Grant Agreement N° 215483

Title:	*Specifications of policies and strategies for distributed and multi-level adaptation*
Authors:	*INRIA, CNR, FBK, UniDue, TUW, UOC, SZTAKI*
Editor:	*Erwan Daubert, Guillaume Gauvrit (INRIA)*
Reviewers:	*Zsolt Nemeth (SZTAKI)*
	Salima Benbernou (Université de Paris Descartes)
Identifier:	*Deliverable # CD-JRA-2.3.8*
Type:	*Deliverable*
Version:	*1*
Date:	*07 March 2012*
Status:	*Final*
Class:	*External*

Management Summary

This deliverable aims to present the research progress of the project partners about distributed and multi-level (also known as cross layer) adaptation.
Previous works from the deliverables CD-JRA-1.2.2, CD-JRA-1.2.3, CD-JRA-2.3.2, CD-JRA-2.3.4, CD-JRA-2.3.6 provide the bases for local adaptation and monitoring. Local adaptations are done on a part of a system taken in isolation from the rest. In this deliverable the adaptation process is extended to take into account the distribution and all the layers (all the part) of a system (hardware, networks, operating systems, middleware, services, workflows...) in order to provide coherent and concurrent adaptations.
The research results are presented through the summaries of joint papers that are classified according to the solutions they provide to manage distributed and multi-level adaptation.

Members of the S-Cube consortium:

University of Duisburg-Essen (Coordinator)	Germany
Tilburg University	Netherlands
City University London	U.K.
Consiglio Nazionale delle Ricerche	Italy
Center for Scientific and Technological Research	Italy
The French National Institute for Research in Computer Science and Control	France
Lero - The Irish Software Engineering Research Centre	Ireland
Politecnico di Milano	Italy
MTA SZTAKI – Computer and Automation Research Institute	Hungary
Vienna University of Technology	Austria
Université Claude Bernard Lyon	France
University of Crete	Greece
Universidad Politécnica de Madrid	Spain
University of Stuttgart	Germany
University of Hamburg	Germany
Vrije Universiteit Amsterdam	Netherlands

Published S-Cube documents

All public S-Cube deliverables are available from the S-Cube Web Portal at the following URL:

http://www.s-cube-network.eu/results/deliverables/

The S-Cube Deliverable Series

Vision and Objectives of S-Cube

The Software Services and Systems Network (S-Cube) will establish a unified, multidisciplinary, vibrant research community which will enable Europe to lead the software-services revolution, helping shape the software-service based Internet which is the backbone of our future interactive society.

By integrating diverse research communities, S-Cube intends to achieve world-wide scientific excellence in a field that is critical for European competitiveness. S-Cube will accomplish its aims by meeting the following objectives:

- Re-aligning, re-shaping and integrating research agendas of key European players from diverse research areas and by synthesizing and integrating diversified knowledge, thereby establishing a long-lasting foundation for steering research and for achieving innovation at the highest level.
- Inaugurating a Europe-wide common program of education and training for researchers and industry thereby creating a common culture that will have a profound impact on the future of the field.
- Establishing a pro-active mobility plan to enable cross-fertilisation and thereby fostering the integration of research communities and the establishment of a common software services research culture.
- Establishing trust relationships with industry via European Technology Platforms (specifically NESSI) to achieve a catalytic effect in shaping European research, strengthening industrial competitiveness and addressing main societal challenges.
- Defining a broader research vision and perspective that will shape the software-service based Internet of the future and will accelerate economic growth and improve the living conditions of European citizens.

S-Cube will produce an integrated research community of international reputation and acclaim that will help define the future shape of the field of software services which is of critical for European competitiveness. S-Cube will provide service engineering methodologies which facilitate the development, deployment and adjustment of sophisticated hybrid service-based systems that cannot be addressed with today's limited software engineering approaches. S-Cube will further introduce an advanced training program for researchers and practitioners. Finally, S-Cube intends to bring strategic added value to European industry by using industry best-practice models and by implementing research results into pilot business cases and prototype systems.

S-Cube materials are available from URL: http://www.s-cube-network.eu/

Contents

Chapter 1

Introduction

This document is part of the deliverable series of S-Cube, the Software Services and Systems Network. Self-* Service Infrastructure and Service Discovery Support (WP-JRA-2.3): Work on service infrastructures will define policies, monitoring and redeployment techniques, for self-adaptive and self-healing services. This will strongly rely on input from WP-JRA-2.2 to design policies for adaptive service composition. In addition, this workpackage will specify and develop registry support mechanisms for service metadata, QoS attributes, service composition, and a federation of service registries.

This deliverable named "Specifications of policies and strategies for distributed and multi-level adaptation" reports on our findings about the reasonable policies to use in multi-level (also called cross-layer) adaptation when applied to distributed systems like Grids and clouds.

1.1 Background

As it was stated in CD-JRA-1.2.1 [5]: "The dynamic nature of the business world highlights the continuous pressure to reduce expenses, to increase revenues, to generate profits, and to remain competitive. This requires Web services to be highly reactive and adaptive. It should be equipped with mechanisms to ensure that their constituent component Web services are able to adapt to meet changing requirements. In fact, services are subject to constant changes and variations. Services can evolve due to changes in structures (attributes and operations), in behavior (when services are interacting) and policies." There is a strong consensus on the fact that services and service infrastructures must be able to perform (self-) adaptation which is the topic of WP-JRA-2.3.

This deliverable focuses on the way to adapt large scale systems taking into account the requirements of the users as well as the constraints of the providers of each part of the systems such as services, middlewares, hardwares, networks... As it is needed to take into account all these elements, it is obviously mandatory to be able to monitor and adapt them. That's why the works described here complete, extend or at least use the existing solutions presented on previous deliverable of the scube project (see the list 1.1.2).

1.1.1 Connections to the JRA-WP-2.3 research architecture

Research work in WP-JRA-2.3 is driven by the Work Package vision that structures the research work internally. Figure 1.1 illustrates the overall research architecture of WP-JRA-2.3: research on service infrastructures is comprised in three threads, Service Discovery, Service Registries and Service Execution. Orthogonally different approaches are separated in three layers.

- Service Discovery Thread (A) - Service discovery is a fundamental element of service-oriented architectures, services heavily rely on it to enable the execution of service-based applications. Novel discovery mechanisms must be able to deal with millions of services. Additionally, these

discovery mechanisms need to consider new constraints, which are not prevalent today, such as Quality of Experience requirements and expectations of users, geographical constraints, pricing and contractual issues, or invocability.

