

Personalization, verification and conformance for logic-based communicating agents

Matteo Baldoni, Cristina Baroglio, Alberto Martelli,
Viviana Patti, Claudio Schifanella, Laura Torasso

Dipartimento di Informatica

Università degli Studi di Torino

C.so Svizzera, 185 — I-10149 Torino (Italy)

Email: {baldoni,baroglio,mrt,patti,schi,ltorasso}@di.unito.it

Viviana Mascardi

Dipartimento di Informatica e Scienze dell'Informazione

Università degli Studi di Genova

Via Dodecaneso, 35 — I-16146 Genova (Italy)

Email: mascardi@disi.unige.it

Abstract— This paper is an overview of the work that we have carried on in the last two years in the context of the MASSIVE project. The main research lines have concerned personalization of the interaction with web services, personalization of courseware, web services interoperability, and integrated environments for agent oriented software engineering. All of them can be seen as applications of different reasoning techniques to a declarative specification of interaction. A declarative specification makes the study of properties easy and allows a fast prototyping of applications. In particular, we applied reasoning about actions and change to the personalized selection and composition of web services and to the construction of courseware that satisfies the user's needs and goals. This kind of reasoning has also been integrated in the DCaseLP MAS prototyping environment. Declarative specifications have also been helpful to face the problem of proving policy conformance in a way that guarantees web service interoperability. Finally, the adoption of process languages for web services for expressing the procedural behavior of adaptive BDI-style agents have been explored.

I. INTRODUCTION

Computational logics and declarative languages are being rediscovered as a tool for some of the most innovative application areas: the Semantic Web and Web Services. By definition, the Semantic Web comprises a machine-shareable representation of knowledge, and it both requires the development of languages for expressing information in a machine-processable form, and the use of inferencing mechanisms that allow a content-aware navigation. The desired result is an overall behavior that is closer to the user's intuition and desire and the possible applications are really various, depending on the kind of resource that is described and on the tasks to be performed. On the other hand, it is getting more and more common describing and realizing applications as sets of cooperative services. This is the case, for example, for manufacturing processes, on-line markets, distributed network management. The traditional approach is based on a functional view, in which the different components require some specific input, and produce some specific output. The system's architecture is based on the principle of static-functional decomposition, where the interactions among the different components are given by their dependencies. Other approaches are, however, being studied which involve describing at a high-level the behavior of the services. The aim is to enable the adoption

of automated reasoning mechanisms for retrieving, composing, invoking services. One of these approaches is the Multi-Agent paradigm, in which the different components dynamically communicate and coordinate with each other, by means of declarative languages, to reach some common (or their own) goal. Among the social aspects, specifically relevant is the ability of expressing behavioral rules, aiming at controlling the organization of the system; communication protocols are the most significant example of such rules. Protocols are used to rule the agents' interaction, therefore, they can be used to check if a given agent can, or cannot, take part into the system, or to check whether the system is behaving as expected. In general, based on this abstraction, open systems can be realized, in which new components can dynamically join the system. The insertion of a new component in an execution context is determined according to some form of reasoning about its behaviour: it will be added provided that it satisfies the body of the rules within the system, intended as a society.

The researches that we have carried on the last two years tackle different aspects related to the Semantic Web and Web Services, in the setting of Multi-Agent Systems. In particular, we have extended the DyLOG language [1], which is the common tool used in all the branches of the research that we have carried on. The extension [2] mainly concerns the introduction of a communication kit aimed at tackling communication in a way that is fully integrated with the representation and reasoning mechanisms of the language. Each of the next sections describes one of the lines of research that we have been pursued and the work carried on in that context. Section II reports about work in the context of personalization of courseware; Section III discusses personalization in the service selection and composition processes; Section IV reports results concerning the proof of interoperability and conformance of services to a global description of their interaction; Section V describes the adoption of process languages for expressing the procedural behavior of adaptive BDI-style agents; Section VI describes an integrated environment for AOSE.

II. PERSONALIZATION OF COURSEWARE

Personalized information systems aim at giving the individual user optimal support in accessing, retrieving, and

storing information. The individual requirements of the user are to be taken into account in such different dimensions like the current task, the goal of the user, the context in which the user is requesting the information, the previous information requests or interactions, the working process s/he is involved in, the level of expertise, the device s/he is using to display the information, the bandwidth and availability of the communication channel, the abilities (disabilities or handicaps) of the user, his/her time constraints, and many, many more. Different research disciplines have contributed to explore personalization techniques and to evaluate their usefulness within various application areas: adaptive hypertext systems, collaborative filtering, recommender systems, artificial intelligence, uncertainty management, and so forth. In this section we will focus on an e-learning scenario and see how reasoning can help personalization in this context, beginning with the annotation of the learning resources. exploit a new level of knowledge thus allowing a better personalization.

