Electronic Communications of the EASST
Volume 27 (2010)

Workshop über Selbstorganisierende, adaptive, kontextsensitive verteilte Systeme (SAKS 2010)

Evaluation of the Technology Agnostic Service Creation Approach

Sian Lun Lau, Niklas Klein, Andreas Pirali, Olaf Droegehorn and Klaus David

12 pages

Guest Editors: Klaus David, Michael Zapf
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Evaluation of the Technology Agnostic Service Creation Approach

Sian Lun Lau, Niklas Klein, Andreas Pirali, Olaf Droegehorn and Klaus David

comtec@uni-kassel.de, http://www.comtec.eecs.uni-kassel.de/
Chair for Communication Technology (ComTec),
University of Kassel, Kassel

Abstract: The current computing and communication services provide convenience and functionality. The creation of these services is however not an easy task. Service development is still mainly technical oriented, where service creation tools are meant for serving and assisting the professional developers. In other words, service creation is not seen as a task for end-users. In our research, we wish to enable service creation for the end-users. This is achieved by introducing the technology agnostic approach into the process. In this paper, we present the conceptual architecture as the proposed solution. Prototype tools were designed and implemented as proof of concept. An evaluation event was held to obtain user feedback on the approach and the prototype tools. This paper presents and discusses the outcome of the evaluation.

Keywords: service creation, technology agnostic, semantic service description, end-users

1 Introduction

Services in the computing and communication technologies play an important role in our daily lives. The services simplify tasks and automate routines, so that we can enjoy the convenience brought. Currently, most services are created by professional developers, who have the necessary technical knowledge. This is due to the fact that service creation is a resource demanding and complex process. Different factors contribute to this, such as the heterogeneity of devices, availability of software platforms on different devices and communication interoperability.

For a person who does not posses the necessary technical know-how, for instance an end-user, it can be rather difficult to create services he needs. In today's service creation cycle, an end-user is only seen as a service consumer. He should only use the services that are provided to him. If the end-user needs new services, he is not able to create them by himself unless he acquires the needed service creation skills, such as programming.

In our research, we have the vision to enable service creation for everyone. Our objective is to remove the technical barriers that might have hindered non-technical users from taking part in the service creation process. The users are the one who know their needs best. Users can make own decisions in creating, obtaining and using their desired services, when the appropriate service creation tools are available. At the same time, the developers can concentrate more on functionality development.

Based on the above vision we have proposed the Technology Agnostic Service Creation (TASC) architecture [LKP+08]. Based on the architecture prototype tools were implemented as

proof of concept. An evaluation was carried out to investigate how the approach can be applied in the designated environment. We wish to obtain feedback that can help to improve the approach and to allow real usage of the TASC approach.

This paper gives a brief overview of the TASC approach, and presents a report on the evaluation of our concepts. The structure of the paper is as follows: Section 2 presents a brief summary of related work in the area of service creation. Section 3 gives an overview on the TASC approach. The evaluation methodology is elaborated in Section 4, and is followed by the evaluation results, feedback and suggestions summary. Section 7 will conclude the paper.

2 State of the art

When it comes to service creation, most people will naturally categorize the tasks of service creation as the developers' responsibilities. This includes using programming and development tools (such as compilers, interpreters and IDEs) to create services with a certain programming language. In order to simplify and to open up service creation tasks to potential users other than professional developers, different technologies and approaches have been introduced.

The idea of reusable software component modules [Szy97] [WS01] can potentially be a good idea to reduce technical complexity of the service creation process. The created modules can be packaged to be applied in a designated service creation tools. One popular approach is the graphical development tool. These tools translate graphical diagrams into programming languages. For instance, the development tool Fujaba [fuj] can generate Java source code from the produced system design that consists of Unified Modelling Language (UML) class diagrams and UML behaviour diagrams. Such approaches simplify the development process, and are able to visually represent the service logic and workflow. But they are still considerably technically demanding. The UML classes and behaviour representations are direct representations of source code level information. These tools are actually enhanced development tools for developers. They enable developers to design services in modelling languages. This can help them to refine the designed service. They are not intended to enable service creation for end-users.