- Service Registry Research Thread (B) - Service registries are tools for the implementation of loosely-coupled service-based systems. The next generation of registries for Internet-scale service ecosystems are emerging, where fault tolerance and scalability of registries is of eminent importance. Autonomic registries need to be able to form loose federations, which are able to work in spite of heavy load or faults. Additionally, a richer set of metadata is needed in order to capture novel aspects such as self-adaptation, user feedback evaluation, or Internet-scale process discovery. Another research topic is the dissemination of metadata: the distributed and heterogeneous nature of these ecosystems asks for new dissemination methods between physically and logically disjoint registry entities, which work in spite of missing, untrusted, inconsistent and wrong metadata.
- Runtime Environment Research Thread (C) - There is an obvious need for automatic, autonomic approaches at run-time. As opposed to current approaches we envision an infrastructure that is able to adapt autonomously and dynamically to changing conditions. Such adaptation should be supported by past experience, should be able to take into consideration a complex set of conditions and their correlations, act proactively to avoid problems before they can occur and have a long lasting, stabilizing effect.

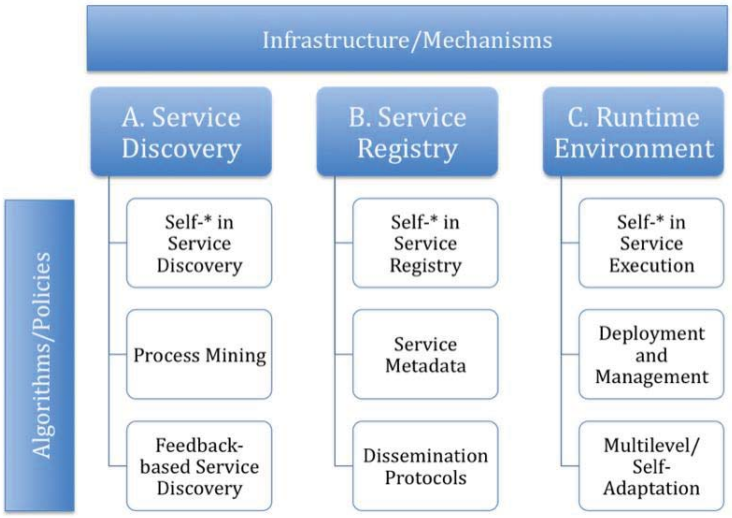


Figure 1.1: WP-JRA-2.3 Research Architecture

In alignment with the lifecycle aspect of the Integrated Research Framework, the presented work, is related mostly to runtime activities hence, to research topics in thread C in Figure 1.1. All these works target the ”Multi-level and self-adaptation” research challenge.

1.1.2 Relationship to other deliverables and workpackages

Some previous deliverables provides various information that fits with the content of this deliverable:

- CD-JRA-1.2.2 [7] untitled “Taxonomy of Adaptation Principles and Mechanisms” provides a comprehensive and holistic overview of the knowledge and concepts in the field of adaptation and monitoring. Those concepts are used and referred to in this CD-JRA-2.3.8 deliverable.

- PO-JRA-1.2.3 [30] untitled "Baseline of Adaptation and Monitoring Principles, Techniques, and Methodologies across Functional SBA Layers" provides some information about the requirements for the multi-level adaptation and monitoring.
- CD-JRA-2.3.2 [8] untitled "Basic requirements for self-healing services and decision support for local adaptation" captures the requirements for the design and realization of self-healing services within the S-Cube architecture. These foundations are used in the following deliverables of the "Infrastructure Mechanisms for the Run-Time Adaptation of Services" research thread.
- CD-JRA-2.3.4 [9] untitled "Decision support for local adaptation" documents research results related to making services adaptable and self-aware while adhering to a set of given specifications and mechanisms; it also investigated the applicability of certain policies to trigger local adaptation mechanisms. This decision support is extended to enable distributed adaptation in the CD-JRA-2.3.8 deliverable.
- CD-JRA-2.3.6 [10] untitled "policies and strategies for adaptation from the infrastructure" describes the elements of infrastructure mechanisms that enable or support the specification of policies and strategies able to be used to adapt the infrastructure. The works presented in the CD-JRA-2.3.8 deliverable build upon and extend these mechanisms to make distributed and multi-level adaptations possible.

As you can see with the relationship to the other deliverables, the works described in this deliverable is clearly related to the works proposed on the WP-JRA-1.2 that define methodologies to build cross-layer adaptation and monitoring systems. It is also related with WP-JRA-1.3 that define "principles, techniques and methods to ensuring end-to-end quality provision and SLA conformance by taking a holistic view on service infrastructure, service composition and coordination, and business process management and by employing SLAs as key elements to guide the integrated, cross-layer monitoring, and adaptation".

1.2 Deliverable Structure

Firstly, section 1.1.2 describes the relationship with other workpackages and deliverables. In section 2.1, we remind the definition of the adaptation, the layers, the multi-level adaptation also known as cross layer adaptation (see the end of section 2.1) and provide some details about each of them. Then, in section 2.2, we provide some concrete example which highlights the need of cross-layer adaptation and especially on distributed systems. In section 2.3, we present very basic schemes to manage cross layer adaptation. Finally, in chapter 3, we present the work done in this workpackage to manage distributed and cross layer adaptation.

Chapter 2

Cross-Layers Adaptations

2.1 Cross-layer adaptation

2.1.1 Adaptation process

To define cross-layer adaptation, we previously need to define what is adaptation. A self-adaptive software makes decisions on its own, using high-level policies; it will constantly check and optimize its status and automatically adapt itself to an evolving context. The figure 2.1 describes an adaptation framework following the autonomic manager described by Kephart and Chess [21].

In this framework, we identify 4 phases called Monitoring (described in [7], Analysis (also called Decision [9, 10], Planning and Execution (MAPE). Each phase triggering the next one: the monitoring gather contextual information used by the analyzing part to decide whether an adaptation is needed or not; from this need, the planning part builds an execution plan to be executed by the execution part.

The contextual information is gathered by probes through events and measures. Events can trigger adaptation while measures are done on demand by the analyzing part when complementary information is needed. The monitoring is not only platform specific, it can also be application or domain specific when adaptation is not due to resources. The application itself can be monitored, for self-healing purpose for example, by a machine learning software, the user or by using ad-hoc metrics.

When receiving an event, the analysis phase chooses if an adaptation is needed by following a specific decision policy and using some information requested to the system. By using decision policy, the analysis phase is generic and can be used for different systems; only the policy is specific to each system adaptation.

Once an adaptation is chosen, the analysis sends a strategy to the planning part. The planning has to work out how to apply the strategy to the system to adapt; the implementation of the planning refers to a guide. This guide provides some information about how to decompose the strategies into elementary tasks to be executed. Like the decision policies, the planning guide may be specific to each adaptation system and so the planning phase is generic.

