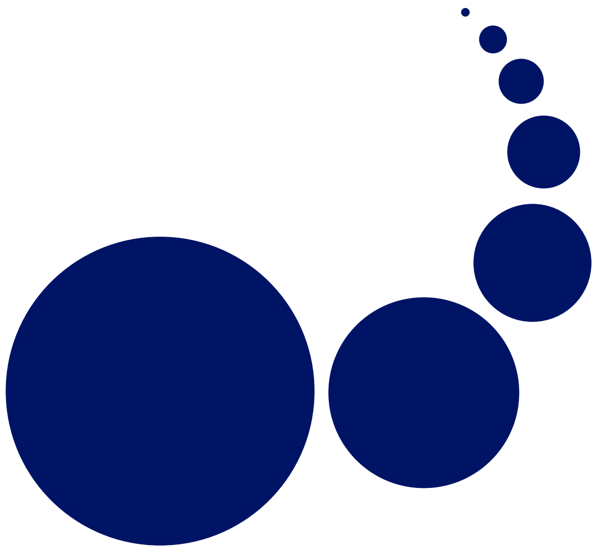
SCALABLE COMPUTING
Practice and Experience

Special Issue: Selected Papers From the 2nd Workshop on Software Services

Editors: Dana Petcu and Jose Luis Vazquez-Poletti



Volume 12, Number 4, December 2011

ISSN 1895-1767

SUBSCRIPTION INFORMATION: please visit http://www.scpe.org

Scalable Computing: Practice and Experience

Volume 12, Number 4, December 2011

TABLE OF CONTENTS

Scalable Computing: Practice and Experience
Volume 12, Number 4, pp. iii–iv. http://www.scpe.org
ISSN 1895-1767
© 2011 SCPE

INTRODUCTION TO THE SPECIAL ISSUE ON SELECTED PAPERS FROM 2ND WORKSHOP ON SOFTWARE SERVICES

Dear SCPE readers,

This December issue of Scalable Computing: Practice and Experience is the final of two issues devoted to Cloud Computing and Applications based on Software Services.

To ensure the high quality of this issue, a reduced number of contributions that were presented at 2nd WoSS (Timisoara, June 6-9 2011) were chosen and an invitation to provide an extended version was sent. The reviewers were the same as in the Workshop so the peer-review process was exhaustive, from the abstract status prior to the Workshop until the final version that you will be reading. Additionally, the priceless feedback gathered during 2nd WoSS definitely increased the contribution quality.

As said in the previous issue, many of the authors come from countries that joined the European Union recently. This demonstrates that the European Research Family is not only growing up but also reunites around the Cloud Computing table, a bleeding edge and promising area which is expected to bring much outstanding outcome.

Dana Petcu,
Computer Science Department
West University of Timisoara
and Institute e-Austria Timisoara, Romania

Jose Luis Vazquez-Poletti,
Distributed Systems Architecture Group
Faculty of Computer Science, Universidad Complutense de Madrid, Spain

Scalable Computing: Practice and Experience
Volume 12, Number 4, pp. 377–384. http://www.scpe.org
ISSN 1895-1767
© 2011 SCPE

TOWARDS A TAXONOMY OF WEB SERVICE COMPOSITION APPROACHES

DESSISLAVA PETROVA-ANTONOVA[*] AND ALEKSANDAR DIMOV[†]

Abstract. Service-Oriented Architecture (SOA) is a well known paradigm for development of flexible and loose coupled software applications using services that are available in a network. The latter provide business functionality through well-defined interfaces that can be dynamically discovered. Services can be aggregated into more complex ones called composite services. Currently, there exist a lot of composition approaches that serve different goals. In order to be able to comprehensively study the web-service composition process, different approaches should be analyzed and organized into appropriate taxonomy framework. This paper presents an overview of current approaches for service composition and further analyzes them toward various aspects of the composition model.

Key words: Composition model, Service-Oriented Architecture, Web services

1. Introduction.

Latest trend in software engineering is presented by the so-called Service-Oriented Architecture (*SOA*). It makes possible development of software intensive systems via loosely coupled services, which provide business functionality through contractually specified interfaces. The promise of services is for increased maintainability and scalability of systems, achieved at predictable time with less efforts and decreased cost of development.

The preferred way to realize SOA is based on web services. The basic web service infrastructure founded on standards like WSDL, SOAP and UDDI provides simple interaction between clients and web services. Functionality of the latter is often combined into composite ones that satisfy specific business goals. In such case, additional composition models, languages, as well as development and run-time environments should be elaborated. Although a lot of efforts for development of web service composition methods are available, most of them aim toward satisfying particular needs. Some composition approaches focus mainly on QoS aspects of the composition [2], [8], [9], while the goal of others is to provide web service orchestration [4], [6], [7], [12], [14]. In such context this paper aims to study the fundamental characteristics and essential aspects of different approaches. For this purpose we extend the number of the dimensions of an existing composition model and provide comparative analysis of different web service composition approaches according to these dimensions.

Some currently existing literature surveys on web service composition approaches directly relate to our work. For instance, in [21], authors present in details a large number of approaches, but only few of them are compared according to the preliminary defined criteria. Rao and SU discuss in [22] only the approaches based on AI Planning. The approaches using Petri nets, statecharts and other techniques for orchestration modeling are not considered. In this paper several categories of orchestration modeling techniques are presented and used as a comparison criterion of presented approaches. Evaluation of QoS Based web service selection techniques for service composition is given in [23]. In this paper, other dimensions of the composition process such as data modeling, transaction modeling and exception handling, are not mentioned. In our work we extend the above mentioned research, by establishing a more sound classification and comparison scheme, which could serve as a basis for a taxonomy framework of the web service composition process.

The rest of the paper abstract is organized as follows: Section 2 introduces the current web service composition approaches, Section 3 analyzes the approaches with respect to aforesaid dimensions and Section 4 concludes the paper.

2. Overview of Web Service Composition Approaches.

This section introduces the key aspects of the current web service composition approaches.

Chang and Lee [2] propose a quality driven web service composition methodology, which evaluates the quality of web services in three dimensions: QoS (quality of services), QoC (quality of contents), and QoD (quality of devices). QoS is related to the quality properties of web services such as performance, interoperability, security, and so on. QoC considers the quality of the context in which the web services work. QoD addresses the quality of devices and the physical environment in which the web services operate. The web service composition methodology uses Multi-Criteria Decision Making solution, called Preference Ranking Organization METHod

[*]Sofia University, Faculty of Mathematics and Informatics, Department of Software Engineering, 1164 Sofia, Bulgaria (d.petrova@fmi.uni-sofia.bg).

[†]Sofia University, Faculty of Mathematics and Informatics, Department of Software Engineering, 1164 Sofia, Bulgaria (aldi@fmi.uni-sofia.bg).

for Enrichment Evaluations (PROMETHEE). The proposed method for web service composition consists of six steps. First, web service composition is described as abstract workflow using Business Process Modeling Notation (BPMN). Next, quality factors from the three dimensions are selected and their quality weights are determined. Then, constraints of quality factors are defined and preference indexes for all quality factors are specified. The preference index expresses the preference of one service over another considering all quality factors. Finally, outranking flows between web service candidates are determined according to the preference indexes. The proposed methodology is implemented in a tool, named Event-driven web service composer, and is evaluated through an example process with five tasks. Twenty five web services and ten event services are developed as candidates for the execution of the process's tasks.

Qiao et al. [3] a broker based architecture for dynamic QoS monitoring and adaptation for composite web services. The web service compositions are described with Business Process Execution Language (BPEL), which is extended with a new construct, called flexPath, allowing definition of multiple execution paths of the BPEL process. The proposed architecture consists of two main components: QoS broker and BPEL compiler. The QoS broker is responsible to monitor and adapt the QoS properties. It keeps track of the execution of the BPEL process based on the schema of the running instance and the current execution point. The values of QoS properties are measured at run-time using a probing technique. The key role of the QoS broker is to decide whether the adaptation has to be triggered and if so, to find a better execution path in order to improve the QoS. The BPEL compiler instruments the BPEL process with a new logic in order to communicate with the QoS broker. The proposed components of the architecture are implemented in a prototype tool and verified through a BPEL process that invokes a number of dummy partner Web services. A disadvantage of the architecture is that it handles only Response time as QoS property. The measured Response time in the broker is not the same as those obtained from user experiences. This drawback is due to the usage a third party monitor.

Zheng and Yan [4] model web service composition problem as a Planning Graph (PG). Each web service in a composition is mapped to an action of a PG. The input parameters of the web service are mapped to the action's preconditions, and its output parameters are mapped to the action's effects. The input parameters of the composition request are mapped to the initial state of a planning problem. The output parameters of the composition request are mapped to goal propositions. The proposed algorithm takes as an input a set of actions, an initial state, and a set of goal propositions. Firstly, it assigns an initial state to a proposition set in level 0 of a simplified PG. Then, the algorithm expands PG iteratively until it reaches a level where a proposition set contains goal propositions or the fixed point level of PG. If the former happens first, then the algorithm outputs a solution. Otherwise, the solution is not produced. The performance of the proposed approach is studied through a sample repository, containing 143 web services. It is compared with the performance of Service Composition Algorithm used in Georgetown Java Software. A drawback of the proposed approach is that it sometimes produces redundant web services.

Yu and Reiff-Marganiec [5] propose a technique for web service composition using Planning as Model Checking (PMC). They translate a message based paradigm to a state oriented one by allowing a single operation in the web service to imply a state, which essentially encapsulates the change after execution of the operation. Each operation is modeled in four aspects, namely quality, domain, purpose and communication. The quality aspect captures non-functional requirements of the web service. The domain aspect describes the area of interest of the operation. The purpose aspect refers to the aim of the operation. The communication aspect defines a message exchange protocol that is used for interaction with the clients and other web services. The model of the web service operation is applied in the design of a composition framework, which has four phases. The first phase specifies the planning goal and describes initial knowledge about the client as input for the next phase. On the second phase, a plan search space is built. The third phase runs the proposed algorithm to search for plans. On the fourth phase, the clients choose the best plan. Then, the composition framework generates an executable plan that is described in BPEL. It monitors the execution of the web service composition and in case of failures automatically revises the executable plan using the alternative ones, obtained in the third phase. A disadvantage of the composition framework is that the clients need to select the best plan instead of the composition framework itself. This can be done automatically based on the non-functional properties of web services.

The AI planning is also applied to the problem of web service composition by Klusch and Gerber [6]. They propose a semantic web service composition planner, called OWLS-XPlan. OWLS-XPlan takes OWL-S web service descriptions, OWL domain description and a planning goal as input and generates a planning sequence of web services, which corresponds to the planning goal. The web service descriptions and domain descriptions are transformed in Planning Domain Definition Language (PDDL). They are used to create a PDDL plan that solves a given problem in the actual domain. OWLS-XPlan is evaluated according to completeness of planning, the average plan length in relation to the complexity of the problem definition and the average plan quality in terms of the average distance of individual plans from the optimal solution of a given problem. Bartalos and Bielikova [7] also propose a solution of semantic web service composition problem. They propose an approach, which creates web services workflows, satisfying a given goal. The goal is described as a pair: concept type and constraint. The concept type is a concept from the ontology used for semantic annotation of web services. The constraint is defined as a first order logic formula. The proposed approach generates a plan containing all possible alternative branches that lead to the goal. The condition determining which branch will be taken is evaluated during execution. The composition process continues until there is no web service which has not specified source for its input data. During the composition, the value restrictions defined in the goal are back propagated based on the pre- and post-conditions of chained services. The proposed approach is evaluated with respect of performance and usefulness.

Cardellini et al. [8] present a web service composition approach, where web service selection is carried out per flows of requests rather than per request. The web service selection problem is scaled to Linear Programming Optimization problem and takes into account QoS properties of the candidate web services. Its solution is used in the design of a service broker, which performs several tasks. First, it defines a business process in BPEL for the requested web service composition and discovers the candidate web services. Next, the service broker negotiates Service Level Agreements (SLAs) with the providers of the candidate web services. During negotiation it establishes the values of QoS properties of each web service in correspondence with a mean volume of requests generated by the broker for that service. Then, the negotiation continues with the requestor establishing the offered QoS level of the composition in correspondence with a mean volume of requests generated by the requestor to the broker. Finally, the selection of concrete web services is realized by solving the optimization problem. The service broker also collects information about web service composition usage. The collected data is used to find out whether a new solution of the optimization problem is required.

Chakhar et al. [9] describe a framework for composite web services selection based on multicriteria evaluation. The proposed solution extends the current web service architecture by adding a Multicriteria Evaluation Component (MEC) in the UDDI registry. MEC takes as input a set of composite web services and a set of evaluation criteria. Its output is a set of recommended composite web services. The set of QoS evaluation criteria is extracted from the SOAP message sent by the client to the UDDI registry in order to find a given web service. They are transformed into quantitative ones by assigning values to the qualitative data. Potential web service compositions are constructed as graphs and then evaluated according to preliminary defined rules. The proposed framework is under development, but its feasibility is shown through an illustrative example.

Aiello et al. [10] propose an algorithm that creates a web service composition, which satisfies the functional requirements of the client. The algorithm is based on an extended Breadth First Search (BFS) algorithm that uses a priority queue with cost based on Response time and Throughput. It is integrated in a system, called RuGQoS, which consist of three components: XML parser, Composition engine and BPEL code generator. The XML parser translates the WSDL, WSLA and OWL descriptions into a set of indexes where each operation name is associated with a web service and its corresponding Response time and Throughput. The Composition engine implements the BFS algorithm. The BPEL code generator converts the output from the Composition engine into business process described in BPEL.

Lecue [11] proposes an approach for web service composition based on the semantic similarities between input and output web service parameters. The semantic similarities are evaluated using semantic links that are specified by statecharts (S). The semantic links are valued with two quality criteria, namely Common description rate and Matching quality. The common description rate provides one possible measure for the degree of similarity between an output parameter of one web service and an input parameter of another one. The matching quality is a value in the range of [0, 1] that can be 1 (Exact), 3/4 (PlugIn), 1/2 (Subsume) or 1/4 (Intersection). The semantic links are also augmented with two QoS properties of web services: Execution price and Response time. Thus, the proposed approach of web service computation optimizes both QoS and the quality of semantic fit.

Sohrabi and McIlraith [12] present a template based web service composition system using Hierarchical Task Networks (HTNs). The system assumes that both web services and composition templates, specified by the clients, are described in OWL-S. OWL-S service profiles and process models are translated into HTNs. The client preferences are described with extension of Planning Domain Definition Language (PDDL3), which allows specification of preferences over how tasks are decomposed as well as over QoS properties of web services. The web service compositions generated by the proposed system adhere to policies and regulations expressed as a subset of Linear Temporal Logic (LTL). Thus, the system guarantees that the synthesized composition preserves certain properties of the world.

Pistore et al. [13] address the web service composition problem by developing a planning technique based on the Planning as Model checking approach. The planning process uses a BPEL4WS description of the external protocols and client requirements for the composition. The external protocols are represented by means of finite state machines due to their nondeterminism and partial observability. The clients requirements are expressed in a goal language, called EaGLe, and are used to navigate the planning. As a result, an executable BPEL4WS process and a monitor are generated. The monitor checks the actual interactions of the BPEL4WS process with the external web services and detects incorrect ones. The proposed planning technique is implemented in a BPM planner and is evaluated on a sample case study.

Farhan et al. [14] propose a framework for web service composition that consists of four main components: Translator, Evaluator, Composer, Execution Engine and Matching Engine. The Translator converts the client request into a form used by the framework. The Matching Engine checks for the requested web services in a web service database. If the requested web services are not found, then it starts to search in UDDI registries. The results of searching are sent to the Evaluator, which evaluates the web services using interface and functionality based rules. After that it sends selected services to the Composer in order to generate a composition. The result composition is executed by the Execution Engine. Finally, the results are sent to the client through the Translator.

Zeng et al. [15] propose a middleware platform for QoS-driven web service composition, called AgFlow. The QoS properties of web services are presented with a multi-dimensional QoS model. The model considers only five QoS properties, namely Execution price, Execution duration, Reputation, Reliability, and Availability, which are also used to evaluate the quality of the final web service composition. The composition is specified as a collection of generic service tasks described in terms of service ontologies and combined according to a set of control-flow and data-flow dependencies. These dependencies are presented as statcharts. AgFlow implements two composition approaches: local optimization and global planning. The local optimization approach performs optimal web service selection for each individual task without considering QoS properties of web services. The global planning approach is based on the preferences specified for the whole composition as well as QoS properties of web services. It uses integer programming in order to compute optimal plan that correspond to the composition. The proposed platform provides an adaptive execution engine, which re-plans the execution of the composition in response to changing QoS. It is implemented as a prototype system and is evaluated over a travel planning application.

Mili et al. [16] address web service composition problem as a function cover problem. The web service composition process starts an algorithm with the strictest interpretation of type equivalence, and then invokes looser versions only if the current ones fail to return appropriate results. The proposed solution is inspired by the programming languages, which handle type equivalence using one of two strategies: name equivalence and structural equivalence.

Che et al. [17] use a XML nets to model control flows and data flows of web service compositions as well as to discover and select web services for that compositions. Control flows can be easily modeled with XML nets due to their graphical nature. The data flow modeling is more complex due to problems that can occur during message passing. One of them is incompatibility between an output message of one web service and an input message of another one. XML nets adjust output and input messages by adding a web service chaining transition as mediator between two web service invocation transitions. The mediating transitions are applicable to message aggregation or disassembly. Web service discovery and selection process is aligned to definition and (re)configuration of XML net process rules. Transition inscriptions are used to define constraints for web service selection.

Lin et al. [18] propose a web service composition technique, in which the web services are described in OWL-S and the client preferences are described in PDDL3. The planning of web service composition combines HTNs with best-first search using a heuristic selection mechanism based on ontological reasoning. Thus, the proposed technique provides web service compositions with a minimal cost of violations of the user preferences.

Ge et al. [19] solve the web service composition problem by using OWL ontology. The candidate web services are semantically matched and composed based on their OWL descriptions. Four cases for check similarity of an output and input parameter from the same ontology are defined. First, if the input and output parameters are the same, then the similarity is maximal. Second, if the output parameter of one service is subsumed by

Table 3.1: Comparison of web service composition approaches.

