The MAIS approach to web service design

Marzia Adorni§, Francesca Arcelli§, Danilo Ardagna#, Luciano Baresi#, Carlo Batini§, Cinzia Cappiello#, Marco Comerio§, Marco Comuzzi#, Flavio De Paoli§, Chiara Francalanci#, Simone Grega§, Paolo Losi§, Andrea Maurino§, Stefano Modafferi#, Barbara Pernici#, Claudia Raibulet§, Francesco Tisato§

[#]Politecnico di Milano, Dipartimento di Elettronica e Informazione [§]Università di Milano Bicocca, Dipartimento di Informatica, Sistemistica e Comunicazione

> Mailing address: Prof. Chiara Francalanci Dipartimento di Elettronica e Informazione Politecnico di Milano Piazza Leonardo Da Vinci, 32 20133, Milano, Italy francala@elet.polimi.it

Abstract. This paper presents a first attempt to realize a methodological framework supporting the most relevant phases of the design of a value-added service. A value-added service is defined as a functionality of an adaptive and multi-channel information system obtained by composing services offered by different providers. The framework has been developed as part of the MAIS project. The MAIS framework focuses on the following phases of service life cycle: requirements analysis, design, deployment, run time use and negotiation. In the first phase, the designer elicits, validates and negotiates service requirements according to social and business goals. The design phase is in charge of modelling services with an enhanced version of UML, augmented with new features developed within the MAIS project. The deployment phase considers the network infrastructure and, in particular, provides an approach to implement and coordinate the execution of services from different providers. In the run time use and negotiation phase, the MAIS methodology provides support to the optimal selection and quality renegotiation of services and to the dynamic evaluation of management costs. The paper describes the MAIS methodological tools available for different phases of service life cycle and discusses the main guidelines driving the implementation of a service management architecture, called reflective architecture, that complies with the MAIS methodological approach.

1 Introduction to the MAIS Project

The design and implementation of multichannel and mobile information systems presents cross-disciplinary research problems: the information system should support adaptivity, since the execution environment is characterized by continuous change, in particular in mobile and ubiquitous systems; it is highly distributed and characterized by a high heterogeneity in both technological platforms and user requirements, and therefore concepts such as stratification and information hiding turn out to be inadequate, since it is almost impossible to identify and implement optimal built-in strategies; non-functional requirements (performance, reliability, security, cost, and, more generally, quality of service) become more and more relevant and the management of system resources cannot be hidden, but has to be visible and controllable at the application level.

The goal of the MAIS project¹ is the development of models, methods, and tools that allow the implementation of multichannel adaptive information systems. Functionalities are provided as services on different types of networks and access devices and are the result of the composition of component services offered by different providers to build a value-added service. In this paper, we present a first attempt to realize a methodological framework supporting the most relevant phases of the design of a value-added service. In particular, we focus on the creation of a service (e.g., analysis, design, development) as an abstract service and its use as a component service.

In MAIS, the design of value-added services is limited to the abstract definition of their functional and non functional features, with no implementation details such as service location and service access protocol. Component services are associated with an abstract value-added service at run time, according to different deployment alternatives. The paper discusses the design of deployment alternatives and of run-time quality control procedures.

Several approaches have been proposed in the literature for the design of web services as composed services and of cooperative information systems based on a service oriented approach. Some approaches focus on the goal-based selection of component services at a conceptual level: in [11] cooperative processes are built on the basis of intentions and strategies in virtual organizations and in [9] a goal-based approach considering non-functional requirements to identify resources and constraints is proposed. Other approaches dynamically select and adapt services based on meta-level descriptions [10] or compose services based on planning and monitoring techniques [12]. Process design in organizations cooperating based on a service oriented approach has been studied by focusing on process control and responsibility in [14] and on the evolution of cooperation processes in [8]. Quality of service is increasingly studied in the service orientation literature [18, 15]. The focus however is more on the representation and monitoring of quality of service aspects rather than on design for quality.

