

Information System for Modular Certification Testing

J. Hodík, J. Vokřínek, M. Jakob

Department of Cybernetics – Agent Technology Center, Faculty of Electrical Engineering,
Czech Technical University in Prague, Technická 2, Praha 6, 166 27, Czech Republic

Abstract

Resources planning system for domain of modular certification testing is presented in this paper. In our domain, there are several independent resources owners and services providers, who together are able to provide their customers with service of certification testing. We are presenting concept of distributed system for supporting negotiation about the certification testing session's life-cycle support, i.e. from the session arrangement, through its execution, to results distribution. Each of the session participants is represented by mutually independent agents supporting all the certification session processes except the act of filling and evaluation of tests.

Keywords:

Agent, Certification Testing, Information System, Contract

1 INTRODUCTION

The electronic contracting support accelerates wide variety of electronic business-to-business (B2B) applications. The parties' obligations are captured by electronic contracts, which explicitly specify (semi-) automated actions of contract parties that have to be performed in electronic or real world under defined conditions. Since the business actors (contract parties) are naturally distributed, autonomous and self-interested, there is great potential to model them as the agents and implement the B2B applications as the multi-agent systems. These advantages are also utilized by the information system for modular certification testing.

A key part of such contract-based system is a process through which the actual behaviour of individual agents is checked for conformance with respective governing contracts. However, such checking requires that relevant information on the behaviour of the parties is captured (both with respect to application processes they execute, and to managing their contractual relationships). The term monitoring has been traditionally used to refer both to the process through which contract-relevant events and states from a running contract-based system are gathered, and the reasoning applied to actually determine their compliance to contracts.

The concept of electronic contracts in agent communities has been introduced also e.g. by Sallé in [1] or by Streitberger in [2], who concentrates on E-Business on Demand. The contract identifies the contract parties and defines the set of conditions of the behaviour of the parties. These conditions usually refer to the state of the environment within which the parties operate (termed *contracting environment*) but can also refer to the state of other contracts. Discussion on structure and semantics of electronic contracts is given by Oren in [3].

The information system presented in this paper has been developed on top of contract-centric middleware presented by Confalonieri et al. in [4] and utilizes the contract execution observation process presented by Biba et al. in [5]. Processing of observed data was implemented by a means of a reasoner based on

augmented transition networks (see [6] for details). In the next chapter we present short overview of the contract, its life-cycle and the analyses. Chapter 3 presents observation and analysis pipeline implemented in the system and chapter 4 describes typical use case of the system and its implementation and usage details.

2 CONTRACT

In a general sense, the contract identifies contract parties, defines quality of service, and set of restrictions on the behaviour of the contract parties etc. Contract structure and semantics is described e.g. in [3]. In this work we concentrate to the last point, which defines what has to be done, may be done, and is prohibited to be done, and under which conditions, i.e. behaviours of the contract parties resulting in a desired outcome of their cooperation. The desired behaviours are defined by set of clauses, which defines what is expected and allowed to be and not to be done by specified contract parties and under specified conditions. The workflow derived from contract clauses (alternative paths are allowed there) constitutes an implementation of the desired behaviours. A deviation from contracted behaviour usually means a breach of contract clauses and possibly a breach of the contract. For minor breaches the document may contain precautions to solve them without terminating the contract.

The behaviour of contract parties is modelled as composed of individual actions, where each action corresponds to an elementary unit of activity. These actions are referenced from contract, which can prescribe under which circumstances a particular action has to or must not be carried out by a specified contract party. Information about actions performed by respective contract parties is therefore essential for determining adherence of contract parties to respective contracts.