Ref.	OM	CM	QM	T	DDAM	EH	SSM	TS
2	BPMN	WSDL	Yes	No	Partial	Yes	Yes	Yes
3	BPEL	WSDL	Partial	No	No	No	Yes	Yes
4	PG	WSDL	No	No	No	No	No	Yes
5	PMC	WSDL	No	No	No	Yes	Yes	Yes
6	AIP	OWL–S,OWL	No	No	No	No	No	Yes
7	OWL–S	SWRL	No	No	Yes	No	No	Yes
8	WS–BPEL	WSDL,SLA	No	Yes	Partial	No	No	No
9	GM	WSDL,UDDI	Yes	Partial	No	Partial	No	Yes
10	GM	WSDL,WSLA,OWL	Partial	No	No	No	No	Yes
11	S	OWL-S,SAWSDL	Partial	No	Yes	No	No	Yes
12	HTN	OWL-S	Yes	No	No	No	Yes	Yes
13	PMC	WSDL	No	No	Yes	Yes	Yes	Yes
14	AIP	WSDL	Yes	No	No	No	Yes	Yes
15	S	WSDL,SLA	Partial	No	No	Yes	Yes	Yes
16	SM	WSDL,UDDI	No	No	No	No	No	Yes
17	XML Nets	OWL–QoS	Yes	No	Yes	No	Yes	No
18	HTN	OWL–S	No	No	No	No	No	Yes
19	OWL	WSDL	No	No	Yes	No	No	Yes

the input parameter of another one, then the similarity value depends on their distance in the ontology. Third, the output parameter of one service subsumes the input parameter of another one and the properties of the parameters are partially satisfied. Fourth, if two parameters do not have subsumption relation or they are come from different ontology, then the similarity value can be obtained by Tversky's feature-based similarity model. The final composition is presented as direct graph and is converted to executable business process described in BPEL4WS.

3. Analysis of Web Service Composition Approaches.

This section presents a comparative analysis of web service composition approaches. It is based on a composition model proposed by Alonso et al. [1]. According to this model, web service composition is characterized with six different dimensions: Component Model (CM), Orchestration Model (OM), Data and Data Access Model (DDAM), Service Selection Model (SSM), Transactions (T), and Exception Handling (EH). The CM considers the type of components that participate in a composition as well as assumptions about them. The OM deals with the way in which web services are composed into more complex services. DDAM is responsible for data exchange among components. The SSM defines whether a particular web service is selected as a component dynamically or statically. Transactions define transactional semantics associated to the composition. Finally, the EH specifies the mechanisms for handling of exceptional situations that is possible to occur during composition invocation.

Evaluation of the composition quality is an important aspect of the composition process. When several candidate web services have the same functionality, their QoS properties are examined in order to select the best one. Thus we identify QoS support as an additional dimension of the composition model, called Quality Model (QM).

Table 3.1 summarizes the comparison of the web service composition approaches. The first column refers to composition approaches. The last column, named Tool Support (TS), shows which of the approaches are implemented as software tools or platforms. They are marked with filled circle. The rest of the columns correspond to the dimensions of composition model presented above.

OM column in Table 3.1 shows the abstractions and languages used for modeling of control flow of the compositions in terms of sequence of operation execution. Here, variety of paradigms exists. Most common are techniques based on AI Planning such as HTNs, PGs, and Semantic Markup for Web Services (OWL-S) and Web Ontology Language (OWL) [4], [6], [7], [12], [14], [18], [19]. PMC is another planning technique that is also applicable to OM [5], [13]. The languages for description of business processes like BPEL and BPMN are also proposed for web service orchestration [2], [3], [8]. Statecharts are abstractions that extend state machines with possibility to define activities while moving between states [11], [15]. HPNs have the ability to model concurrency of the systems, analyze concurrent behavior, and express the dynamically changing software. Here, they are presented by XML nets, which provide advantages in the description of process objects and inter-organizational exchange of XML structured data [17]. Graph Models (GM) are another way for design of OMs,

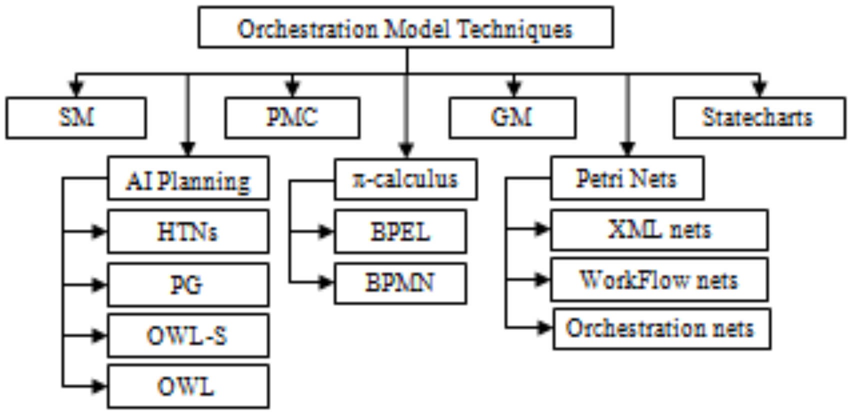


Fig. 3.1: Classification scheme of web service

where web services are presented with graph nodes and the sequence of their execution is shown by the graph edges. They are applied in [9] and [10]. Signature Matching (SM) is a technique used in [16] in order to compose web services. It is instance of more general problem, called function realization problem [20]. A classification scheme that summarizes web service orchestration techniques is shown on Figure 3.1.

The CM column shows the standards and languages used for description of composition components. Currently, most of the composition approaches (along with the analyzed ones here) assume that components are WSDL services based on standards such as HTTP, SOAP and WS-Transaction. As shown in Table 3.1, some approaches rely on additional standards like SLA and OWL-S in order to enrich the composition model with QoS and semantic data. They take into account the QoS properties of candidate web services in order to produce composition that aggregates web services with the maximum quality. This aspect of the composition process is presented in column QM of the table. Here, Ÿesm̈eans that the proposed QM is capable to covers all QoS properties. The P̈artialv̈alue shows that given approach supports QM with limited number of QoS properties. We call such approaches QM limited. For example, the QM proposed in [3] concerns only Response time.
QMs can be further classified according to the formalism that is applied to the composition when QoS properties are taken into account. For example the QM proposed in [2] is based on Multi Criteria Decision Making. It considers quality in three aspects: QoS that is related to web service execution, Quality of Contents (QoC) referring to the quality of the content with which the web services work, and Quality of Devices (QoD), concerning physical environment. The T column shows whether given approach consider transactional behavior of web service composition. Since the underlying middleware is often responsible for providing transactional capabilities, approaches presented in Table 3.1 do not consider this dimension of the composition model. For example, the compensation logic for roll back of web service activities may be handled by engine that implements WS-Transaction specifications.
The DDAM column shows which approaches compose web services in terms of data types and data transfer. According to [1] the data of given web service composition can be divided into application-specific data and control flow data. Data exchanged by web service messages is called application-specific data. Data that is used for evaluation of branching conditions is called control flow data. Data transfer method can follow blackboard approach or explicit data flow approach. The first one relies on a blackboard, which is a collection of variables defining the output and input of each web service activity. The explicit data flow allows developers to specify that the input data of an activity should be taken from the output data of previously executed activities. For instance, the DDAM presented in [17] uses data type definitions of XML Schemas. The proposed approach solves a problem with message passing, where an output message of a web service may be incompatible with the required input message of another WS. The approach uses XML nets to create mediating transition between two web service invocation transitions. In [19] the similarity of an output and input parameter of web service activities are checked according to preliminary defined rules using OWL.
The EH column indicates approaches that take into account unexpected behavior of web service compositions. A possible solution is to use conditional branch that checks the result from invocation of an activity for failures

or timeout according to which an activity will be terminated if the timeout expires. Another solution is to associate exception handling logic to an activity or group of activities. Rule-based languages are also applicable to the exception handling problem. For example, the approach presented in [13] uses Computation Tree Logic (CTL) to model web service composition failures. A global planning approach is used in [15] to reconfigure composition execution in case of one or several failures during web service invocations.
The SSM column show which approaches support dynamic binding of web services in a composition. Dynamic binding have advantage on dealing with web services that change their URIs.

4. Conclusion. Composition of web services is appealing tactic to enable even more powerful business processes and enrich software applications. A lot of web service composition approaches exist in the literature differing in the applied techniques and strategies for web service orchestration, data modeling, transaction support, QoS awareness and exception handling. In this paper we analyze a number of such approaches and distinguish them according to their essential characteristics.
Directions for future work include further developing the proposed classification into ontological framework for web service composition process and also a composition meta-model. This will help software architects to choose a particular approach best suited for a given application domain.

Acknowledgments. The work presented in this paper was partially supported by the FP7 Specific Programme 'Capacity' - Research potential under Grant No. 205030 and project BG 051PO001-3.3.04/13 of the HR Development OP of the European Social Fund 2007-2013.

REFERENCES

[1] G. Alonso, F. Casati, H. Kuno and V. Machiraju, *Web services, Concepts Architectures and Applications*, Springer-Verlag Berlin Heidelberg (2004).

[2] H. Chang and K. Lee, *Quality-Driven Web Service Composition Methodology for Ubiquitous Services*, Journal of Information Science and Engineering 26(6) (2010), pp. 1957–1971.

[3] M. Qiao, F. Khendek, A. Serhani, R. Dsouli and G. Roch, *An Architecture for Automatic QoS Adaptation for Composite Web Services*, Journal of Web Services Practices, Vol. 4, No.1 (2009), pp. 18–27.

[4] X. Zheng and Y. Yan, *An Efficient Syntactic Web Service Composition Algorithm Based on the Planning Graph Model*, 8th IEEE Int. Conf. on Web Services (2008), pp. 691–699.

[5] H. Q. Yu and S. Reiff-Marganiec, *Semantic Web Services Composition via Planning as Model Checking*, Technical report CS-06-003, University of Leicester (2006).

[6] M. Klusch and A. Gerber, *Evaluation of service composition planning with QWLS-XPlan*, Int. Conf. on Web Intelligence and Intelligent Agent Technology (2006), pp. 117–120.

[7] P. Bartalos and M. Bielikova, *Fast and Scalable Semantic Web Service Composition Approach Considering Complex Pre/Postconditions*, Int. Workshop on Web Service Composition and Adaptation (2009), pp. 414–421.

[8] V. Cardellini, E. Casalicchio, V. Grassi and F. Lo Presti, *Flow-Based Service Selection for Web Service Composition Supporting Multiple QoS Classes*, IEEE Int. Conf. on Web Services (2007), pp. 743–750.

[9] S. Chakhar, S. Youcef, V. Mousseau, L. Mokdad and S. Haddad, *Multicriteria Evaluation-Based Conceptual Framework for Composite Web Service Selection*, http://www.lipn.univ-paris13.fr/ youcef/BookQoS.

[10] M. Aiello, E. Khoury, A. Lazovik, P. Ratelband, *Optimal QoS-Aware Web Service Composition*, IEEE Conf. on Commerce and Enterprise Computing (2009), pp. 491–494.

[11] F. Lecue, *Optimizing QoS-Aware Semantic Web Service Composition*, 8th Int. Semantic Web Conference, LNCS Vol. 4273, (1989), pp 375–391.

[12] S. Sohrabi, S. A. McIlraith, *Optimizing Web Service Composition While Enforcing Regulations*, 8th Int. Semantic Web Conference, USA (2009), pp. 601–617.

[13] M. Pistore, F. Barbon, P. Bertoli, D. Shaparau and P. Traverso, *Planning and Monitoring Web Service Composition*, Lecture Notes in Computer Science Vol. 3192 (2004), pp. 106–115.

[14] H. K. Farhan, M. Y. Javed, B. Saba and H. K. Sikandar, *QoS Based Dynamic Web Services Composition and Execution*, Int. Journal of Computer Science and Information Security, Vol. 7, Issue 2, USA (2010), pp. 147–152.

[15] L. Zeng, B. Benatallah, A.Ngu, M. Dumas, J. Kalagnanam and H. Chang, *QoS-aware Middleware for Web Services Composition* IEEE Transactions on Software Engineering, Vol. 30 Issue5 (2004), pp. 311–327.

[16] H. Mili, G. Tremblay, A. Caillot and R. B. Tamrout, *Web service composition as a function cover problem*, MCeTech Montreal Conference on eTechnologies, Canada (2005), pp. 73–85.

[17] H. Che, Y. Li, A. Oberweis and W. Stucky, *Web Service Composition Based on XML Nets*, Hawaii Int. Conf. on Systems (2009), pp. 1–10.

[18] N. Lin, U. Kuter and E. Sirin, *Web service composition with user preferences*, 5th European semantic web conference on semantic web: research and applications, LNCS Vol. 5021, Spain (2008), pp. 629–643.

[19] J. Ge, Y.Qiu and S. Yin, *Web Services Composition Method Based on OWL*, Int. Conf. on Computer Science and Software Engineering, China (2008), pp. 74–77.

[20] H. Mili, O. Marcotte and A. Kabbaj, *Intelligent Component Retrieval for Software Reuse*, Third Maghrebian Conference on Artificial Intelligence and Software Engineering, Rabat, Morocco (1994), pp. 101–114.

[21] S. Dustdar and W. Schreiner, *A survey on web services composition*,International Journal of Web and Grid Services, Vol. 1, No. 1 (2005), pp. 1–30.
[22] J. Rao and X. Su, *A Survey of Automated Web Service Composition Methods*, 1st Int. Workshop on Semantic Web Services and Web Process Composition, USA (2004), pp. 43–54.
[23] M. Sathya, M. Swarnamugi, P. Dhavachelvan and G. Sureshkumar, *Evaluation of QoS Based Web- Service Selection Techniques for Service Composition*, Int. Journal of Software Engineering, Vol. 1 Issue 5 (2011), pp. 73–90.

Edited by: Dana Petcu and Jose Luis Vazquez-Poletti
Received: November 1, 2011
Accepted: November 30, 2011

Scalable Computing: Practice and Experience
Volume 12, Number 4, pp. 385–390. http://www.scpe.org
ISSN 1895-1767
© 2011 SCPE

CHALLENGES IN CLOUD COMPUTING[*]

KLAUS-DIETER SCHEWE[†], KÁROLY BÓSA[‡], HARALD LAMPESBERGER[§], JI MA[¶], MARIAM RADY[‖], AND BORIS VLEJU[**]

Abstract. Though cloud computing is considered mature for practical application, there is a need for more research. The identified challenges primarily concern client-cloud interaction and cloud interoperability. As to the former one, we highlight the needs of clients, contracting and legal aspects, and missing foundations as necessary fields of investigation. For the latter one clouds are considered to constitute repositories of services, so the challenge is to realize web-scale, service-oriented, distributed computing.

Key words: cloud computing, research, service clouds

AMS subject classifications. 68N99, 68Q85, 68U99

1. The Case for Research in Cloud Computing.

Currently, “cloud computing” is the most often used buzzword in computing, and many providers (Amazon, Google, Microsoft, IBM, etc.) of cloud services (IaaS, SaaS, PaaS, DaaS, . . .) emphasize the many benefits of outsourcing application into a (private or public) cloud. In other words, it is suggested that cloud computing represents a mature technology that is ready to be massively used. It is our conviction that that cloud computing still requires lots of fundamental research.

In particular, most of the offerings in cloud computing are provider-centric. For instance, a client (or tenant) may rent a certain piece of infrastructure, load and execute a piece of software on it, pay for the use, and leave the cloud without leaving permanent traces. Certainly, there are many computing-intensive applications, e.g. web crawling, image processing, machine learning, etc. that fit well into such a scenario. However, if we think of a multi-user database application, its usefulness decreases significantly.

First, we have to cope with interaction, which implies massive transfer of data from and to a cloud leading to a performance problem. Second, in such applications it is usually required that the use of the database is transparent to the database owner, i.e. credentials of user must be maintained by the client and not the cloud provider. Third, if several such applications are combined, the problem of cloud interoperability arises, for which the state of the art in cloud computing is not yet well prepared.

On these grounds we identified two major areas of investigation. First, our research aims at a significant improvement of client-cloud interaction. Second, we envision that cloud computing will only unfold its full potential, if the scope is widened to web-scale distributed computing.

2. Facilitation and Improvement of Client-Cloud Interaction.

2.1. The Forgotten Tenant.

Among many other problems in cloud computing we identified the lack of client-orientation as a serious problem that needs to be addressed in research. This subsumes the problems of identity of tenants, access rights, adaptivity to the needs of clients, and more. For instance, in the database application scenario above it would be indispensible to keep knowledge of users and their rights in the authority of the client instead of in the cloud. The immediate consequence of such an approach is that cloud applications become hybrid and distributed, as parts of data and software will reside on promise, while others reside in the cloud.

[*]This paper is a short position paper describing a research view on cloud computing in accordance with the research agenda of the recently established Austrian Christian-Doppler Laboratory for Client-Centric Cloud Computing.

[†]Software Competence Centre Hagenberg, Hagenberg, Austria, kd.schewe@scch.at & Johannes-Kepler-University Linz, Christian-Doppler Laboratory for Client-Centric Cloud Computing, Linz, Austria, kd.schewe@cdcc.faw.jku.at

[‡]Johannes-Kepler-University Linz, Christian-Doppler Laboratory for Client-Centric Cloud Computing, Linz, Austria, k.bosa@cdcc.faw.jku.at

[§]Johannes-Kepler-University Linz, Christian-Doppler Laboratory for Client-Centric Cloud Computing, Linz, Austria, h.lampesberger@cdcc.faw.jku.at

[¶]Johannes-Kepler-University Linz, Christian-Doppler Laboratory for Client-Centric Cloud Computing, Linz, Austria, j.ma@cdcc.faw.jku.at

[‖]Johannes-Kepler-University Linz, Christian-Doppler Laboratory for Client-Centric Cloud Computing, Linz, Austria, m.rady@cdcc.faw.jku.at

[**]Johannes-Kepler-University Linz, Christian-Doppler Laboratory for Client-Centric Cloud Computing, Linz, Austria, b.vleju@cdcc.faw.jku.at

As adaptivity describes a property of a system to adapt itself to different contexts, we aim at adaptivity to preferences of clients, restrictions arising from channels, and specific requirements for end-devices, in particular mobile devices. Preferences of clients refer to the way they interact with cloud servers. This already applies to the selection of services, for which we need customer-specific preference rules giving weights to providers and quality of service attributes. Our aim is to investigate the full range of possible selection preferences and to come up with languages for the expression of selection preferences for clients.

Analogously, client preferences may impact on the way parts of a cloud are handled that are owned by them or for which a client has exclusive access rights. In this case preferences can affect the location and method of storage as well as access protection; clients may even have preferences for particular protocols that are to be applied in these cases. Here, we also want to investigate, which options should be made available for clients to express preferences.

Furthermore, client preferences affect the content, functionality and the presentation of selected services. We aim at extending the rewriting-based approaches to automatically adapt choices in plots [8], which at the moment are restricted to high-level specifications. In particular, this is expected to lead to adaptation algorithms that take a specification of a (composed) service and a set of preference rules as input and produce a modified service that respects the client preferences.

2.2. Contracting and Legal Aspects. The creation of applications, in which larger portions of software are services that are offered somewhere by some service provider, is not only a scientific and technological challenges, it also implies the need to handle the relationship between service providers and service users. In a service-centric system the use of a services replaces the purchase or lease of data or a software product. The use of other's hardware as a platform replaces the purchase and maintenance of hardware by clients. The use of software services also brings with it the benefit of not being bothered by new releases, bug fixes, etc., which become part of the service.