A *learning object* can profitably be used if the learner has a given set of prerequisite competences; by using it, the learner will acquire a new set of competences. It is, therefore, appropriate to interpret learning objects as *actions*. The idea that we have proposed is to introduce at the level of the learning objects, some additional annotation for describing both their *pre-requisites* and their *effects* and to do this by exploiting standard representation languages, like LOM, and *ontologies*, for using terms with a clear and sharable meaning.

The proposed annotation expresses a set of *learning dependencies* between *ontological terms*, dependencies which can be expressed in a declarative formalism, and can be used by a reasoning system. So, given a set of learning objects, each annotated in this way, it is possible to use the standard planners, developed by the Artificial Intelligence community (for instance, the well-known Graphplan [3]), for building the reading sequences.

General-purpose planners search a sequence of interest in the whole space of possible solutions and allow the construction of learning objects on the basis of any learning goal. This is not always adequate in an educational application framework, where the set of learning goals of interest is fairly limited and the experience of the teachers in structuring the courses and the learning materials is important. This kind of constraint cannot be exploited by a general-purpose planner, being related to the strategy adopted by the teacher. The ideal solution is to express them as *rules* that specify an overall structure in terms of ontological terms (competences). We will call such rules *learning strategies*.

Given a set of learning strategies, it is possible to build a learning object by refining a general rule according to specific requirements and, in particular, by choosing those components that best fit the user. An emblematic example is preparing the material for a basic computer science course: the course may, in fact, have different contents depending on the kind of student to whom it will be offered (e.g. a Biology student, rather than a Communication Sciences student, rather than a Computer Science student). In particular, having a learning

strategy and a set of annotated learning objects, it is possible to apply *procedural planning* for assembling a reading path that is a sequence of learning resources that are annotated as required by the strategy. Opposite to general-purpose planners, procedural planning searches for a solution in the set of the possible executions of a learning strategy.

Since the strategy is based on competences, rather than on specific resources, the system might need to select between different courses, annotated with the same desired competence, which could equally be selected in building the actual learning path. This choice can be done based on external information, such as a user model, or it may be derive from a further interaction with the user. Decoupling the strategies from the learning objects results in a greater flexibility of the overall system, and simplifies the reuse of the learning objects. As well as learning objects, also learning strategies could be made public and shared across different systems. Results of these researches in the context of Massive are reported in [4], [5].

III. PERSONALIZATION OF THE INTERACTION WITH WEB SERVICES

In the last years distributed applications over the World-Wide Web have obtained wide popularity and uniform mechanisms have been developed for handling computing problems which involve a large number of heterogeneous components, that are physically distributed and that interoperate. These developments have begun to coalesce around the *web service* paradigm, where a service can be seen as a component available over the web. Each service has an interface that is accessible through standard protocols and that describes its *interaction capabilities*, and it can be *combined* and *integrated* with others to develop new applications over the web.

In this scenario, one of the needs that have inspired recent research [6] is the study of declarative descriptions of web services, aimed at allowing forms of automated interoperation that include, on the one hand, the automation of tasks like matchmaking and execution, on the other, the automation of service *selection and composition*, in a way that is customized w.r.t. the *user's goals and needs*, a task that can be considered as a form of *personalization* [5]. Indeed, selection and composition not always are to be performed on the sole basis of general properties of the services themselves and of their interactive behavior, such as their category or their functional compositionality, but they should also take into account the user's intentions (and purposes) which both motivate and constrain the search or the composition. As a quick example, consider a service that allows buying products, alternatively paying cash or by credit card: a user might have preferences on the form of payment to enact. In order to decide whether or not buying at this shop, it is necessary to single out the specific course of interaction that allows buying cash. This form of personalization can be obtained by applying *reasoning techniques* on a description of the service process. Such a description must have a well-defined meaning for all the parties involved. In this issue it is possible to distinguish three necessary components:

- web services capabilities must be represented according to some declarative formalism with a well-defined semantics, as also recently observed by van der Aalst [7];
- automated tools for reasoning about such a description and performing tasks of interest must be developed;
- in order to gain flexibility in fulfilling the user's request, reasoning tools should represent such requests as *abstract goals*.

The approach that we propose in [8] inherits from the experience of the research community that studies MAS and, in particular, logic-based formalizations of interaction aspects. Indeed, communication has intensively been studied in the context of formal theories of agency [9], [10] and a great deal of attention has been devoted to the definition of standard agent communication languages (ACL), e.g. FIPA [11] and KQML [12]. Recently, most of the efforts have been devoted to the definition of formal models of interaction among agents, that use *conversation protocols*. The interest for protocols is due to the fact that they improve the interoperability of the various components (often separately developed) and allow the verification of compliance to the desired standards.