Then the plan of actions is sent to the executing part. Its role is to execute each action specified in the plan, taking into account the execution of the system to adapt. To do so, the execution phase interact with the system to adapt and may intercept the execution flow to execute adaptation actions.

The system described on the framework may represent something different according to points of view. In the case of Service-Based Applications (SBA), the system is most of time viewed as the application itself but it can also encompass the underlying architecture that host the services. Indeed this architecture is composed by a set of heterogeneous machines. A machine is able to communicate with some others in order to allow service(s) interoperation (communication). A machine can range from small devices like smartphones to more powerful ones like clouds, including personal computers and laptops. Moreover machines are running specific operating systems and some middlewares providing all the necessary core services needed such as communication facilities and resource brokering and they are

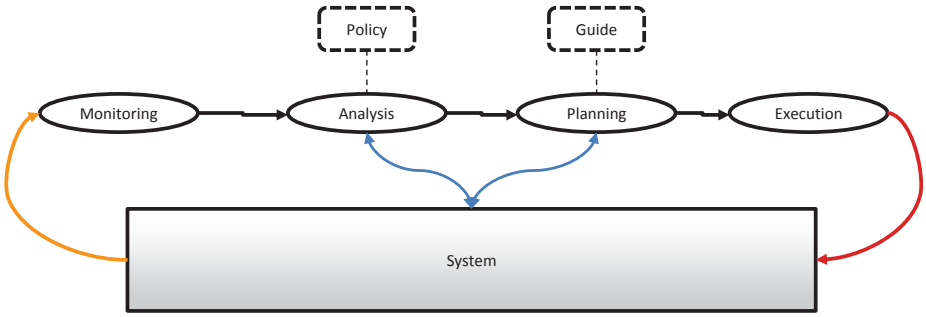


Figure 2.1: General adaptation framework

distributed which means that operating systems and middlewares do not provide seamless data sharing. Indeed, delays can vary while communicating between machines and there will be some failures among machines. At last, applications are not only composed by services and business processes but may also include humain beings.

2.1.2 Adaptation capabilities and layers

Every layer composing a system can be adapted in various ways. These ways can be classified according to several criteria (some of them are also described on [8]. The simplest one, "parameter adaptation" consists in changing the value of a parameter, for example to modify the frame rate on a video or to change the time between harvesting and processing in the Scube Wine production case study [6]. "Functional adaptation" that consists in changing a piece of code without visible logical consequences for the outside (the users). This means that this adaptation must not modify the interface for the service. For instance, one can change the bytecode of a Java service to improve its performance. Oppositely, "behavioural adaptation" consists in changing the algorithm of an element while having some visible effect for the outside. Then, "structural adaptation" concerns the modification of a composition of elements (such as services), by changing one or several links between them. For example, this can be used to replace a service by another.

All these adaptation capabilities may require various information and may also be executed with different techniques depending on what is adapted. Indeed, the adaptation of a parameter of the Operating System (the scheduling algorithm for example) doesn't use the same operations as the adaptation of a parameter of a service, because these capabilities don't have the same providers and impact different layers.

2.1.3 Adaptation layers

There exist various definition of layers. For example a layer may be defined for each level of the software stack (hardware, OS, middleware, services, workflow) but it is also possible to define layers according to the properties of the distribution of the system (one machine, one cluster, a set of connected computers on a same network, on differents networks, ...) because it can provide different adaptation capabilities. Moreover some layers may represent a set of layers due to the heterogeneity of the top layer. For example, it is possible to use different middlewares to host services and these middlewares may have different adaptation capabilities. At last, in the deliverable 2.3.2 [8], the authors define 3 other layers with the introduction of "human (or not) in the loop" properties (a human may replace the analysis phase in the adaptation process) and the need of coordination (e.g. services should be able to agree before any adaptation) before to adapt a set of related services.

Each layer is not clearly independent of the others because an adaptation at a specific layer may

impact some others. For example, migrating a service form one server to another can impacts at least to layers: at the service layer, the composition between services has to take into account the new address of the service and the infrastructure layer may have to deploy a host (virtual machine or middleware) for the service. Such an adaptation can be useful to do load balancing and thus improve the QoS.

The cross layer adaptation can be defined in multiple ways. First of all, it can be seen as a set of adaptation that are applied at the same time on different layers. In that case, there are multiple adaptation process (MAPE) where each have access to the information of a single layer. It can also be seen as a single adaptation that impact multiple layers. In that case, there is only one adaptation process but each part of it have access to information of multiple layers.

We always speak about cross layer adaptation instead of multi-level adaptation because cross layer adaptation explicits that adaptations on differents layers can have impacts and/or interactions with the others while multi-level only explicits there are some adaptations at different layers.

2.2 The needs of cross layer adaptations

Each level can and must adapt itself to fulfill the SLA constraints. Thus the cross layer adaptation is a strong issue to ensure the consistency and the coordination between all the adaptations. Indeed, when there are multiple adaptation systems, the consequences of an adaptation at a specific level may have impacts on the others. For example, the modification of the scheduling algorithm to promote a specific software (the one that are the most used) may decrease the CPU time allowed to others and so damage their quality of services. Moreover, the cross layer adaptation allows to optimize the adaptation itself by reducing the number of actions to execute or by selecting more appropriated actions at different levels. The figures 2.2 to 2.4 details a usecase showing the difference between clearly independent adaptations and cross layer adaptation.

At the beginning, there are three applications that are executed on top of two machines. Each of these applications are service-based and execute on top of middlewares (middleware 1 and 2 are equivalent). App2 is made up of three services called *Core* which provides the set of functionality of the application, *MM1* which is an active monitoring manager that looks at some sensors on a network and allows to get aggregate values of them, and *Cache* that is used as a cache between the Core and MM1 because MM1 needs times to compute an aggregate value whereas this value doesn't evolve to much.

