Coordination and Adaptation Techniques: Bridging the Gap Between Design and Implementation

Report on the WS WCAT at ECOOP'06

Steffen Becker[1], Carlos Canal[2], Nikolay Diakov[3], Juan Manuel Murillo[4], Pascal Poizat[5], and Massimo Tivoli[6]

[1] Universität Karlsruhe (TH), Institute for Program Structures and Data Organization
sbecker@ipd.uka.de
[2] Universidad de Málaga, GISUM Software Engineering Group
canal@lcc.uma.es
[3] Centrum voor Wiskunde en Informatica (CWI)
nikolay.diakow@cwi.nl
[4] Universidad de Extremadura, Quercus Software Engineering Group
juanmamu@unex.es
[5] IBISC FRE 2873 CNRS - Université d'Evry Val d'Essonne
ARLES Project, INRIA Rocquencourt, France
Pascal.Poizat@inria.fr
[6] Università degli Studi dell'Aquila, Software Engineering and Architecture Group
tivoli@di.univaq.it

Abstract. Coordination and Adaptation are two key issues when developing complex distributed systems. Coordination focuses on the interaction among software entities. Adaptation focuses on solving the problems that arise when the interacting entities do not match properly. This is the report of the third edition of the WCAT workshop, that took place in Nantes jointly with ECOOP 2006. In this third edition, the topics of interest of the participants covered a large number of fields where coordination and adaptation have an impact: models, requirements identification, interface specification, extra-functional properties, automatic generation, frameworks, middleware, and tools.

1 Introduction

The new challenges raised by complex distributed systems have promoted the development of specific fields of Software Engineering such as Coordination [1]. This discipline addresses the interaction issues among software entities (either considered as subsystems, modules, objects, components, or web services) that collaborate to provide some functionality.

A serious limitation of currently available interface descriptions of software entities is that they do not provide suitable means to specify and reason on the

M. Südholt and C. Consel (Eds.): ECOOP 2006 Ws, LNCS 4379, pp. 72–86, 2007.

interacting behaviour of complex systems. Indeed, while the notations commonly used provide convenient ways to describe the typed signatures of software entities, they offer a quite limited support to describe their interaction protocol or concurrent behaviour.

To deal with such problems, a new discipline, Software Adaptation, is emerging [2]. Software Adaptation promotes the use of adaptors - specific computational entities with a main goal of guaranteeing that software components will interact in the right way not only at the signature level, but also at the protocol, Quality of Service, and semantic levels. In particular, Adaptation focuses on the dynamic/automatic generation of adaptors. In this sense, models for software adaptation can be considered as a new generation of coordination models. An introduction to Software Adaptation, its state-of-the-art, the description of the main research lines in the field, and some of its open issues can be found in [3].

This report summarizes the third edition of the WCAT workshop, that took place in Nantes jointly with ECOOP 2006. WCAT'06 tried to provide a venue where researchers and practitioners on these topics could meet, exchange ideas and problems, identify some of the key issues related to coordination and adaptation, and explore together and disseminate possible solutions. The topics of interest of the workshop were:

- Coordination Models separating the interaction concern.
- Identification and specification of interaction requirements and problems.
- Automatic generation of adaptors.
- Dynamic versus static adaptation.
- Enhanced specification of components to enable software composition and adaptation.
- Behavioral interfaces, types, and contracts for components, coordinators and adaptors.
- Formal and rigorous approaches to software adaptation.
- The role of adaptation in the software life-cycle.
- Patterns and frameworks for component look-up and adaptation.
- Metrics and prediction models for software adaptation.
- Prediction of the impact of software adaptation on Quality of Service.
- Extra-functional properties and their relation to coordination and adaptation.
- Aspect-oriented approaches to software adaptation and coordination.
- Coordination and adaptation middleware.
- Tools and environments.
- Coordination and adaptation in concurrent and distributed object-oriented systems.
- Interface and choreography description of Web-Services.
- Using adaptors for legacy system integration.
- Industrial and experience reports.
- Surveys and case studies.

The rest of this report is organized as follows: In Section 2 we enumerate the contributions received, and also the participants of the workshop. Then, Section 3, presents a comparative outline of these contributions. Section 4 summarizes the results of the three discussion groups that worked during the workshop, and Section 5 presents the conclusions of the workshop. Finally, we provide some references to relevant works on coordination and adaptation.