2.1 Contract Life-Cycle

There are three basic steps in the life-cycle of the contract (e.g. in [7]). They are:

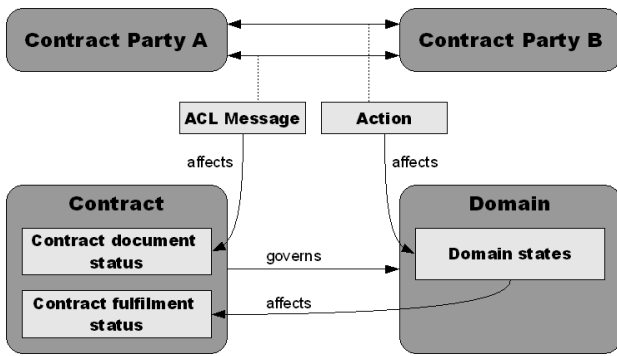


Figure 1: Observation in contract-based systems

- **Contract formation** during that the contract parties initiate the contract formation negotiation and negotiate the suggested contracts. This phase is ended by contract agreement.
- **Contract execution** during that the value-adding processes to those the contract is concluded are performed. When all duties are fulfilled or exception prohibiting the contract conclusion fulfillment appears, this phase is terminated.
- **Contract dissolution** during that the contract is finished. In case of unsuccessful finishing of the contract the “exception management” is handled in this phase.

Often these basic steps are extended by several steps for specific processes especially during the first phase (plan creation, team forming, and schedule negotiation) and second phase (concentration to modification of already concluded contract).

2.2 Contract Observation and Monitoring

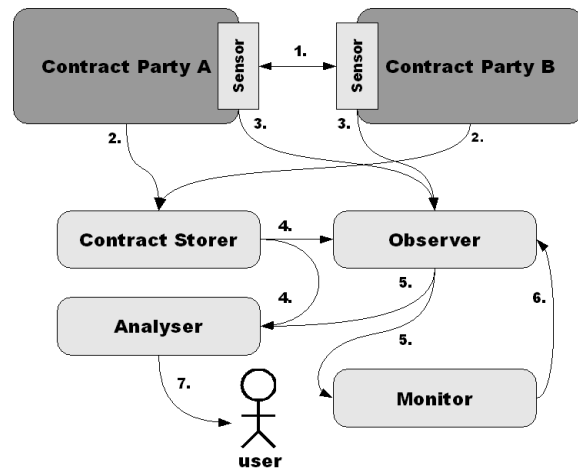
In order to be able to determine contract fulfilment, it is necessary to know two categories of information: (i) information about the behaviour of respective contract parties, and (ii) information about the content and status of the contract [5]. These two categories correspond to a two-level model of contract-based systems, distinguishing between the lower *domain level* and the *contract level* layered on top of it. On the top level the contract is negotiated between the agents using ACL (agent communication language) messages; on the lower level the agents influence the environment by action performances according to the contracted behaviours.

Information about contract-regulated actions is captured at the domain-level in order to determine fulfilment; information about contract-affecting communication between agents is captured at the contract level in order to track which contracts are currently active (and against which the behaviour of contract parties should be checked for fulfilment). For details of the processes and related type of information see Figure 1.

Monitoring of the contract execution consists of reasoning above both the data observed at the domain and action results. For the mapping to the contract clauses the observations are required to hold at least the information about (i) Subjects to them the observation relates (e.g. ID of agent invoking the action), and (ii) Objects to them the observation relates (e.g. parameters with them the action is invoked).

2.3 Specifics of Contract Analysis

The first dimension (along which analysis in contract-based systems can be categorized) is the level of analysis. We distinguish three levels:



1. Messaging and actions invocation between Contract Parties
2. Contract proposal and contract life-cycle changes storing
3. Messages and actions observations reports
4. Contract document download by observation and analysing tools
5. Providing of aggregated and pre-processed contract related data
6. Configuration of Monitor, observation reports, and clauses statuses
7. Front-end of contract observation, visualisation and analysis

Figure 2: Interactions between the observation pipe-line agents

- **Contract level** concerning contracts and their instantiations;
- **Domain level** concerning states and actions taking place in the domain; and
- **Fulfilment level** concerning fulfilment state of clauses and contracts; this information is derived from the previous two.