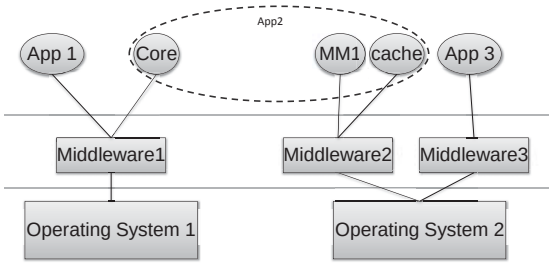
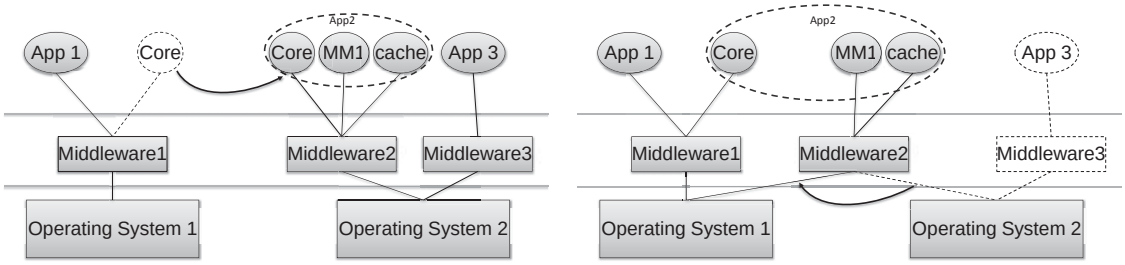


Figure 2.2: State of the usecase before adaptation

During the execution of the applications, the adaptation system at the middleware level decides to migrate the core service (Figure 2.3(a)) to reduce bandwidth used and so increases the response time between core and MM1 (or cache). Meanwhile, the App3 is stopped on the resource 2 and the adaptation system at the resource level decides to migrate the middleware 2 on resource 1 (Figure 2.3(b)).

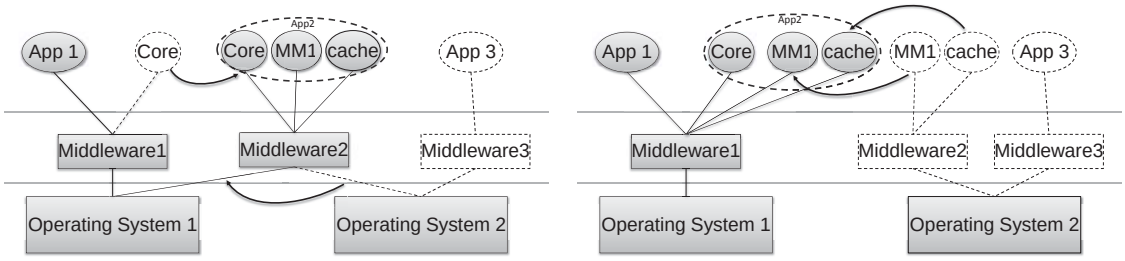
If the two adaptations are seen as independent, then the core will be migrated on the middleware 2 and this middleware 2 will be migrated on the resource 1 (Figure 2.4(a)) while the migration of the middleware 2 would have been enough. This shows why taking into different layers in a an adaptation is so important. Furthermore many possibilities exist to do these two adaptations and maybe one of them



(a) State of the usecase after service layer adaptation (b) State of the usecase after OS layer adaptation

Figure 2.3: State of the usecase after single adaptation

is more efficient than the others. For exemple, if a service migration between middlewares is less time consuming than middleware migration on top of resources, it will be better to migrate MM1 and Cache to the middleware 1 instead of migrate the complete middleware 2 (Figure 2.4(b)).



(a) State of the usecase after both layer adaptations (b) State of the usecase after cross layer adaptation

Figure 2.4: Independant layer adaptations and cross layer adaptation

2.3 How to manage multi-level/cross layer adaptations

Managing multi-level/cross layer adaptation can be done with different solutions according to the behavior of each adaptation system and the layers. All these approaches have some common requirements to be able to manage cross-layer. These requirements are described on [30]. One of the solutions consists to manage all the layers with only one adaptation system that is able to get all the needed information of all the elements in each layers to adapt the overall system. The complexity of this kind of solutions lies in the way to build such a global description of the system to have enough information to be able to take decisions.

Another solution consists to coordinate adaptation systems. Thus each adaptation system must ask validation and agreement to the other adaptation systems before to enact its adaptations. The complexity of this kind of solutions lies in the way to coordinate the adaptation systems. Indeed this solution is interesting when already existing adaptation systems must be used and when these adaptation systems have different implementations and different communication languages that are difficult to make interoperable.

Chapter 3

Contributions

3.1 Overview of the contributions

The core of this deliverable is constituted by a collection of published papers attached as appendices. These papers summarize research work carried out in or related to JRA-WP-2.3 according to its long-term goals, with a focus on distributed and multi-level adaptation. Mainly two complementary paradigms emerge to manage cross-layer adaptations: the approach based on chemical programming and another one based on distributed frameworks. This section presents the contributions in this order, followed works either using a different perspectives or targeting a specific problem.

Table 3.1 summarizes the contributed papers by the problem they address and the global approach taken to this end.

Problem addressed	Approach	Ref.	Paper
Dynamic workflow instantiation and execution	Chemical metaphor	Section 3.2	C. Di Napoli, M. Giordano, J.L. Pazat, and C. Wang. A chemical based middleware for workflow instantiation and execution. *Towards a Service-Based Internet*, 2010
The chemical machine: an interpreter for the Higher Order Chemical Language	Chemical metaphor	Section 3.3	V. Rajcsányi and Z. Németh. The chemical machine: an interpreter for the higher order chemical language. In *Proceedings of the Euro-Par Workshops 2011, Bordeaux*. Springer, 2012
Coordinated adaptations	Framework	Section 3.4	F. André, E. Daubert, and G. Gauvrit. Distribution and Self-Adaptation of a Framework for Dynamic Adaptation of Services. In *The Sixth International Conference on Internet and Web Applications and Services (ICIW)*, pages 16–21, St. Maarten, 03 2011. IARIA
Prevent SLA violations	Framework	Section 3.5	E. Schmieders, A. Micsik, M. Oriol, K. Mahbub, and R. Kazhamiakin. Combining sla prediction and cross layer adaptation for preventing sla violations. In *Proceedings of the 2nd Workshop on Software Services: Cloud Computing and Applications based on Software Services, Timisoara, Romania*, 2011
Cross layer monitoring and adaptation	Framework	Section 3.6	C. Zeginis, K. Konsolaki, K. Kritikos, and D. Plexousakis. Ecmaf: An event-based cross-layer service monitoring and adaptation framework. In *5th Workshop on Non-Functional Properties and SLA Management in Service-Oriented Computing (NFPSLAM-SOC'11) co-located with ICSOC 2011*. Springer, 2011
SLA enforcement	Framework	Section 3.7	A. Lage Freitas, N. Parlavantzas, and J-L. Pazat. A QoS Assurance Framework for Distributed Infrastructures. In *MONA'10: Proceedings of The Third International Workshop on Monitoring, Adaptation and Beyond*. ACM, 2010
Distributed adaptation strategies enforcement	Framework	Section 3.8	A. Lage Freitas and J-L Pazat. A Self-Adaptable Approach for Easing the Development of Grid-Oriented Services. In *IEEE International Conference on Computer and Information Technology (CIT 2010)*, Bradford, UK, 06 2010
Cross-layer adaptation manager	Manager	Section 3.9	A. Zengin, A. Marconi, and M. Pistore. Clam: cross-layer adaptation manager for service-based applications. In *Proceedings of the International Workshop on Quality Assurance for Service-Based Applications*, QASBA '11, pages 21–27, New York, NY, USA, 2011. ACM
SLA enforcement	Decision procedure	Section 3.10	P. Leitner, W. Hummer, B. Satzger, and S. Dustdar. Stepwise and Asynchronous Runtime Optimization of Web Service Compositions. In *Proceedings of the 12th International Conference on Web Information System Engineering (WISE 2011)*, 2011
On-demand service deployment and execution in federated Clouds	Architecture	Section 3.11	A.C. Marosi, G. Kecskemeti, A. Kertesz, and P. Kacsuk. Fcm: an architecture for integrating iaas cloud systems. In *CLOUD COMPUTING 2011, The Second International Conference on Cloud Computing, GRIDs, and Virtualization*, pages 7–12, 2011
Enhanced monitoring for adaptive scheduling in federated Clouds	Architecture	Section 3.12	A. Kertesz, G. Kecskemeti, M. Oriol, A. Marosi, X. Franch, and J. Marco. Integrated monitoring approach for seamless service provisioning in federated clouds. In *In PDP 12: Proceedings of the 20th Euromicro International Conference on Parallel, Distributed and Network-Based Computing*. IEEE CS, 2012