In the field of End-User Programming, there are also efforts in providing end-users the means to create services without in-depth knowledge of a specific programming language. Tools such as the AgentSheet [Rep93], iCAP [SD03] and Topiary [LHL04] provide developers the possibility to create applications without code writing. These tools focus on simulation and prototypes for the designated environment, hence do not provide the solution in areas like service deployment, life cycle management and semantic service description. However, the graphical interface approach used is in line with our idea.

With the emergence of web services and service oriented architecture, there are also other service creation examples. Reusable service components can be created as open and standardized elements. With protocols such as SOAP and UDDI, one can search, locate and use desired components in a given environment. Several works have been carried out to investigate how easy service creation and composition can be enabled. Among them are JOpera [PHA06] and Pilote [AS01]. Most of them provide the possibilities to model the workflow and process using BPEL visually and allow execution via an underlying BPEL engine. These approaches demonstrated how one can use visual tools to compose services with available web services in a given

environment. But users are still required to know and to understand the web services and their features directly. The above approaches focus on visual BPEL workflow representation. Users are not aided directly in expressing their service needs and desire semantically. They will still need a sufficient amount of technical know-how in order to perform service creation tasks.

Since the start of Web 2.0 services, the concept of mash-ups was introduced. The idea is to aggregate and delegate data from different web services to create new services. In the meantime, there are mash-up web editors that assist the mash-ups creation process, such as Yahoo! Pipes [1]. These tools enable the creation of web-data processing and presentation applications. From our observation, the mash-up concepts are comparatively limited because they are more towards content aggregation and manipulation. There is also no semantic-based approach to locate and to use potential Web 2.0 services.

The investigation of the various approaches above shows that the direction of technology agnostic approach has its potential in bringing service creation to common users. The migration from manual coding to visually aided composition indicates simplification without sacrificing functionality. However, most of the approaches are still technically demanding and are complex for people who have no programming knowledge. There are also no solutions where users can freely express their service needs without directly understanding the provided components or building blocks.

3 Technology Agnostic Service Creation (TASC) approach

From our observation, the main obstacles in enabling service creation for the end-users are the complexity and technical issues. Without sufficient know-how, they are not able to create services that can be used to answer to their daily needs. The TASC approach provide service creation tools that enable the end-users to create services using non-technical expressions. If the supporting service creation platform is capable of processing these expressions, it can trigger and execute the needed functionality. In this section, we presents a brief overview of the TASC approach. Details of this approach can be found in [LKP+08].

In the TASC approach, services are defined as a composition of Service Building Blocks (SBB). The concept of SBB is similar to the concept of enablers, specified by the Open Mobile Alliance (OMA) [All]. Each SBB provides a specific function. Therefore, if we can provide a possibility for an end-user to compose a desired service by putting the needed SBB together in a non-technical way, this will be the answer to the question above. Contrary to today's service creation approaches, the professional developers are responsible for the SBB development. The end-users are responsible to create services according to their own needs.

We use a simple scenario to explain our concept. An end-user wants to publish his location on the micro-blogging site Twitter[2]. If he is at home, the service should post "At home". If he reaches his office, the service will update his Twitter page with "At work". The challenge is: how can the end-user create a service that performs the above tasks, without any programming knowledge or the need to acquire technical know-how?

We assume that the end-users can express their service needs in a non-technical way. They

[1] http://pipes.yahoo.com/pipes/, last visited on 10th January 2010
[2] http://www.twitter.com/, last visited on 10th January 2010

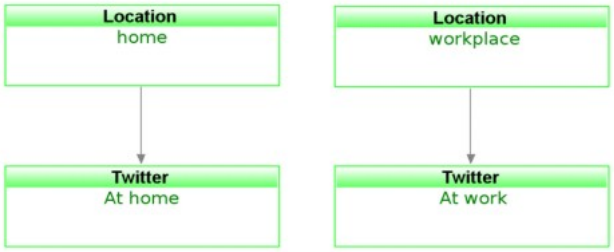

Figure 1: The visualization of a service using SBBs

can use terms and logical relationships to describe the nature of the desired services. Hence the Visual Service Editor (VSE) was developed as a prototype tool that simplifies this process. End-users can use the graphical blocks and expressions to "draw-a-service" (Fig. 1). Most of the time, we foresee the expressions can be described using the "If...Else..." relationship. Therefore we chose RuleML [rul] as the representation language for the "drawn" services.