The basic idea is to consider a service as a software agent and the problem of composing a set of web services as the problem of making a set of software agents interact and cooperate within a multiagent system (or MAS). This interpretation is, actually, quite natural, and shared in proposals that are closer to the agent research community and more properly set in the *Semantic Web* research field [13], [14]. Among the other proposals, let us recall the OWL-S [15] (formerly DAML-S) experience. In [13] the goal of providing greater expressiveness to service description in a way that can be *reasoned about* has been pursued by exploiting agent technologies based on the *action metaphor*. In particular, at the level of abstraction of the process model, a service is described as atomic, simple or composite in a way inspired by the agent language GOLOG and its extensions [16], [17], [18]; therefore reasoning techniques supported by the language are used to produce composite and customized services.

On this line, we have studied the possible benefits provided by a *declarative description* of their communicative behavior, in terms of personalization of the service selection and composition. Indeed we claim that a better personalization can be achieved by focussing on the abstraction of web services as entities, that communicate by following predefined, public and sharable interaction protocols and by allowing agents to reason about high level descriptions of the *interaction protocols* followed by web services. We model the interaction protocols provided by web services by a set of logic clauses, thus at high (not at network) level. The language we have used for describing conversation protocols, is based on an extension of the agent programming language DyLOG [1], [19].

Having a logic specification of the protocol, it is possible to reason about the effects of engaging specific conversations. In particular, we propose to use techniques for *reasoning about actions* for performing the automatic selection and composition of web services, in a way that is customized w.r.t.

the users's request. Communication can, in fact, be considered as the behavior resulting from the application of a special kind of actions: *speech acts*. The reasoning problem that this proposal faces can intuitively be described as looking for an answer to the question "Is it possible to make a deal with this service respecting the user's goals?". Given a logic-based representation of the service policies and a representation of the customer's needs as abstract goals, expressed by a logic formula, logic programming reasoning techniques are used for understanding if the constraints of the customer fit in with the policy of the service.

Our proposal can be considered as an approach based on the process ontology, a *white box* approach in which part of the behavior of the services is available for a rational inspection. A description of the communicative behavior by policies is definitely richer than the list of input and output, precondition and effect properties usually taken into account for the matchmaking. Actually, the approach can be considered as a *second step* in the matchmaking process, which narrows a set of already selected services and performs a *customization* of the interaction with them.

Moreover the idea of focussing on abstract descriptions of the communicative behavior is, actually, a novelty also with respect to other proposals that are set in the Semantic Web research field. The deductive process on communication policies can exploit more semantic information: in fact, it does not only take into account the pre- and post-conditions, as in OWL-S proposal, it also takes into account the complex communicative behavior of the service.

IV. WEB SERVICE INTEROPERABILITY

According to *Agent-Oriented Software Engineering* [20], a distinction is made between the global and the individual points of view of interaction. The *global* viewpoint is captured by an *abstract protocol*, expressed by formalisms like AUMML, automata or Petri Nets. The *local* viewpoint, instead, regards one of the agents and is captured by its policy; being part of the agent's implementation, the policy is usually written in some executable language. Having these two levels of description it is possible to decide whether an agent can take a role in an interaction. In fact, this problem can be read as the problem of proving if the agent's policy *conforms* to the abstract protocol specification.

A similar need of distinguishing a global and a local view of the interaction is recently emerging also in the area of *Service Oriented Architectures*. In this case a distinction is made between the *choreography* of a set of services, i.e. a global specification of the way in which they should interact, and the concept of *behavioral interface*, seen as the specification of the interaction from the point of view of the individual service. The recent W3C proposal of the choreography language WS-CDL [21], well-characterized and distinguished from languages for business process representation, like BPEL, is emblematic.

Taking this perspective, choreographies and agent communication protocols undoubtedly share a common purpose. In

fact, they both aim at expressing *global interaction protocols*, i.e. rules that define the global behavior of a system of cooperating parties. The respect of these rules guarantees the interoperability of the parties (i.e. the capability of *actually* producing an interaction), and that the interactions will satisfy given requirements.

In this context, one problem that becomes crucial is the development of formal methods for verifying if the behavior of a service respects a choreography. The applications would be various. A choreography could be used *at design time* (a priori) for verifying if the internal processes of a service enable it to participate appropriately in the interaction. At *run-time*, choreographies could be used to verify if everything is proceeding according to the agreements. A choreography could also be used unilaterally to detect exceptions (e.g. a message was expected but not received) or help a participant in sending messages in the right order and at the right time.