Table 3.1: Contributed papers by addressed problem.

3.2 A Chemical Based Middleware for Workflow Instantiation and Execution

The paper [13] proposes a middleware architecture for dynamic workflow instantiation and execution. The middleware integrates late binding service selection mechanisms with a workflow execution module. The selection mechanisms are carried out at two levels. The first occurs at the composition request level and it allows for the selection of abstract services by taking into account global QoS requirements on the composition of services, such as the overall delivery time and the total price. An abstract service is an interface specifying a required functionality of the workflow, and the corresponding QoS offered by a provider for that functionality without referring to a specific service implementation. The global QoS requirements usually concern the application and/or business levels.

The second selection mechanism takes place at the concrete binding level, and it allows to select actual service implementations, i.e. concrete services, based on some local QoS dimensions (i.e. referring to a single service), such as performance, reliability. These requirements usually concern system and/or middleware levels. The middleware architecture implementation is based on the chemical programming model, so that the dynamic selection of both abstract and concrete services is realized as an autonomous and always running process that can react to environmental changes affecting system behaviour in the form of chemical perturbations.

The middleware can be used as a component of an adaptation architecture making it possible to process adaptation needs coming from different levels. In the present work strategies for each level are not specified, since the main contribution is to model the dynamic selection of abstract and concrete services as a chemical process always running and always ready to process any change in the environment. In such a way adaptation needs coming from different levels can be continuously processed and the corresponding adaptation strategies can be inserted dynamically in the chemical system for each coming need, so providing support at the infrastructure level for multi-level adaptation.

3.3 The chemical machine: an interpreter for the Higher Order Chemical Language

Previous and ongoing research work has proven the viability of non-conventional approaches, particularly the chemical modeling to service composition and enactment. One of the research questions is how the chemical model described in the Higher Order Chemical Language (HOCL) can be realized and executed. This work focuses solely on creating an interpreter for HOCL that enables native execution of the entire language; the applicability of the HOCL to service composition has been investigated and published in other works, e.g. [12] [4] and reported in [10]. The major obstacle of realizing the chemical model is its intrinsic semantic difference from von Neumann architectures and imperative languages.

Such an unconventional computing model requires a full reconsideration of implementation means: we completely rejected all techniques related to imperative languages instead, carefully analyzed related models and distilled similar features that can be re-used in a chemical framework. In this way the concept of the chemical interpreter (i) follows the notion of abstract interpreter engines of declarative languages such as Prolog, Lisp, where HOCL is transformed into an intermediate language that is interpreted by the engine; (ii) is based on a production system that lends its state-of-the-art pattern matching mechanism nevertheless, needs modifications to support the hierarchical notion of knowledge base of the chemical semantics and fulfill other technical challenges; and (iii) several additional mechanism ensure the proper chemical behaviour.

A proof-of-concept implementation of the interpreter was realized that supports the entire HOCL language and has a graphical user interface and a basic support for tracing and debugging. The interpreter with its additional tools is able to support more complex experiments in the realm of the chemical model.

3.4 Distribution and Self-Adaptation of a Framework for Dynamic Adaptation of Services

The paper [1] presents a generic framework for self-adaptation of services and service based applications. The basic steps of an adaptation framework are Monitoring, Analysis, Planning and Execution, following the MAPE model proposed in [21] and presented in section 2.1.1. This work intend to improve this basic framework by refining each step of the MAPE model, in particular by providing elements that cope with the distribution of the application and the infrastructure layer. The adaptation system can itself be distributed for the purpose of scalability or to better match the heterogeneity of the environment. Moreover, it can be adaptable, allowing to take into account unforeseen situations.

This system called SAFDIS for Self-Adaptation For DIstributed Services fully exploits the advantages of the framework concept [32]. It gives a frame, paradigms and rules to develop and implement adaptation mechanisms, as well as the liberty and the flexibility for the developer to specialize its system according to its specific needs. Using this framework, the task of developing concrete adaptation systems for some applications, services or infrastructures will be facilitated as many of the different elements that may be composed in adaptation systems are exposed, their interfaces clearly defined, the relationships between them coherently specified.

The SAFDIS framework is build as a set of services, providing functionalities useful to build an adaptation system. Not all functionalities are necessarily needed for each instantiation of an adaptation system. For instance we provide a *negotiator* service to negotiate the adaptation decisions when SAFDIS is distributed on several nodes; this service is not useful when SAFDIS is build as a unique centralized adaptation system.

In the context of the S-Cube infrastructure, the work presented in the paper addresses the Multi-level Adaptation research challenge of WP2.3, since it proposes a framework for building self-adaptable services that is aware of the different layers. Moreover, it enables users of this framework to propose and fine-tune adaptation policies and strategies for their service-based applications.