As illustrated in the Fig. 1, the desired service mentioned in the scenario above can be created using 4 SBBs. Two different SBB categories were defined in our approach. Firstly, we have the State SBBs, which are used as context providers. Secondly, we have the Action SBBs, which triggers a desired action. The State SBBs provide information updates that are used as conditions to trigger Action SBBs. In our scenario the location SBB is an example of State SBB, whereas the Twitter SBB is an Action SBB.

Besides the VSE, we have also proposed another tool that aims to simplify the development of SBBs for a given Service Execution Environment (SEE). We observed that for different SEEs there are usually specific requirements and development routines to follow. These processes are usually recurring, and can be automated or assisted. The prototype plug-in we have developed is targeted on our SEE - the Wireless-Internet Platform (WI-P) [WDD04]. By using the plug-in, WI-P specific interfaces are created automatically upon creation of a new SBB development project. Tasks such as packaging, deployment and service description generation are also assisted with wizards and menus. The purpose of this tool is to reduce the workload of developers in creating SBBs for a given SEE. In this way they can focus on the SBB functionality development and let the plug-in to handle the recurring processes. One can even reuse the created codes to develop SBBs for multiple SEEs, when corresponding plug-ins are available. The plug-in for professional developers is found on Layer 1.

The realisation of the TASC depends on the Semantic SBB Discovery and Description layer (Layer 2). This layer is responsible in translating and linking the technology agnostic expressions to the designated SBBs. The use of a semantic SBB description enables the functionality to be described in a meaningful high-level sense. The Semantic SBB Discovery searches for the best-matched SBBs so that the SEE can execute them at runtime. Currently we use only simple keywords and domain terms to describe SBBs.

In the SEE we use the Business Rule Evaluator (BRE) component to search and trigger SBBs. It uses a rule engine to evaluate the produced service in RuleML. The component first locates the corresponding State SBBs to gather necessary information for the rule evaluation. The outcome

of the rule evaluation is used to trigger the designated Action SBBs. As an example from the scenario, the BRE queries a SBB that provides location information. If the location SBB returns "home", the BRE will then request the Twitter SBB to post the message "At home".

4 Evaluation event

There were altogether 25 participants in the evaluation event. The event was divided into three parts. The participants are from diversified backgrounds. There are some developers among them. They provide evaluation from a developer's point of view for potential end-users. We also had some participants from the business sector. They are part of our targeted audience - They can be potentially the user group who wishes to create services for their needs without the requirement of technical know-how.

In the first part of the event, participants were presented with the core concepts of the service creation approaches, such as an introduction to the TASC approach, semantic service description and discovery. The goal was to allow the participants to obtain the necessary background knowledge. The prototype service creation tools were introduced next. The tools were placed at different stations, and the participants were invited to test them in three groups. Moderators were assigned to ensure that the participants understood the functions and features of the tools, and to guide them in using these tools. The participants were encouraged to ask questions in order to provide them better understanding on the realisation of the TASC approach.

We selected some scenarios and the participants were requested to realise these scenarios using the prototype tools. For example, they were expected to create a SBB using the IDE Plug-in that performs a Twitter update. With the created SBB, the participants then used the VSE to compose a service that sends a message via the Twitter SBB upon location change. When service composition was completed, the participants tested the deployment process to see whether the service was working as expected.

The last part of the event was the evaluation. Instead of using the conventional questionnaire approach, meta-planning [Wil80] was used to provide a more intuitive and efficient evaluation. Before the event intended evaluation areas were first identified and selected. The evaluation questions were then generated in forms of charts and boxes. The participants answered them on flip charts, where answers were given in forms of voting, drawings or writings. Facilitators led and guided this process. This approach allows us to have discussions during the answering session. This enabled a two-way communication to sum up the evaluation opinions and critics.