2 Contributions and Workshop Participants

To enable lively and productive discussions, prospective participants were required to submit in advance a short position paper, describing their work in the field, open issues, and their expectations on the workshop. From the contributions received, we decided to invite ten position papers. Both the Call for Papers, and the full text of these papers can be found at the Website of the workshop:

`http://wcat06.unex.es`

The list of accepted papers, together with the names and affiliations of their authors is as follows:

- Software Adaptation in Integrated Tool Frameworks for Composite Services
  Nikolay Diakov (*), and Farhad Arbab
  {`nikolay.diakow, farhad.arbab`}`@cwi.nl`
  CWI, The Netherlands
- Coordination and Adaptation for Hierarchical Components and Services
  Pascal André, Gilles Ardourel (*), and Christian Attiogbé
  {`Pascal.Andre, Gilles.Ardourel, Christian.Attiogbe`}`@univ-nantes.fr`
  University of Nantes, France
- Towards Unanticipated Dynamic Service Adaptation
  Marcel Cremene (*) (`cremene.marcel@com.utcluj.ro`)
  Technical University of Cluj-Napoca, Romania
  Michel Riveill (`riveill@unice.fr`)
  University of Nice, France
  Christian Martel (`christian.martel@univ-savoie.fr`)
  University of Savoie, France
- Dynamic Adaptation Using Contextual Environments
  Javier Cámara, Carlos Canal (*), Javier Cubo, and Ernesto Pimentel
  {`jcamara, canal, cubo, ernesto`}`@lcc.uma.es`
  University of Málaga, Spain
- Aspect-Oriented Approaches for Component Coordination
  Lidia Fuentes, and Pablo Sánchez (*)
  {`lff,pablo`}`@lcc.uma.es`
  University of Málaga, Spain

- Towards Unification of Software Component Procurement Approaches
  Hans-Gerhard Gross (*) (h.g.gross@tudelft.nl)
  Delft University of Technology, The Netherlands
- On Dynamic Reconfiguration of Behavioral Adaptations
  Pascal Poizat (*) (poizat@lami.univ-evry.fr), University of Evry, France
  Gwen Salaün (Gwen.Salaun@inrialpes.fr), INRIA Rhône-Alpes, France
  Massimo Tivoli (*) (tivoli@di.univaq.it), University of L'Aquila, Italy
- Capitalizing Adaptation Safety: a Service-oriented Approach
  Audrey Occello (*), and Anne-Marie Dery-Pinna
  {occello, pinna}@essi.fr
  University of Nice, France
- Safe Dynamic Adaptation of Interaction Protocols
  Christophe Sibertin-Blanc (*) (sibertin@univ-tlse1.fr)
  University of Toulouse 1, France
  Philippe Mauran, and Gérard Padiou
  {mauran, padiou}@frenseeiht.fr
  Institut de Recherche en Informatique de Toulouse
  Pham Thi Xuan Loc (phamtxloc@yahoo.com)
  Can Tho University, Vietnam
- An Aspect-Oriented Adaptation Framework for Dynamic Component Evolution
  Javier Cámara, Carlos Canal (*), and Javier Cubo
  {jcamara, canal, cubo}@lcc.uma.es
  University of Málaga, Spain
  Juan Manuel Murillo (*) (juanmamu@unex.es)
  University of Extremadura (Spain)

Authors that were in fact present at the workshop are marked with and asterisk in the relation above. Apart from those, also attended the workshop, without presenting a paper:

- *Steffen Becker* (sbecker@ipd.uka.de)
  Technical University of Karlsruhe, Germany
- *Jérémy Buisson* (jbuisson@irisa.fr)
  IRISA, France
- *Fabien Dagnat* (Fabien.dagnat@enst-bretagne.fr)
  Ecole Nationale Supérieure des Télécommunications de Bretagne, France
- *Leonel Gayard* (leonel.gayard@ic.unicamp.br)
  State University of Campinas, Brasil
- *Arnaud Lanoix* (lanoix@loria.fr)
  LORIA, France
- *Inès Mouakher* (mouakher@loria.fr)
  LORIA, France

In total, seventeen participants coming from seven different countries attended the workshop.