The goal of this paper is to provide a first integrated view of design aspects which are not considered in an integrated methodological framework in the literature. In particular, we focus on service selection and on the representation of quality requirements at a system level.

The paper is organized as follows: Section 2 presents the MAIS methodological framework. Sections 3 to 7 describe each component of our methodological framework, while the last section draws conclusions and outlines future work.

¹ MAIS (Multichannel Adaptive Information Systems), web site www.mais-project.it

2 The MAIS Methodological Framework

The life cycle of web services, both simple and complex, is composed of a series of methodological phases, from requirements analysis to service monitoring at run time. Figure 1 reports the phases of the MAIS methodological framework:

- requirements analysis;
- design;
- deployment;
- run time use and negotiation.

In the first phase, the designer elicits, validates and negotiates web service requirements according to social and business goals. Services are supposed to be provided to users through different distribution channels. The inputs of this phase are domain requirements, QoS requirements, user profiles, and architectural requirements for different distribution channels. The output of this phase is a set of functional and non-functional requirements, which is taken as input by the subsequent design phase (see Figure 1). The MAIS methodological framework described in this paper assumes that an informal description of functional and non-functional requirements is available and provides support starting from the design phase.

The design phase (Section 3) is in charge of modelling services with an enhanced version of UML, augmented with new features developed within the MAIS project (e.g., Abstract Interaction Unit [5]). At this stage of the methodology, the designer is interested in defining a high-level description of the whole system. Therefore, starting from functional and non-functional requirements, the designer identifies the information and operating services that will be supplied in a multi-channel fashion and the corresponding distribution channels. The result of this phase is a set of MAIS-UML diagrams that will be used in the following phases. Design is also supported by the evaluation of the management costs of services with a varying level of QoS. This evaluation, described in Section 4, allows the analysis of different service scenarios and the selection of the most profitable approach to service management for the MAIS brokering architecture.

The deployment phase (Section 5) considers the network infrastructure. The MAIS methodology provides an approach to implement and coordinate the execution of complex services built from multiple services of different providers. The input of this phase is a MAIS-BPEL description, describing a composition of abstract services augmented with QoS and coordination definitions automatically derived from the MAIS-UML diagrams. The output is a set of MAIS-BPEL specifications.

In the run time use and negotiation phase (Section 6), the MAIS project proposes two different tools supporting the adaptive and context-aware use of web services. The first one is based on the optimal selection and quality renegotiation of services based on a set of abstract descriptions of services and QoS requirements. The second one is in charge of supporting the negotiation and dynamic evaluation of managements costs allowing the maximization of MAIS brokering profits.

The first tool allows workflow engines to invoke the best service satisfying a set of requirements according to the specific execution context and end-user profile. The concepts of abstract services and concrete services are distinguished. An abstract service is a non-invokable service specifying the functional interface of the service and its QoS requirements. A concrete service is a completely described service, i.e., an invokable service, inheriting the functional interface and QoS requirements of a corresponding abstract service, but specifying additional implementation details (e.g., access protocol). This distinction allows the designer to define a generic description of web services at design time without paying attention to implementation problems. Implementation problems can be solved at run time, when the right (and optimal) selection and invocation of web services is realized.



Fig. 1. MAIS contributions to the web service life cycle

The second tool evaluates the returns of the MAIS brokering service for each concrete service. This supports run time decisions on the most profitable degree of QoS improvement that the MAIS brokering architecture can implement to meet user requirements. The MAIS architecture can improve QoS in several ways. For example, it can improve the quality of a data set requested by a user by complementing the information provided by the supplier of the concrete service with higher-quality information from additional sources. These improvements increase QoS, but also involve additional costs. Profits are maximized when the returns from higher QoS are greater than QoS improvement costs. Section 6 describes the MAIS methodological approach to these cost-benefit evaluations. The MAIS project has also proposed a reflective architecture that can support the run time selection of services and QoS negotiation. The term "reflective" indicates the ability of a system to adapt dynamically to user requirements by using appropriate metadata. In this paper, we present the main guidelines driving the implementation of a reflective architecture, to show how it is possible to design and realize a reflective middleware even in a fully distributed environment (Section 7).