However, what can be expected by a service user requires contracts that state explicitly, what a service offers, when it will be available, which performance can be expected, which maintenance is guaranteed, which security mechanisms are in place, how privacy is guaranteed, and how much the service costs. It has to contain regulations that apply in case one of these assertions of service is violated. Such service contracts may replace licensing agreements. As such there must be different classes of contracts depending on whether individuals use the service or groups. It also has to prescribe whether the client or the provider takes care of group-specific regulations. On the other hand, contracts must also contain rules that prescribe health conditions on the side of the client, rules, the level of security and privacy the client has to guarantee when using the service.

Our aim is to investigate the full range of regulations that have to be handled in service contracts. Leaving aside purely legal regulations such as the responsible court in case of disagreement or claims, the question is, to what degree the ingredients of service contracts can become part of the service description, in which case they could become part of a QoS ontology.

2.3. Security and Privacy – What Clients Need. The first problem area in security and privacy is connected to the management of access rights. This leads to two research problems we intend to investigate. The first one is concerned with the checking of access rights, for which we intend to develop adequate methods. Though this appears to be rather straightforward, the remaining problem is to keep track of dependencies between group rights and rights of group members.

The second problem is dedicated to inferences on access rights. For this we have to take into account that ownership should imply access rights, that membership in a group should imply that access rights of the group also apply to its members, that the right to execute a service operation should imply the right to execute the underlying view, that the right to use a composed service should imply the right to use the component services, etc.

More generally, access rights can be considered as permissions in a deontic logic. Therefore, we believe it is advisable to investigate not only permissions, but also prohibitions, obligations, and triggered actions. Thus, we aim at embedding access rights into a complex deontic logic and to study inferences in that logic. The goal s to ensure that all implied access rights must be explicitly granted. A particular difficulty arises from rights to grant access and revoke rights.

With respect to privacy tenants have to be protected against malicious users as well as against the cloud provider. With respect to the first of these hazards we pick up on the idea of specification of secrets, i.e. for data stored on a cloud it has to be specified who is permitted to see the data and who is not. Therefore, we first have to investigate data models in more detail in order to be able to identify at which granularity level such secret specifications should be applied. For instance, in case data is organised in relations, secrets may apply to records, clusters of records, or even individual attributes. For tree-based data organisation, e.g. in case XML is used as the data model, secrets might apply to subtrees rooted at particular elements, leaf elements and attributes, or aggregated tree portions that are defined by some query expression or algebra term.

Next we have to be aware that secrets may nonetheless be discovered by means of inferences. Therefore, we investigate, which queries or sequences of queries would be necessary to retrieve a secret that cannot be queried directly. We intend to also distinguish between exact detection of a secret, detection with a tolerable error, and detection with a high probability. These results should give an indication which access rights may need to be restricted to exclude the detection (or almost detection) of secrets by means of inferences. We will also investigate the alternatives of blocking certain queries, if the result could be used to discover secrets and the application of “lying strategies”. However, also the rejection of queries and inconsistency of query results can be used as valuable information inferences can be based upon.

The second problem is somehow inverse, as we want to allow a customer to retrieve some data from a cloud – access rights are assumed to be granted – without being able to see the actual query. It is well known that anonymity can only be guaranteed in an efficient way, if data is replicated. We therefore intend to study replication strategies and develop algorithms for query execution that preserve anonymity.

2.4. Epistemiological and Formal Foundations. Besides the forgotten tenant there is a serious lack of formal foundations in cloud computing starting from the simple fact that key notions such as service are not defined. Therefore, we address the fundamental research question how a uniform formal model for clouds must look like without any bias to particular languages and technology. We further investigate what a cloud has to offer to enable effective and efficient search for services as well as effective and efficient of multiple access to services by multiple tenants. The basic research component will build upon our previous research on Abstract State Services (AS^{2}s) described above, which has to be extended in various ways.

According to our understanding a cloud is primarily a repository of services, so we first have to specify the notion of service. The model of Abstact State Services (AS^{2}s) [5] starts from the assumption that a service should combine a hidden part, and a public part that is exported to service users. The hidden part is assumed to be a database-based transactional system, for which a universal model of database transformations [9] is adopted. The visual public part is represented by a collection of views, each of which is extended by a set of transaction-oriented service operations that link to the hidden database part. The whole model adopts the theory of Abstract State Machines (ASMs) [3], in particular referring to the so-called ASM thesis [4, 2].

3. Cloud Interoperability.

3.1. Clouds as Repositories of Services. Roughly speaking the research topics presented in the previous section address participation barriers in cloud computing. Clients, i.e. companies will only engage in cloud computing, if they have full control of their “outsourced” data and software, they can leave the cloud and engage elsewhere at any time, and the clent-cloud interaction is secure and reliable. There have to be formal and contractual guarantees for all this.

Nonetheless, even with all problems disussed so far being solved there remain risks, inless there is a trusted third party involved. If such a trusted party does not exist, we believe that distribution andreplication will still offer possibilities to remove participation barriers. Anyway, securing availability of cloud services will require some form of replication.

Furthermore, we argue that cloud computing can only unfold its complete strength, if services from different cloud are exploited in an interoperable way. With this in mind let us first look back at the services typically offered by a cloud.

- *Infrastructure as a Service* (IaaS) is the simplest form of cloud computing emphasizing mainly the use of hardware by the tenants. The costs are based on actual usage rather than on a priori fixed payment model. A known example is Amazon’s elastic cloud (EC2).
- *Platform as a Service* (PaaS) makes computing platforms available to tenants, thus parts or all of the software of tenants are outsourced to a cloud, which is maintained by some cloud provider. The provider takes care of maintenance of hard- and software, and thus guarantees a smooth running of the tenants’ applications, and the tenants pay for this service. In particular, ad-hoc usage of service components is part of the model. Examples are Microsoft Azure and Google’s AppEngine.

- *Software as a Service* (SaaS) emphasizes that the services provided are in fact software indicating a shift from software licensing to leasing. The software can be made available through web services [1].
- Similarly, *Data as a Service* (DaaS) emphasizes that the services provided are in the form of data. This facet, however, is not well present in actual cloud computing offers.

We propose to look at this "cloud stack" from the angle of ownership and usage rights (and obligations). In the IaaS model the basic (hardware) infrastructure is owned by the cloud provider and leased to the client on the grounds of some cloud usage contract. Every piece of software uploaded to the leased cloud infrastructure, however, is owned by the client. So, at the end the client will build up software and data services on the cloud. In most cases such services will be used only by the client who owns these services, but it is no problem to assume that services are also made available to be used by others, in the extreme case by everyone. In this sense the client of the cloud would also become a service provider.

Similarly, in the PaaS model the basic infrastructure plus development platforms (i.e. more than in the IaaS model) are owned by the cloud provider, but again this is used by the client to build up services to be used by himself or any other (defined) larger community. Finally, in the PaaS (and similarly the DaaS) model everything on the cloud is owned by the provider, but used by the client.

Thus, looking at the "cloud stack" from this angle turns a cloud into a repository of (data and software) services, each of which have an owner, a community of users and many regulations regarding rights and obligations of providers and users. The orthogonal classificaton into private and public clouds does not change the view, it only impacts on the way the cloud is organised as a repository of services.

We therefore like to address the interoperability of clouds by means of looking at such repositories of services. In [5] a formal model – called Abstract State Service (AS^2) – for (data and software) services was developed, which can serve as a basis for these investigations. Without repeating formal details here an AS^2 is composed of two layers: a hidden data layer and a view layer on top of it. Both layers combine static and dynamic aspects. Data services are formalised by views, which in the extreme case could be empty to capture pure functional services.

3.2. Service Clouds. In [6] this formal model was extended to a formal model of a "service cloud", which combines several AS^{2}s with an ontology that describes them "semantically", i.e. some form of description logic is exploited to describe each service by a combination of three different aspects. Without going into formal details a service description comprises:

- a *functional* description of input- and output types as well as pre- and post-conditions telling in technical terms, what the service operation will do,
- a *categorical* description by inter-related keywords telling what the service operation does by using common terminology of the application area, and
- a *quality of service* (QoS) description of non-functional properties such as availability, response time, cost, rights, obligations, etc.

The functional description is needed to actually use the service, once it has been selected, while the categorical description is needed to locate candidate services. The QoS description is not needed for service discovery, but can be exploited to select among alternatives.

The reason for the use of description logics (since its very beginnings over 30 years ago) is that they enable the definition of concepts by necessary and sufficient conditions, and the logics are kept so simple that classification, i.e. determining subsumption relationships, is decidable. Thus, a search requires a definition of the service sought by means of a complex concept. The well-known classification algorithms for description logics then can be used to determine all instances (in the ABox) matching the complex concept.

The idea of a service ontology is already omnipresent is the web services community. Languages such as WSDL [12], OWL [11] and UDDI [7] have been introduced to capture description, publishing and search of services. However, we like to remark that whatever the description in an ontology looks like, it is indispensible that a high-level description of the service (e.g. by means of ASMs as exploited in [5]) is made available as well so that a service interpreter can be used to check the suitability of a service.

3.3. Web-Scale Distributed Computing. Service clouds can be used to build up new applications that are composed mainly out of available services. In this way we obtain distributed systems. The interaction between different components could be supported by protocols such as SOAP [10]. We argue that building such web-scale distributed applications (as illustrated in Figure 3.1 and making them again available as services present a real challenge for cloud computing and web services.

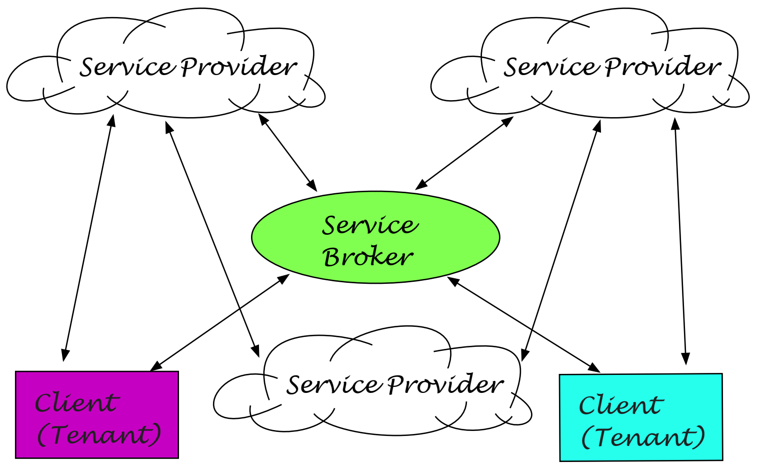


Fig. 3.1: Clouds in Web-Scale Distributed Systems

The specification and instantiation of large-scale distributed systems exploiting services presents a problem that goes beyond service composition, even more beyond the composition of service operations. One way to create such an application would be to start from a set of known services that are composed and extended by local components.

The other way is to start from a specification of the composed specification, which can be taken as an AS^2, in which most service operations are yet unknown. We may only know a categorical description for them. That is, besides search for services we also need a notion of matching services. The matching problem becomes particularly interesting, when we consider that services sought may be overlapping. For instance, when combining several booking services, each of them may contain a service operation for payment as well as one for gathering personal data. It would be not a very interesting composed applications, if such overlaps were not integrated – however, this is done in almost all approaches to srvice compositon and orchestration, in particular when BPEL is used or this purpose. First attempts to develop a theory of matching services were done in [6].

4. Conclusion. In this short position paper we outlined our view on which research challenges should be addressed in cloud computing. Naturally, these research directions describe the problems we want to address ourselves.

We identified the need to deal more carefully with concerns of cloud clients regarding loss of control, insufficient contractual guarantees, and security and privacy. Our objective is to achieve transparency in the sense that cloud-based applications should be seen by the client, as if there were no cloud involved. We envision a communicating distributed system with components located in the cloud and others on the client-side. This system will be adaptive to various cloud providers and to devices and preferences defined by the client. The key idea of our intended work plan is to specify such a system and to enlarge it step-by-step to capture access control, identification, transparent services, privacy preservation, etc.

We further plan to address cloud interoperability, which means to set up broker technology to publish and locate services in clouds, to compose them, and to run distributed applications across multiple clouds. The key idea for this part of our intended work plan is to extend the before-mentioned specification in a way that external access is enabled and controlled, and an architecture for handling such requests from outside is developed.

REFERENCES

[1] G. Alonso et al., eds., *Web Services: Concepts, Architectures and Applications*, Springer-Verlag, 2003.
[2] A. Blass and J. Gurevich, *Abstract state machines capture parallel algorithms*, ACM Transactions on Computational Logic, 4 (2003), pp. 578–651.
[3] E. Börger and R. Stärk, *Abstract State Machines*, Springer-Verlag, Berlin Heidelberg New York, 2003.
[4] J. Gurevich, *Sequential abstract state machines capture sequential algorithms*, ACM Transactions on Computational Logic, 1 (2000), pp. 77–111.

[5] H. Ma, K.-D. Schewe, B. Thalheim, and Q. Wang, *A theory of data-intensive software services*, Service Oriented Computing and Its Applications, 3 (2009), pp. 263–283.
[6] ———, *A formal model for the interoperability of service clouds*, 2011. submitted for publication.
[7] *Universal description, discovery and integration (UDDI)*. http://www.uddi.org.
[8] K.-D. Schewe, B. Thalheim, and Q. Wang, *Customising web information systems according to user preferences*, World Wide Web, 12 (2009), pp. 27–50.
[9] K.-D. Schewe and Q. Wang, *A customised ASM thesis for database transformations*, Acta Cybernetica, 19 (2010), pp. 765–805.
[10] *Simple object access protocol (SOAP)*. http://www.w3c.org/TR/soap.
[11] *Web ontology language (OWL)*. http://www.w3c.org//OWL/.
[12] *Web services description language (WSDL)*. http://www.w3c.org/TR/wsdl.

Edited by: Dana Petcu and Jose Luis Vazquez-Poletti
Received: November 1, 2011
Accepted: November 30, 2011

Scalable Computing: Practice and Experience
Volume 12, Number 4, pp. 391–401. http://www.scpe.org

ISSN 1895-1767
© 2011 SCPE

LOAD-BALANCING METRIC FOR SERVICE DEPENDABILITY IN LARGE SCALE DISTRIBUTED ENVIRONMENTS

FLORIN POP[*], MARIUS-VIOREL GRIGORAS[†], CIPRIAN DOBRE[‡], OVIDIU ACHIM[§] AND VALENTIN CRISTEA[¶]

Abstract. Today we live in a World-as-a-Service society, where electronic services are used everywhere, to support from large collaborations to ever more scalable business applications. In this era, specific service capabilities such as dependability and availability are more than ever needed to sustain ever more critical business- and service-oriented quality metrics. For example, today businesses need to know they can rely on a banking service called from used inside a complex client-oriented application. Service composition could not work unless services can reliably expect results from one another. In this context, we present a load-balancing metric designed to provide increased levels of availability for a wide-range of services running inside globally-scale distributed environments. We propose the use of "'service containers"', special mechanisms designed to encapsulate sets of replicated and distributed services. These mechanisms make faults transparent for the end-user based on smart decisions based on monitored information and specific metrics. We propose a set of service-oriented metrics designed to increase the use of resources in such a distributed approach. The proposed container-based approach is part of an architecture designed to increase dependability (resilience, fault tolerance, security) of a wide-range of distributed systems. We present evaluation experiments over real-world scenarios. We show how the proposed container-based mechanism can not only deliver increased tolerance to failures, but also it leads to improvements in the time-of-response, scalability or load-balancing. The solution is able to cope with various scenarios involving different failure-injection patterns in the system. The novelty of proposed solution is represented by the new metric for multi-criteria load-balancing within a fault tolerant distributed environment based on replication.

Key words: Web Services, Dependability, Load-Balancing, Service Replication, Distributed Systems.

AMS subject classifications. 68M14, 68M15, 68Q10

1. Introduction. Distributed systems play an important role as a background support in many areas such as trade, communication, science, government and even entertainment. The Web service architecture for large distributed environments is preferred today by businesses to support the interaction with users, because of advantages such as ergonomic behavior, portability, availability and transparent access from different platforms. All Web Service elements and corresponding designs are ever more important for current distributed systems [6]: standard structure of messages sent between clients and services, description method, discovery method, invocation method and development.

With the increasing interest for distributed systems, also customers' requirements become complex and are today addressed by specific SLAs (Service Level Agreements). For example, users can be interested in invoking a specific service which answers correctly at their requests in a reasonable time with a specific degree of accuracy. Also, time availability of service is very important: as the service is available to customers over time, the service will be more used. Services that fill these requirements are part of the dependable services category [13].

The concept of dependability [1] for large scale distributed systems (LSDS) is very complex and has been refined over many years of research. Dependability addresses concepts such as availability, reliability, safety, confidentiality, integrity and maintainability. *Availability* is the property of a system to offer correct service on demand. *Reliability* is the property of a system to offer correct service continuously. *Safety* is the absence of catastrophic effects on the environment, even in case of incorrect service. *Confidentiality* is the absence of unauthorized disclosure of information. *Integrity* is the absence of improper system state alteration. *Maintainability* is the property of a system to be easily repaired and modified. *Security* is the concurrent existence of availability to authorized users only, confidentiality, and integrity. Availability and reliability are essential in distributed systems, especially systems considered at large scale because customers expect system be operational despite technical problems and work as its specification even though some components were damaged.

[*]Computer Science Department, Faculty of Automatic Control and Computers, University POLITEHNICA of Bucharest, Romania, Email: florin.pop@cs.pub.ro, Principal Author
[†]Computer Science Department, Faculty of Automatic Control and Computers, University POLITEHNICA of Bucharest, Romania, Email: viorel.grigoras@cti.pub.ro
[‡]Computer Science Department, Faculty of Automatic Control and Computers, University POLITEHNICA of Bucharest, Romania, Email: ciprian.dobre@cs.pub.ro, Corresponding Author
[§]Oracle TSBU, Bucharest, Romania, Email: ovidiu.achim@oracle.com
[¶]Computer Science Department, Faculty of Automatic Control and Computers, University POLITEHNICA of Bucharest, Romania, Email: valentin.cristea@cs.pub.ro

In this context, the QoS could be considered as a quantitative measure of service dependability. There are many ways to increase the QoS [13], such as: bug fixing before deployment, mechanisms for fault tolerance, mechanisms for recovering from error, etc. In the development process, most developers choose the solution less expensive and faster in detriment of quality for service device. By contrast, service providers want to offer reliable services to customers, to ensure a certain level of QoS.

In this paper we present a solution to satisfy the service dependability in LSDS. We propose a balancing metric to increase availability and a reputation model designed to satisfy QoS requirements. The proposed solution is part of an architecture designed to optimize the access to services with respect to availability and scalability. The solution stands out by the support given to services providers, by implementing a system as a service container.