3 Comparative Summary of the Contributions

The position papers presented in the workshop covered a wide number of issues related to coordination and adaptation, both from the theoretical and the practical point of view.

3.1 Coordination Models

Some of the papers focused on coordination techniques for software composition and interaction. Coordination Models and Languages provide mechanisms to specify the coordination constraints of a software system. Such constraints involve all the dependencies between the computational entities of the system, that is, their interactions. Thus, Coordination Languages could be good tools to support adaptation when the subject of the adaptation is the way in which software entities interact.

Coordination Models can be classified in two categories [1] named *Endogenous* and *Exogenous*. Such categories are motivated by the different mechanisms used by coordination models to deal with the coordination constraints of the system.

Endogenous coordination languages provide primitives that are incorporated within the computational entities for their coordination. The most representative language in this category is Linda [4]. If the interaction protocol is to be adapted, it could be done adapting the behaviour of the primitives to access the tuple space and the pattern matching mechanism.

In contrast, exogenous coordination languages place the coordination primitives in entities external to those to be coordinated. Such clear separation between processes functionality and interaction protocols provides a better support for adaptation. Hence, adapting the interaction protocols can be managed manipulating the coordinators. Design time adaptation can be easily managed recoding the interaction protocols implemented by coordination processes. Moreover, runtime adaptation is also supported.

Following this approach, Diakov [5] presented an adaptation framework for the construction of composite Web Services (WS) for distributed computing environments, developed in collaboration with F. Arbab. Their position was that current Web Service description languages, such as WSDL or WSBPEL do not sufficiently address compositionality at the level of their formal semantics and dynamic execution model. Thus, they applied their previous theoretical results in formal techniques for exogenous coordination of software components, namely using the coordination model Reo [6]. The major issues addressed are: *(i)* the implication of synchrony in the way in which software is designed, *(ii)* the bridging of protocols with fundamentally different coordination models, and *(iii)* the enriching of theoretical models so that they become usable in practical applications. They also presented a tool framework for WS composition that includes visual editing and a graphical user interface.

Combining both Coordination and Aspect-Oriented Software Development (AOSD) techniques, Sánchez [7] presented an Aspect-Oriented approach for component coordination that complements their work on adaptation developed

for the preceding edition of the workshop [8]. His position was that the coordination protocol that governs the interchange of messages among interacting software components is usually entangled with the base functionality of those components, especially in the field of endogenous coordination. In this work, the authors advocate for the use of Aspect-Oriented techniques for the separation of coordination patterns and components' functionality, in order both to reuse them and to make composition easier. Their paper includes a survey on some recent publications that combine CBSD and AOSD with success, including component and aspect models such as MALACA, programming languages such as JAsCo (Java Aspect Components), and aspectized component platforms such as CAM/DAOP [9], which gives a broad view of the technologies required and the current state of the art in aspect-oriented coordination.

3.2 Software Adaptation

The rest of the works presented addressed specifically different adaptation issues. In his paper, Gross [10] presented his proposal towards the unification of software component procurement and integration. Component procurement deals with how to map customer component requirements to provider's component specifications. His position was that the mechanisms currently available only deal with lower levels of abstraction, close to the implementation level. The goal of his research work is to elevate typical component featuring mapping mechanisms from the implementation level up onto the design and requirements engineering levels. He outlined a research agenda that includes *(i)* the formalization of component specifications equipped with provided and required behaviour models, in order to automate the currently manually performed component mapping and integration effort; *(ii)* the definition of an integrated requirements specification method based on natural language; *(iii)* the derivation of model artifacts from the these requirements specifications documents; and *(iv)* the development of mapping mechanisms to other commonly used component specification notations such as UML.

3.3 Adaptation of Behaviour

Also dealing with how to express component specifications, Ardourel [11] presented a Behavioural IDL (BIDL) for the coordination and adaptation of hierarchical components and services. The paper presented an approach in which a service is like a formal operation with a signature (name and parameters), a contract (pre/post-conditions), a signature interface (with provided and required services, from which the service's hierarchical structure can be inferred), and a BIDL that describes the dynamic evolution of the service. The proposal is exemplified by means of a classical Automatic Teller Machine example. In this setting the kind of adaptation that can be performed includes both message and parameter names, message ordering, and one-to-many relations between clients and servers.