Figure 1 summarizes how the MAIS approach supports the web service life cycle. In the following sections, we describe the MAIS contribution for each methodological phase. The reader will be referred to MAIS papers and reports to find more detailed descriptions of the methodological components described in the next sections.

3 Service Specification and Compatibility Analysis

Research work on service design has started from the definition of a methodology for the redesign of existing services, described in [19] and [20]. This redesign methodology is based on existing specifications of services and information on new requirements. The service redesign methodology considers several aspects of the information on new requirements, including communication channels and technologies, user profiles, and quality of service (see Figure 1). The redesign methodology has also reconsidered traditional development processes to take into account new requirements (see [19] and [20]). The output of the methodology are enhanced UML diagrams that describe services in terms of functional and non functional properties.

Recently, a revised version of the methodology that considers design in addition to redesign has been proposed. The revised methodology is composed by 3 macrophases: functional service modeling, high-level redesign, and context adaptation. The functional service modeling phase aims at modelling functional service requirements as a set of UML diagrams. These diagrams highlight the logical and operational structure of services. The main objective of the second phase, high-level redesign, is to redesign existing services according to new requirements. QoS requirements are modelled by means of appropriate quality dimensions and metrics extracted from the MAIS QoS registry, which provides a structured list of QoS dimensions and corresponding metrics. QoS requirements are quantified with B_k values that represent the quality level that the service must provide for the k^{th} quality dimension. Finally, QoS constraints are modeled by using an extension of UML proposed by OMG [21]. The enhanced UML diagrams, that are the output of this phase, define services at an abstract level, i.e., without considering specific technologies or user characteristics. Such diagrams will be exploited to actually implement and deploy the service in subsequent methodological phases.

The context adaptation phase takes into account the actual target environment in order to evaluate technological and user requirements. An abstract QoS requirement is verified if contextual technical characteristics (for example, the actual device or the network connection) provide quality values greater than or equal to threshold B_k .

The value of each quality dimension can be quantified by considering ideal quality values associated with the profile of the requesting user. Therefore, a comparison between the level B_k of each quality dimension defined in the high-level redesign

phase (service quality request) and the ideal level B_u associated with the profile of the requesting user (user quality request) allows the compatibility analysis between user requirements and service characteristics. An overall evaluation of compatibility on multiple quality dimensions can be performed by using QoS trees, where each node represents a quality dimension. The MAIS methodology provides a bottom-up quality evaluation approach based on the Simple Additive Weighting technique [20, 22]. If design assumptions are compatible with quality requirements, the context adaptation phase is completed, otherwise the set of violated constraints is provided.

4 Broker-Provider Negotiation and Dynamic Evaluation of Management Costs

The MAIS methodology assumes the existence of a broker between providers and users. The broker has two conflicting goals: to maximize the satisfaction of user requirements and to achieve maximum possible returns from its brokering role. The broker is supposed to be paid by each provider every time a service of that provider is supplied to a user. Payment is quantified as a percentage of price. The value of this percentage is the output of a negotiation process between the broker and the provider occurring when the provider subscribes the brokering service. The broker can also increase the quality of a service offered by a provider by complementing the service in several ways, whose discussion is out of the scope of this paper.

