aim at generating a CA-STs adaptor model from a contextual mapping. Validation techniques are then helpful to check that the adaptor is as expected by preventing inconsistencies or faults. If not, another mapping may be proposed. Finally, once the adaptor passes the test, the corresponding abstract adaptor service (AWF/ABPEL) is generated making use of a formal model. Last, the developer can complete this adaptor service (WF/BPEL) by including the abstracted information previously. So we generate an intermediate piece to make possible the contextual composition and adaptation of those services requested by the user.

The remainder of the paper is organised as follows. Firstly, in Section 2, is presented a background to adaptive service composition, context-aware computing and validation techniques, and is discussed about some related works. Section 3 briefly presents a motivating scenario of a rent-a-car system, by illustrating the need to generate a transformation model capable of adapting components and services requested in a context-aware application. In Section 4, we present the proposal introducing our behavioural model, as well as describing briefly the contextual composition. Finally, Section 5 concludes this paper with a summary and a discussion about the improvements getting using this new model in the context-aware applications, as well as a plan for future work.

2 Background

Most of the current systems are developed as a selection, adaptation and composition of pre-existing components and services rather than as a programming of applications from scratch. The use of *Commercial-Off-The-Shelf* (COTS) components and services is promoted by Component-Based Software Engineering (CBSE) [32] and Service-Oriented Architecture (SOA) [11, 22] respectively. They require a certain degree of adaptation in order to avoid mismatching behaviours, since the interfaces of the constituent components of a system do not always fit one another.

This adaptation is performed by adaptors which are automatically built from an abstract description (i.e. adaptation mapping) of how the mismatch can be solved with respect to the component behavioural interfaces. Software Adaptation (SA) [5] focuses on the automation of the adaptation process, by enabling components with mismatching behaviours to inter-operate. It supports run-time features that occur as devices and applications move from

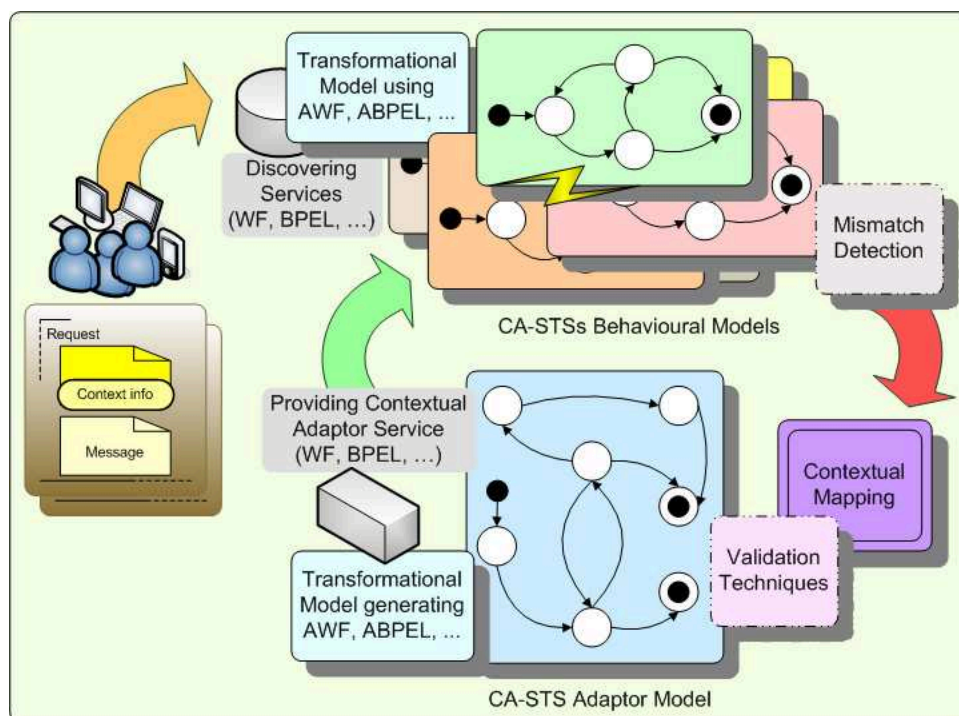


Fig. 1. Context-Aware Adaptive Service Composition Model

network to network, modifying or extending their behaviour, and enhancing the flexibility and maintainability of systems.