3.5 Combining SLA Prediction and Cross Layer Adaptation for Preventing SLA Violations

Service-based Applications (SBA) are deployed in highly dynamic and distributed settings, where various parts of the constituent components – services and their infrastructure – are controlled by different third parties. In such a loosely coupled environment, adaptation capabilities are needed to manage deviations and unforeseen situations which might lead to negative consequences (e.g. contractual penalties). Current approaches either focus on cross-layer-adaptation or the prevention of SLA violations. In contrast to this, the approach presented in this paper combines both. The paper presents an architecture as a generic framework for the management of arising problems during service execution. Multiple adaptation mechanisms are available to react on adaptation needs, acting on different layers of the SBA (including e.g. the composition layer and the infrastructure layer). The final goal of the cross-layer adaptation capability is to avoid the violation of agreed Service Level (in SLAs) and thus ensure the benefits of SBAs for both customers and providers. The novelty of the approach is the exploitation of all SBA layers (BPM, SCC and SI) for the prevention of SLA violations. The identification of adaptation needs is based on SLA prediction, which uses assumptions on the characteristics of the running execution context. Multiple adaptation mechanisms are available to react on the adaptation need, acting on different layers of the SBA. The adaptation strategy chooses the right adaptation mechanism, coordinated by a multi-agent community. In the future, we plan to further generalize the mechanism for handling the adaptation management cycle of detection, selection and enactment, and to incorporate more components, event types and adaptation capabilities.

3.6 ECMAF: An Event-Based Cross-Layer Service Monitoring and Adaptation Framework

The paper [34] outlines a cross-layer monitoring and adaptation framework that is based on monitored events. In addition, a layer-based taxonomy of adaptation-related events and a meta-model describing the dependencies among components of a cross-layer system are introduced to pinpoint the need for such a type of framework.

Although several techniques have been proposed towards monitoring and adaptation of Service-Based Applications (SBAs), few of them deal with cross-layer issues. This paper proposes a framework, able to monitor and adapt SBAs across all functional layers. This is achieved by using techniques, such as event monitoring and logging, event-pattern detection, and mapping between event patterns and appropriate adaptation strategies. In addition, a taxonomy of adaptation-related events and a meta-model describing the dependencies among the SBA layers are introduced in order to capture the cross-layer dimension of the framework. Finally, a specific case study is used to illustrate its functionality.

3.7 A QoS Assurance Framework for Distributed Infrastructures

Maintaining the promised QoS properties is a major concern for service providers in order to avoid losses and penalties. On the one hand, most research on QoS assurance in SOAs targets composite services, where managing QoS typically involves replacing services by other services more suitable for the composition [18]. However, such work does not address how individual, atomic services guarantee QoS properties, which unavoidably requires controlling the underlying infrastructure. On the other hand, further work that addresses QoS assurance of atomic services such as [2, 19, 17], fails to address the SLA life-cycle. This paper proposes the Qu4DS framework which addresses QoS assurance for atomic services by taking into account the complete SLA life-cycle. In particular, atomic services that build on large-scale distributed infrastructures, such as clusters, grids or clouds. Importantly, the framework supports dynamic adaptation, i.e., support for monitoring and automatically modifying service behavior and resource usage.

Currently, the Qu4DS framework supports providers in enforcing quality properties in distributed environments. The framework has three main features. First, Qu4DS provides flexible support for dynamic adaptation, necessary for maintaining agreed SLAs in the face of fluctuating environmental conditions. Second, the framework tackles the complete SLA life-cycle, from contract negotiation to service termination. Lastly, the framework lies on a modular design, extensible structure, cleanly separating the different QoS management functions and service implementations. In order to increase its applicability, Qu4DS builds on the standard SAGA API that provides a uniform and consistent interface to the most commonly used distributed functionality. A prototype of the framework has been developed; this prototype uses the XtreemOS [11] grid and leverages the MAPE Autonomic Computing control loop [21]. Furthermore, the paper presents initial experimental results that provide evidence that Qu4DS can effectively reduce SLA violations in dynamic environments.

3.8 A Self-Adaptable Approach for Easing the Development of Grid-Oriented Services

The Service-Oriented Architecture (SOA) proposes a support which addresses service composition as well as basic and high-level service management features. However, the SOA was not conceived to take into account non-trivial computational capabilities for services that require high-performance computing. On the other hand, grid computing [15] leverages low-cost and heterogeneous resources in order to provide a distributed infrastructure for high-performance computing in large-scale. Grids precisely define resource access through the management of Virtual Organizations (VO) in a dynamic fashion.

Thereby, grids offer an infrastructure suitable for supporting services that have high-performance computing needs. In order to use the grid, such services rely on the grid job abstraction by submitting and managing them through the grid interface. However, in spite of interesting efforts as the Simple Grid API (SAGA) [16] that addresses to ease the use of grids, grid usage still remains complex. This drawback becomes an inconvenience for the development of grid-oriented services.

This paper proposes a self-adaptable support for easing the conception of grid-oriented services. The self-adaptive aspect leverages Dynaco (Dynamic Adaptation for Components) [3] by following its component-based design. The architecture proposed takes advantage of the iPOJO [14] component model owing to its dynamic and flexible capabilities for developing and maintaining services on top of OSGi [29] platforms. iPOJO allows to separate the functional code from non-functional concerns which are modularly addressed in handlers. Thereby, this paper proposes an iPOJO handler which automatically addresses the management of jobs on grids. Moreover, the XtreemOS [11] grid platform is used as case study.

3.9 CLAM: Cross-layer Adaptation Manager for Service-Based Applications

CLAM [35] is a cross-layer adaptation management platform where adaptation actions are analyzed and validated for their consistency with the overall service-based system (SBS). It relies on two pillars:

1- Cross-layer system meta model: It is basically a directed graph where nodes represent the system elements from different layers and the edges represent the relations between these elements. Apart from system elements and their relationships, we have the tools associated with the elements. They are the analysis and adaptation mechanisms that are available for the SBS: (i) Analyzers check the compatibility of a new adaptation for a system element that they are associated with. (ii) Solvers get an incompatibility problem triggered by an analyzer and try to propose an adaptation to handle the problem. (iii) Enactors implement final adaptation strategies validated by CLAM.

2- Rules and the algorithm: We perform the entire adaptation analysis as a continuous execution of predefined rules. These rules and the overall algorithm determine how to navigate the SBS meta model to identify the system elements affected by an adaptation, and subsequently how to decide which analyzers and solvers to invoke, and finally how to gradually construct a cross-layer adaptation tree upon receiving results from those external tools. Cross-layer adaptation tree is basically the output of the analysis process where the branches correspond to alternative adaptation strategies validated by CLAM.

3.10 Stepwise and Asynchronous Runtime Optimization of Web Service Compositions

Providers of service compositions often guarantee important customers so called service level agreements (SLAs), which are essentially collections of target qualities (service level objectives, SLOs) and monetary penalties that go into effect if the promised target quality cannot be achieved. Hence, providers of service compositions have strong incentives to prevent cases of SLA violation. One promising approach to achieve this is predicting violations at runtime, before they have actually occurred, and using adaptation to prevent these violations [27, 20]. An important part of runtime adaptation is deciding which adaptations to apply, which we have modeled as an optimization problem in other work [25]. However, a limitation of the approach presented in that papers is that it is inherently assumed that the decision problem can be solved in time, before the first adaptation has to be applied. Even using fast meta-heuristics this is not guaranteed, especially so for shorter service compositions.