The aim of the service provider *i* of the service *j* and the broker in the preliminary negotiation phase is to set the value of a triple $\langle p_{ij}, perc_{ij}, q_{ij} \rangle$ where p_{ij} is the price paid by the user for the service, $perc_{ij}$ is the percentage on the price due by the service provider to the broker, comprised between 0 and 1, and q_{ij} is the aggregate value of QoS (see Section 3) with which the service will be provided ($0 \leq q_{ij} \leq 1$). The negotiation process is defined by the Negotiation Protocol, the Negotiation Objectives, and the participants' Decision Model, as discussed in [23]. A utility function *V* is defined, evaluating how much an offer is worth to a participant. Such utility function is:

- $V = \frac{p_{ij}}{perc_{ij} \cdot q_{ij}}$ for the provider, which is interested in maximizing its revenue;
- $V = p_{ij} \cdot q_{ij} \cdot perc_{ij}$ for the broker, which is interested in maximizing both its revenue and user satisfaction.

The broker can increase the service quality level q_{ij} to a quality level q_{ij}^* . In order to provide an example, let us consider a user that requires a data quality level equal to Q_j . If the service provider can offer a quality $q_{ij} < Q_j$, the broker can increase the quality level by improving the data provided with other data retrieved from certified external sources. In general, in order to increase the quality level of a service, the broker will incur an extra cost $c^*(q_{ij}^*)$, but can also provide the service to the customer at a higher price $p^*(q_{ij}^*)$. Formally, the goal of the broker is to maximize the function: $W_{Broker}*U_{Broker}(q)+W_{User}*U_{User}(q)$, where U_{Broker} and U_{User} indicate the broker and user utility functions, while W_{Broker} and W_{User} are two weights such that $W_{Broker}+W_{User}=1$, which establish the relative importance of broker returns and user satisfaction. Figure 2 shows a sample user utility function. If the quality level provided by the MAIS platform equals the quality level \overline{q} required by the end-user, then the user utility function reaches its maximum. Vice versa, the broker's utility function is expressed as the net revenue from service provisioning, i.e. $U_{Broker}(q)=p^*(q)-p+p*percc^*(q)$. As will be discussed in Section 6 the maximization problem is NP-hard if the platform has to guarantee global constraints for the execution of complex services built from simple services from multiple providers.



Fig. 2. Sample User utility function

5 Process Partitioning

The execution of a complex service in a mobile environment, with different devices connected through different network technologies, needs new strategies with respect to the traditional solutions adopted for centralized workflows. These solutions rely on a single engine that knows and controls all system resources, while mobility demands a decentralized execution carried out by a federation of heterogeneous devices. These requirements lead to a new strategy that stresses the independency among actors, to minimize interaction and knowledge sharing, and, thus, increases reliability.

The MAIS methodology proposes a set of formal partitioning rules that transform a unique workflow into a set of federated workflows that can be executed by different engines. Our partitioning approach is based on graph transformation [BH02], where a typed graph defines the types of nodes and edges that can be used to create graphs and transformation rules manipulate these graphs. The left-hand side L of a rule defines the pre-conditions that must hold on the graph to enable the rule, while the right-hand side R describes the post-conditions, that is, the modifications on the graph after applying the rule.

The rules read a MAIS-BPEL specification of the original workflow, along with the description of the topology of the network infrastructure (i.e., the list of available engines). The result is a set of MAIS-BPEL specifications that represent the local processes (views) of each engine.

The partitioning framework is implemented as a Web service, called Partitioner, based on AGG, an existing general-purpose graph transformation tool. This module

receives a GXL file, representing the original MAIS-BPEL description, and produces a set of GXL files representing the local views for the orchestrators. Consequently, we first translate the original MAIS-BPEL description into GXL, by means to XSL technology, and then we re-translate GXL files into a MAIS-BPEL description.

The feasibility of our transformation depends on the assumptions that partitioning rules define a graph transformation system that exposes a functional behavior, i.e., is confluent and terminates. Moreover, the execution flow of the original workflow has to be preserved. We check the first hypothesis by exploiting the critical pair analysis capabilities supplied by AGG [7]. Our rules have no conflicts such as the ones described before; thus our graph transformation system has a functional behaviour [6]. Currently our proof of the second hypothesis is based on the observation that partitioning rules only add activities to synchronize the different sub-workflows, which do not alter the execution flow.