Mismatch among services can be presented at four different levels: signature, behavioural, service and semantic levels [8]. The most common mismatches occur at the behavioural level. It is due to an incompatibility in the order of the exchanged messages among services, which can lead to deadlock situations. But industrial platforms only deal suitably with the signature level (e.g. CORBA's IDL [28]). In addition, the nature of pervasive systems produces inconsistent situations, which must be dealt.

Our proposal focuses on the behavioural interoperability level, extending interfaces with a description of their protocol and dealing with the different compositional issues between them taking context-dependencies into account. Windows Workflow Foundation (WF) [30] which belongs to .NET Framework 3.5 and is supported by Visual Studio 2008, as well as BPEL [1] and Netbeans Enterprise, are using to relate our transformation contextual model with realistic and complex examples. Their workflow-based graphical support and the automation of the code generation make the implementation of services easier. Besides, they have capabilities for developing workflows in different scenarios, from simple sequential workflows to realistic and complex workflows with human interactions. It has been demonstrated that services implemented on existing platforms can be modeled using our behavioral formal model [13, 14].

There exist different approaches tackling model-based adaptation at the behavioural interoperability level, and aim at generating adaptors which are used to solve mismatch in a non-intrusive way. However, only a few of them have been carried out using some particular platform [2, 4, 17, 19, 25].

Bastide et al. [2] show that the OMG's style to specify services in CORBA does not guarantee interoperable and substitutable implementations. Gaspari and Zavattaro [17] illustrate how CORBA models for requesting invocation can be mapped into a message passing architecture using the base of a process algebra.

The work presented by Brogi and Popescu [4] outlines a methodology for the automated generation of adaptors capable of solving behavioural mismatches between BPEL processes. They use YAWL workflow [33] as an intermediate language, and once the adaptor workflow is generated, they use lock analysis techniques to check if a full adaptor has been generated or only a partial one (some interaction scenarios cannot be resolved). Our adaptation approach may reorder messages in between services when required [8]. This ability is needed to ensure a correct interaction when communicating entities have messages which are not ordered as required.

Inverardi and Tivoli [19] tackle the automatic synthesis of connectors in COM/DCOM environments, by guaranteeing deadlock-free interactions among components. Compared to this proposal, we may match different name messages using our correspondences, which is very useful within context-aware systems. In addition, this approach does not use any mapping language for the adaptor specification, so restrict the adaptor to possible non-deadlocking.

Finally, Benatallah et al. [25] present techniques, as well as a tool providing semi-automated support for identification and resolution of mismatches between service interfaces and protocols, and generate adaptation behavioural specifications. They deal with data, and base on SCA architecture implementing the services and the generated adapter as SCA components.

In general, respect to the aforementioned works, we focus on modeling the behavioural composition, not only preventing mismatch problems, but also taking into account dependencies of contexts in order to control states of inconsistency.

On the other hand, recently, some works are researching about context-aware computing, as well as built pervasive applications [24, 26, 27] to demonstrate the usefulness of this technology. There have even been significant achievements in the architectural support of context-aware applications [6, 29]. However, at the behavioural level, the composition and adaptation of software entities within the context-aware pervasive systems has only briefly been dealt within some of these works.

We focus on a formal model that transforms services (implemented in specific platforms) in a standard notation (CA-STS) to be able to compose and adapt them when required. In this sense, we use the Software Adaptation approach, which reuses services and tackles the interoperability issues which exist at the different levels of component interaction.

Related to our motivating scenario (presented in Section 3), in [31] the authors have designed and implemented a Tourist Guide system that is mobile and context sensitive. However, they focus on the design and usability issues, and not about service composition.

Last, different works propose validation techniques in the service composition field [12, 15, 16, 36]. Even, some of them use semantic ontology languages to compose the services. However, any of them tackle the context-aware composition.

Kim and Choi [20] suggest a context infrastructure to provide semantic interoperability in a ubiquitous computing environment. They only evaluate the performance of the context infrastructure, not controlling possible inconsistent situations nor faults. With our proposal we are planning to solve these problems by providing context dependencies.

A model to compose services and validate their correctness using Petri nets is presented by Luo et al. [21]. They check behavioral properties based on simulation of the model. Compared to our idea, we model the service composition with a strategy by means of CA-STS that may be easily validated. In addition, their model currently doesn't control the changes of contexts at run-time.