The paper is structured as follows. Related works are described in Section 2. Section 3 considers the service based architecture and highlights the technology solution chosen in order to implement the proposed architecture. Implementation details are explained in Section 4. We present in Section 5 the load-balancing metric used for replicated services. In Section 6 are presented the performance tests, scenarios used and performance results. The paper ends with conclusions and some possible future works.

2. Related Work. The research topics in distributed systems address the problem of building a reliable and highly-available distributed service by replicating the service on two or more hosts using the primary-backup approach. Three implementations are possible: one that makes heavy use of the notification interface defined in OGSI, one that uses standard grid service requests instead of notification, and one that uses low-level socket primitives [17]. The solution presented in [18] consider the problem of supporting fault tolerance for adaptive and time-critical applications in heterogeneous and unreliable grid computing environments. This is supported by techniques for implementing k-resilient objects, distributed objects that remain available, and whose operations are guaranteed to progress to completion, despite up to k site failures [3]. There is a commercial solution for enterprise-grade system that provide fault-tolerant virtual machines, based on the approach of replicating the execution of a primary virtual machine (VM) via a backup virtual machine on another server [12, 16].

There are some approaches to fault tolerance assuring at TCP level [10] considering a splicing mechanism. There are three enhancements to the TCP splicing mechanism: (1) Enable a TCP connection to be simultaneously spliced through multiple machines for higher scalability; (2) Make a spliced connection fault-tolerant to proxy failures; and (3) Provide flexibility of splitting a TCP splice between a proxy and a back-end server for further increasing the scalability of a web server system.

In [11] the authors present a system that ensure services dependability oriented on highly available and scalable Grid services. The services are classified in three separate categories: (i) services to store/query structured data in a scalable fashion; (ii) services to communicate in a scalable fashion; (iii) services to (partially) hide the effects of scale. The solution for availability ensuring for Grid Services is used in XtreemOS project.

Accurate failure prediction in Grids is critical for reasoning about QoS guarantees such as job completion time and availability. Statistical methods can be used but they suffer from the fact that they are based on assumptions, such as time-homogeneity, that are often not true. In particular, periodic failures are not modeled well by statistical methods. In [7] is presented an alternative mechanism for failure prediction in which periodic failures are first determined and then filtered from the failure list. The remaining failures are then used in a traditional statistical method.

Another project that consider load-balancing and fault tolerance is presented in [2]. It is based on comparison between Checkpoint-recovery and balancing using Intra-cluster load-balancing, Intra-grid load-balancing solutions. The project propose a fault detector that detects the occurrence of resource failures and a fault manager that guarantees that the tasks submitted are completely executed using the available resources. Model used in implementation of this fault tolerance technique is existing Intra-cluster and Intra-grid load-balancing model [15].

A project that aims to define and to implement a framework at the application level for development of grid dependable services is DIGS [4, 5]. In this project is described the structure of a *Proxy* that must combining several identical services from functionality view in another "better" service. A better service means a service with a higher level of dependability. The starting point objective was to design a mechanism to intercept transparent SOAP messages exchanged between client and server. A client sends messages to proxy like this would be a real service. Proxy intercepts the message and processes it based on a fault tolerance model and it then sends to a real service. Messages from the server to the client are transmitted also via proxy, which can process the response, if any, depending on the fault tolerance implemented model.

3. System Architecture. The main improvement of our solution is a new metric for load-balancing and fault tolerance. The proposed load-balancing metric considers a wide range of monitored parameters to compute the state of the services container. The proposed system architecture is presented in Figure 3.1. For example, the Monitor Service will respond with a "better" replica because will be capable to take real time decisions based on specific information from each container of a replica.

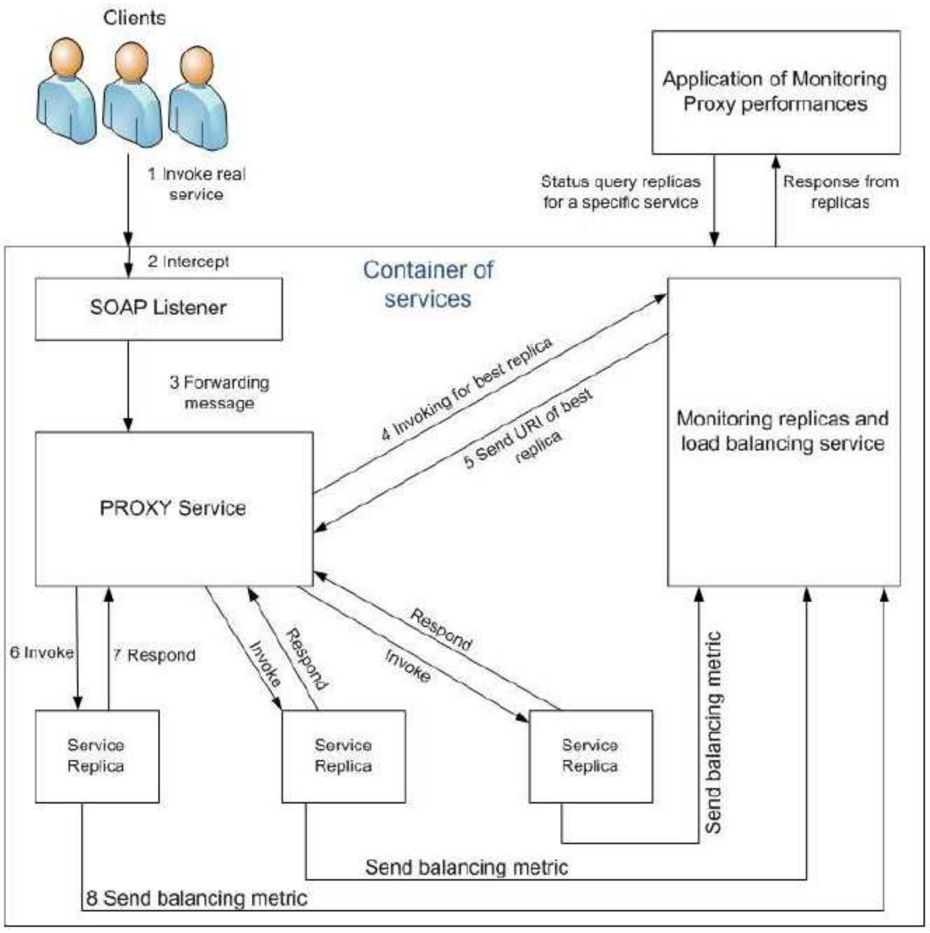


Fig. 3.1: System Architecture

The proposed architecture implements the *ProxyPattern* design pattern [8]. The main objective of this pattern is to define a proxy object, what is positioned between the client and the real service for controlling the access at the real service and for realizing certain processing each time a service is accessed. The service container functions similarly to a proxy for replicated services. All messages delivered to the real services are intercepted by the container (with the help of a SOAP listener) and are forwarded to one of the replicas service via the Proxy Service. All actions performed by the container are transparent to clients and services. Each client will consider during communication that it is connected to the real service, although in fact replicas are the one responding at requests like they came from the client-side, not from another service side.

The SOAP Listener is a system component responsible with monitoring all SOAP messages received by the services container. This component monitors SOAP messages and verifies if these messages are to be redirected to another proxy service or to the real invoked service. Specific situations when a message is transmitted to the real invoked service are:

- Messages that have been forwarded by proxy to a replica of the invoked service implemented in the same container with Proxy, must not be sent to the proxy anymore, to exclude an infinite loop. These

messages are marked by Proxy with a specific attribute in the SOAP message header.
- SOAP Messages that have as target some specific special services like Admin Service (because this service is used local by providers for services deployment and monitoring) and Load-Balancing Service.

The MONITORING AND LOAD-BALANCING SERVICE centralizes all information about replica services and chooses the best replica of a service to be invoking by Proxy Service. The best replica for a specific service is chosen according to the rules described in Section 4, in order to realize a better Load-Balancing for replicas. The proposed solution is based on monitoring information from each services container. The monitoring information is collected from container and sent to a Proxy Monitor. The Proxy Monitor will further calculate a balancing metric which is then used to choose the best replica to which to route the message.

When the PROXY SERVICE receives the response submitted to a replica, it invokes a method from the monitoring service for new information about this replica. The monitoring service has information about the total number of requests received from the clients for a specified service, the total number of processing errors transmitted to clients, as well as the total number of denied requests because of the replicas load. Concurrent access of each function to monitoring information is synchronized, to ensure exclusive access.

The PROXY SERVICE receives SOAP messages from LISTENER and invokes replicas of the service for sending a response to client and hiding eventually errors. When receiving a SOAP message from Listener, the Proxy Service makes a request to the Monitoring and Load-Balancing Service to find the best replica of the service for which is intended the message and to signal the receiving of a new request from the client side. The received replica from this request is used for redirecting of SOAP message. If the selected replica generates an exception, then the monitoring service is again interrogated to find another replica, which can be used for the actual request. This algorithm is repeated a number equal of steps, if all requests generate an exception or it stops then when a request return an invalid answer. According to the final response of the request (valid response or exception), Proxy Service invokes the monitoring service to update local information and to send the final result to the client.

4. Load-Balancing Metric.

To achieve the goal proposed in this paper, making services more reliable , we implemented in each container a method to calculate a metric which helps to assure load-balancing and fault tolerance. The load-balancing functionality of replicas optimizes resources utilization and minimizes the response time of a request.

4.1. Balancing Metric.

Each replica of a service has a balancing level calculated according to the information gathered by monitoring tool. The balancing metric is a value between 0 and 100. The balancing metric is calculated using the following formula:

$$BalancingMetric = \sum_{i=1}^{6} P_i * B_i$$

In this formula we consider some criteria for load-balancing, each criterion having a weight in the $BalancingMetric$ formula.

B_1 represents the measure of system usage. It considers the number of threads in the system. *Live threads* represents the number of threads which are alive (it runs on the processor or fallows to be scheduled by the operating system waiting at a blocking condition); the term where this parameter is located is calculated versus the total threads supported by that container (default in Apache Tomcat total threads is equal with 200). The B_1 term is:

$$B_1 = 1 - \frac{LiveThreads}{TotalThreads}$$

B_2 represents a measure of system quality. *Number of errors* represents total number of errors reported by services container. It grows when a request for a replica in that container is finished with error; the term where this parameter is located is calculated versus the total number of requests. *Number of requests* represents total number of requests and help to calculate the second term. We increment this number when receive a new request. The B_2 term is:

$$B_2 = 1 - \frac{ErrorsNo + 1}{RequestsNo + 1}$$

B_3 shows the proccessors usage in the system, considering the Java Virtual Machine. *Number of processors* represents total number of processors exited in system that has the services container; the term that contained this parameter is calculated dividing this number at maximum number of processors, that is a generic number, it represents the maximum number of processors that a system can have (if we chose this number 8, so a system can be maxim an opteron system); this parameter tells us how much we can parallel services from container a better throughput. *JVM CPU Usage* represents as percent how much JVM usage occupies from all processors. Maximum value for this parameter is ProcsNo*100%; when this sixth term tends to 1 means that all processors are almost in sleep mode, so JVM has no process in running mode and when tends to 0.5 means that from all processors, only one is "half" free, mean that OS can schedule and other processes and the throughput has still an acceptable value. The B_3 term is:

$$B_3 = 1 - \frac{1}{ProcsNo} \frac{JVMCPUUsage(\%)}{100}$$

B_4 represents a way to measure how total free space we have from total memory. When this report will tend to 0, Garbage Collector (GC) will try to delete all unreferenced objects or memory scheduler will try to allocate extra memory for total memory or max memory. Both operations have a high overhead. *Free memory* represents total number of mega bytes free from JVM. *Total Memory* represents total number of mega bytes allocated from Max Memory. Operating system allocates a total number of bytes from JVM. This memory is named Commited JVM Memory, is a fix number throughout runtime. From this Commited JVM Memory, the memory scheduler from java allocates a fix percent for Max memory. Usual, this percent is 60% 70%. All resources are allocated at runtime in total memory, which at first tends to zero and then go to max memory value. When max memory value is equal with total memory value, GC will "clean" the system, removing all allocated resources which are unreferenced. If GC will remove anything, memory scheduler will allocate extra memory for Max memory and this parameter will increase (see Figure 4.1). We have the following:

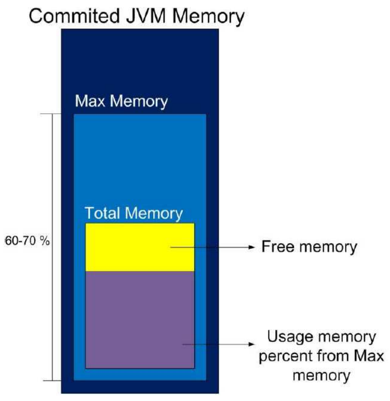


Fig. 4.1: JVM Memory

1. Usage memory value from total memory = Usage percent * Max memory
2. Total memory usage percent * Max memory = Free memory
3. Total memory Usage memory value from total memory = Free memory

So, the B_4 term is:

$$B_4 = \frac{FreeMemory}{TotalMemory}$$

B_5 represents the proportion of CPU utilization. The B_5 term is:

$$B_5 = \frac{ProcsNo}{MaxProcsNo}$$

B_6 represents the measure of overhead introduced by swap memory. *Free Swap* represents total number of free mega bytes located on the swap partition. Is important to consider and this parameter because swap-in and swap-out operations have a high overhead also. The B_6 term is:

$$B_6 = \frac{FreeSwap(\%)}{100}$$

The values for weights were considered as follows:

P_1	P_2	P_3	P_4	P_5	P_6
28	27	25	10	5	5

In the $BalancingMetric$ formula the importance of weights decrease from P_1 to P_6. P_1, P_2 and P_3 are the most important weights because found formula for balancing metric wanted to represent and a trust level (P_1 and P_2), but also because operations of GC and memory scheduler have not a very height overhead in a services container context. The greater is number of requests and the smaller is number of errors, the second term of the sum will approach to P_2 value and will keep this value a period of time as long as possible. This value will decrease very low, because in the worst case at a request that finished unsuccessfully increases both the number of requests and number of errors. P_3 is also a very important weight because even if we have very few requests is enough to have just one request which lead our processors on maxim load and already the system will have a higher response time. P_4 has a smaller weight because operation of allocation memory has not a very high overhead and also P_5 and P_6 have smaller weights because term of P5 is always a constant in that container and at swap-in and swap-out operations system rarely reach.

4.2. Reputation Function for Service Replica. Reputation-based function addresses one of the most critical issue in Grid Environment: QoS for grid-based services. Simulation and, in the future, implementation of Reputation solution over Proxy in Service Environments, will constantly follow to ensure QoS in term of execution times and performance. Solution will permanently monitor the "quality" of the grid and how it's responding to consumers' requests and will propose ways to choose either for "best (shortest) execution time" or for"most determinist execution time". In the first approach, solution will try, based on execution history, to provide with the best service endpoint in term of execution performance. Second scenario will cover those consumers that care more about a deterministic execution time rather than the shortest execution time. Allocation of Services is limited due to little information available for the Monitor. The replica selection algorithm doesn't know whether a particular service is running a job or if has or not multiple jobs queued. Moreover, it doesn't have a history of the completion times of the jobs. This makes difficult the prediction of the performance of Web Services. Allocation of services in grid can be improved with a help of a reputation system.

For replica selection we proposed a dedicated service to compute reputations. The service is a centralized one, which receives information with the completion times of the jobs or information about services failures and provides a degree of reputation, based on a metric for the QoS offered. When choosing an evaluation function for reputation calculation, a few issues there must be taken into consideration. Firstly, trust decays with time, according with the history of the quality of service provided. If, over time, the quality decreases (job execution times increase or job success rate decrease) then the function must be able to penalize it by a drop in reputation. Also, in the case of performance improvements, the reputation will grow.

This proposed solution based on reputation improves the service selection in the case of the same values for balancing metric for different replicas.

5. Test Scenarios and Experimental Results. We have realized experimental tests to highlight each term behavior of the proposed metric for balancing. The test conditions consider Lambda probe [9] integrated module write in log files every 30 seconds, balancing metric calculated every approximate 30 seconds (30 s some sync times), an event takes place every 15 seconds (an event can be a request/error, event of GC or of memory scheduler, increasing/decreasing free memory etc). In all test we have used the same Tomcat container that mean that we consider the same resources. The experiments consider the following scenarious:

- Empty container
- Threads Alive
- Successful Requests
- Error requests
- Best Replica
- Many Requests

5.1. Empty container. The first scenario intends to measure the balancing metric evolving in time into an empty container. No service is running on this container, only default services from Apache Tomcat and services from monitoring tool Lambda Probe. Time of measuring is 30 minutes.

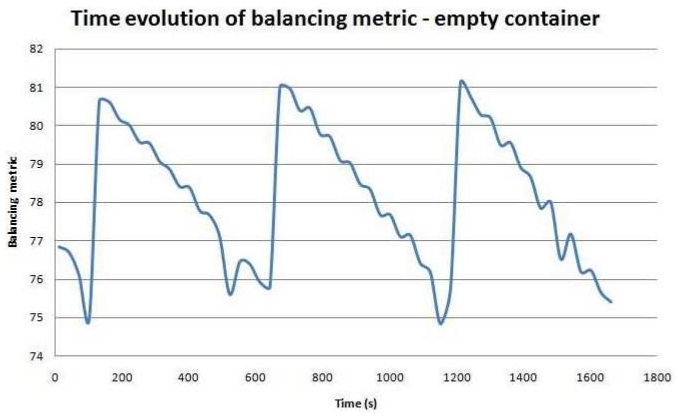


Fig. 5.1: Time evolution of balancing metric empty container

As is illustrated in Figure 5.1 the balancing metric varies between 75 and 81 with a periodic behavior at approximately 10 minutes. This period is due to the actions of GC which cleans references to unused objects. These actions determine the increasing of the free memory value and therefore the value of fourth parameter.

5.2. Threads Alive. The second scenario intends to measure the balancing metric evolving in time when increasing number of threads alive. Time of measuring is approximately 15 minutes. For testing we have used services with a longer uptime than the period of testing because we wanted to avoid altering the requests parameter which consequently would have increased the balancing metric.

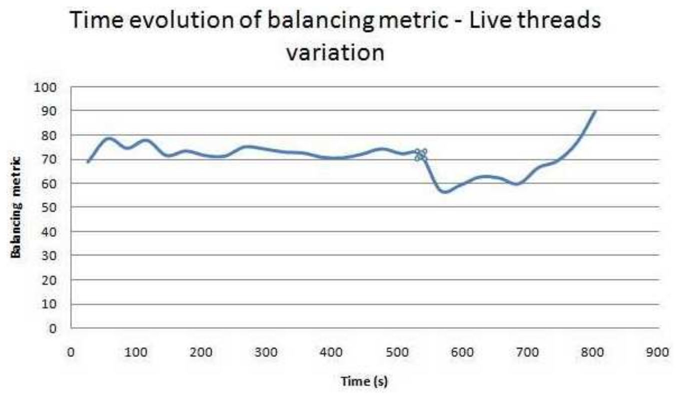


Fig. 5.2: Time evolution of balancing metric live thread variation

As is illustrated in Figure 5.2 the balancing metric varies between 70 and 80 before the moment of busy threads increasing. After this moment, we have increased the number of busy threads and the balancing metric decrease down to 55. After 10 minutes (600 seconds) the value of balancing metric has slowly increased but the busy threads remained the same number. The balancing metric increased because the GC has cleaned the unused references objects and possibly memory provider has allocated extra memory for total memory (value of balancing metric reached 90).