In the last years the agent community already started to face the two above mentioned kinds of conformance w.r.t. MASs [22] (e.g. see [23], [24], [25], [26] for *a priori* conformance, and [27] for *run-time* conformance). In the web service community the problem of conformance is arising only recently [28] because so far the focus has been posed on the specification of single services and on standards for their remote invocation. The new interest is emerging due to the growing need of making services, that are heterogeneous (in kind of platform or in language implementation), to interoperate. Therefore, there is a need of giving more abstract representations of the interactions that allow to perform reasoning in order to select and compose services disregarding the specific implementation details. Given our experience in the area of MASs, where the heterogeneity of the components is a fundamental characteristic, we agree with the observation by van der Aalst [7] that there is a need for a more declarative representation of the behaviour of services.

In this line, the work in [25], [26] about conformance of agent implementations w.r.t. protocol specifications has been adapted to the case of web services in [29]. In particular, in [29] we focus on testing *a priori conformance* and develop a framework based on the use of formal languages. In this framework a global interaction protocol (a choreography), is represented as a finite state automaton, whose alphabet is the set of messages exchanged among services. It specifies permitted conversations. Atomic services, that have to be composed according to the choreography, are described as finite state automata as well. Given such a representation we capture a concept of conformance that answers positively to all these questions: *is it possible to verify that a service, playing a role in a given global protocol, produces at least those conversations which guarantee interoperability with other conformant service? Will such a service always follow one of these conversations when interacting with the other parties in the context of the protocol? Will it always be able to conclude the legal conversations it is involved in?* Technically, the conformance test is based on the acceptance of both the service behavior and the global protocol by a special finite

state automaton. Briefly, at every point of a conversation, we expect that a conformant policy never utters speech acts that are not expected, according to the protocol, and we also expect it to be able to handle any message that can possibly be received, once again according to the protocol. However, the policy is not obliged to foresee (at every point of conversation) an outgoing message for every alternative included in the protocol (but it must foresee at least one of them).

The interesting characteristic of this test is that it guarantees the interoperability of services that are proved conformant *individually* and *independently* from one another. By interoperability we mean the capability of an agent of actually producing a conversation when interacting with another. The conformance test has been proved *decidable* when the languages used to represent all the possible conversations w.r.t. the policy and w.r.t. the protocol are *regular*.

The application of our approach is particularly easy in case a logic-based declarative language is used to implement the policies. In logic languages indeed policies are usually expressed by Prolog-like rules, which can be easily converted in a formal language representation. In [26] we show this by means of a concrete example where the language DyLOG [1], based on computational logic, is used for implementing the agents' policies. On the side of the protocol specification languages, currently there is a great interest in using informal, graphical languages (e.g. UML-based) for specifying protocols and in the translation of such languages in formal languages [30], [31]. By this translation it is, in fact, possible to prove properties that the original representation does not allow. In this context, in [25] we have shown an easy algorithm for translating AUML sequence diagrams to finite state automata thus enabling the verification of conformance. Of course, having a declarative representation of the choreographies as well, would help the proof of these properties in the context of the web services.

V. WEB SERVICE PROCESS LANGUAGES FOR BDI-STYLE AGENTS

The adoption of process languages for (semantic) WSs as a means for specifying the behaviour of agents and MASs is envisaged by a growing number of researchers working in the MAS community. For example, in [32] P. Buhler and J. M. Vidal discuss a technique for providing agent software with dynamically configured capabilities described with DAML-S, that can represent atomic or orchestrated WSs. In [33], the same authors advance the idea that BPEL can be used as a specification language for expressing the initial social order of a MAS, which can then intelligently adapt to changing environmental conditions. K. Sycara, M. Paolucci, J. Soudry, and N. Srinivasan suggest to extend the OWL-S Model Processing Language by adding to it a new statement called `exec` that takes a process model as input and executes it in order to support a broker agent in both discovery and mediation [34]. More recently, C. Walton [35] proposes to decompose agents into a stub that executes Agent Interaction Protocols and is responsible for communication between agents, and a body

which encapsulates the reasoning processes, and is encoded as a set of decision procedures. Both the stub and the body are implemented as WSs.

Our approach to the specification of the agents' behavioural knowledge by means of process languages for WSs is driven by our previous research on cooperative BDI agents, and thus differs from all the existing proposals discussed so far. In [36], we discuss the idea that BDI-style agents [37] can be extended with a built-in mechanism for retrieving plans from cooperative agents (thus becoming "CooBDI" agents), for example when no local plans suitable for achieving a certain desire are available. This feature turns out to be useful in many application fields such as: Personal Digital Assistants (PDAs), whose limited physical resources make dynamic loading and linking of code necessary; Self-repairing agents, namely agents situated in a dynamically changing software environment and able to identify the portions of their code that should be updated to ensure their correct functioning in the evolving environment; Digital butlers, i.e. agents that assist a human user in some task such as managing her/his agenda, filtering incoming e-mail, retrieving interesting information from the web; digital butlers adapt their behaviour to the user's needs by cooperating both with more experienced digital butlers, and with the assisted user.