Vukovic presents in [34] a complete approach focus on the recomposition of the composite service during its execution, according to changes in the context. This proposal provides a failure-tolerant solution. But it has still some limitations about the control of user preferences, as well as the control of independent requests, which we pretend to support using our model.

3 Motivation

This section introduces a scenario illustrating realistic and complex problems used in the real-world by end-users, whose purpose is to give information requested by the client by taking into account context information likely changing without intervention of the user. These daily life applications may give rise to inconsistent behaviours. So these problems have motivated our research in this field and in this work we propose a model (Section 4) to solve problems emerged in those systems.

We present a rent-a-car system that consists of services giving information to the user about different cars to be rented, depending on the user preferences, as well as the conditions of the environment where the user is requesting the services, and the capabilities of the services and device used by the user. These conditions will make the system contexts change, so our model should be able to control possible inconsistency problems generated by the wrong interpretations of the contexts.

Multiple contextual information is used in real world services to improve their features. In our simple scenario, a user is requesting information about renting a car and the services have to give the response by using explicit and implicit contextual information. Ideally a response should include details of car rental companies closest to the user and use the user preferences, not only using the explicit parameters given by the user (model of car, renting duration or authentication), but also capturing implicit context information (type of device, location, language or currency) to obtain explicit data through it and know when the composition has to change.

Therefore, our goal consist of modeling these systems to make easier the interaction between user and services, but maintaining the features of the context-aware systems. So a user could request a service giving only some common parameters called explicit context information. Thus, the services will have to control the implicit context information using an adaptor service modeled in CA-STS. Our previous proposal [7] tackles only explicit context information given by parameters, but here we pretend to infer implicit context dependencies from requesting message of the user.

Real services of pervasive applications can be aware of some context information, but in addition, they can support this information in different ways. So we need to extend the notion of a third-party connector with context-awareness features. In such a way our contextual adaptor could give the most suitable and consistent responses depending on both context and possibilities of interconnected services.

Returning to our scenario, between the services that constitutes it, we can find a car description service composed by an explanation about each car, as well as a picture and the price for each one. In addition, there exists a car rental location service, which indicates where is the nearest car rental center, by means of a street plan or simply textual indication according to the device used for the requesting.

If the user is traveling and requests a car in a country different to his/her country, we should control possible inconsistency problems due to different ways of interpretation of the context information. Each service should use the context information to generate a suitable response according to theirs features.

For instance, the car description service will use the language and currency of the user to generate a response, and the location service will use the user location information to

localise the nearest center. But if the user language, captured as a implicit context, is not supported by our description service, then this service will have to return the explanation in a default language defined. In addition, this service should show the price using the currency corresponding to the user's country. The currency forms part of the implicit context information. On the other hand, the location service will respond by using the language of the user's location, which is different to the default one and to the language of the country where is being requested the service. This service is going to show the result in different ways depending on the device. So at the same time in the service composition is needed to take into account different values related to a context, depending on the conditions, the user preferences, and the features of the services and device.

In Section 4, we present briefly our model where we add context-awareness features to the STS model trying to solve the problems expose here, and sketch the steps to perform in the adaptive service composition.

4 Model based on Context-Awareness

Here we describe our approach to compose services depending on the explicit and implicit contexts. Services are equipped with external interfaces (signature and behaviour), giving information about their functionality. We represent the behaviour of the services by means of a *Symbolic Transition System* (STSs) [18], which provides an expressive and graphical notation to specify adaptation policies. Any formalism to describe dynamic contextual behaviour may be expressed in terms of an standard notation like STS.

We propose an extension of STS, called *Context-Aware Symbolic Transition System* (CA-STS), to support context-awareness features.

4.1 Adding Context-Awareness to STSs

A *Context-Aware Symbolic Transition System* (CA-STS) is a tuple (A, S, I, F, C, T) where: A is an alphabet (set of messages), $S \subseteq Id$ is a set of states (Id stands for a set of identifiers), $I \in S$ is the initial state, $F \subseteq S$ are final states, C is a set of conditions depending on the context, and $T \subseteq S \times A \times S$ is the transition function.