5.3. Successful Requests. This test scenario intends to measure the balancing metric evolving in time when increasing number of successful requests. Time of measuring is approximately 11 minutes. For testing we have used services with an empty method named nothing because we wanted to avoid altering the busy threads parameter which consequently would have decreased the balancing metric.

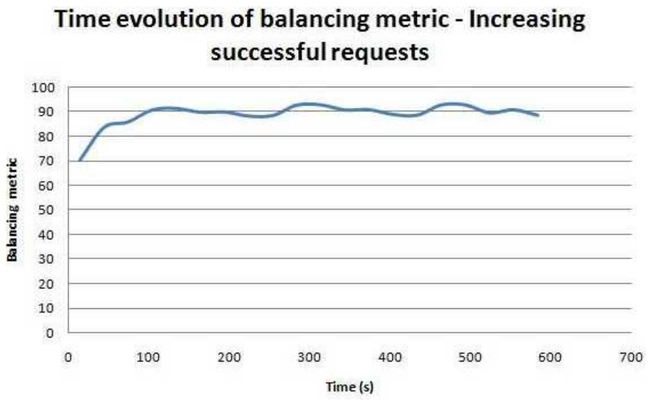


Fig. 5.3: Time evolution of balancing metric Increasing successful requests

As is illustrated in Figure 5.3 the balancing metric starts at 70 and increasing to 90, where is established. The second term of balancing metric represents also a trust parameter. The higher of this parameter is, the higher is trust level represented by P2 value. Default in our implementation number of requests is 1. We have made this initiation because we wanted to give half of trust to every replica. If we will have error requests (like in the next scenario) and default number of requests is 0, the second term of sum will be always 0 and the monitor service will choose at every moment first replica that has generated errors because it will not have a smaller value.

We can reach at a moment to every replica having the same value and this value cant decrease because the errors number is much higher than requests number. This means that B_2 term tends to 0. In this case we can choose the first replica, for example, even if we still send requests to it, the value of its balancing metric will not decrease, but this “problem” does not make us to choose another replica, because all other replicas are second term tended to 0, so these arent more confidence.

5.4. Error requests. This test scenario intends to measure the balancing metric evolving in time when increasing number of error requests. Time of measuring is approximately 12 minutes. For testing we have used services with a method that not exists on the container because we wanted to avoid altering the busy threads parameter which consequently would have decreased the balancing metric or other parameters like JVM CPU Usage.

As is illustrated in Figure 5.4 the balancing metric starts at 70 and increasing to 80 because JVM cpu usage begins to decrease. After about 100 seconds we begin to increase the request number that finished with errors. The balancing metric starts to decrease down to 62 where begins to established because the value of second term (P2) cant decrease more and the rest of terms are constantly (only GC alters sometimes the free memory percents and balancing metric varies by 3 or 4 percents.

This parameter is part of the trusted component (P2). As we can see in Figure 5.5, number of errors represents an important role in value of trust level. If a container begins to have errors it immediately decreases the value of trust (second parameter P2) and then even if it receives a high number of request that successfully

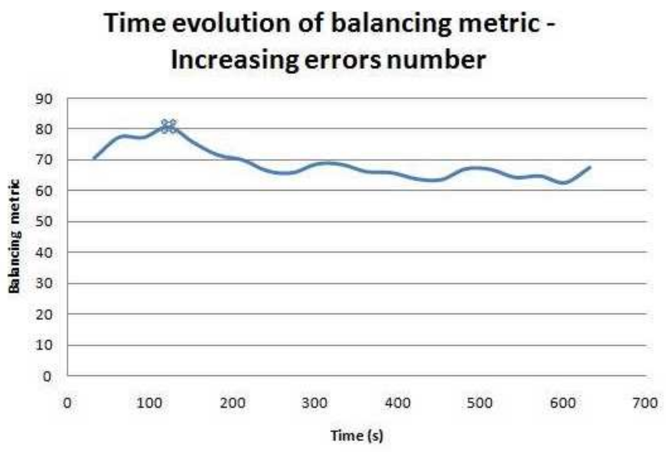


Fig. 5.4: Time evolution of balancing metric Increasing error requests

resolve them, the value of trust level increases very hard. In conclusion, it is recommended for a container to have the smallest number of errors from the first moments of activity.

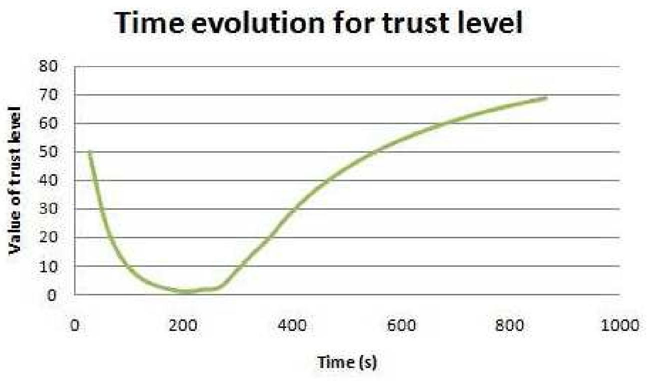


Fig. 5.5: Evolution of trust level

5.5. Best Replica. This test scenario intends to measure the balancing metric evolving in time in a proxy system that forwarded the received request to the best replica of the moment, with the highest value of the balancing metric.

At the beginning moment replica with the best value of balancing metric is blue replica. Proxy monitor will forward to it the received request. Because blue replica finish with success all requests, it still receive requests from monitor Service. From 300 moment, blue replica finish with errors all requests and in this manner its balancing metric value decrease down to 79. In this moment replica with best value of balancing metric is green replica. It will receive all requests from monitor service and from the same reason, its value of balancing metric will increase up to 85, when at 450 (seconds) moment this value will decrease because received requests finished with errors. But the green replica still receive requests from monitor service, because it has the highest value of balancing metric (see Figure 5.6).

5.6. Many Requests. This test scenario intends to measure the balancing metric evolving in time when services container receives very many requests and its resources are tested to the fullest. As is illustrated in Figure 5.7 the balancing metric starts at 77 and increasing to 79 because JVM cpu usage begins to decrease. After about 60 seconds we begin to increase the request number that finished with errors and the requests that requires extra resources like cpu usage and are IO intensive. After this all requests finish. This scenario repeats

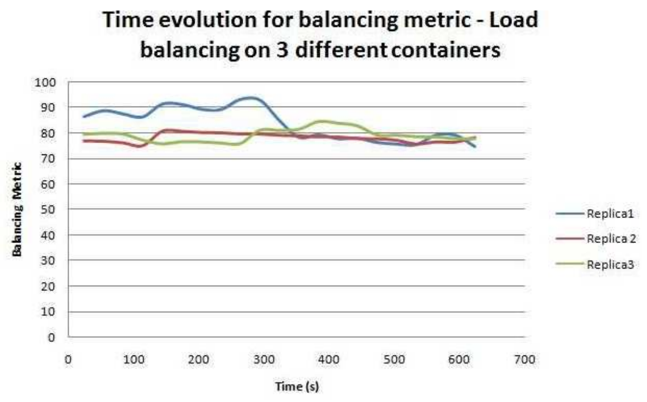


Fig. 5.6: Evolution of trust level

for two times between 250 and 350 and respectively 350 and 420 periods.

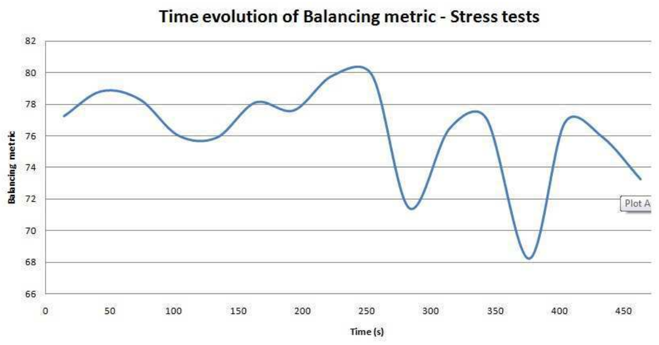


Fig. 5.7: Balancing metric - stress tests

6. Conclusions. Many of today's applications follow the distributed computing paradigm in which parts of the application are executed on different network-interconnected computers. Examples include Web browsing and searching, Internet banking, enterprise applications (for accounting, production scheduling, customer information management), and grid applications (for data intensive or compute intensive processing). The last classes of applications are in the area of Scientific Computing. In this paper we presented a service-based architecture that uses a proxy and services replicas and we present the load-balancing metric used by proxy in order to select a service replica. Dependability remains a key element in the context of application development based on services and is by far one of the most important issues still not solved by recent research efforts.

The work presented in this paper is concerned by increasing reliability and availability, particularly in Grids and Web-based distributed systems. The presented solution presented is based on encapsulation of replicated distributed services in a container for masking the possible defects that may occur. Thus, for a client, a failure occurred is imperceptible, presented service being designed for masking possible errors and in transparent mode to redirect any received requests to another functional service/replica. Our solution uses a load-balancing system that ensures the use of resources more efficiently and reduces the time response.

Using a service by clients assume to know only the service address implemented in the same container with proxy service. This address remains unchanged even if other replicas of the service join in the system or leave from the system. Using transparent replication makes the system more scalable.

In the future we consider the possibility of extending the service implemented in several ways. First we will consider implementing a mechanism for Proxy service replication to eliminate the possibility of becoming a service insertion mechanism defects, that is to replicate and proxy service. Also, we want to integrate two services: one to calculate the administrative distance between nodes where there are services (this is a maximum flow problem) and another for detecting a fall service/container. First service can help Monitor Service to take better decisions because the balancing metric will have a more realistic value and the second service will find possible failures in the system architecture and after proxy service will be capable to replicate dynamically the invoked request. Another important aspect will be focused on reputation function. We will consider a quantitative approach of this model and we will combine the balancing metric with reputation in the process of replica selection.

Acknowledgments. The research presented in this paper is supported by national project: "SORMSYS - Resource Management Optimization in Self-Organizing Large Scale Distributes Systems", Project CNCSIS-PN-II-RU-PD ID: 201 (Contract No. 5/28.07.2010). The research is co-founded by national project "DEPSYS - Models and Techniques for ensuring reliability, safety, availability and security of Large Scale Distributes Systems", Project "CNCSIS-IDEI" ID: 1710 (618/15.01.2009).

REFERENCES

[1] Algirdas Avizienis, Jean-Claude Laprie, Brian Randell, and Vytautas. Fundamental Concepts of Dependability, 2000.

[2] J. Balasangameshwara and N. Raju. A Decentralized Recent Neighbour Load-Balancing Algorithm for Computational Grid. *International Journal*, 1.

[3] K. P. Birman, T. A. Joseph, T. Raeuchle, and A. El Abbadi. Implementing fault-tolerant distributed objects. *IEEE Trans. Softw. Eng.*, 11:502–508, June 1985.

[4] G. Dobson. Using ws-bpel to implement software fault tolerance for web services. In *Software Engineering and Advanced Applications, 2006. SEAA'06. 32nd EUROMICRO Conference on*, pages 126–133. IEEE, 2006.

[5] G. Dobson, S. Hall, and I. Sommerville. A Container-Based Approach to Fault Tolerance in Service-Oriented Architectures. In *International Conference of Software Engeneering.* Citeseer, 2005.

[6] Ian Foster, Carl Kesselman, Jeffrey M. Nick, and Steven Tuecke. Grid services for distributed system integration. *Computer*, 35(6):37–46, 2002.

[7] Woochul Kang and Andrew Grimshaw. Failure prediction in computational grids. In *ANSS '07: Proceedings of the 40th Annual Simulation Symposium*, pages 275–282, Washington, DC, USA, 2007. IEEE Computer Society.

[8] Huawen Li and Qingjie Wang. Proxy pattern informatization research based on saas. In *Proceedings of the 2009 IEEE International Conference on e-Business Engineering*, ICEBE '09, pages 518–521, Washington, DC, USA, 2009. IEEE Computer Society.

[9] X. Lu, Q. Yue, Y. Zou, and X. Wang. An Experimental Analysis for Memory Usage of GOS Core. In *Parallel and Distributed Computing, Applications and Technologies, 2008. PDCAT 2008. Ninth International Conference on*, pages 33–36. IEEE, 2008.

[10] Manish Marwah, Shivakant Mishra, and Christof Fetzer. Fault-tolerant and scalable tcp splice and web server architecture. In *Proceedings of the 25th IEEE Symposium on Reliable Distributed Systems*, pages 301–310, Washington, DC, USA, 2006. IEEE Computer Society.

[11] Guillaume Pierre, Thorsten Schütt, Jörg Domaschka, and Massimo Coppola. Highly available and scalable grid services. In *WDDM '09: Proceedings of the Third Workshop on Dependable Distributed Data Management*, pages 18–20, New York, NY, USA, 2009. ACM.

[12] Daniel J. Scales, Mike Nelson, and Ganesh Venkitachalam. The design of a practical system for fault-tolerant virtual machines. *SIGOPS Oper. Syst. Rev.*, 44:30–39, December 2010.

[13] Ian Sommerville and Guy Dewsbury. Dependable domestic systems design: A socio-technical approach. *Interact. Comput.*, 19(4):438–456, 2007.

[14] Michal Szymaniak, Guillaume Pierre, and Maarten van Steen. Versatile anycasting with mobile ipv6. In *AAA-IDEA '06: Proceedings of the 2nd international workshop on Advanced architectures and algorithms for internet delivery and applications*, page 2, New York, NY, USA, 2006. ACM.

[15] B. Yagoubi and M. Medebber. A load-balancing model for grid environment. In *Computer and information sciences, 2007. iscis 2007. 22nd international symposium on*, pages 1–7. IEEE, 2008.

[16] Dmitrii Zagorodnov, Keith Marzullo, Lorenzo Alvisi, and Thomas C. Bressoud. Practical and low-overhead masking of failures of tcp-based servers. *ACM Trans. Comput. Syst.*, 27:4:1–4:39, May 2009.

[17] Xianan Zhang, D. Zagorodnov, M. Hiltunen, K. Marzullo, and R. D. Schlichting. 2004. Fault-tolerant grid services using primary-backup: feasibility and performance. In Proceedings of the 2004 IEEE International Conference on Cluster Computing (CLUSTER '04). IEEE Computer Society, Washington, DC, USA, 105-114.

[18] Qian Zhu and Gagan Agrawal. Supporting fault-tolerance for time-critical events in distributed environments. *Sci. Program.*, 18:51–76, January 2010.

Edited by: Dana Petcu and Jose Luis Vazquez-Poletti
Received: November 1, 2011
Accepted: November 30, 2011

Scalable Computing: Practice and Experience
Volume 12, Number 4, pp. 403–415. http://www.scpe.org

ISSN 1895-1767
© 2011 SCPE

MULTI-AGENT SYSTEMS FOR ACCESS CONTROL IN DISTRIBUTED INFORMATIONS SYSTEMS

ANETA PONISZEWSKA-MARANDA*

Abstract. The modern information systems evaluate very quickly. The information is more and more distributed through the networks or federation of numerous information systems located in different places on the globe. Also, the control domain of information system is very important in the times of very fast networks, telecommunication protocols and telecommunication equipment.

The paper presents the proposal of cooperation between the information systems and multi-agent systems. It is necessary to assure the cooperation of local data resources and create the coherent structure for intelligent agents. The dynamic process of conflict solving can be realized by using different techniques that come from the multi-agent systems.

Key words: access control, distributed information systems, access control models, multi-agent systems

AMS subject classifications. 68N02, 68T02

1. Introduction. The modern information systems evaluate very quickly. The information is more and more distributed through the networks or federation of numerous information systems located in different places on the globe. Also, the control domain of information system is very important in the times of very fast networks, telecommunication protocols and telecommunication equipment.

Not less important is the data protection against improper disclosure or modification in the information systems. This requirement is always obligatory in the development process of information system and its security domain as well in the other phases of information system lifecycle. But nowadays the information engineering, as well as another information science domains, changes very quickly and the new products appear every day. Also the access modes to the information have been changed. The new protocols appear to exchange the information. All these changes provoke the new security problems against which the existing models or architectures of access control should make a stand.

The appearance of new business models for the organization and enterprise activities in the network and the appearance of new protocols for information exchange provoke that the information is more and more distributed and the traditional access models are insufficient to solve the problems of information control. Also, it is important to protect the information against the non-controlled utilization and on the other hand we would like to control the usage and the diffusion of this information. It gives the possibility to specify how it can be used and specify the utilization constraints. All these new problems are connected with the usage control. This new term in security domain necessities the mechanisms to apply the usage control that should be distributed in the information systems.

Development of information systems should answer more and more to the problems of federated data sources and the problems with heterogeneous distributed information systems [13]. The assurance of access data security realized in federated information systems with loose connection among local data sources is hard to achieve mainly for two reasons: local data sources are heterogeneous (i.e. data, models, access security models, semantics, etc.) and local autonomy of systems does not allow to create a global integrated security schema. To solve such problems we propose to use the intelligent agents that can assist in the process of real-time access by defined and undefined users to the data stored in different systems, subsystems or applications of federated information systems. Each of these systems or subsystems can be secured by a different security policy and the agents can help in the process of security policy integration on a global level.

This paper presents the proposal of cooperation between the information systems and multi-agent systems using the interactions between the system agents. It is necessary to assure the cooperation of local data resources and create the coherent structure for intelligent agents. It can be made by using the unified security model to exchange the data and to access them. The model based on the roles can assure the homogeneity of local security models and allows the description of local models. The dynamic process of conflict solving can be realized by using different techniques that come from the multi-agent systems.

The paper is structured as follows: section 2 presents the access control policies and models - traditional access control models, models based on role concept and models based on usage concept that particularly

*Institute of Information Technology, Technical University of Lodz, Poland (anetap@ics.p.lodz.p)

interesting for distributed information systems. Section 3 deals with the security problems of distributed information systems and presents the concept of agents and multi-agent systems. Section 4 presents the proposition of architecture based on multi-agent approach for secured cooperation in distributed information systems.

2. Access control policies and models. The security policies of a system generally express the basic choices taken by an institution for its own data security. They define the principles on which the access is granted or denied. Access control imposes the constraints on what a user can do directly, and what the programs executed on behalf of the user are allowed to do. In information system the access control is responsible for granting direct access to system objects in accordance with the modes and principles defined by the protection policies.