6 Optimal Service Selection and Quality Renegotiation

The goal of this phase is to select a set of services satisfying requirements from a registry of available services at runtime. Usually, a set of functional equivalent services can be selected, i.e., services which implement the same functionality but differ in their quality parameters [37]. Therefore, service selection raises an optimization problem. In the work presented in [17], two main approaches have been proposed: local and global optimization. The former selects at run time the best candidate service which supports the execution of a running high level activity. The latter identifies the set of candidate services which satisfy end-user preferences for an entire application. The two approaches allow the specification of Quality of Service (QoS) constraints at a local and global level, respectively. A local constraint allows the selection of a service according to a required characteristic. For example, a service can be selected such that its price or its execution time are lower than a given threshold. Global constraints are constraints such as "The overall execution time of the application has to be lower than 3 seconds" or "total price has to be lower than 2\$."

Note that the end-user is mainly interested in global constraints. For example, a user is typically concerned with the total execution time of the application instead of the execution time of individual activities. Furthermore, service composition could be transparent to the end-user. In the MAIS methodology, we have implemented a global approach for service selection and optimization. The problem of service composition with QoS constraints has been modelled as a mixed integer linear programming problem. The problem is NP-hard, since it is equivalent to a Multiple Choice Multiple Dimension Knapsack problem [16, 22, 18]. The optimization problem is solved by formulating a mixed integer linear programming (MILP) model which is solved with CPLEX, a state of the art commercial solver, which implements a branch and cut technique [18]. In the MAIS methodology, negotiation, service selection, optimization, and service execution are interleaved. Re-optimization is performed periodically, if the end-user changes the service channel or if a service invocation fails.

In order to evaluate the effectiveness of our approach, we have compared our solutions with those provided by the local optimization approach proposed in [17]. Results have shown [16] that the global optimization provides better results, since bounds for quality dimensions can be always guaranteed and the value of the quality dimensions can be improved by 10-70%.

7 Implementation Guidelines for a QoS-Oriented Reflective Architecture

The methodological framework for the definition of adaptive services introduced in the previous sections is supported by an underlying reflective architecture. Generally, services rely on a logical layer (e.g., OS and middleware) exploiting functional features of the system components (e.g., devices and network services). Architectural reflection [26, 28] introduces a reflective layer allowing applications to observe and control non-functional features of system components at execution time, thus supporting adaptability. A reflective layer is causally connected [28] to the logical layer. The reflective architecture [24, 25, 29] models the quality of service [27] of system components by means of reflective objects (R_Objects).



Fig. 3. QoS Extension Pattern

Neither components nor their QoS can be defined in an absolute way. For example, an application may observe just the maximum screen resolution of an end-user channel in terms of qualitative, domain-dependent QoS (e.g., "low", "medium", "high"). A different application may observe and/or control both specific devices (e.g., a desktop monitor, a wall screen, and a projector) and their "*pixel x pixel*" resolution. Therefore, a general mechanism for defining R_Objects and their QoS according to domain requirements is needed.

The QoS extension pattern of Figure 3 highlights that an R_Aggregate is a reflective object whose QoS is causally connected via a QoSStrategy to the QoS of a collection of R_Elemental reflected objects. The *mapUp()* method of the QoSStrategy defines how QoS of an aggregate is obtained by exploiting the QoS of its elemental components. The *mapDown()* method defines how the QoS of an aggregate is mapped onto the QoS of its elemental components. Figure 4 shows how the general extension pattern fits into the reflective architecture. R_Objects at the *Base Reflective Layer* are causally connected to the Logical Layer components. They expose measurable QoS values which can be observed and/or controlled via platform-dependent mechanisms.

R_Objects at the *Extended Reflective Layer* model higher-level, domain-oriented abstractions. For example, the maximum resolution of a laptop is computed as the maximum resolution of all the display components it is connected to (e.g., wall monitor, desktop, hand-on device monitor). The bandwidth of the extended network service can be controlled by selecting one among several service providers.