In our model, a *label* corresponding to a transition of a CA-STS, is either the internal action (τ) or a tuple (SI, M, C, D, PL, CL) where: SI is a service identifier, M is the message name, C are the conditions of the message, D stands for the direction of messages ($!$ and $?$ represent emission and reception respectively), PL is either a list of data terms for emissions, or a list of variables for receptions, both of them given explicitly (it can include explicit contexts), and CL is a list of implicit contexts obtained from the message, and which will be had as information making such contexts explicit.

Events relative to the emission and reception of messages represent the service composition. They may come with a set of data terms whose types respect the operation signatures, as well as conditions depending on which context the system is in.

We assume the services of our motivating scenario presented in Section 3 have been implemented on existing platforms, such as WF or BPEL, and apply our model on these services by obtaining the corresponding CA-STSs. In Figure 2 we depict our scenario to a general list of parameters, conditions and implicit contexts, and receiving as result the description of the cars (*description*) as well as the location of the nearest center to rent some of those cars (*location*).

Specifically, *params* indicates the parameters as well as the explicit contexts; *context_info* is the implicit context obtained from the message; *conditions* are the conditions of the message; and *context_params* are the contextual parameters returned as result depending on the conditions, the user preferences (context information) and the capabilities of the

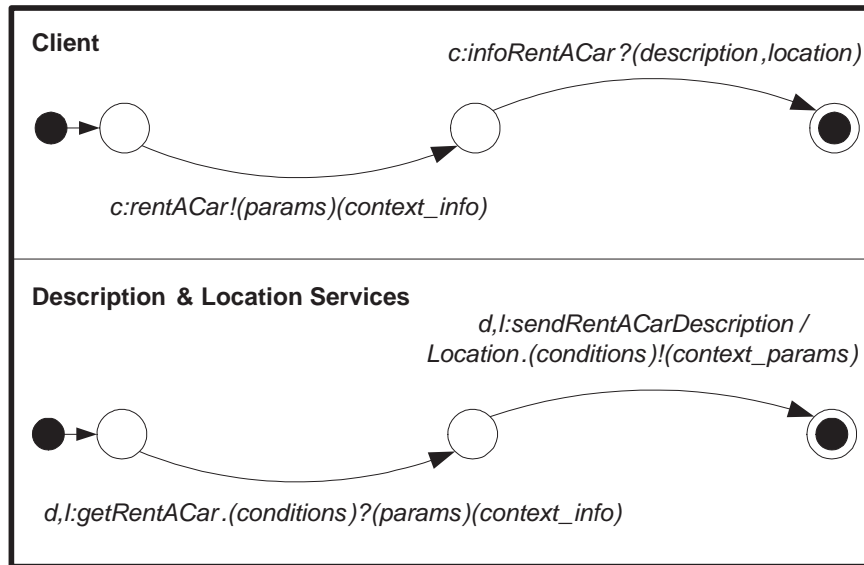


Fig. 2. CA-STS for the motivating example of renting a car

services and device used by the user. Note we only use one CA-STS representation to show both description and location services, since we are representing our scenario in a general case without going into any further details respect to the parameters, conditions or implicit context information for each one.

4.2 Context-Aware Service Composition

This section describes briefly the process of our approach to define correspondences in between service messages of a system taking contextual information into account. To complete our approach, we explain the generation of an adaptor protocol which will work as an intermediate piece connecting services.

When there exist services which cannot be directly used together due to mismatching problems, a mapping or contract have to be given to work the cases of mismatch out. Our contract is called *contextual mapping*, and it is made up of several constituents as is explained in [8], besides of including the implicit context information in the mapping.

We generate a CA-STS adaptor model from the descriptions of service interfaces (CA-STSs behavioural models) and the contract. The adaptor acts as a third-party component that is in charge of coordinating all the services involved in the system with respect to a set of correspondences defined in the contract. To obtain the adaptor we will use an extension with value-passing protocols of the algorithm presented in [23], and using this new model we need to control the implicit contexts.

Once we have generated the adaptor model (CA-STS adaptor model), we need to apply validation techniques. We are working on checking our composition model to prove the usefulness of supporting context dependencies in adaptive service composition. As we have already mentioned, the incorporation of context-awareness features gives rise to inconsistency situations. These situations have to be controlled by performing validation respect to properties such as determinism, deadlock or liveness.