2.1. Access control models. It is possible to distinguish two categories of security policies of the information systems: discretionary security policy and mandatory (non-discretionary) security policy. We can find some access control models based on these policies [1, 2, 3]:

Discretionary security model - manages the users' access to the information according to the user identification and on the rules defined for every user (subject) and object in the system. For each subject and object in a system there are authorization rules that define the access modes of the subject on the object. The access modes: read, write and execute, are verified for each user and for each object. The access to the object in the specific mode is granted only to the subjects for whom an authorization rule exists and is verified. Otherwise it is denied. "Discretionary" means that users are allowed to grant and revoke access rights on particular objects. This implies decentralization of the administration control through ownership [1, 2].

Mandatory (non-discretionary) security model - manages the access to data according to classification of the subjects and objects in a system. Each user and each object of a system are assigned to specific security levels. The access to data is limited by the definition of security classes. Subjects and objects in a system are related to security classes, and the access of a subject to an object is granted if the relation between the classes of the subject and the object is verified [1, 2].

Role-Based Access Control model - RBAC model - regulates the access of users to the information on the basis of the activities that the users perform in a system. This model requires identification of roles in a system. The role can represent competency to do a specific task and it can embody the authority and responsibility. The roles are created for various job functions in an organization and the users are assigned to the roles based on their responsibilities and qualifications. The user playing a role is allowed to execute all access modes to which the role is authorized. RBAC model provides support for several important security principles (notably least privilege, privilege abstraction and separation of duties), but does not dictate how these should be put into practice [3].

Extended RBAC model - the complexity of organizations gave rise to the idea of extending the standard RBAC model to ensure a finer decomposition of the responsibilities in an enterprise. To this aim, the notion of a function has been introduced. Since a person employed in an enterprise may have many responsibilities in it, he may be attached to an entire set of roles. Each role defined in the extended RBAC model makes it possible to realize a specific task associated with the enterprise process. At the same time, every role can contain many functions that a user can take, and therefore it is possible to choose functions in the system that are necessary for a given role. Therefore, the classical RBAC model was extended by addition of some elements to express more complex elements of information system secured by security model (Fig. 2.1) [4, 5].

Usage Control (UCON) model - it is based on the three decision factors: authorizations,, obligations and conditions that have to be evaluated for the usage decision. It consists of eight main components: subjects, objects, subject attributes, object attributes, rights, authorizations, obligations, and conditions. Subjects, objects and rights can be divided into several detailed components with different perspectives. The UCON strategy is characterized by two features: mutability and continuity. Mutability means the mutability of subject and object attributes - with a mutability property the attributes can be either mutable or immutable. The mutable attributes can be modified by the actions of subjects and the immutable attributes can be modified only by the administrative actions. The continuity means that a decision can be made even after an access (Fig. 2.2) [10].

2.2. Access control approach for dynamic information systems. Modern information systems are very often dynamic, distributed and heterogeneous in different aspects. They can contain many different components, applications, located in different places in a city, in a country or on the globe. Each of such components can store the information, can make this information available to other components or to different users. The

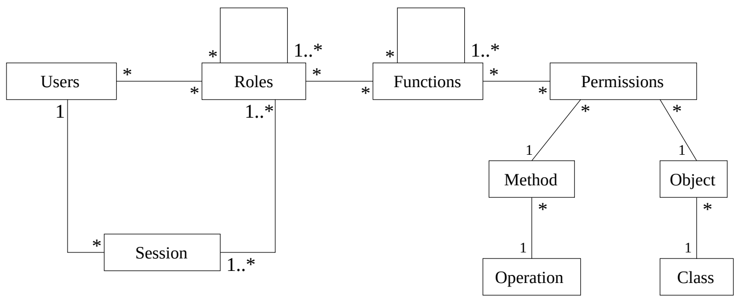


Fig. 2.1: Elements of extended RBAC model

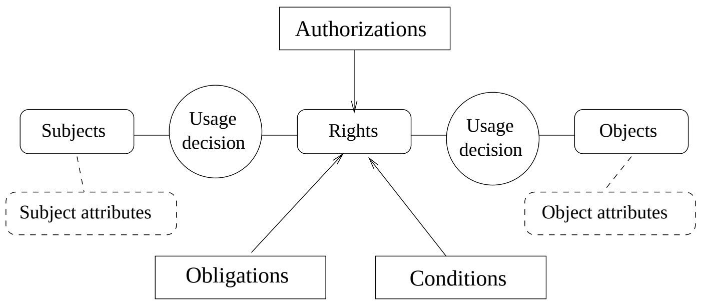


Fig. 2.2: Elements of Usage Control model

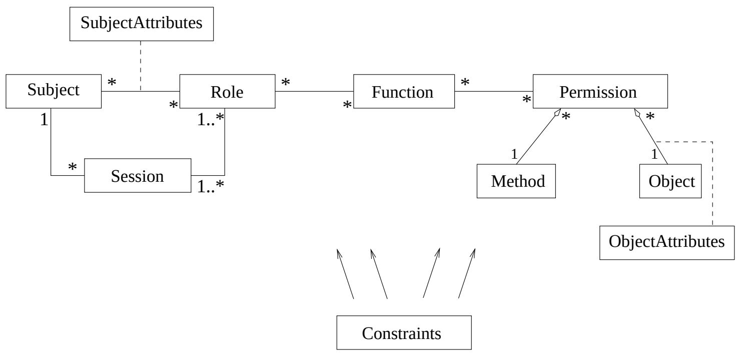


Fig. 2.3: Usage Role-based Access Control approach

authorized users accessing the information can change this information, its status, role or other attributes at any time. These changes can cause the necessity of modifications in security properties of accessed data on access control level. Such modifications are dynamic and often should be realized ad hoc because other users from other locations can request the access to the information almost at the same time.

The new access control approach was based on two access control models: extended RBAC model [2, 3] and UCON model [10]. It was named Usage Role-based Access Control (URBAC) (Fig. 2.3) [17]. The term usage means usage of rights on information system objects. The ”rights” include the rights to use particular objects and also to delegate the rights to other subjects.

The core part of URBAC model essentially represents the extended RBAC model. Subjects can be regarded as individual human beings. They hold and execute indirectly certain rights on the objects. Subject permits to formalize the assignment of users or groups of users to the roles. Subject can be viewed as the base type of all users and groups of users in a system. The aggregation relation SubjectGroup that represents an ordering relation in the set of all system subjects can assign subjects to the groups.

A Role is a job function or a job title within the organization with some associated semantics regarding the authority and responsibility conferred on a member of the role. The roles are created for various job functions in an organization. The direct relation is established between roles and subjects that represent the users or groups of users. It is also possible to define the hierarchy of roles, represented by aggregation relation RoleHierarchy, which represents the inheritance relations between the roles.

The association relation between roles and subjects is described by the association class SubjectAttributes that represents the additional subject attributes (i.e. subject properties). Subject attributes provide additional properties, describing the subjects that can be used for the usage decision process, for example an identity, enterprise role, credit, membership, security level.

Each role allows the realization of specific task associated with an enterprise process. A role can contain many functions that a user can apply. A role can be viewed as a set of functions that this role can take to realize a specific job. It is also possible to define the hierarchy of functions, presented by aggregation relation named FunctionHierarchy, which represents the inheritance relations between the functions.

Each function can perform one or more operations that this function needs to be defined as a set of permissions. To perform an operation one has the access to required object, so necessary permissions should be assigned to corresponding function. The permission determines the execution right for a particular method on the particular object. Very often the constraints have to be defined in assignment process of permissions to the object. Such constraints are represented first of all by the authorizations. Authorization is a logical predicate attached to a permission that determines the permission validity depending on the access rules, object attributes and subject attributes. A constraint determines that some permission is valid only for a part of the object instances. Therefore, the permission can be presented as a function $p(o, m, c)$, where o is an object, m is a method which can be executed on this object and c is a set of constraints which determine this permission. Taking into consideration a concept of authorization, the permission can be presented as a function $p(o, m, A)$, where A is a set of the authorizations determining this permission. The constraints defined on permission can be also determined by the obligations and conditions.

The objects are the entities that can be indirectly accessed or used by the users. The relation between objects and their permissions are additionally described by association class ObjectAttributes that represents the additional object attributes (i.e. object properties) that cannot be specified in the object's class and they can be used for usage decision process. They can be also mutable or immutable as subject attributes do.

The security constraints can be defined for each main element of the model presented above and also for the relationships among the elements. The concept of constraints is described widely in the literature [3, 4, 11, 12]. It is possible to distinguish different types of constraints, static and dynamic that can be attached to different model elements. The URBAC approach distinguishes the following general types of constraints:

- Authorizations - constraints defined for the permissions, basing on access rules defined by enterprise security policy but also basing on objects' attributes and subjects' attributes.
- Obligations - the subject can be associated with the obligations which represents different access control predicates that describe the mandatory requirements performed by a subject before (pre) or during (ongoing) the access.
- Conditions - session is connected with the set of conditions that represent the features of a system or application. They can describe the current environmental or system status and states during the user session that are used for the usage decision.
- Constraints on roles and on functions. The most popular type in this group of constraints is Separation of Duty (SoD) constraints [3, 4, 11, 12].
- Constraints on relationships between the model elements [3, 4, 11, 12].

3. Access control for dynamic distributed information systems. The components of the information systems, i.e. applications, databases are typically distributed and heterogeneous. The topology of these systems is dynamic and their content is changing sometimes very quickly or rapidly and it is difficult for a user of an application to obtain the correct information or for the enterprise to maintain the consistent information. The information systems are large and complex in several meanings [13]:

- they can have many components, i.e. applications, databases,
- they can have huge content of the number of concepts and of the amount of the data about each concept,
- they can be geographically distributed,
- they can have a broad scope, i.e. coverage of a major portion of a significant domain.

In distributed information system each local component can be secured by another access control model. The objectives of the security policy in cooperative information systems are to respect the local security model of each system (each model specifies the security principles of a local system) and to control the indirect security connected with the global cooperation level: a member of a local system may in another local system access only the equivalent information according to his local profile. Each system can have other security policy for describing the access control rules to access its data. This situation can involve some difficulties and heterogeneities in definition of the global security model.

In order to better allow defining of the access control of distributed information systems it is necessary to have more expressive access control model. This model should allow the greatest structure of security policy to make possible the decomposition of this policy and make easy its definition. It should be possible to express more that the simple authorizations but also the interdictions or obligations that should be fulfilled in order to obtain the access to the information system. Also the model should allow expressing of the rules assigned to the conditions on the system state or on the access context of a system [15, 16].

Four major techniques exist for handling the huge size and complexity of such enterprise information systems: modularity, distribution, abstraction and intelligence. A very reasonable solution is to use the intelligent, distributed modules which are the components of the entire information system. Using this concept, the intelligent agents or computational agents can be distributed and embedded throughout the enterprise. The agents could act as intelligent programs working for the applications, as active information resources, "actors" that surround and buffer conventional components, or as the on-line network services. These agents should have the great knowledge about information system resources that are local to them and they should cooperate with other agents to provide the global access to the information in the data flow from and to the information system. Multi-agent systems are the best way to characterize and design the distributed information systems because of the large size of systems, their dynamism and the needs of formulation and implementation of the global principles and solutions.

Some definitions of an agent or a multi-agent system can be found in literature [7, 8]:

"An *agent* is a computer system or application that is situated in some environment and that is capable of autonomous actions in this environment in order to meet its design objectives."

"An *intelligent agent* is one that is capable of flexible autonomous actions in order to meet its design objectives: reactivity, pro-activeness and social ability."

Agents operate and exist in some environment that typically is both computational and physical. The environment might be open or closed, it might or not contain other agents. At times, the number of agents may be too numerous to deal with individually and it is more convenient to deal with them collectively as a society of agents. An agent has the ability to communicate. This ability is part perception (the receiving of messages) and part action (the sending of messages). Agents communicate in order to achieve better goals for themselves or for the system in which they exist.

Multi-agent system is composed of multiple interacting software components known as agents, which are typically capable of cooperating to solve the problems that are beyond the abilities of any individual member [7, 9].

A multi-agent system consists of a number of agents that interact with one-another. In the most general case, the agents will be acting on behalf of the users with different goals and motivations. To successfully interact, they will require the ability to cooperate, coordinate and negotiate with each other, much as people do. Multi-agent environment provides an infrastructure specifying the communication and interaction protocols for the agents. It is typically open and contains the agents that are autonomous and distributed and may be self-interested or cooperative.

3.1. Related works. The concept of agents and multi-agent systems used for security of information systems, in particular for access control can be found in some works presented in the literature.

Kagal in [21] proposes solution based on trust management with the use of agent concept that can be applicable to distributed informations system. This solution involves developing a security policy, assigning credentials to entities, verifying that the credentials fulfill the policy, delegating trust to third parties, and reasoning about users access rights. However, this proposition does not take into consideration the dynamic aspects of security and access control.

In [22] Varadharajan, Kumar and Mu propose a security agent based approach for the authorization aspects in distributed environment. They define the security agents used to capture the privileges and a part of security policy in distributed authorization. The introduce some concepts - the principles make use of the agents to carry out their requests on the remote hosts the targets verify the authenticity of security agent and its privileges and use them together with their local security policy to grant or deny the requests. The operations of authorization system is described using these security agents and the use of agents to support dynamic decision making is also presented. This proposition does not concern the dynamic and heterogeneous access control.

Antonopoulos et all. in [18] propose distributed access control architecture that is based on the concept of distributed, active authorization entities (lock cells). The combinations of these entities can be referenced by an agent to provide input and/or output access control. The authors presents how these authorization entities can be used to implement security domains and how they can be combined to create composite lock cells. However, this architecture does not concern the dynamic aspects of information system security.

Seleznyov in [19, 20] presents conceptual architecture for an autonomic middle-ware component designed to provide the application-independent access control, named ADAM. This component can be used in large-scale dynamic computing environments. Such environments do not allow to determine the centralized access control policy because of the complexity of trust relationships. The architecture is based on multi-agent system. The agents dynamically organize themselves into cooperating distributed communities that mediate between users and devices (collectively known as trustees) and network resources (principals). Authorization decisions in ADAM are based on negotiations between two agents: user agents and authorization agents. The user-agent contains information about its legal owner, including secret keys, and certificates required for the user authentication. Authorization agents protect network resources by ensuring that only valid users can obtain access to them. They are enforce the security policies and procedures of the resources that they manage. This solution allows to manage the dynamic information systems on the access control level but do not concern the problems of heterogeneous informations system. Furthermore, it functions on total absence of explicitly stated organizational policy.

In [23] Weippl et all. presents the project SemanticLIFE that stores an individual's entire digital life and makes it available to co-workers. The presented security scheme is based on implementing role-based access controls through the usage of database systems. Security policies offer a more flexible and much robust way for security administration than in the project ADAM.

Belsis et all. present in [24] the proposition of system architecture for knowledge management from the point of view of security and access control. This architecture is based on Role-base Access Control model and use the concepts of intelligent agents. This solution proposes a distributed, resilient knowledge management architecture to handle knowledge assets exchange between autonomous domains in distributed information systems. This proposition represents the domain of static access control and does not take up the aspect of heterogeneous information systems.

4. Access control architecture for dynamic distributed information systems. The agent approach in the information system security can be considered as follows: the agents can be used to assist the security policies already defined in a federation of distributed information systems, to preserve the data security on the higher level or to solve the security problems attached to the real-time access desired by some users. The agents can use the security rules already defined in a system to decide whether or not to give the users the access to the desired data or to the part of it.

The distributed information systems can also base on the communication between the agents and on the communication of external users with the agents. The agents exchange the data between themselves or with users. The systems communicate with each other by means of agents, which exchange the information, search, explore the data from one system to another. We can distinguish some types of agents in this situation: search agents (explore agents), exchange agents, communication agents (Fig. 4.1).

The incorporation of the local data sources on the global level in federation of cooperative information systems can be divided into two phases:

- definition and representation of local data exported to the global level using the corresponding descriptive elements to obtain the *described local schema*,
- allocation of these local schema on the global level and their assignment to the particular security agents.

In general, there are: *security objects* (passive data entities) and *security subjects* (active entities like users). These elements can be described by the *Security Entities (SE)* that are initiated from SE classes (i.e. Data, User, Application, System). The system SE classes describe the general security strategy of the local system.

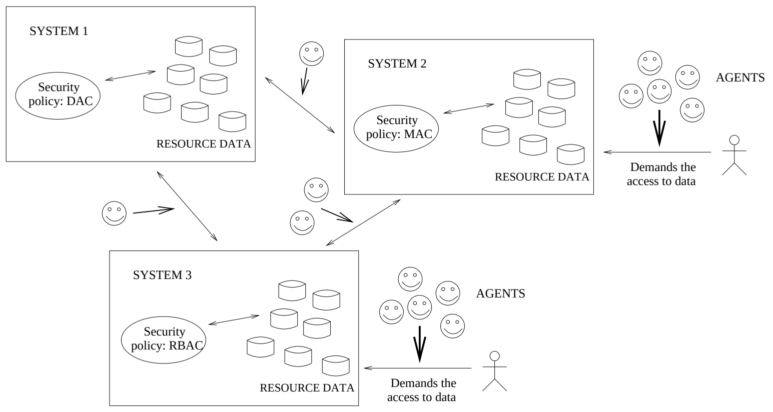


Fig. 4.1: Cooperative information system with the agents

The local security authorization units like groups (DAC models), MAC "containers", roles (RBAC models, URBAC model) or subjects (UCON model) can be described by *Access Model Entities (AME)*. Additionally, one more structure can be defined, *Information Entities (IE)* that represents all these elements on the global level and assures the homogeneous representation of each local information entity.

Taking into consideration the representation of security elements on two levels in the security aspects of cooperative information systems given above, we can enrich the process of incorporation of the local data sources on the global level as follows (Fig. 4.2):

- definition of the local data exported to the global level using the corresponding descriptive elements to obtain the *described local schema*,
- description of each data element from the exported local data schema using the security structures of semantic model given above,
- representation of these local schema using the semantics of unified security model (for example based on roles),
- allocation of the local security schema on the global level and their assignment to the particular security agents.

Therefore, the security architecture for the federation of cooperative information systems can be defined on four levels. The first level, representing the definition of local data exported to the global level using the corresponding descriptive elements, contains the exported data schema joined with the local data.

The main stages of this process (i.e. the creation of system application Model and creation of user profiles based on the access control models) for three types of access control models (MAC, DAC and eRBAC) are presented in [14].