We have implemented the ideas behind the CooBDI theory by means of WS technologies, obtaining what we named "CooWS" agents [38]. A CooWS agent adopts the following metaphor inspired by CooBDI.

- **Beliefs.** The variables local to the BPEL processes that constitute the body of the agent's plans can be considered as a metaphor for the agent's beliefs local to that plan, that are not explicitly represented.
- **Desires.** Desires may be either messages structured according to the FIPA ACL standard (<http://www.fipa.org/>), or unstructured Java strings.
- **Actions.** There are two kinds of actions: those that may appear inside the BPEL specification of the agent's plan body (that, in turn, may be delivery of ordinary events; achievement of new desires; and invocation of existing WSs by means of the BPEL `invoke` statement), and those that must be executed in case of success or failure of the achievement of a desire. Cooperative requests for plans are managed transparently to the agent, and do not belong to the set of actions that can be programmed by the user.
- **Plans.** Plans are defined by a unique plan identifier, a trigger (the desire for which the plan has been defined); a body (a BPEL process); and an access specifier (which may assume one of the three values *OnlyTrusted(Set)* – the plan may be shared only with the agents in the trusted agents set –, *Private* – the plan is private to the agent –, and *Public* – the plan may be shared with any agent).
- **Intentions.** An intention contains a stack of desires, a boolean attribute defining the intention's state (either active or suspended), and the success and failure actions. The set of plans currently available to the agent for man-

aging a given desire, is associated with the corresponding desire on the stack. The set of these "relevant" plans is generated by exploiting the cooperation mechanism (transparent to the user), thus retrieving both local and external plans useful for achieving the desire.

- **Events.** There are three kinds of events: cooperation, ordinary, and achieve events. A cooperation event is either a request, characterised by the desire for which the request has been issued, or a provide event, characterised by the set of plans that are relevant for the desire appearing in the corresponding request. Ordinary events consist of the reception of messages from other agents, while achieve events implement the plan nesting mechanism.

The implementation of the CooWS platform, downloadable from the web site <http://coows.altervista.org>, relies entirely on opensource tools that include ActiveBPEL, Apache Tomcat and Axis, jUDDI, UDDI4J, and MySQL. In order to validate the feasibility of our approach, we are currently working on the implementation of digital butlers that query Google (which can be accessed as a web service) to arrange travels and to organise meetings for their principals. The plans available to the digital butlers do not cover all the requests that may arrive from their principals, and the lack of plans for coping with an incoming request fires the collaborative exchange of plans.

In the future, we are willing to explore: 1) the ability to integrate an ontology into the system, so that matching between desires and triggers of plans can become more sophisticated than a simple comparison of strings; 2) the ability to dynamically update the set of trusted partners following reputation mechanisms such those described in [39].

VI. INTEGRATED ENVIRONMENTS FOR AGENT-ORIENTED SOFTWARE ENGINEERING

The correct and efficient engineering of heterogeneous, distributed, open, and dynamic applications is one of the technological challenges faced by Agent-Oriented Software Engineering (AOSE). The lack of mature methodologies, tools, and environments for agent-based system development limits the effectiveness and impact of AOSE [40].

MAS development requires engineering support for a diverse range of non-functional properties, such as understandability of the MAS at various conceptual levels, integrability of heterogeneous agent architectures, usability, re-usability, and testability. Creating one monolithic AOSE approach to support all these properties is not feasible. Rather, we expect different approaches to be suitable for modelling, verifying, or implementing various properties. By providing the MAS developer with an integrated set of languages and tools, and allowing for the choice of the most suitable language/tool to model, verify, or implement each property, we could make a step towards a modular approach to AOSE [41].

DCaseLP [42], [43] provides a prototyping environment where agents specified and implemented in a given set of languages can be seamlessly integrated. It also provides an AOSE methodology to guide the developer during the analysis

of the MAS requirements, its design, and the development of a working MAS prototype.

DCaseLP supports UML and AUML (<http://www.auml.org/>) for the specification of the general structure of the MAS, and Jess (<http://herzberg.ca.sandia.gov/jess/>), Java and tuProlog (<http://lia.deis.unibo.it/research/tuprolog/>) for the implementation of the agents.

As discussed in [42], DCaseLP adopts an existing multi-view, use-case driven and UML-based method in the phase of requirements analysis. Once the requirements of the application have been clearly identified, the developer can use UML and/or AUML to describe the interaction protocols followed by the agents, the general MAS architecture and the agent types and instances. Moreover, the developer can automatically translate the UML/AUML diagrams, describing the agents in the MAS, into Jess rule-based code. The Jess code obtained from the translation of AUML diagrams must be manually completed by the developer with the behavioural knowledge which was not explicitly provided at the specification level. The developer does not need to have a deep insight into rule-based languages in order to complete the Jess code, since he/she is guided by comments included in the automatically generated code.