Fig. 4. MAIS Reflective Layers exploiting QoS Strategies

8 Conclusions

This paper discusses a first attempt to define a methodological framework supporting the most relevant phases of the design of a value-added service, that is a functionality of an adaptive and multichannel information system obtained by composing services offered by different providers. The discussion has focused on the following phases of the life cycle of web-services: requirements analysis, design, deployment, run time use and negotiation. Current work is focusing on the use of specific requirement techniques to elicit user requirements and usage scenario [1], and to extend our proposal including other contributes of the MAIS project such as design and deployment of context-aware data-intensive web applications [2], techniques for evaluating the usability of interfaces [4], and tools for adaptive interfaces [3].

Acknowledgements

This work has been supported by the Italian MIUR-FIRB Project MAIS. The authors acknowledge the contribution of all MAIS participants to this work in many discussions in project meetings.

References

- 1. D. Bolchini and J. Mylopoulos. From Task-Oriented to Goal-Oriented Web Requirements Analysis. In *Proceedings of the 4th International Conference on Web Information Systems Engineering, WISE'03, Rome, Italy,* December 2003.
- S. Ceri, P. Fraternali and A. Bongio. Web Modeling Language (WebML): a Modeling Language for Designing Web Sites. In *Proceedings of the 9th International World Wide* Web Conference, WWW 2000, Amsterdam, The Netherlands, May 2000.
- R. Torlone and P. Ciaccia. Management of User Preferences in Data Intensive Applications. In Proceedings of the 11th Italian National Conference on Advanced Data Base Systems, SEBD 2003, Cetraro, Italy, June 2003.
- 4. E. Bertini, M. Billi, L. Burzagli, T. Catarci, F. Gabbanini, P. Graziani, S. Kimani, E. Palchetti, and G. Santucci. Evaluation of the Usability and Accessibility wrt Channels, Devices, and Users. MAIS technical report 7.3.5, 2005.
- 5. E. Bestini and G. Santucci. Modeling internet based applications for designing multidevice adaptive interfaces. In *Proceedings of the working conference on Advanced Visual Interfaces, AVI 2004, Gallipoli, Italy,* May 2004.
- 6. L. Baresi, A. Maurino, and S. Modafferi. Workflow Partitioning in Mobile Information Systems. In *Proceedings of the IFIP TC 8 Working Conference on Mobile Information Systems (MOBIS), Oslo, Norway*, pages 93-106, September 2004.
- J.H. Hausmann, R. Heckel, and G. Taentzer. Detection of conflicting functional requirements in a use case-driven approach: a static analysis technique based on graph transformation. In *Proceedings of the International Conference on Software Engineering, ICSE 2002, Orlando, FL, USA*, pages 105-115, May 2002.
- K. Baïna, B. Benatallah, F. Casati, and F. Toumani. Model-Driven Web Service Development. In Proceedings of the 16th International Conference on Advanced Information Systems Engineering, CAiSE 2004, Riga, Latvia, pages 290-306, June 2004.
- E. Colombo, C. Francalanci, and B. Pernici. Modeling Cooperation in Virtual Districts: A Methodology for E-Service Design. *Int. Journal of Cooperative Information Systems*, 13(4):369-411, 2004.
- 10. F. Casati and M. Shan. Dynamic and adaptive composition of e-services. *Information Systems*, 6(3), 2001.
- R.S. Kaabi, C. Souveyet, and C. Rolland. Eliciting service composition in a goal driven manner. In Proceedings of the 2nd International Conference on Service Oriented Computing, ICSOC 2004, New York City, NY, USA, pages 308-315, November 2004.
- A. Lazovik, M. Aiello, and M.P. Papazoglou. Planning and Monitoring the Execution of Web Service Requests. In *Proceedings of the 1st International Conference on Service Oriented Computing, ICSOC 2003, Trento, Italy,* pages 335-350, November 2004.
- 13. D. Menasce. Composing Web Services: A QoS View., *IEEE Internet Computing*, 8(6), November/December 2004.