This creation is possible with regards to the features of access control concepts [1] and the concepts of access control models. It can be realized by the automatic transformation of XML files, containing the application elements in approach of concepts of access control models (DAC/MAC/eRBAC/UCON/URBAC), to the XML file(s) containing these elements using the common concepts. First of all, it is necessary to create the DTD (Data Type Definition) files for these XML files to define their structures. The root elements of DTD files for each access control model are given as follows:

- DTD file for DAC model:
  $<!ELEMENT\ DAC(user+, object+, operation+, userIdentification*)>$
- DTD file for MAC model:
  $<!ELEMENT\ MAC(user+, object+, operation+, authorizationLevel*,$
  $classificationLevel*, category*)>$
- DTD file for eRBAC model:
  $<!ELEMENT\ eRBAC(user+, role+, function+, permission+, method+,$
  $object+, operation+, class+, constraint*)>$
- DTD file for UCON model:

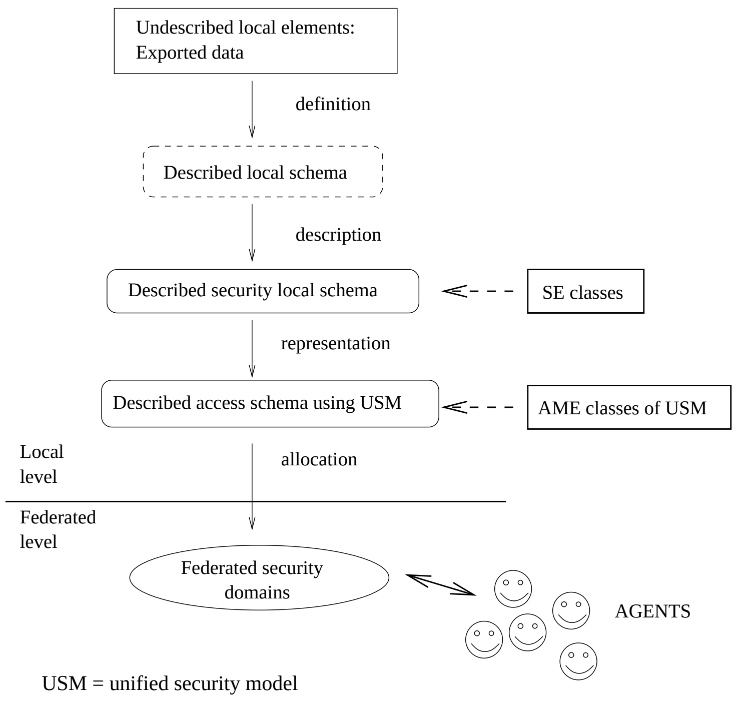

Undescribed local elements:
Exported data
definition
Described local schema
description
Described security local schema
SE classes
representation
Described access schema using USM
AME classes of USM
Local level
allocation
Federated level
Federated security domains
AGENTS
USM = unified security model

Fig. 4.2: Security incorporation process of data on the federation level

$<!ELEMENT\ UCON(subject+, subjectAttr+, object+, objectAttr+,$
$right+, authorization*, obligation*, condition*)>$

- DTD file for URBAC model:
  in general:
  $<!ELEMENT\ URBAC(subject+, subjectAttr*, role+, function+,$
  $permission+, method+, object+, objectAttr*, constraint*)>$
  in details:
  $<!ELEMENT\ URBAC(user+, userGroup*, subjectAttr*, role+,$
  $function+, permission+, method+, object+, objectAttr*, authorization*,$
  $obligation*, condition*)>$

The second level of presented above process of incorporation of local data sources on the global level is composed of the set of semantic descriptive elements and the security structures.

In an information system the access control is responsible for granting direct access to the system objects in accordance with the modes and principles defined by the protection policies. An access control system defines: the *subjects* (active entities of a system) that access the *information* (passive entities) executing different *actions*, which respect the access rules. The subjects can describe the *users* or the processes that have access to the data stored in a system. The information, i.e. the data, determines the system *objects* on which the actions represented by the most popular *operations*, i.e. read, write, delete, execute, can be performed. Therefore, it is possible to distinguish three main sets of elements describing the access control rules: **subjects**, **objects** and **operations** and two additional sets: **subject attributes** and **object attributes**.

We propose to represent the elements of access control models, i.e. DAC, MAC, eRBAC, UCON and URBAC, with the use of these sets of concepts and additionally the concept of constraints (Fig. 4.3, Fig. 4.4 and Fig. 4.5).

The root element of DTD file containing the common concepts for describing the security elements of each access control model is as follows:

$<!ELEMENT\ commonModel(subject+, subjectAttr*, object+, objectAttr*,$
$operation+, constraint*)>$

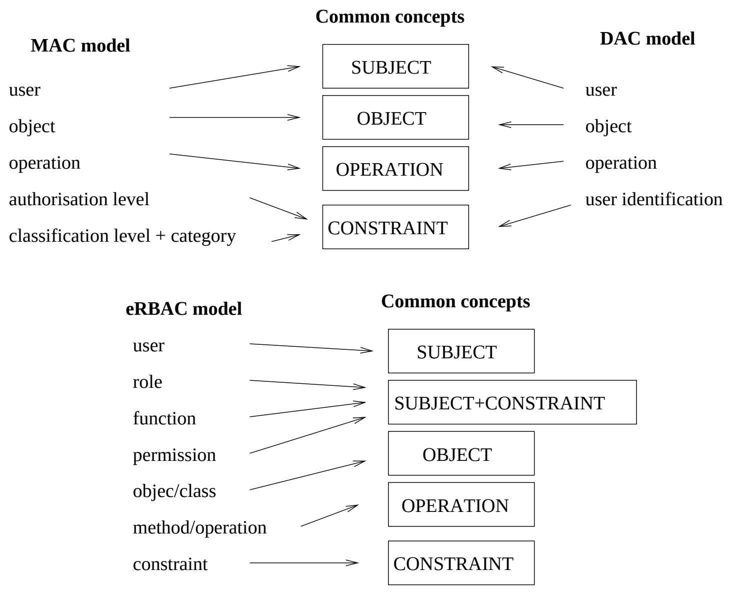


Fig. 4.3: Common concepts for access control models (DAC, MAC and eRBAC)

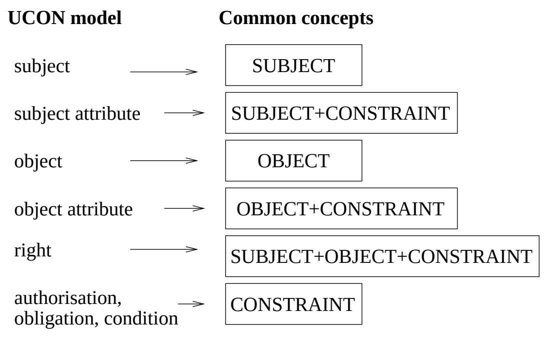


Fig. 4.4: Common concepts for UCON model

It describes the common concepts of heterogeneous security systems. The XML files based on such DTD file can be intended for the security administrator(s) to manage the federation of heterogeneous security systems.

The third level of the presented process contains the security common elements based on the semantics of unified security model. This unified security model can be based on the extended RBAC model or on the UCON model. The question is: which model or which strategy is good or quite enough to manage the access control in distributed information system, that can change sometimes very quickly and the topology of these systems is dynamic? Traditional access control models, trust management, DRM (digital right management) or usage control? Maybe the chosen security strategy should be extended by adding of the new elements to support the administration of information system security?

Actually, it seems that the most proper model to support the security of dynamic information systems is the UCON model. In this model the security policy is dynamic because it can change during the information access. The dynamic change of security policy can be translated by the change of the values of subject attributes or object attributes there are mutable attributes. The modification of an attribute can be realized before the information access, during the information access or at the end of the access.

However, the unified security model should be determined to have one common notion to express the access control elements from different models and levels.

The last level of the presented process contains the security agents and their connections with the security

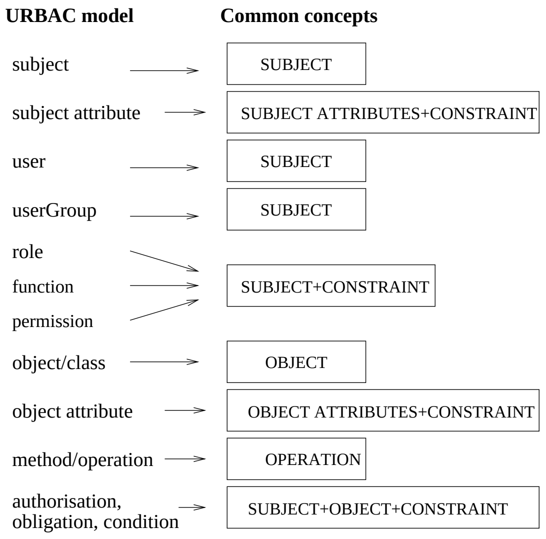


Fig. 4.5: Common concepts for URBAC model

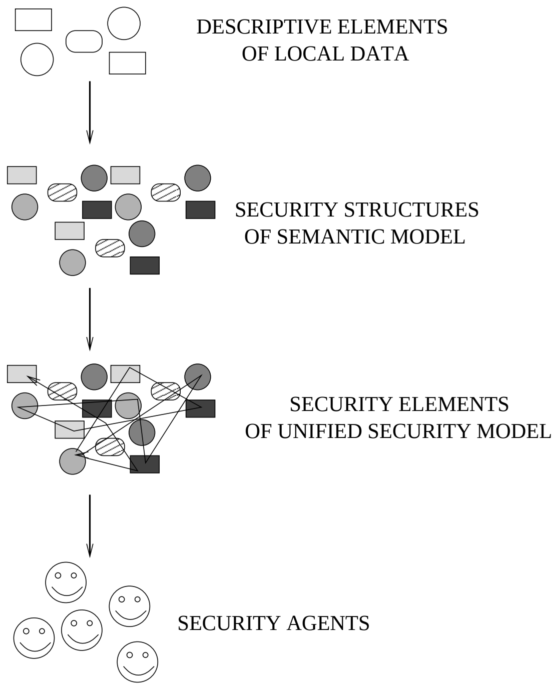

DESCRIPTIVE ELEMENTS OF LOCAL DATA
SECURITY STRUCTURES OF SEMANTIC MODEL
SECURITY ELEMENTS OF UNIFIED SECURITY MODEL
SECURITY AGENTS

Fig. 4.6: Security architecture for federation of cooperative information systems

domains containing the elements that came from the local levels (Fig. 4.6). These agents are specialized in different tasks - it is possible to distinguish different types of agents, e.g. management agents, security agents, semantic agents or organization agents.

The agents cooperating with the federation security domains on the global levels manage the different systems functions, particularly on the access control level (Fig. 4.7):

- *security agents* are responsible for the management of global security domains,
- *semantic agents* manage the local semantic domains and the global semantic domains composed of the Information Entities and their relationships,

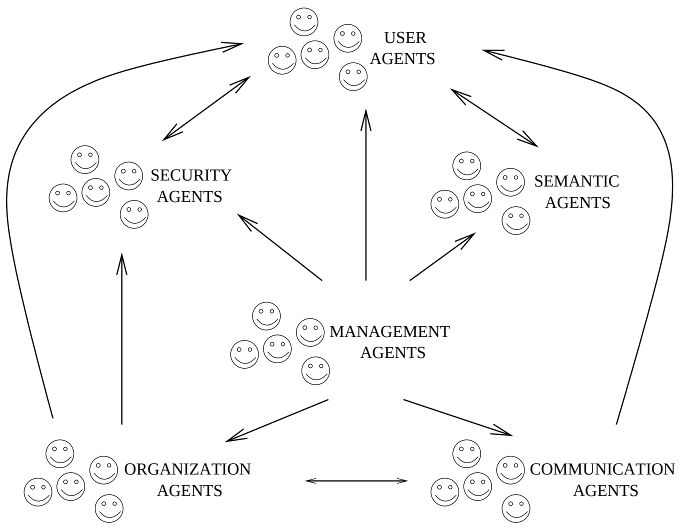


Fig. 4.7: Security agents and their relations in the federation

- *user agents* manage the system users and their rights on the global level of the federation,
- *organization agents* are responsible for the relations among the elements in the federation and the knowledge database of the security domains,
- *communication agents* are responsible for the communications on the level of local systems and on the global level and manage the security alerts generated during the occurrence of the security problems and access control problems
- and the *management agents* are responsible for the proper functioning of other types of the agents.

These agents have specific competences and they communicate with each other exchanging the information concerning different aspect of the security domains in the federation and the users of different cooperative systems in the federation.

4.1. Implementation of presented approach. A multi-agent system for access control of distributed information systems was developed to implement the presented approach. The proposed system is used to support the methodology of information project management. It manages and assigns dynamically an access to code repositories basing on current developer's status and basing on progress of works in particular phase of software development.

The main purpose of the system is to manage and monitor an access to code repositories of project groups that use a specific software creation methodology. The use of chosen methodology results in the following situations:

- project team assign dynamically the tasks to the developers according to their preferences and qualifications,
- members of project team should be focused only on one project,
- client can actively take part in process of product development and in consequence he has an access to particular resources of development company,
- reduction of costs in programming companies provokes that use of cloud computing and data stored in clouds becomes more and more popular and profitable,
- several times a company has a few departments in different countries and the servers storing the data are distributed.

The exemplary situations presented above cause the application of intelligent system monitoring an access to resources of development company operating in distributed environment. The monitoring of access rights to resources should be realized with the use of intelligent agents from multi0agent systems assigning appropriate access rights to company resources.

5. Conclusions. Since the information systems are more open nowadays, which means also that more information is easily accessible to users, the task of better protection of confidential information becomes of essential importance. The logical security (i.e. access control) concerns the access control management based on the identification, authentication and authorization, counteracting the data modification or theft and wrong access to the data and programs. The traditional access control models are insufficient in distributed information systems, especially to express the policy of usage control. We need to have the unified security model to specify the general permissions, interdictions and obligations of an information system and to define the security rules dependent on an application context.

The presented paper focuses on the access control security in cooperative information systems. The proposed approach has to treat the cooperation of open and evaluative information systems and has to guarantee the respect of various local security policies on the global level. To solve these problems we propose to use the concepts of intelligent agents with their principles and abilities. This solution can preserve the control of data flow in the cooperative systems with respect of all security rules defined in each local system. The approach of multi-agent systems can be used in different domains of distributed information systems, e.g. electronic commerce, travel applications, public administration, management of university, management of hospital network.

REFERENCES

[1] S. Castano, M. Fugini, G. Martella and P. Samarati, *Database Security*, ACM Press, Addison-Wesley, 1994.

[2] R. S. Sandhu and Q. Munawer, *How to do Discretionary Access Control Using Roles*, Proceeding of 3rd ACM Workshop on Role-Based Access Control, 1998.

[3] D. Ferraiolo, R. S. Sandhu, S. Gavrila, D. R. Kuhn and R. Chandramouli, *Proposed NIST Role-Based Access Control*, ACM, Transactions on Information and Systems Security (TISSEC), Vol 4, No 3, 2001.

[4] G. Goncalves and A. Poniszewska-Maranda, *Role engineering: from design to evaluation of security schemas*, Journal of Systems and Software, Elsevier, Vol. 81, 2008.

[5] A. Poniszewska-Maranda, G. Goncalves and F. Hemery, *Representation of extended RBAC model using UML language*, LNCS, Proceedings of SOFSEM 2005: Theory and Practice of Computer Science, Springer-Verlag, 2005.

[6] B. Lampson, M. Abadi, M. Burrows and E. Wobber, *Authentication in Distributed Systems: Theory and Practice*, ACM Transactions on Computer Systems, 1992.

[7] M. Wooldridge, *An Introduction to MultiAgent Systems*, John Wiley & Sons, 2002.

[8] G. Weiss, *Multi-Agent Systems*, The MIT Press, 1999.

[9] M. Singh and M. Huhns, *Readings in Agents*, Morgan-Kaufmann Pub., 1997.

[10] J. Park and R. Sandhu, *The UCON ABC Usage Control Model*, ACM Transactions on Information and System Security, Vol 7, No 1, February 2004.

[11] G.-J. Ahn, *The RCL 2000 Language for Specifying Role-Based Authorization Constraints*, ACM Transactions on Information and Systems Security, 1999.

[12] G.-J. Ahn and R. Sandhu, *Role-based Authorization Constraints Specification*, ACM Transactions on Information and Systems Security, 2000.

[13] A. Ouksel and C. Naiman, *Coordinating Context Building in Heterogeneous Information Systems*, Journal of Intelligent Information Systems, No 3, Kluwer, Academic Publishers, 1994.

[14] A. Poniszewska-Maranda, *Conception Approach of Access Control in Heterogeneous Information Systems using UML*, Journal of Telecommunication Systems, Springer-Verlag, Vol 44, 2010.

[15] D. Basin, J. Doser and T. Lodderstedt, *Model Driven Security: from UML Models to Access Control Infrastructures*, ACM Transactions on Software Engineering and Methodology, Vol. 15, 2006.

[16] D. Basin, J. Doser and T. Lodderstedt, *Model Driven Security*, Engineering Theories of Software Intensive Systems, Springer, 2005.

[17] A. Poniszewska-Maranda, *Implementation of Access Control Model for Distributed Information Systems using Usage Control*, P. Bouvry et al. (Eds.): SIIS 2011, LNCS 7053, pages 54-67, Publisher: Springer, Heidelberg (2011).

[18] N. Antonopoulos, K. Koukoumpetsos and A. Shafarenko, *Access control for agent-based computing: a distributed approach*, Internet Research, Vol. 11 Issue: 1, pp.55 - 64 (2001).

[19] A. Seleznyov and S. Hailes, *Distributed Knowledge Management for Autonomous Access Control in Computer Networks*, International Conference on Information Technology: Information Assurance and Security, USA (2004).

[20] A. Seleznyov, M.O. Ahmed and S. Hailes, *ADAM: An Agent-based Middleware Architecture for Distributed Access Control*, The Twenty-Second International Multi-Conference on Applied Informatics: Artificial Intelligence and Applications, pages 200 - 205, Innsbruck, Austria (2004).

[21] L. Kagal, T. Finin and A. Joshi, *Trust-Based Security in Pervasive Computing Environments*, Computer (2001).

[22] V. Varadharajan, N. Kumar and Y. Mu, *Security Agent Based Distributed Authorization: An Approach*, Proceedings of the 21st National Information Systems Security Conference (NISSC), USA, pp. 315-328 (1998).

[23] E. Weippl, A. Schatten, S. Karim and A. Tjoa, *SemanticLIFE Collaboration: Security Requirements and solutions security aspects of semantic knowledge management*, Proceedings of Practical Aspects of Knowledge Management (PAKM), Austria (2004).

[24] P. Belsis, S. Gritzalis and Ch. Skourlas, *Security Enhanced Distributed Knowledge Management Architecture*, Proceedings

of I-KNOW, Austria (2005).

Edited by: Dana Petcu and Jose Luis Vazquez-Poletti
Received: November 1, 2011
Accepted: November 30, 2011

Scalable Computing: Practice and Experience
Volume 12, Number 4, pp. 417–421. http://www.scpe.org

ISSN 1895-1767
© 2011 SCPE

THE GRID AND CLOUD COMPUTING FACILITIES IN LITHUANIA

VAIDAS GIEDRIMAS[*], AUDRIUS VARONECKAS[†] AND ALGIMANTAS JUOZAPAVICIUS[‡]

Abstract. Nowadays ICT are faced the paradigm shift to the cloud computing. Even already distributed environments such as grids are challenged to change in order to meet new paradigm. This paper presents Lithuanian grid computing achievements, including National grid initiative (LitGrid), and our cloud computing potential. As an example two particular cloud-related applications are presented.