One of the big issues of the workshop was how to address dynamic adaptation. Several papers presented the proposals of their authors to this problem. For

instance, Canal [12] presented an approach for dynamic adaptation based on the use of contextual environments. In this work it is shown how to perform adaptation following flexible polices. The goal is to describe a context-dependent mapping between the interfaces of the components being adapted, as opposed to static mappings presented in the authors' previous work [13]. Mappings are described using a map calculus that defines a small set of operations over environments and mappings, designed to express various encapsulation policies, composition rules, and extensibility mechanisms. On the other hand, component's interface description is achieved by means of a BIDL based on process algebraic notation (namely the π-calculus). From that, an algorithm would automatically develop an adaptor for the components involved, following the flexible adaptation policies specified in the mapping. The proposal is exemplified by means of a Video-On-Demand system.

A similar approach, also dealing with dynamic behavioural adaptation was that of Poizat [14], who presented a proposal for the dynamic reconfiguration of behavioural adaptations. His position was that, since the development of software adaptors is costly, it is crucial to make adaptor reconfiguration possible when one wants to modify or update some parts of a running system involving adaptors. In an initial approach, the problem of dynamically reconfigurable adaptors was presented, and some ideas for a solution to this problem were sketched. The proposal is built on previous work [15] in which component behavioral interfaces are specified by labelled transition systems (LTS), and in order to find out if a system made up of several components presents behavioural mismatch, the synchronous product of these LTS is computed, and then the absence of deadlock is checked on it. Then, an abstract description of an adaptor is specified again by an LTS which, put into a non-deadlock-free system, keeps it deadlock-free. From this starting point, the possible changes that can be applied dynamically to a system and may result in incompatible behavior (namely, component addition, upgrading, or suppression) are identified, and some hints on how to deal with the automatic handling of the reconfiguration process in order to maintain the system deadlock-free are given.

An aspect-orientated approach was also advocated by Murillo [16]. AOSD provides techniques to deal with the separation of crosscutting concerns along the software life cycle. Within that, Aspect Oriented Programming (AOP) is focused on the implementation step proposing programming languages supporting the separated codification of system features that crosscut several software entities (classes, subprograms, processes,...). The code specifying the separated properties is located in a new kind of module commonly called an *aspect*. In addition, one should also specify how aspects affect software entities. In order to obtain the original system behaviour, this information is used to weave the code inside aspects with the software entities. Such weaving can be done both statically, before the application starts, or dynamically, when the application is already running.

Murillo presented an adaptation framework for dynamic component evolution. The approach combined both aspect-oriented programming with reflection

and with adaptation techniques in order to support and speed up the process of dynamic component evolution by tackling issues related to signature and protocol interoperability. The goal was to provide the first stage of a semi-automatic approach for syntactical and behavioural adaptation. Murillo pointed out that when performing component adaptation it is necessary to have information that is only available at runtime. Moreover, if one wants to take advantage of this information, he or she has to find a way to apply it at runtime as well. His position was that aspect-oriented techniques serve for the purpose, and two different strategies may be considered: *(i)* Dynamic aspect generation, allowing runtime generation, application and removal of aspects implementing adaptation concerns. However, this is not a realistic scenario yet, considering the state-of-the-art on aspect-oriented languages and runtime platforms. Moreover, the computational overhead caused by these additional tasks may be too heavy for the system. *(ii)* Dynamic adaptor management, a more reasonable approach with respect to to overhead, in which the aspects that manage adaptation are precompiled, and an adaptation manager is able to retrieve, interpret and use the dynamic information required for runtime adaptation.

3.4 Unanticipated Adaptation

The important issue of unanticipated orientation was raised by the work of Cremene et al. [17]. Indeed, the adaptation proposals described above deal with dynamism either by assuming predefined adaptation policies or by passing through a phase of system re-design. In contrast, Cremene's presentation dealt with unanticipated service adaptation at runtime. Their position is that automatically solving unanticipated adaptation is a must, since the context (user profile, physical resources, and other elements) may change while the service is running. However, the current service adaptation approaches require the prediction of all the possible variations in the context, and need to specify beforehand rules for each possible situation. The authors presented a solution based on a context-service common representation that enables them to discover the adaptation rules and strategies rather than to fix them a priori. For that, they use a service-context description that explains how the service and the context interact. This description is based on a three-layered model, including a context layer, a component layer and a profile layer. They are currently developing a prototype of service adaptation based on their proposals.