The evaluation ended with the summary and discussion of the evaluation results. The interaction helped not only us to understand the comments and feedback from the participants, but also clarified some uncertainties concerning our approaches. Besides that, the participants also discussed and presented some suggestions for the next steps and further potential research focuses. These evaluation results will be presented in the coming sections.

5 Evaluation questions and results

This section will present the evaluation results. Each subsection consists of a question put forward to the participants and a summary of the respective answers and feedback. The first two

questions focused on the prototype tools, while the rest of the questions intended to investigate the general ideas of the service creation approach. The questions were presented in chart or table form, where participants can freely provide their answers accordingly. In the following sub-sections the questions and answers will be elaborated. The colours of the provided feedback indicate answers from the three different groups of participants.

5.1 Q1: How do you rate the usability of the Visual Service Editor?

The participants were expected to write their answers in the given table. The usability evaluation was divided in four categories: Strength, Weaknesses, Opportunities and Threats. As seen in

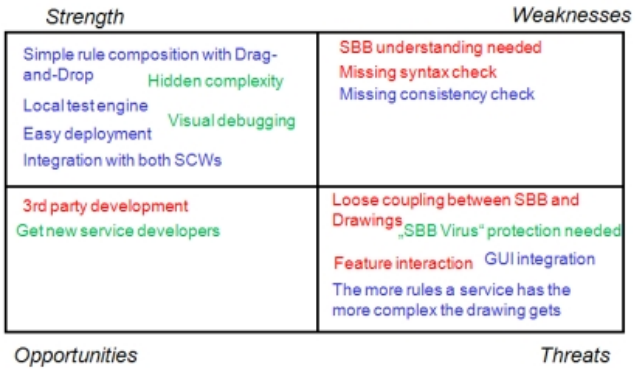

Figure 2: Evaluation results on the VSE usability

Fig. 2, there was mixed feedback concerning the usability of the VSE. Some agreed that such a graphical tool can provide simple rule composition and hide the complexity. The abilities to deploy, test and debug composed services were also noted as strength. This was observed from the rule-based service execution and triggering, since composed services can be tested immediately. The participants also mentioned about opportunities in SBBs development, where third party development and new service developers can be encouraged and increased through the proposed approaches.

The usability evaluation revealed some potential issues. The graphical approach currently has a lack of syntax and consistency check for SBBs. This was seen as a crucial issue if the amount and complexity of SBBs increase. At the same time, several possible threats were also highlighted by the participants. Among them were potential security threats, drawing complexity and loose coupling between SBBs and the drawn services.

5.2 Q2: What percentage of workload do you expect to save through the usage of the plug-in?

As presented in Fig. 3, the participants were requested to mark on the given chart to indicate the amount of workload saving estimation. The maximum savings were indicated at around 70-80%. A group did indicate that the user interface and plug-in concept each saves up to 30-35%.

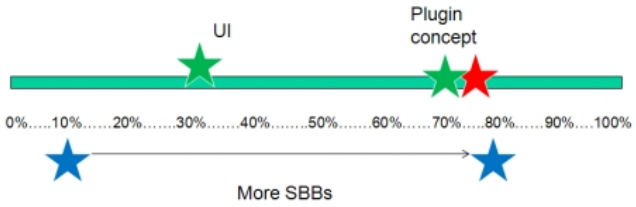

Figure 3: Evaluation results on the workload savings

Another group gave a variable from 10-80%. Because they foresee that the saving percentage will increase as more SBBs are developed.

Generally the plug-in approach was well received. This showed that the participants welcome the idea of resource saving tools that can simplify and accelerate the development process.