Finally, if our adaptor model is validated correctly, then we have to generate an abstract adaptor following the guidelines presented in [9]. Otherwise, we will need to generate other adaptor model, using a different contextual mapping. The final abstract adaptor has to be refined, by requiring intervention of the designer, to add the information that was initially abstracted.

5 Conclusion and Future Work

This work presents a proposal to model adaptive service composition taking into account context-awareness features. The main improvement respect to our previous work [7] consists of inferring the implicit context dependencies from the user's requesting message. To do that we need to extend the STS model making use of such implicit context information.

We have described the problem of modeling composition of context-aware pervasive systems without taking the context-dependencies into account. Thus, in such a case, the service composition of those systems may reach inconsistent states. So we need to detect not only mismatching problems, but also the inconsistency situations or faults.

Once we have defined our transformation model, we need to validate the composition with respect to a set of properties such as determinism, deadlock or liveness. We plan to apply validation and assessment techniques, which it is needed to evaluate our proposal.

In addition, we pretend to extend our model by incorporating contextual ontologies [3], so that we will get to enrich the semantic of our model.

Acknowledgements

We would like to acknowledge to our colleagues of University College London, with whom we are working on validating the model here presented, since in our discussions we have obtained some ideas to improve our proposal. This paper has been partially supported by project TIN2007-67134 funded by the Spanish Ministry of Innovation and Science, and project P06-TIC-02250 funded by Junta de Andalucía.

References

1. T. Andrews et al. *Business Process Execution Language for Web Services (WSBPEL)*. BEA Systems, IBM, Microsoft, SAP AG, and Siebel Systems, 2005.
2. R. Bastide, O. Sy, D. Navarre, and P. A. Palanque. A Formal Specification of the CORBA Event Service. In *Proc. of FMOODS'00*, pages 371–396. Kluwer, 2000.
3. P. Bouquet, F. Giunchiglia, F. van Harmelen, L. Serafini, and H. Stuckenschmidt. Contextualizing Ontologies. *Journal of Web Semantics*, 1(4):325–343, 2004.
4. A. Brogi and R. Popescu. Automated Generation of BPEL Adapters. In *Proc. of ICSOC'06*, volume 4294 of *LNCS*, pages 27–39, 2006.
5. C. Canal, J.M. Murillo, and P. Poizat. Software Adaptation. *L'Objet*, 12(1):9–31, 2006. Special Issue on Coordination and Adaptation Techniques for Software Entities.
6. H. Chen, T. Finin, and A. Joshi. An Intelligent Broker for Context-Aware Systems. In *Proc. of UbiComp'03*, 2003.
7. J. Cubo, C. Canal, and E. Pimentel. Towards a Model-Based Approach for Context-Aware Composition and Adaptation: A Case Study using WF/.NET. In *Proc. of MOMPES'08*, pages 3–13. IEEE Computer Society, 2008.
8. J. Cubo, G. Salaün, J. Cámara, C. Canal, and E. Pimentel. Context-Based Adaptation of Component Behavioural Interfaces. In *Proc. of COORDINATION'07*, volume 4467 of *LNCS*, pages 305–323, 2007.