3.5 Safe Adaptation

The remaining two papers addressed safety issues. Sibertin-Blanc [18] presented an approach to safe dynamic adaptation of interaction protocols between the actors of a computer-aided learning system based on the notion of a moderator. In this proposal, moderators are components managing interactions described and formalized by means for Petri nets. Then, dynamic adaptation is performed by specific transformations of the net representing the moderator. These transformations permit to satisfy adaptation demands, insofar as these changes do

not alter the integrity of the base system. The flexibility of the approach is illustrated by a real case study for the adaptation of a protocol for controlling accesses to documents during an examination.

Finally, Occello et al. [19] presented a service-oriented approach for capitalizing adaptation safety. They assume that the different adaptation mechanisms for modifying system's structure and components' behaviour dynamically may lead the application to an unsafe state. Current approaches facing this problem can be divided into *(i)* formal approaches, that provide deep theoretical results but are generally not linked with an implementation; and *(ii)* practical solutions, that lack of formal foundations and cannot be reused in other platforms. From this description of the problem, their proposal consists of computing the required adaptation at model level in order to make adaptation generic, and thus reusable. The authors presented the *Satin* safety service that can be queried by technological platforms to determine whether the adaptations to be done at the platform level are safe or not. This work makes benefits of a model driven engineering (MDE) approach for validation purposes. Among the open issues of this work was the development of a methodology for model validation.

4 Discussion Groups

Invited participants made five-minute presentations of their positions, followed by discussions that served to identify a list of open issues in the field. We divided these issues into three categories or groups: *Fundamental Issues*, *Incremental Composition and Testing of Dynamic Architectures*, and *Industrial Knowledge Transfer*. Participants were then assigned to one of them, attending to their interests. The task of each group was be to discuss about a subset of the previously identified issues.

4.1 Fundamental Issues

This group was formed by Jeremy Buisson, Carlos Canal, Marcel Cremene, Fabien Dagnat, Inès Mouakher, Audrey Occello, and Pablo Sánchez. The topics assigned to the group covered a number of general and fundamental issues on adaptation and coordination, including component models, in particular semantic models, exogenous (external) coordination vs. endogenous (intracomponent) coordination, context compliance, and context dependent reasoning. These topics lead the discussion to other even more general issues on components, aspects, interfaces, coordination, etc.

At the end of the session, each participant was asked to write down a conclusion or *motto*, which afterwards was discussed and voted for acceptance/rejection by the rest of the group.

- *"Coordination is more than just message coordination"*. This issue was raised by the common perception that most of the works presented in the workshop (and also in the literature) on Coordination deal with message interaction. Moreover, adaptation techniques also address mainly message passing issues,

both at the signature and behaviour levels. However, Coordination should also address other issues such as, for instance, memory consumption. Such issues can profoundly affect the interaction of components and the performance and global behaviour of the whole system.

- *"Existing component models do not contain enough elements in order to define components interaction with the environment".* Indeed one of the classical issues in Software Adaptation and also in CBSD is how to effectively model and represent the dependencies and interrelations both among the components of the system, and between the system and its environment. This problem was also raised by the next *motto*.
- *"The four levels of contracts (syntactical, behavioural, quality of service, semantics) are enough to represent adaptation requirements, but we do not have language tools to express them".* As we have already mentioned, current component models and IDLs only focus on the signature of the messages exchanged. Many research proposals in the field address also behavioural specification using BIDLs, but many other important properties of components (for instance, QoS and functionality) lack a sufficient description.
- *"Component should declare all their dependencies in their interfaces and this declaration should be as abstract as possible".* Only by means of abstract, platform-independent interface descriptions would it be possible to perform third-party and interplatform composition and adaptation.
- *"Context dependency is complementary to contract specification".* That is, the more explicitly the context dependencies of a component are described in its interface, the less context-dependent would the component become, since all these dependencies could be then addressed with composition and adaptation techniques.
- *"Context is a component that should be contracted".* We can take this *motto* as the conclusion/summary of the group discussions, since it tries to drive attention and incitate discussion by being controversial, paving the path for future discussions on these and other related issues.