5.3 Q3: Accessing the "Rule-Idea"

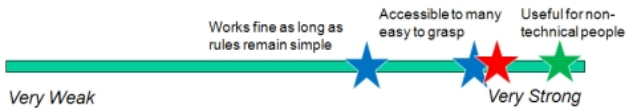

Figure 4: Evaluation results on the viability of the "draw-a-service" concept

Figure 5: Evaluation result on the acceptance of the rule approach

There were two parts in this question. The first part was to evaluate the viability of the 'draw-a-service' concept (Fig. 4). Most participants rated the concept as strong to very strong. It was seen as a useful approach especially for non-technical people. The comment on keeping rules simple was also valuable, since it revealed the usability issue in the concept. The participants did presume that the current idea might not work as expected, when rules become more complex. This gave us an important pointer to be looked at in the future.

The second part asked for the opinion on whether the 'rule representation' approach is inline with the participants' idea of processing software creation tasks (Fig. 5). We received a rather mixed result. Two groups showed acceptance of the rule representation approach, since they could foresee how the approach would fit into their service creation tasks. A remark was made, stating that the approach requires rethinking by service developers. A group gave a neutral feedback (group blue). This was due to the fact where the group saw the approach as hard to debug. This point was similar to the "Weaknesses" part in Q1 (Fig. 2).

5.4 Q4: How do you assess the general definition of service in the TASC approach?

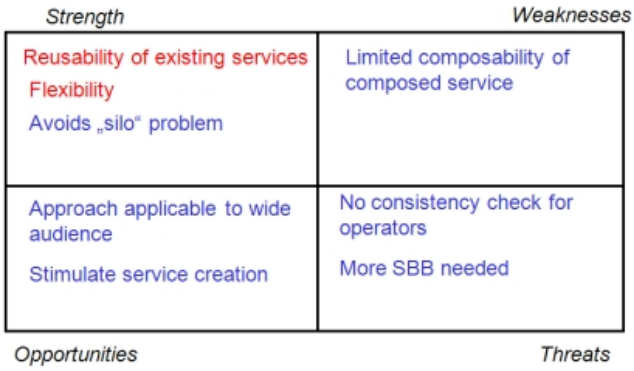

Figure 6: Evaluation result on the proposed general definition of service

The definition of service in our approach brought also some mixed feedback. The strength of the idea was the clear definition of SBBs and their relationships to services. The participants saw the benefits of reusability and flexibility for services. With the approach, the service creation tasks can be provided to wider audience and will stimulate the service creation cycle.

On the other hand, some saw weaknesses in limited ability in reusing composed services. They proposed services to be re-used in a simpler representation. At the same time some also commented that more SBBs are needed in helping the success of the approach. The lack of consistency check for operators in defined service could also be a possible threat to the idea.

5.5 Q5: Which are the most rivalling ideas with regard to the TASC?

Table 1: Evaluation result on the rivalling ideas

General approaches	Projects	Products (academic + market)
MDA	MAMS	Yahoo! Pipes
	SPICE	Google Mashup
	OPUCE	
	SMS	
	LOMS	

The summarized result was as illustrated in Table 1. Model-Driven Architecture (MDA) [Gro] was mentioned as a rivalling idea. However, MDA is a formalisation of a technical development process. It is not intended for end-users and they will not be able to understand the complexity of the different modelling layers in MDA. MDA can generate code out of graphical models but it is not intended to run this code without further looking in the code.

At the second column, the enlisted projects were known to us. These projects have direct and indirect focus on the issues of service creation. However they do not tackle issues on end-user service creation. Projects such as SPICE [3] and SMS [4] deal mainly on supporting telcos in providing user-centric services. Some of the proposed service creation tools still require certain level of technical knowledge, such as data flow definition. In our opinion such approaches might be still complex for end-users. The TASC concept can be seen as a complementing approach to the solutions from these projects. Yahoo! Pipes [5] and Google Mashup [6] are some prominent Web 2.0 [O'R05] service creation products. However the Web 2.0 services focus more on content aggregation and manipulation, and do not provide semantic based service creation.

The comparison made here by the participants gave us a good insight on current related work. This will assist us in reviewing and improving our approach and the prototypes.