In this paper, we improve on this by proposing an asynchronous and step-wise optimization model, in which we do not generate a decision for all adaptations at once. Instead, we run the optimization in parallel to the execution of the composition. At so-called decision points we make a decision for only

those adaptations that absolutely need to be decided at that moment, update the optimization problem according to the made decisions, and continue optimizing for all remaining possibilities for adaptation. Note that the contribution presented here is not specific to the approach presented in [25]. Much more, the same ideas are applicable for other runtime adaptation approaches facing similar problems as well, e.g., [20].

3.11 FCM: an Architecture for Integrating IaaS Cloud Systems

Highly dynamic service environments require a novel infrastructure that can handle the on demand deployment and decommission of service instances. Cloud Computing offers simple and cost effective outsourcing in dynamic service environments and allows the construction of service-based applications extensible with the latest achievements of diverse research areas, such as Grid Computing, Service-oriented computing, business processes and virtualization. Virtual appliances (VA) encapsulate metadata (e.g., network requirements) with a complete software system (e.g., operating system, software libraries and applications) prepared for execution in virtual machines (VM). Infrastructure as a Service (IaaS) cloud systems provide access to remote computing infrastructures by allowing their users to instantiate virtual appliances on their virtualized resources as virtual machines.

Nowadays, several public and private IaaS systems co-exist and to accomplish dynamic service environments users frequently envisage a federated cloud that aggregates capabilities of various IaaS cloud providers. These IaaS systems are either offered by public service providers (like Amazon or RackSpace) or by smaller scale privately managed infrastructures. Therefore there is a need for an autonomic resource management solution that serves as an entry point to this cloud federation by providing transparent service execution for users. The following challenges are of great importance for such a mediator solution: varying load of user requests, enabling virtualized management of applications, establishing interoperability, minimizing Cloud usage costs and enhancing provider selection.

To address these issues, this paper proposes a layered architecture that incorporates the concepts of meta-brokering, cloud brokers and automated, on-demand service deployment. The meta-brokering component allows the system to interconnect the various cloud brokers available in the system. The cloud broker component is responsible for managing the virtual machine instances of the particular virtual appliances hosted on a specific infrastructure as a service provider. Our architecture organizes the virtual appliance distribution with the automatic service deployment component that can decompose virtual appliances to smaller parts. With the help of the minimal manageable virtual appliances the Virtual Machine Handler rebuilds these decomposed parts in the IaaS system chosen by the meta-broker. As a result, the cloud broker component uses the VM Handler to maintain the number of virtual machines according to the demand.

3.12 Integrated Monitoring Approach for Seamless Service Provisioning in Federated Clouds

Cloud Computing offers simple and cost effective outsourcing in dynamic service environments, and allows the construction of service-based applications using virtualization. By aggregating the capabilities of various IaaS cloud providers, federated clouds can be built. Managing such a distributed, heterogeneous environment requires sophisticated interoperation of adaptive coordinating components. However, user demands are frequently overextending the boundaries of a single cloud system. In these cases, they need to handle the differences between the various cloud providers and have to negotiate their requirements with multiple parties. Federated clouds aim at supporting these users by providing a single interface on which they can transparently handle the different cloud providers as they would do with a single cloud system.

This paper proposes an architecture to construct federated cloud systems that not only offers a single interface for its users but it automatically manages their virtual machines independently from the currently applied cloud system. We argue that efficient cloud selection in federated clouds requires a cloud monitoring subsystem that determines the actual health status of the available IaaS systems. We present an architecture that incorporates the concepts of on-demand service deployment, cloud brokering and meta-brokering, supported by an integrated monitoring solution. The meta-brokering component allows the system to interconnect the various cloud brokers available in the system. It is also responsible for selecting a proper execution environment managed by a cloud broker. This selection process relies on a sophisticated monitoring component, which provides up-to-date service availability and infrastructure reliability based on specific monitoring metrics. The cloud broker component is responsible for managing the virtual machine instances of the particular virtual appliances hosted on a specific infrastructure as a service provider. Our architecture also organizes virtual appliance distribution with its automatic service deployment component that can decompose and deliver virtual appliances in smaller parts. We also provided a simplified scenario for exemplifying the operation of our proposed solution using a minimal metric monitoring service. Our future work targets the evaluation of the proposed architecture with ordinary services deployed at different cloud providers.

Chapter 4

Conclusion

This deliverable presented our results related to elements and aspects of infrastructure level mechanisms to specify adaptation policies and strategies for distributed and multi-level adaptation. These mechanisms are targeting the objectives of the "Infrastructure Mechanisms for the Run-Time Adaptation of Services" research thread of the work package. As the last deliverable of this research thread, the proposed mechanisms build upon its previous deliverables, mainly the deliverable CD-JRA-2.3.6 "Specifications of policies and strategies for adaptation".

The deliverable is a collection of scientific papers published in conference proceedings and organized along the research directions of WP-JRA-2.3. The papers all have been peer reviewed which ensures that the papers represent significant contributions to service-based system research and they demonstrate progress in the WP. The positioning of the papers within the adaptation framework, their relationship to the WP-JRA-2.3 research goals and vision and to other research WPs are exposed in Section 3. These research directions are focused around various solutions to manage cross layer and distributed. First, we have applied the chemical metaphor in 3.2 to reveal the possibilities of composing services while taking into account the multi-level or cross-layer properties and the distribution of the infrastructure. Next, we have demonstrated that framework-based approach [1, 34, 23, 24, 35, 28, 22] can be extended to manage and enact cross-layer and distributed dynamic adaptation of services based applications.

The papers presented in this deliverable proposed mechanisms that focused on distributed and multi-level aware components of service based infrastructures. They demonstrate that viable approach exists to build self-adaptive and self-healing services in the dynamic environment of the future internet.

Bibliography

[1] F. André, E. Daubert, and G. Gauvrit. Distribution and Self-Adaptation of a Framework for Dynamic Adaptation of Services. In *The Sixth International Conference on Internet and Web Applications and Services (ICIW)*, pages 16–21, St. Maarten, 03 2011. IARIA.

[2] Siegfried Benkner and Gerhard Engelbrecht. A Generic QoS Infrastructure for Grid Web Services. *Advanced International Conference on Telecommunications / Internet and Web Applications and Services, International Conference on*, 0:141, 2006.

[3] Jérémy Buisson, Françoise André, and Jean-Louis Pazat. Dynamic adaptation for Grid computing. In *EGC '05: Proceedings of The European Grid Conference*, pages 538–547, Amsterdam, June 2005.