4.2 Incremental Composition and Testing of Dynamic Architectures

The group was formed by Gilles Ardourel, Leonel Gayard, Juan Manuel Murillo, Pascal Poizat, and Christophe Sibertin-Blanc. The main issue assigned to the group was how (and when) to address dynamic adaptation, an underlying topic in most of the papers presented in the workshop.

First, the group started by agreeing and setting the context of the problem. It falls obviously in the grounds of CBSD and, in general, in any other context where the integration of different components or software units is required (for instance, in team development). One less obvious but yet significant field of application is *"the jungle of"* Pervasive Computing, as it was named by some of the participants, and in general, any system which has a high degree of dynamism in its architecture, and changing environment conditions. In this context,

adaptation is not only required to make components match, but it must also collaborate with service discovery and software composition to build new emerging services from a jungle of components.

Then, if we assume that the architecture of the system can be dynamic (with components continuously entering and leaving the system, possibly evolving themselves, and with dynamic reconfiguration of component relations), the adaptation process needs to be also inherently dynamic. This establish a series of links with implementation issues, such as adaptative middleware [20], and runtime overload.

As a consequence, it was agreed that there was a need for *incremental* adaptation: when something changed in the system, not everything had to be re-computed. Incremental adaptation would require to adapt the adaptors themselves, or alternatively, to make them evolve attending to the changing conditions of the system. Techniques for this kind of adaptation have to be developed.

The group also discussed how the incremental development of the system relates to the need for compositionality. It is a fact that some properties of software can be lifted from the local (components) to the global (system), but this seems to require some kind of compositionality of properties. If that could be expressed, incremental adaptation would be solved by using hierarchical adaptors (i.e. adaptors of adaptors). However, some properties (for instance, deadlock freedom) are not compositional in that sense, and this means that an additional specific step of system-wide adaptation has to be computed.

4.3 Industrial Knowledge Transfer

The group —formed by Steffen Becker, Nikolay Diakow, Hans-Gerhard Gross, Arnaud Lanoix, and Massimo Tivoli— has been assigned to discuss on several issues. The overall question has been why there is no widespread use of adaptation techniques in industrial practice although there exist quite several working and useful algorithms. As a consequence, the main issue dealt with was what can be done to transfer this scientific knowledge into industry. The group came up with a discussion on the stakeholders involved in to transfer process, the reasons for them to actually do knowledge transfer, the problems faced, why this transfer is performed insufficiently and what means can help to change the situation.

Initially, the group identified three stakeholders mostly involved in a knowledge transfer process: academic institutions, research units in companies, and production units in companies. In the first category there are usually universities and technology transfer institutes which are usually associated to one or more university institutes. They develop research ideas and report on them in academic publications. Research units in companies also develop new ideas but are mainly concerned with customizing research results of academic institutions in order to make them applicable in their companies operational goals. Finally, the production units are involved with their daily work. They have the possibility to report on their problems either to their research units or to universities. One of the most often used way of transferring knowledge goes from academic institutions over research units into production. But there is also a direct link

between universities and production which is used implicitly by people coming from universities into production practice. This is often the case with interns, but also when students finish their studies and go into industry.

The next question was to investigate why companies and universities are motivated to do knowledge transfer. This motivation can be used to increase the transfer amount. For companies there are short term and long term goals driving them to participate in technology transfer. Two major factors have been identified: business advantages and prestige/vanity. Business advantages directly result gaining a higher efficiency often by the possibility to decrease costs using the new knowledge. Other factors can result from the ability to produce the same products but with increased quality attributes or by getting the same products faster into a market (decreased time-to-market). Prestige and vanity which can be used in marketing and project acquisition come from the fact that companies can show that they have experience in new fields of technology. This is often supported by the use of current buzz-words to communicate their expertise.