The agents obtained by means of the manual completion of the Jess code are integrated into the JADE (Java Agent Development Framework, (<http://jade.tilab.com/>)) middleware. By integrating Jess into JADE, we were able to easily monitor and debug the execution of Jess agents thanks to the monitoring facilities that JADE provides. A recent extension of DCaseLP, discussed in [43], has been the integration of a Prolog implementation: tuProlog. The choice of tuProlog was due to two of its features:

- 1) it is implemented in Java, which makes its integration into JADE easier, and
- 2) it is very light, which ensures a certain level of efficiency to the prototype.

By extending DCaseLP with tuProlog we have obtained the possibility to execute agents, whose behavior is completely described by a Prolog-like theory, in the JADE platform. For this purpose, we have developed a library of predicates that allow agents specified in tuProlog to access the communication primitives provided by JADE: asynchronous send, asynchronous receive, and blocking receive (with and without timeout). Finally, a methodological integration of DyLOG into DCaseLP has been proposed in [44]. So far, the integration of DyLOG into DCaseLP is only “methodological” in the sense that it extends the set of languages supported by DCaseLP during the MAS engineering process and augments the verification capabilities of DCaseLP, without requiring any real integration of the DyLOG working interpreter into DCaseLP. Nevertheless, DyLOG can also be used to directly specify agents and execute them inside the DCaseLP environment, in order to exploit the distribution, concurrency, monitoring and debugging facilities that DCaseLP offers.

We have already tested – on a toy application – the ability of Jade, Jess and tuProlog agents to be integrated into the same MAS and to communicate with each other. Currently, we are developing a much more sophisticated application in the electronic auctions field, whose basic building block are described in [45].

VII. CONCLUSIONS

Mainstream research in Web Services (WS) is looking at two main aspects: first, formally describing interactions among services (possibly over long periods of time and having multiple real-world effects, including legally binding actions); second, finding and combining services (e.g., by extending the simple catalogue contained in UDDI repositories with semantically rich descriptions and using the latter for automated composition via planning and for formal verification). As observed in AgentLink III, 2004, and by M. N. Huhns, 2002, much work made in the intelligent agents area can be applied to these issues.

One of the problems that we have studied is the verification of the a priori conformance of the communication policy of an agent (or web service) w.r.t. a general interaction protocol specification, that rules a system of cooperating parties. The interesting characteristic of the test that we have proposed is that it guarantees the interoperability of services that are proved conformant *individually* and *independently* from one another. It emerged that the application of our approach is particularly easy in case a logic-based declarative language is used to implement the policies.

For what concerns the specification languages, the modelling languages commonly used in the “requirements specification” and “software design” phases proposed in the AOSE community, like AUML, are not declarative and, as such, they do not provide any automatic proof mechanism. In this context it is interesting to study translations between modelling languages and languages with a formalized semantics to enable the use of the automatic proof mechanisms associated to them. For instance in [26] we have proposed the use of finite state automata as formal representation of protocols which supports the proof of conformance and an algorithm for translating a subset of AUML into finite state automata has been proposed in [25]. However this is just a first step and more research should be devoted to the issue of the transformation from semi-formal to formal specification languages.

We have studied how the above approach applies to some concrete domain such as web services and e-learning. In particular web services are an example of a highly dynamic application domain where a challenging problem that we have studied is the development of formal methods for verifying if the behavior of a single service respects a choreography. More specifically the problem consists in deciding if the internal processes of a service enable it to participate appropriately in the interaction encoded by a choreography. Another related problem that it would be interesting to address is the use of choreographies at *run-time* to verify that everything is proceeding according to the agreements. In this context a

choreography could also be used unilaterally to detect exceptions (e.g. a message was expected but not received) or help a participant in sending messages in the right order and at the right time. Also in this case there are logic techniques developed in the agent community that can be adapted to tackle the problem in the web service domain [27].