9. J. Cubo, G. Salaün, C. Canal, E. Pimentel, and P. Poizat. A Model-Based Approach to the Verification and Adaptation of WF/.NET Components. In *Proc. of FACS'07, ENTCS*, 2007. To appear.
10. A.K. Dey and G.D. Abowd. Towards a Better Understanding of Context and Context-Awareness. In *Proc. of Workshop on the What, Who, Where, When and How of Context-Awareness 2000*, pages 304–307, 2000.
11. T. Erl. *Service-Oriented Architecture (SOA): Concepts, Technology, and Design*. Prentice Hall, 2005.
12. H. Foster, W. Emmerich, J. Kramer, J. Magee, D. Rosenblum, and S. Uchitel. Model Checking Service Compositions under Resource Constraints. In *Proc. of ESEC/FSE'07*, pages 3–7. ACM Press, 2007.
13. H. Foster, S. Uchitel, and J. Kramer. LTSA-WS: A Tool for Model-based Verification of Web Service Compositions and Choreography. In *Proc. of ICSE'06*, pages 771–774. ACM Press, 2006.
14. X. Fu, T. Bultan, and J. Su. Analysis of Interacting BPEL Web Services. In *Proc. of WWW'04*, pages 621–630. ACM Press, 2004.
15. C. Gao, R. Liu, Y. Song, and H. Chen. A Model Checking Tool Embedded into Services Composition Environment. In *Proc. of GCC'06*, pages 355–362. IEEE Computer Society, 2006.
16. J. Garcia-Fanjul, C. de la Riva, and J. Tuya. Generation of Conformance Test Suites for Compositions of Web Services Using Model Checking. In *Proc. of TAIC-PART'08*, pages 127–130. IEEE Computer Society, 2006.
17. M. Gaspari and G. Zavattaro. A Process Algebraic Specification of the New Asynchronous CORBA Messaging Service. In *Proc. of ECOOP'99*, volume 1628 of *LNCS*, pages 495–518, 1999.
18. A. Ingólfssdóttir and H. Lin. *A Symbolic Approach to Value-passing Processes*. Handbook of Process Algebra, Elsevier, 2001.
19. P. Inverardi and M. Tivoli. Deadlock-Free Software Architectures for COM/DCOM Applications. *Journal of Systems and Software*, 65(3):173–183, 2003.
20. E. Kim and J. Choi. A Semantic Interoperable Context Infrastructure using Web Services. In *Proc. of ICCSA'07*, volume 4706 of *LNCS*, pages 839–848, 2007.
21. N. Luo, J. Yan, M. Liu, and S. Yang. Towards Context-Aware Composition of Web Services. In *Proc. of GCC'06*, pages 494–499. IEEE Computer Society, 2006.
22. B. Margolis. *SOA for the Business Developer: Concepts, BPEL, and SCA*. Mc Press, 2007.
23. R. Mateescu, P. Poizat, and Gwen Salaün. Behavioral Adaptation of Component Compositions Based on Process Algebra Encodings. In *Proc. of ASE'07*, IEEE Computer Society Press, 2007. To appear.
24. S.B. Mokhtar, D. Fournier, N. Georgantas, and V. Issarny. Context-Aware Service Composition in Pervasive Computing Environments. In *Proc. of RISE'05*, volume 3943 of *LNCS*, pages 129–144, 2006.
25. H. Motahari, B. Benatallah, A. Martens, F. Curbera, and F. Casati. Semi-Automated Adaptation of Service Interactions. In *Proc. of WWW'07*, pages 993–1002. ACM Press, 2007.
26. J. Muñoz, V. Pelechano, and J. Fons. Model Driven Development of Pervasive Systems. *ERICIM News*, 58(0):50–51, 2004.
27. A. Nicoara and G. Alonso. PROSE - A Middleware Platform for Dynamic Adaptation. *Demo presented at AOSD'05*, 2005.
28. OMG. *CORBA Component Model Specification, v. 4.0*. www.omg.org/technology/documents/formal/components.htm, Object Management Group, 2006.
29. D. Salber, A.K. Dey, and G.D. Abowd. The Context Toolkit: Aiding the Development of Context-Enabled Applications. In *Proc. of CHI'99*, pages 434–441. ACM, 1999.
30. K. Scribner. *Microsoft Windows Workflow Foundation: Step by Step*. Microsoft Press, 2007.
31. T. Simcock, S. Peter Hillenbrand, and Bruce H. Thomas. Developing a location based tourist guide application. In *ACSW Frontiers'03*, pages 177–183. Australian Computer Society, Inc., 2003.
32. C. Szyperski. *Component Software: Beyond Object-Oriented Programming*. Addison-Wesley, 2nd edition, 2003.

33. W.M.P. van der Aalst and A.H.M. ter Hofstede. YAWL: Yet Another Workflow Language. *Information Systems*, 30(4):245–275, 2005.
34. M. Vukovic. Context Aware Service Composition. Technical Report UCAM-CL-TR-700, University of Cambridge, 2007.
35. www.omg.org/docs/omg/03-06-01.pdf. *Object Management Group, MDA Guide version 1.0.1*, 2003.
36. H. Q. Yu and S. Reiff-Marganiec. Semantic Web Services Composition via Planning as Model Checking. Technical Report CS-06-003, University of Leicester, 2006.