5.6 Q6: Which factors could be supportive or threatening for the market success of the overall concepts of TASC?

Figure 7: Evaluation result on the market success potential

From the economical point of view (Fig. 7), our approach was foreseen to benefit third party service providers. The service creation tools can allow them to provide SBBs for easy service creation. One should also take note of the spam threat that can exist in such an implementation.

In the category of social, the Authentication, Authorization and Accounting (AAA) approach was seen as necessary. This will tackle issues regarding trust and privacy issues in service creation and usage. Newer target groups were also expected, since the nature of the proposed concepts allows sharing of created services.

From the technological point of view, common vocabulary was highlighted. This was due to the fact when one wishes to create services using own vocabulary, he must be sure that the keywords used are known in the system. We agreed that through the use of semantic description

[3] http://www.ist-spice.org, last visited on 10th January 2010
[4] http://www.ist-sms.org/, last visited on 10th January 2010
[5] http://pipes.yahoo.com/, last visited on 10th January 2010
[6] http://editor.googlemashups.com/, last visited on 10th January 2010

and discovery, common semantic can be a good alternative. Different vocabulary with the same semantic can be linked to the desired SBB. For a successful implementation of TASC approach in future SEE, a group also noted that in-depth knowledge is needed.

6 Feedback and suggestions

6.1 The TASC approach

The TASC approach gained positive response based on the participants' feedback. The hands-on tests on different service creation tools were successful, given the fact that the participants were able to understand the concepts behind the different tools. The idea of bringing easy service creation to non-developers was a new concept to the participants and they welcome the idea.

The TASC approach was seen as acceptable in area of service creation. Participants were quick in adapting to the concept, as they were able to use the VSE with non-technical semantics to create services. The participants also agreed that technicalities could be a hurdle for the non-developers, and the proposed approach successfully tackled this issue with the provided solution.

The realisation of service creation via "draw-a-service" brought also positive feedback. Most of the participants agreed that this concept has high potential in enabling service creation for non-technical users. The usage of rules to express service needs was also well accepted by the participants. The use of technology agnostic keywords for semantic service discovery was seen as a novel approach where end-users can use technology agnostic terms to create services.

The participants also welcomed the idea of platform independent implementation. The proof of concept implementations was based on Java and .Net web services. This demonstrated that the proposed approaches were not limited to only certain technologies. The choice of rule language is also not fixed, where future implementations can freely select the most suitable rule representation needed for the designated platform.

6.2 The TASC tools

During the evaluation event, the participants were willing to test various prototypes and to ask questions in order to understand the TASC tools and approaches. The implementation of an IDE plug-in for SBB developers gained positive feedback. The main benefits of having such a plug-in were the time and resource saving factors. The participants found the assistance and wizards provided by the plug-in useful. The participants also made good remarks on the VSE. The ability to graphically "draw-a-service" without prior knowledge of the provided service environment was regarded as a feasible feature. The hands-on tests of the VSE also brought valuable suggestions for future improvements.

6.3 Suggestions

The technology agnostic process of service creation can be simplified and improved with the help of graphical debugging and monitoring. Consistency checks for the rules can inform the end-user while drawing a service, if there are mistakes such as wrong compositions and misspelled entries. In-depth research also should be carried out on the area of SBB interaction in our service

creation solution. This addresses the unresolved problems of how services and SBBs could interact and how information could be exchanged. Besides that, exception handling should also be considered as one of the next steps. This means that the end-user could be informed and advised when an evaluated rule has an error.

A self learning dictionary of the defined and used tags as well as meta tags can be added. This will assist the end-users in describing SBBs using their own vocabularies. The dictionary can be improved and modified during the execution of the SBBs or during the service creation process using the provided VSE. The dictionary is expected to further improve the implementation of the technology agnostic idea in the area of semantic service description and discovery.

Other questions which have to be researched are the issues on trust and privacy for the SBBs. These are important challenges because the SBBs used within a service rule are not obligatory connected to concrete implementations of SBBs but are characterised by technology agnostic descriptions. The implementation of trust and privacy will reduce the risk of accidental execution of potentially malicious SBBs from unknown parties.