Scientific people are motivated to do technology transfers for scientific and social reasons. The scientific value of doing knowledge transfer comes from the possibility to do validations, verification or proof-of-concept studies of the developed ideas. Additionally, after doing such studies there is usually a lot of gained new experiences. This kind of feedback can afterwards lead to the development of enhancements of the existing methods or even to the initiation of new research. The social factor comes from the fact that scientists get to know people and develop long term cooperations.

Given the above highlighted motivations for the participants in knowledge transfers, there is the question why there are problems in doing technology transfers. The main issue here is how to convince the companies that a newly developed idea is worth trying. The ideas developed against this issue contained the two suggestions. Either the transfer should be done in small steps or in pilot projects for new buzz-word related technologies. In performing small introductory steps those steps should be selected which are easy to introduce but show their usefulness in large parts of the companies. After a successful initial step of this kind larger projects can be initiated as follow up projects. Buzz-word related projects may be larger because they usually target at the prestige motivation of companies.

To get started into the discussion what can be done to alter the current situation the question was how can scientific people convince companies of the usefulness of their ideas. The group came up with several organizational issues which can help. In order to reduce the costs and hence the risk for companies, student projects are well suited. Additionally, also well defined reference projects with little financial risk can be suited to get a knowledge transfer started. An other means to reduce the financial risk of larger projects can be to participate in governmental projects which cover most of the costs. A different means is to found a spin-off company in which research ideas are evolved into a mature state. Additionally, given a spin-off company the spin-off has to deal with the economic risks alone and not a third-party.

A different issue is that many of the developed ideas in science take a while to communicate and comprehend. During a scientific presentation, company representatives do not get the impression that a research idea is useful. This can often be altered when a tool is available. Tools are able to hide the complexity of algorithms and hence, are better suited to communicate the benefits and the possible automation of the methods. A remaining problem is that even if a tool exists, for productive use often tool customizations are still needed.

Finally, the group discussed the idea that presenting research ideas with more suited, problem-oriented models can also help. For example, in adaptation methods often difficult to understand models like process algebras are applied. This kind of complexity should be hidden by using different models in the interaction with the end users of the methods. In our example, graphical languages to model protocols are better suited for communication than the same model written as process algebra. In order to realize tools which offer easy to understand problem or domain oriented models, model transformations (like in MDA) can help to automatically come from the user representation to the internal algorithm oriented representation. Additionally, model transformations can also be applied in the opposite direction. In this direction, they transform the results of the algorithms into a user understandable way. In tool prototypes it would be therefore a task of the researches to implement in their tools also such a user understandable model and presentation. Moreover, having such a transformation approach in place it is also easier to integrate several different analysis methods in a single tool which also increases the acceptance probability when doing industry presentations.

5 Conclusion of the Workshop

Finally, a plenary session was held, in which each group presented their conclusions to the rest of the participants, followed by a general discussion. During this session attendants were asked for their general impression about the third edition of WCAT. While the first two editions of WCAT [21,22] tried to define adaptation and its limits with reference to other domains such as evolution or maintenance, during this third edition, people agreed that they had been able to discuss on more precise and technical issues. These issues were namely the differences between techniques for adaptation at runtime and design time, and mechanisms to implement or generate adaptors. Implementation, the third part of the detect-correct-implement adaptation triplet, is often eluded in adaptation proposals.

Some future work for next editions of WCAT can be seen in the following issues for which several complementary domains should be investigated:

- the relation with ontologies and automatic service composition approaches [23] for the automatic specification of adaptation mappings;
- the relation with distributed and real-time domains for the detection of adaptation needs, and the online application of adaptation (either using design time or runtime techniques);

– the relation with AOSD and MDA techniques, languages and component models for the implementation of adaptors: should we use weaving (AOSD) or refinement (MDA)?

Finally, the community interested in adaptation and coordination identified the task of developing a common suite of example problems found in adaptation and coordination. It is envisioned to use this suite in the future to estimate strengths and weaknesses of proposed adaptation methods, to compare methods with each other, and finally, to validate the usefulness of the proposed approaches.