[4] Manuel Caeiro, Zsolt Németh, and Thierry Priol. A chemical model for dynamic workflow coordination. In *PDP*, pages 215–222, 2011.

[5] State of the art report, gap analysis of knowledge on principles, techniques and methodologies for monitoring and adaptation of sbas. (available on Scube website).

[6] Collection of industrial best practices, scenarios and business cases. (available on Scube website).

[7] Taxonomy of adaptation principles and mechanisms. (available on Scube website).

[8] Basic requirements for self-healing services and decision support for local adaptation. (available on Scube website).

[9] Decision support for local adaptation. (available on Scube website).

[10] Policies and strategies for adaptation from the infrastructure. (available on Scube website).

[11] Toni Cortes, Carsten Franke, Yvon Jégou, Thilo Kielmann, Domenico Laforenza, Brian Matthews, Christine Morin, Luis Pablo Prieto, and Alexander Reinefeld. XtreemOS: a Vision for a Grid Operating System. Technical report, XtreemOS Consortium, May 2008.

[12] C Di Napoli, M. Giordano, Z. Németh, and N Tonellotto. Adaptive instantiation of service workflows using a chemical approach. In *Euro-Par Workshops*, pages 247–255, 2010.

[13] C. Di Napoli, M. Giordano, J.L. Pazat, and C. Wang. A chemical based middleware for workflow instantiation and execution. *Towards a Service-Based Internet*, 2010.

[14] Clement Escoffier, Richard S. Hall, and Philippe Lalanda. iPOJO: an Extensible Service-Oriented Component Framework. *SCC '07: Proceedings of The IEEE International Conference on Services Computing*, pages 474–481, July 2007.

[15] Ian Foster. What is the Grid? - A Three Point Checklist. *GRIDtoday*, 1:22–25, 2002.

[16] Tom Goodale, Shantenu Jha, Hartmut Kaiser, Thilo Kielmann, Pascal Kleijer, Andre Merzky, John Shalf, and Christopher Smith. A Simple API for Grid Applications (SAGA). Technical Report GFD-R-P.90, Open Grid Forum, 2008.

[17] Peer Hasselmeyer, Bastian Koller, Lutz Schubert, and Philipp Wieder. Towards SLA-Supported Resource Management. In *HPCC '06: Proceedings of the 2006 International Conference on High Performance Computing and Communications*, pages 743–752. Springer, 2006.

[18] Julia Hielscher, Andreas Metzger, and Raman Kazhamiakin. Taxonomy of Adaptation Principles and Mechanisms. Technical Report Deliverable # CD-JRA-1.2.2, S-CUBE Consortium, 2009.

[19] Jose Antonio Parejo and Pablo Fernandez and Antonio Ruiz-Cortés and José María García. SLAWs: Towards a Conceptual Architecture for SLA Enforcement. In *Services, IEEE Congress on*, volume 0, pages 322–328. IEEE Computer Society, 2008.

[20] Raman Kazhamiakin, Branimir Wetzstein, Dimka Karastoyanova, Marco Pistore, and Frank Leymann. Adaptation of Service-Based Applications Based on Process Quality Factor Analysis. In *MONA+*, pages 395–404, 2009.

[21] Jeffrey O. Kephart and David M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, 2003.

[22] A. Kertesz, G. Kecskemeti, M. Oriol, A. Marosi, X. Franch, and J. Marco. Integrated monitoring approach for seamless service provisioning in federated clouds. In *In PDP 12: Proceedings of the 20th Euromicro International Conference on Parallel, Distributed and Network-Based Computing*. IEEE CS, 2012.

[23] A. Lage Freitas, N. Parlavantzas, and J-L. Pazat. A QoS Assurance Framework for Distributed Infrastructures. In *MONA'10: Proceedings of The Third International Workshop on Monitoring, Adaptation and Beyond*. ACM, 2010.

[24] A. Lage Freitas and J-L Pazat. A Self-Adaptable Approach for Easing the Development of Grid-Oriented Services. In *IEEE International Conference on Computer and Information Technology (CIT 2010)*, Bradford, UK, 06 2010.

[25] P. Leitner, W. Hummer, and S. Dustdar. Cost-Based Optimization of Service Compositions. *IEEE Transactions on Services Computing (TSC)*, 2011. To appear.

[26] P. Leitner, W. Hummer, B. Satzger, and S. Dustdar. Stepwise and Asynchronous Runtime Optimization of Web Service Compositions. In *Proceedings of the 12th International Conference on Web Information System Engineering (WISE 2011)*, 2011.

[27] Philipp Leitner, Anton Michlmayr, Florian Rosenberg, and Schahram Dustdar. Monitoring, Prediction and Prevention of SLA Violations in Composite Services. In *ICWS'10*, pages 369–376, 2010.

[28] A.C. Marosi, G. Kecskemeti, A. Kertesz, and P. Kacsuk. Fcm: an architecture for integrating iaas cloud systems. In *CLOUD COMPUTING 2011, The Second International Conference on Cloud Computing, GRIDs, and Virtualization*, pages 7–12, 2011.

[29] OSGi Alliance. OSGi Service Platform Core Specification. Release 4, Version 4.1, April 2007.

[30] Baseline of adaptation and monitoring principles, techniques, and methodologies across functional SBA layers. (available on Scube website).

[31] V. Rajcsányi and Z. Németh. The chemical machine: an interpreter for the higher order chemical language. In *Proceedings of the Euro-Par Workshops 2011, Bordeaux*. Springer, 2012.

[32] Dirk Riehle and Thomas Gross. Role model based framework design and integration. In *In OOPSLA 98: Proceedings of the 1998 Conference on Object-Oriented Programming Systems, Languages, and Applications*, pages 117–133. ACM Press, 1998.

[33] E. Schmieders, A. Micsik, M. Oriol, K. Mahbub, and R. Kazhamiakin. Combining sla prediction and cross layer adaptation for preventing sla violations. In *Proceedings of the 2nd Workshop on Software Services: Cloud Computing and Applications based on Software Services, Timisoara, Romania*, 2011.

[34] C. Zeginis, K. Konsolaki, K. Kritikos, and D. Plexousakis. Ecmaf: An event-based cross-layer service monitoring and adaptation framework. In *5th Workshop on Non-Functional Properties and SLA Management in Service-Oriented Computing (NFPSLAM-SOC'11) co-located with ICSOC 2011*. Springer, 2011.

[35] A. Zengin, A. Marconi, and M. Pistore. Clam: cross-layer adaptation manager for service-based applications. In *Proceedings of the International Workshop on Quality Assurance for Service-Based Applications*, QASBA '11, pages 21–27, New York, NY, USA, 2011. ACM.