REFERENCES

- [1] M. Baldoni, L. Giordano, A. Martelli, and V. Patti, "Programming Rational Agents in a Modal Action Logic," *Annals of Mathematics and Artificial Intelligence, Special issue on Logic-Based Agent Implementation*, vol. 41, no. 2–4, pp. 207–257, 2004.
- [2] M. Baldoni, C. Baroglio, A. Martelli, and V. Patti, "Reasoning about interaction protocols for customizing web service selection and composition," *The Journal of Logic and Algebraic Programming*, 2005, accepted for publication after major revision.
- [3] A. Blum and M. Furst, "Fast planning through planning graph analysis," *Artificial Intelligence*, vol. 90, pp. 281–300, 1997.
- [4] M. Baldoni, C. Baroglio, V. Patti, and L. Torasso, "Reasoning about learning object metadata for adapting SCORM courseware," in *Int. Workshop on Engineering the Adaptive Web, EAW'04: Methods and Technologies for Personalization and Adaptation in the Semantic Web, Part I*, L. Aroyo and C. Tasso, Eds., Eindhoven, The Netherlands, August 2004, pp. 4–13.
- [5] M. Baldoni, C. Baroglio, and N. Henze, "Personalization for the Semantic Web," in *Reasoning Web*, ser. LNCS Tutorial, vol. 3564. Springer, 2005, pp. 173–212.
- [6] A. Barros, M. Dumas, and P. Oaks, "A critical overview of the web services choreography description language(ws-cdl)," *Business Process Trends*, 2005, <http://www.bptrends.com>.
- [7] W. M. P. van der Aalst, M. Dumas, A. H. M. ter Hofstede, N. Russell, H. M. W. Verbeek, and P. Wohed, "Life after BPEL?" in *Proc. of WS-FM'05*, ser. LNCS, vol. 3670. Springer, 2005, pp. 35–50, invited speaker.
- [8] M. Baldoni, C. Baroglio, A. Martelli, and V. Patti, "Reasoning about interaction protocols for web service composition," M. Bravetti and G. Zavattaro, Eds. Elsevier Science Direct, 2004, pp. 21–36, vol. 105 of Electronic Notes in Theoretical Computer Science.
- [9] F. Dignum and M. Greaves, "Issues in agent communication," in *Issues in Agent Communication*, ser. LNCS, vol. 1916. Springer, 2000, pp. 1–16.
- [10] F. Dignum, Ed., *Advances in agent communication languages*, ser. LNAI, vol. 2922. Springer-Verlag, 2004.
- [11] FIPA, "Communicative act library specification," FIPA (Foundation for Intelligent Physical Agents), Tech. Rep., 2002.
- [12] T. Finin, Y. Labrou, and J. Mayfield, "KQML as an Agent Communication Language," in *Software Agents*, J. Bradshaw, Ed. MIT Press, 1995.
- [13] J. Bryson, D. Martin, S. McIlraith, and L. A. Stein, "Agent-based composite services in DAML-S: The behavior-oriented design of an intelligent semantic web," 2002. [Online]. Available: citeseer.nj.nec.com/bryson02agentbased.html
- [14] K. Sycara, "Brokering and matchmaking for coordination of agent societies: A survey," in *Coordination of Internet Agents*, A. O. et al., Ed. Springer, 2001.
- [15] OWL-S, "<http://www.daml.org/services/owl-s/1.1/>," 2004.
- [16] H. J. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R. B. Scherl, "GOLOG: A Logic Programming Language for Dynamic Domains," *J. of Logic Programming*, vol. 31, pp. 59–83, 1997.
- [17] G. D. Giacomo, Y. Lesperance, and H. Levesque, "Congolog, a concurrent programming language based on the situation calculus," *Artificial Intelligence*, vol. 121, pp. 109–169, 2000.
- [18] S. McIlraith and T. Son, "Adapting Golog for Programmin the Semantic Web," in *5th Int. Symp. on Logical Formalization of Commonsense Reasoning*, 2001, pp. 195–202.
- [19] M. Baldoni, C. Baroglio, A. Martelli, and V. Patti, "Reasoning about self and others: communicating agents in a modal action logic," in *Proc. of ICTCS'2003*, ser. LNCS, vol. 2841. Springer, 2003, pp. 228–241.
- [20] M. P. Huget and J. Koning, "Interaction Protocol Engineering," in *Communication in Multiagent Systems*, ser. LNAI, H. Huget, Ed., vol. 2650. Springer, 2003, pp. 179–193.
- [21] WS-CDL, "<http://www.w3.org/tr/2004/wd-ws-cdl-10-20041217/>," 2004.
- [22] F. Guerin and J. Pitt, "Verification and Compliance Testing," in *Communication in Multiagent Systems*, ser. LNAI, M. Huget, Ed., vol. 2650. Springer, 2003, pp. 98–112.
- [23] U. Endriss, N. Maudet, F. Sadri, and F. Toni, "Protocol conformance for logic-based agents," in *Proc. of the 18th International Joint Conference on Artificial Intelligence (IJCAI-2003)*, G. Gottlob and T. Walsh, Eds. Morgan Kaufmann Publishers, August 2003, pp. 679–684.