For the market success of a service creation tool, it is necessary to study the requirements that enable open service market places. Another potential step in this area is to further push the technology agnostic service creation approach towards standardization. In this way, the openness of the approaches can be kept, while different parties involved in the service provision and creation chain can also profit from the approach. More SBBs should also be made available for further tests and usages so that the approach can be evaluated and used by more users.

7 Conclusion and future work

In our work we have developed a prototype service creation environment that aims to enable service creation for everyone. An evaluation event was held to allow participants to test and evaluate our prototype tools and service creation approach. Generally the evaluation process has brought up fruitful and good suggestions. Potential extensions were proposed in order to further improve the concepts. The challenges ahead were also identified.

As future work we will investigate issues such as consistency, feature interaction, concurrent execution. At the same time, more SBBs will be developed for additional use cases. We also intend to investigate further semantic service description and discovery methods to improve SBB discovery for our prototype tools.

Acknowledgment

The work performed is partly funded by the German “Bundesministerium für Bildung und Forschung” (BMBF). The authors thank the evaluation participants for their valuable feedback.

Bibliography

[All] O. M. Alliance. OMA Enabler and Reference Releases. Available online at http://www.openmobilealliance.org/Technical/released_enablers.aspx, last visited on 10th

January 2010.

[AS01] T. Aubonnet, N. Simoni. PILOTE: a service creation environment in next generation networks. In *Intelligent Network Workshop, 2001 IEEE*. Pp. 36–40. May 2001.

[fuj] Fujaba Tool Suite. Available online at http://www.fujaba.de, last visited on 10th January 2010.

[Gro] O. M. Group. MDA Model-Driven Architecture Guide. Available online at http://ormsc.omg.org/mda_guide_working_page.htm, last visited on 10th January 2010.

[LHL04] Y. Li, J. I. Hong, J. A. Landay. Topiary: a tool for prototyping location-enhanced applications. In *UIST '04: Proceedings of the 17th annual ACM symposium on User interface software and technology*. Pp. 217–226. ACM, New York, NY, USA, 2004.

[LKP+08] S. L. Lau, N. Klein, A. Pirali, I. Koenig, O. Droegehorn, K. David. Making Service Creation for (Almost) Everyone. In *ICT-Mobile Summit 2008*. Stockholm, June 2008.

[O'R05] T. O'Reilly. What is Web 2.0. *Design Patterns and Business Models for the Next Generation of Software* 30, 2005. Available online at http://www.oreilly.de/artikel/web20.html, last visited on 10th January 2010.

[PHA06] C. Pautasso, T. Heinis, G. Alonso. JOpera: Autonomic Service Orchestration. *IEEE Data Eng. Bull.* 29(3):32–39, 2006.

[Rep93] A. Repenning. *Agentsheets: a tool for building domain-oriented dynamic, visual environments*. PhD thesis, Boulder, CO, USA, 1993.

[rul] The Rule Markup Initiative. Available online at http://www.ruleml.org, last visited on 10th January 2010.

[SD03] T. Sohn, A. Dey. iCAP: an informal tool for interactive prototyping of context-aware applications. In *CHI '03: CHI '03 extended abstracts on Human factors in computing systems*. Pp. 974–975. ACM, New York, NY, USA, 2003.

[Szy97] C. Szyperski. *Component Software: Beyond Object-Oriented Programming (ACM Press)*. Addison-Wesley Professional, December 1997.

[WDD04] B. Wuest, O. Droegehorn, K. David. The Fame2 Platform Concept: Moving Platforms to the Mobile. In *Proc. 5th Int. Conference on Internet Computing (IC'04)*. Pp. 423–430. LasVegas, USA, 2004.

[Wil80] R. Wilensky. Meta-Planning. In *AAAI*. Pp. 334–336. 1980.

[WS01] R. Weinreich, J. Sametinger. *Component-based software engineering: putting the pieces together*. Chapter Component models and component services: concepts and principles, pp. 33–48. Addison-Wesley Longman Publishing Co., Inc., 2001.