Key words: cloud computing, software services, LitGrid, optimization

1. Introduction.

Nowadays ICT are faced the paradigm shift to the cloud computing. Even already distributed environments such as grids are challenged to change in order to meet new paradigm. The solutions for migration into the cloud or to achieve efficient side-by-side usage of the grid and cloud technologies (e.g. StratusLab project [11]) are on the focus on world's scientific research.

This paper presents Lithuanian grid computing achievements in the cloud-computing context. As an example two particular cloud-related applications are presented.

The main aim of this paper is to present current position of Lithuanian National Grid Initiative (NGI) – LitGrid – and its potential in the cloud computing context.

The paper is organized as following. Section 2 outlines main distributed computing projects in Lithuania. Section 3 presents two optimization problems and in Lithuanian's distributed applications for its solving. Finally, conclusions are made and further steps are discussed.

2. Main Distributed Computing Projects in Lithuania.

2.1. LitGrid.

There are several private and public distributed computing infrastructures in Lithuania. Each university and some colleges has a cluster. One of the most powerful of them is private cluster constituted of 36 CPUs named Vilkas at Vilnius Gediminas Technical University (http://vilkas.vgtu.lt).

In order to connect the clusters and to improve the accessibility to computing resources in academic sector the program "Lithuanian distributed and parallel computing and e-services network (LitGrid)" has been started from 2006's. LitGrid now is not only grid as infrastructure but research community and powerful NGI. Currently it operates over 500 processors, has over 30 TB storage capacity, about 90 users, serves numerous scientific research areas. Lithuanian Grid functionality includes:

- deployment, maintenance and upgrading of the grid infrastructure and e-services;
- analysis, design, test deployment of the cloud computing technology, especially in relation to grids;
- serve for new users, their training tasks;
- creation, maintenance and deployment of the grid certificating procedure;
- presentation of the grid activities and capabilities to the society, to potential users, to public sector and business, developing new grid projects.

LitGrid consist of 13 academic institutions as partners: most active Lithuanian universities, research institutes and colleges. Academic and Research Network in Lithuania (Litnet) provides the network infrastructure with 1Gbps throughput connection between sites. The special communication center to support distributed computing data exchange has been implemented. LitGrid infrastructure is now based on gLite and ARC middleware, however several other operational systems are under consideration to be deployed (Globus, UNICORE, CREAM). Three sites are running central services (WMS, VOMS, LFC) for Lithuanian Grid. The important task for LitGrid is to participate in EGI organization and in the FP7 project EGI-InSPIRE. Such activities grant access to new technologies, developments, innovations, enables to take part in support and development procedures. The essential links for Grid are relations to Scandinavian countries grids, as well as joint efforts with Belarus academic institutions.

[*]Siauliai University, Visinskio 15, LT-77156 Siauliai, Lithuania, (grid@su.lt)
[†]Vytautas Magnus University, Vileikos 8, LT-44404 Kaunas, Lithuania, (a.varoneckas@if.vdu.lt)
[‡]Vilnius University, Naugarduko 24, LT-03225 Vilnius, Lithuania, (algimantas.juozapavicius@mif.vu.lt), http://www.litgrid.lt/

2.2. DiSCC. The biggest element of the Lithuanian NGI is Digital Science and Computing Center (DiSCC) of the Vilnius University, Faculty of Mathematics and Informatics (MIF) established together with two business partners - BAIP and TVM. The mission of DiSCC is to provide a translational environment for the transfer of the knowledge and research results to specific areas of the economics and to innovative digital products, maximally shortening the path from generation and exploration of new scientific ideas to computationally intensive market products.

DiSCC pays special attention to cooperation projects between scientific and public or private institutions, their partnership and elaborated feasibilities, in search for effective digitalization areas of Lithuanian economy and public sector, identification of such areas and implementation of suitable cooperation projects.

DiSCC also has a task to manage efficiently, elaborate and use computing resources of the MIF, for the purposes of the students' studies and for research work, for the implementation of newest trends in information technologies. The priority in this field belongs to grid, cloud and HPC computing technologies, to what they can offer to the science and the studies, as well as to public sector or business.

DiSCC is expected to be translated into international center, attractive for computer scientists, industry, foreign partners and for the state as a customer, helping them to digitize areas of the activities by using most suitable scientific and technological initiatives.

DiSCC offers for open access the supercomputer constituted of 1500 CPU cores and 600 TB of storage.

2.3. Main Litgrid applications . LitGrid directly or indirectly serves as a infrastructure for the national and international distributed applications. The main domains of Lithuanian applications are as follows (in the alphabetic order):

- Astronomy (the analysis of star's spectrum);
- Environmental research (the investigation of the Baltic sea ecosystem etc.);
- E-learning;
- Hydrodynamics (the modeling of the floodgates, barrages etc.);
- Health care (epilepsy diagnostics, orthopedics, radiology, fluorescence microscopy, biomarkers);
- Humanities (the analysis of text databases, language processing etc.);
- Materials science (nuclear physics, quantum chemistry, etc.);
- Mathematics (optimization, number theory etc.).

As a resource provider and/or partner LitGrid has been involved in the following projects:

- COST P19 – Multiscale modeling of the materials [10].
- PjezoAdapt - Development and research of mechatronic nanometer resolution multi-dimensional displacements generation / measurement systems
- ITER – the International Thermonuclear Experimental Reactor project [7].
- GridTechno – the Lithuanian national project of the applications of the grid technlogies for various research areas. [15].

LitGrid in the cooperation with Sweden, Estonia, Latvia, Poland and Switzerland (CERN) have been participated in the BalticGrid and the BalticGrid-II projects [1].

2.4. The migration towards the cloud. The migration to the cloud computing in Lithuanian academic sector is already started. Small clouds based on various platforms (OpenNebula, Windows Azure, EC2 etc.) has been created in VU, SU and other Lithuanian universities for the test purposes as well as for the scientific research.

Kaunas University of technology (KTU) [8] in cooperation with Vilnius University now is working on the project *Enabling the Researchers with the Gigabit Network Technologies*. One of the tasks of this project is the investigation of three virtualisation platforms:

- virtual machines and resource pools based on VMWare technologies;
- virtual machines based on open-source technologies (Xen,KVM,VirtualBox etc.);
- jail virtual machines (OpenVM,lxc, BSD jail etc.).

The process of KTU's information system migration into VMWare vSphere-based platform is already started.

Vilnius Gediminas Technical University (VGTU) is developing the workplace of the researcher based on cloud technology. Private cloud infrastructure was build based on Eucalyptus open source solution, XEN hypervisor and Rocks Clusters platform. Java Typica framework and JetS3t Toolkit for Amazon's S3 online storage service were used to develop the flexible interface to the user. ANSYS and MATLAB are considered as an interesting and promising cases of the user software [9].

Lithuanian cloud-related research is not limited to local projects only. Lithuania, as a partner of BalticGrid project has been involved in the BG subproject – BalticCloud [5], aimed at developing cloud infrastructure in Baltic states and Belarus. The infrastructure is based on open-source solutions and is available for research and teaching activities within the partner states. Now the BalticCloud is part ECEE initiative [4].

The cloud computing gains the popularity in the commercial sector as well as in academic. There are several companies which provide complete IaaS or PaaS solutions (e.g. [2, 13]). The customers of CC providers are small and medium business companies as well as the government of Lithuania. One of the biggest applications in public sector is the cloud computing solutions for the electronic census of Lithuanian population, performed this year.

The association of Lithuanian ICT companies *InfoBalt* [6] have performed the survey of its members (N=78), mainly SMEs (71% of the respondents). This survey shows the following prospects in Lithuanian cloud computing market:

- 33% of the respondents alredy provide CC services, 6% have plans to start such services this (2011) year and 15% ave plans to start next two years;
- Most popular CC services are related with CRM (33%) and ERP(30%) but almost 52% of the respondents have named other small services, e.g. e-mail, data managment etc.;

3. Optimization problems and grid-oriented applications in Lithuania. The researchers of LitGrid community develops and uses a large set of the grid-oriented applications. Part of these applications can be adopted (or already are under adopting) to the cloud computing, e.g. materials science applications [12]. It is not possible to describe all applications in one short paper. In this section examples of developed applications in Vilnius University Institute of Mathematics and Informatics (VU IMI), Vytautas Magnus University (VMU) and Siauliai University (SU) were taken.

3.1. VU IMI and VMU approach. Visualization of the multidimensional data is a large scale numerical problem which is time consuming and difficult to solve on a single personal computer. The Grid is an excellent candidate for providing the infrastructure needed for solving such problems.

Real world objects usually are described by multidimensional data sets. Visual representation can be very useful to grasp the structure of such data sets. There are a lot of the visualization techniques, which can be used for multidimensional data visualization. One of the powerful multidimensional data visualization techniques is multidimensional scaling (MDS). Multidimensional scaling is an exploratory technique for data analysis, widely usable in different applications, e.g. psychometrics, market analysis, data mining, visualization of general multidimensional data, visualization of the observation points in interactive global optimization.

Let a set of n multidimensional vectors (representing the considered objects) have to be visualized in a $p = 2$ dimensional space. Pairwise dissimilarities measured between all pairs of the objects are denoted by $\delta_{ij}, i, j = 1, ..., n$. It is supposed that dissimilarities are symmetric $\delta_{ij} = \delta_{ji}$ and $\delta_{ii} = 0$. The points $x_1, x_2, ..., x_n$ that constitute a set of n objects in p dimensional space should be found fitting pairwise distances of points to given pairwise dissimilarities $\delta_{ij}, i, j = 1, ..., n$. The fitness criterion, called STRESS function should be minimized:

$$\min S(X) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} w_{ij} (d(x_i, x_j) - \delta_{ij})^2$$

where $X = (x_1, x_2, ..., x_n), x_i = (x_{i1}, x_{i2}, ..., x_{ip}, d(x_i, x_j)$ denotes the distance between the points x_i and x_j, and w_{ij} denotes weights.

Quality of the projection of the multidimensional data into low-dimensional space depends on $d(x_i, x_j)$, and therefore choice of the metrics of the embedding space is very important. Pairwise dissimilarities between pairs of the object may be considered as a distance in multidimensional original space, and may be estimated using different norms in $\Re^p$. The most widely used distance measure is a Minkowski distance [3]:

$$d(x_i, x_j) = (\sum_{k=1}^{p} |x_{ik} - x_{jk}|^r)^{1/r}.$$

Parameter r influences the quality of the projection into low-dimensional space. Usually two well known special cases of Minkowski distance are used in MDS: Euclidean distance when $r = 2$, and city-block distance when

$r = 1$. The points $X = (x_1, x_2, ..., x_n)$ found by means of the minimization of STRESS function using different distance metrics are different nonlinear projections of the set of the objects in original multidimensional space to the lower dimensional embedding space.

MDS is a difficult global optimization problem. Although STRESS is defined by an analytical formula, which seems rather simple, its minimization is difficult. The function normally has many local minima. When city-block distances are used, STRESS can be non differentiable even at the minimum point [16]. The minimization problem is high dimensional (number of variables is $N = n \times m$) global optimization problem. When computing power of usual computers is not sufficient to solve a problem, the high performance parallel computers, clusters of the computers, computational grids and cloud computing may be helpful. An algorithm is more applicable in case its parallel implementation is available, because larger practical problems may be solved by means of parallel computation. Parallel version of genetic algorithm with multiple populations for MDS with Euclidean distances has been implemented in grid [14].

3.2. SU approach. There are three main distributed computing-related research areas of the Siauliai University:

- Interoperability of the e-learning and grid and cloud computing systems.
- Numerical modeling and investigation of the systems. Modeling of the temperature, stability and optimal design of the constructions (shells, plates, sticks) is investigated. New technology of numerical modeling of the constructions of composite materials using grid and cloud computing is being developed.
- Software synthesis for grid, cloud computing and other distributed systems. The synthesis distributed software to solve scientific problems of the numerical modeling and optimization of the systems.

On the umbrella of third research area new project at Siauliai University is started from the October 1 of 2010s. The main goal of this project is to develop the environment for scientific software synthesis using grid, cloud and wiki-oriented technologies.

We are stating the hypothesis that using together wiki-based technologies, software synthesis methods and the power of the grid/cloud infrastructure scientific software can be developed more rapidly and the quality of the software will be better. The project consist of three main stages:

1. The development of the portal for the wiki-based mass-collaboration. This portal will be used as the UI enabling scientists to specify the problems for software development, to rewrite/refine the specifications and software artifacts given by other researchers, to contribute all software developing for particular domain process. As the target domain for soft-ware development we have chosen the set of the statistical simulation and optimization problems. In the future the created environment can be applied to other domains.
2. The development of the model of the interoperability bridge between wiki-based portal and the LitGrid or other grid/cloud-based infrastructure. For this purpose currently the private OpenNebula-based cloud is created at SU.
3. To refine existing methods for software synthesis using the power of distributed computing infrastructures.

The results of the project will have direct positive impact in the scientific software development, because of the bridging two technologies, each of them promise good performance. The power of the wiki-technologies will ensure the ability of the interactive collaboration on software developing using the terms of particular domain. On the other hand the bridge of new environment and the grid/cloud infrastructure will give the possibility to use all power of distributed computing infrastructures.

Conclusions and Future Work. Lithuanian infrastructure for the distributed and parallel computing, including grid and cloud computing, is modern, advanced and continuous growing. The scientific LitGrid community is growing and the variety of applications is increasing. The applications from very different domains are observed, starting with the nuclear physics, ending with the public sector.

The migration to the cloud computing in Lithuanian academic sector is already started and first very promising results are already achieved. As a test cases for this migration scientific problems of optimization methods are selected. Our future work includes the continuous migration to cloud computing platform and investigation of the interoperability of grid and cloud technologies as well as the interoperability of different clouds. Lithuanian National Grid Initiative is open for collaboration for other NGI's, universities and business

Aknowledgements. The authors of this paper would like to address special thanks to all indirect contributors of this paper, especially Dr. D.Mazeika (VGTU), Dr. K. Paulikas (KTU), Prof. L.Sakalauskas (SU)

and Mr. A.Pleckaitis (InfoBalt association).

REFERENCES

[1] BalticGrid: Balticgrid-ii web site. `http://www.balticgrid.org/`
[2] BlueBridge: Bridge to cloud. `http://www.bridge2cloud.com/`
[3] Cox, T., Cox, M.: Multidimensional Scaling. Chapman and Hall/CRC (2001)
[4] ECEE: Enabling clouds for escience. `http://www.scientific-cloud.org/`
[5] Edlund, A.: Balticcloudproject. `http://cloud.balticgrid.eu/`
[6] INFOBALT: Infobalt association. `http://www.infobalt.lt/sl/index_en.php`
[7] ITER: The way to the energy. `http://www.iter.org/`
[8] KTU: Kaunas university of technology. `http://en.ktu.lt/`
[9] Mazeika, D., Kaceniauskas, A., Pacevic, R., Katkevicius, T.: Engineering application on university private cloud. EGI User Frorum 2011 abstract session.
[10] P19, A.: Multiscale modelling of materials. `http://www.ipm.cz/costp19/`
[11] StratusLab: Enhancing grid infrastructures with virtualization and cloud technologies. `http://stratuslab.eu/`
[12] Tamuliene, J., Vaisnoras, R., Badenes, G., Balevicius, L.: Point of view on magnetic properties of con (n=6,8,10,12) based on quantum chemistry investigations
[13] TEO: The cloud computing-based voice telephony services. `http://www.teo.lt/en/press-en/2733`
[14] Varoneckas, A., Zilinskas, A., Zilinskas, J.: Parallel multidimensional scaling using grid computing: assessment of performance. Information technology and control 37, 52–56 (2008)
[15] VGTULPC: Gridtechno project. `http://lsl.vgtu.lt/en`
[16] Zilinskas, A., Zilinskas, J.: Two level minimization in multidimensional scaling. Journal of Global Optimization 38, 581–596 (2007)

Edited by: Dana Petcu and Jose Luis Vazquez-Poletti
Received: November 1, 2011
Accepted: November 30, 2011

AIMS AND SCOPE

The area of scalable computing has matured and reached a point where new issues and trends require a professional forum. SCPE will provide this avenue by publishing original refereed papers that address the present as well as the future of parallel and distributed computing. The journal will focus on algorithm development, implementation and execution on real-world parallel architectures, and application of parallel and distributed computing to the solution of real-life problems. Of particular interest are:

Expressiveness:
- high level languages,
- object oriented techniques,
- compiler technology for parallel computing,
- implementation techniques and their efficiency.

System engineering:
- programming environments,
- debugging tools,
- software libraries.

Performance:
- performance measurement: metrics, evaluation, visualization,
- performance improvement: resource allocation and scheduling, I/O, network throughput.

Applications:
- database,
- control systems,
- embedded systems,
- fault tolerance,
- industrial and business,
- real-time,
- scientific computing,
- visualization.

Future:
- limitations of current approaches,
- engineering trends and their consequences,
- novel parallel architectures.

Taking into account the extremely rapid pace of changes in the field SCPE is committed to fast turnaround of papers and a short publication time of accepted papers.

INSTRUCTIONS FOR CONTRIBUTORS

Proposals of Special Issues should be submitted to the editor-in-chief.

The language of the journal is English. SCPE publishes three categories of papers: overview papers, research papers and short communications. Electronic submissions are preferred. Overview papers and short communications should be submitted to the editor-in-chief. Research papers should be submitted to the editor whose research interests match the subject of the paper most closely. The list of editors' research interests can be found at the journal WWW site (`http://www.scpe.org`). Each paper appropriate to the journal will be refereed by a minimum of two referees.

There is no a priori limit on the length of overview papers. Research papers should be limited to approximately 20 pages, while short communications should not exceed 5 pages. A 50–100 word abstract should be included.

Upon acceptance the authors will be asked to transfer copyright of the article to the publisher. The authors will be required to prepare the text in $\LaTeX 2_\varepsilon$ using the journal document class file (based on the SIAM's `siamltex.clo` document class, available at the journal WWW site). Figures must be prepared in encapsulated PostScript and appropriately incorporated into the text. The bibliography should be formatted using the SIAM convention. Detailed instructions for the Authors are available on the SCPE WWW site at `http://www.scpe.org`.

Contributions are accepted for review on the understanding that the same work has not been published and that it is not being considered for publication elsewhere. Technical reports can be submitted. Substantially revised versions of papers published in not easily accessible conference proceedings can also be submitted. The editor-in-chief should be notified at the time of submission and the author is responsible for obtaining the necessary copyright releases for all copyrighted material.