- M. Mecella, F. Parisi-Presicce, and B. Pernici. Modeling E -service Orchestration through Petri Nets. In Proceedings of the 3rd VLDB International Workshop on Technologies for E-Services, TES'02, Hong Kong, China, pages 38-47, August 2002.
- A. Abhijit, A. Patil, A. Swapna, A. Oundhakar, A. P. Sheth, and K. Verma. Meteor-s web service annotation framework. In *Proceedings of the 13th International World Wide Web Conference, WWW 2004, New York City, NY, USA*, pages 553-562, May 2004.
- D. Ardagna and B. Pernici. QoS Evaluation in Web Service Selection. Politecnico di Milano, Technical Report n. 2005.20, 2005.
- L. Zeng, B. Benatallah, M. Dumas, J. Kalagnamam, and H. Chang. QoS-Aware Middleware for Web Services Composition. *IEEE Transactions on Software Engineering*, 30(11):315-327, May 2004.
- 18. L. Wolsey. Integer Programming. John Wiley & Sons, 1998.
- M. Comerio, F. De Paoli, C. De Francesco, A. Di Pasquale, S. Grega, and C. Batini. A Redesign Methodology for Multi-channel Applications in the Zootechnical Domain. In *Proceedings of the 12th Italian Symposium on Advanced Database Systems, SEBD 2004, S.Margherita di Pula, Italy*, June 2004.
- M. Comerio, F. De Paoli, S. Grega, C. Batini, C. Di Francesco, and A. Di Pasquale. A service re-design methodology for multi-channel adaptation. In *Proceedings of the 2nd International Conference on Service Oriented Computing, ICSOC 2004, New York City, NY,* USA, November 2004.
- UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms. OMG report, September 2004.
- W.Y. Lum and F.C.M. Lau. User-Centric Content Negotiation for Effective Adaptation Service in Mobile Computing. *IEEE Transaction on Software Engineering*, 29(12):1000-1111, December 2003.
- 23. N.R. Jennings, A.R. Lomuscio, S. Parsons, C. Sierra, and M. Wooldridge. Automated Negotiation: prospects, methods, and challenges. *International Journal of Group Decision and Negotiation*, 10(2):199-215, 2001.
- M. Adorni, F. Arcelli, C. Raibulet, M. Sarini, and F. Tisato. Designing an Architecture for Multichannel Adaptive Information Systems. In *Proceedings of the 2004 International Conference on Software Engineering Research and Practice, SERP 2004, Las Vegas, Nevada, USA*, pages 652-658, June 2004.
- F. Arcelli, C. Raibulet, F. Tisato, and M. Adorni. Architectural Reflection in Adaptive Systems. In *Proceedings of the 16th International Conference on Software Engineering & Knowledge Engineering, SEKE 2004, Banff, Alberta, Canada,* pages 74-79, June 2004.
- W. Cazzola, A. Savigni, A. Sosio, and F. Tisato. Rule-Based Strategic Reflection: Observing and Modifying Behaviour at the Architectural Level. In *Proceedings of the 14th IEEE International Conference on Automated Software Engineering*, pages 263-266, 1999.
- D. Chalmers and M. Sloman. A Survey of Quality of Service in Mobile Computing Environments. *IEEE Communications Surveys*, 1999.
- 28. P. Maes. Concepts and Experiments in Computational Reflection. In Proceedings of the Conference on Object-Oriented Programming Systems, Languages and Applications, OOPSLA'87, pages 147-155, 1987.
- 29. F. Tisato et al. The MAIS Reflected Architecture. MAIS Technical Report, November 2004.
- D. Bianchini, V. De Antonellis, B. Pernici, and P. Plebani. Ontology-based methodology for e-Service discovery. *Journal of Information Systems*, Elsevier Science (to appear).