- [24] —, "Logic-based agent communication protocols," in *Advances in agent communication languages*, ser. LNAI, vol. 2922. Springer-Verlag, 2004, pp. 91–107, invited contribution.
- [25] M. Baldoni, C. Baroglio, A. Martelli, V. Patti, and C. Schifanella, "Verifying protocol conformance for logic-based communicating agents," in *Proc. of 5th Int. Workshop on Computational Logic in Multi-Agent Systems, CLIMA V*, ser. LNCS, no. 3487, 2005, pp. 192–212.
- [26] M. Baldoni, C. Baroglio, A. Martelli, and V. Patti, "Verification of protocol conformance and agent interoperability," in *Pre-proc. of Sixth International Workshop on Computational Logic in Multi-Agent Systems, CLIMA VI*, 2005, pp. 12–27.
- [27] M. Alberty, M. Gavanelli, E. Lamma, P. Mello, and P. Torroni, "Specification and verification of agent interactions using social integrity constraints," in *Proc. of the Workshop on Logic and Communication in Multi-Agent Systems, LCMAS 2003*, ser. ENTCS, W. van der Hoek, A. Lomuscio, E. de Vink, and M. Wooldridge, Eds., vol. 85(2). Eindhoven, the Netherlands: Elsevier, 2003.
- [28] N. Busi, R. Gorrieri, C. Guidi, R. Lucchi, and G. Zavattaro, "Choreography and Orchestration: a synergic approach for system design," in *Proc. the 3rd Int. Conf. on Service Oriented Computing*, 2005.
- [29] M. Baldoni, C. Baroglio, A. Martelli, V. Patti, and C. Schifanella, "Verifying the conformance of web services to global interaction protocols: a first step," in *Proc. of 2nd Int. Workshop on Web Services and Formal Methods, WS-FM 2005*, ser. LNCS, no. 3670, 2005, pp. 257–271.
- [30] L. Cabac and D. Moldt, "Formal semantics for auml agent interaction protocol diagrams," in *Proc. of AOSE 2004*, 2004, pp. 47–61.
- [31] R. Eshuis and R. Wieringa, "Tool support for verifying UML activity diagrams," *IEEE Trans. on Software Eng.*, vol. 7, no. 30, 2004.
- [32] P. Buhler and J. M. Vidal, "Semantic web services as agent behaviors," in *Proc. of Agentcities: Challenges in Open Agent Environments*, 2003.
- [33] —, "Adaptive workflow = web services + agents," in *Proc. of the Int'l Conference on Web Services*, 2003, pp. 131–137.
- [34] K. Sycara, M. Paolucci, J. Soudry, and N. Srinivasan, "Dynamic discovery and coordination of agent-based semantic web services agents," *IEEE Internet Computing*, vol. 8, no. 3, pp. 66–73, 2004.
- [35] C. Walton, "Uniting agents and web services," *AgentLink News*, vol. 18, pp. 26–28, 2005.
- [36] D. Ancona and V. Mascardi, "Coo-BDI: Extending the BDI model with cooperativity," in *Post-proc. of DALT'03*, 2004, pp. 109–134.
- [37] A. S. Rao and M. P. Georgeff, "Modeling rational agents within a BDI-architecture," in *Proc. of KR'91*, 1991, pp. 473–484.
- [38] L. Bozzo, V. Mascardi, D. Ancona, and P. Busetta, "CooWS: Adaptive BDI agents meet service-oriented computing," in *Proc. of the Int'l Conference on WWW/Internet*, 2005.
- [39] J. Sabater, "Trust and reputation for agent societies," *IIIA Monographs*, vol. 20, 2003.
- [40] AgentLink III, "Agent technology roadmap: Overview and consultation report," 2004.
- [41] T. Juan, M. Martelli, V. Mascardi, and L. Sterling, "Customizing AOSE methodologies by reusing AOSE features," in *Proc. of AAMAS'03*, 2003, pp. 113–120.
- [42] E. Astesiano, M. Martelli, V. Mascardi, and G. Reggio, "From Requirement Specification to Prototype Execution: a Combination of a Multiview Use-Case Driven Method and Agent-Oriented Techniques," in *Proc. of SEKE'03*, 2003, pp. 578–585.
- [43] I. Gungui and V. Mascardi, "Integrating tuProlog into DCaseLP to engineer heterogeneous agent systems," in *Proc. of CILC 2004*. Available at <http://www.disi.unige.it/person/MascardiV/Download/CILC04a.pdf.gz>.
- [44] M. Baldoni, C. Baroglio, I. Gungui, A. Martelli, M. Martelli, V. Mascardi, V. Patti, and C. Schifanella, "Reasoning about agents' interaction protocols inside DCaseLP," in *Proc. of DALT 2004*, 2004, pp. 112–131.
- [45] D. Roggero, F. Patrone, and V. Mascardi, "Designing and implementing electronic auctions in a multiagent system environment," 2005, DISI Technical Report.