References

1. Arbab, F.: What Do You Mean Coordination? Bulletin of the Dutch Association for Theoretical Computer Science (NVTI) (1998)
2. Canal, C., Murillo, J.M., Poizat, P.: Report on the First International Workshop on Coordination and Adaptation Techniques for Software Entities. In: Object-Oriented Technology. ECOOP 2004 Workshop Reader. Volume 3344 of Lecture Notes in Computer Science., Springer (2004) 133–147
3. Canal, C., Murillo, J.M., Poizat, P.: Software adaptation. L'Objet **12-1** (2006) 9–31
4. N. Carreiro, D.G.: Linda in Context. Communications of the ACM **32** (1989) 133–147
5. Diakov, N., Arbab, F.: Software adaptation in integrated tool frameworks for composite services. [24] 9–14
6. Arbab, F.: Reo: A channel-based coordination model for component-composition. Mathematical Structures in Computer Science **14-1** (2004) 329–366
7. Fuentes, L., Sánchez, P.: Aspect-oriented approaches for component coordination. [24] 43–51
8. Fuentes, L., Sánchez, P.: AO approaches for Component Adaptation. [22] 79–86
9. Pinto, M., Fuentes, L., Troya, J.M.: A dynamic component and aspect-oriented platform. The Computer Journal **48-4** (2005) 401–420
10. Gross, H.G.: Towards unification of software component procurement approaches. [24] 53–59
11. André, P., Ardourel, G., Attiogbé, C.: Coordination and adaptation for hierarchical components and services. [24] 15–23
12. Cámara, J., Canal, C., Cubo, J., Pimentel, E.: Dynamic adaptation using contextual environments. [24] 35–42
13. Bracciali, A., Brogi, A., Canal, C.: A Formal Approach to Component Adaptation. Journal of Systems and Software **74-1** (2005) 45–54
14. Poizat, P., Salaün, G., Tivoli, M.: On dynamic reconfiguration of behavioral adaptations. [24] 60–69
15. Canal, C., Poizat, P., Salaün, G.: Synchronizing behavioural mismatch in software composition. In: Formal Methods for Open Object-Based Distributed Systems (FMOODS'06). Volume 4037 of Lecture Notes in Computer Science., Springer (2006) 63–77
16. Cámara, J., Canal, C., Cubo, J., Murillo, J.M.: An aspect-oriented adaptation framework for dynamic component evolution. [24] 91–99

17. Cremene, M., Riveill, M., Martel, C.: Towards unanticipated dynamic service adaptation. [24] 25–34
18. Sibertin-Blanc, C., Xuan-Loc, P.T., Mauran, P., Padiou, G.: Safe dynamic adaptation of interaction protocols. [24] 81–90
19. Occello, A., Dery-Pinna, A.M.: Capitalizing adaptation safety: a service-oriented approach. [24] 71–79
20. Agha, G.A., ed.: Special Issue on Adaptative Middleware. Volume 45(6) of Communications of the ACM. ACM Press (2002)
21. Canal, C., Murillo, J.M., Poizat, P., eds.: First International Workshop on Coordination and Adaptation Techniques for Software Entities (WCAT'04). Held in conjunction with the 18th European Conference on Object-Oriented Programming (ECOOP'2004). Available at http://wcat04.unex.es/. (2004)
22. Becker, S., Canal, C., Murillo, J.M., Poizat, P., Tivoli, M., eds.: Second International Workshop on Coordination and Adaptation Techniques for Software Entities (WCAT'05). Held in conjunction with the 19th European Conference on Object-Oriented Programming (ECOOP'2005). Available at http://wcat05.unex.es/. (2005)
23. Mokhtar, S.B., Georgantas, N., Issarny, V.: Ad hoc composition of user tasks in pervasive computing environments. In: Software Composition. (2005) 31–46
24. Becker, S., Canal, C., Diakov, N., Murillo, J.M., Poizat, P., Tivoli, M., eds.: Third International Workshop on Coordination and Adaptation Techniques for Software Entities (WCAT'06). Held in conjunction with the 20th European Conference on Object-Oriented Programming (ECOOP'2006). Available at http://wcat06.unex.es/. (2006)
25. Ciancarini, P., Hankin, C., eds.: First International Conference on Coordination Languages and Models, (COORDINATION '96). Volume 1061 of Lecture Notes in Computer Science., Springer (1996)
26. Arbab, F.: The IWIM model for coordination of concurrent activities. [25]