In addition, we can distinguish three time horizons of the analysis:

- **History** concerning the past events and their location in time;
- **Current status** concerning the contract based systems as it is; and
- **Trends** deduced from all accessible data.

In parallel to time horizons there another two perspectives on visualization:

- **State-concentrated** concerning world/things as they are, and
- **Action-concentrated** concerning world/things as they change.

Although dual (one can be in principle derived from the other), the two perspectives may require quite a different way of presenting the information to the user. Implementation-wise, the level of information provided by either of the perspectives may differ and both can use different information sources.

3 OBSERVATION AND ANALYSIS PIPELINE

Data are gathered, collected, and processed by set of specialized agents together forming the *Observation and Analysis Pipeline* in the contract environment [5]. The pipeline consists of (the interactions between the agents are depicted on the Figure 2):

- **Sensor**, an interface implemented by Contract Parties. The Sensor is responsible for observation of messages and activities, and reporting them to the Observer. The Sensor may be integrated to the message transport layer to enable automated monitoring, or invoked on the Contract Parties' will.

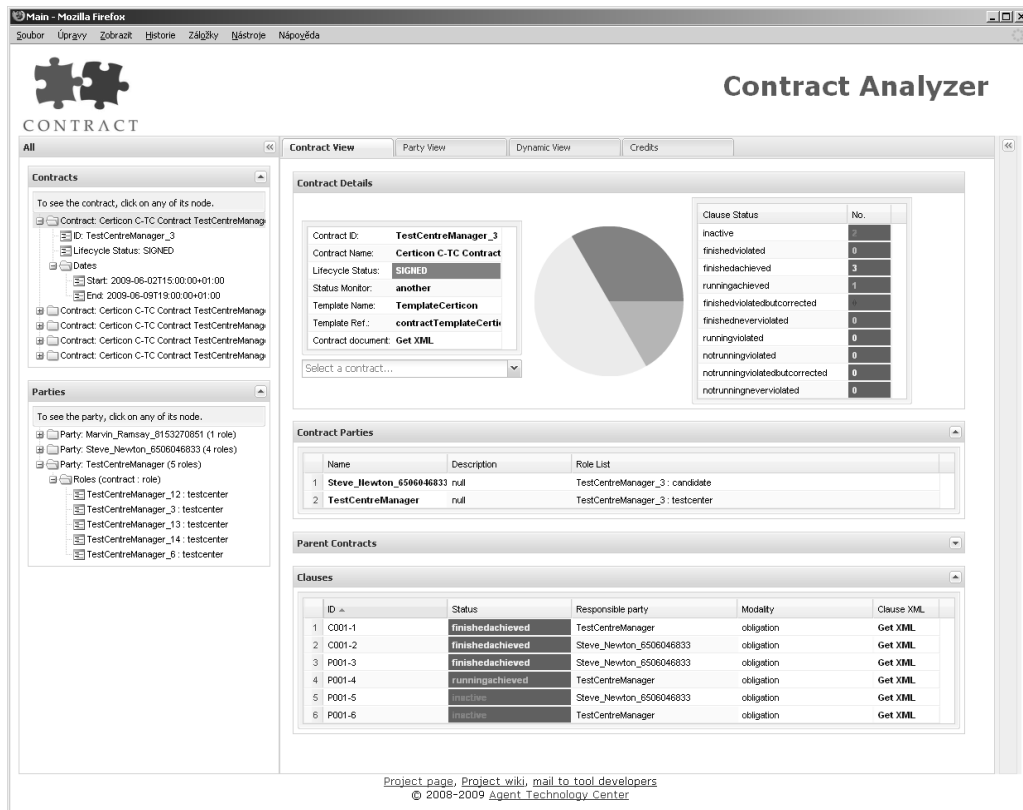


Figure 3: Screenshot of the Analyzer GUI – frontend of the pipeline

- **Observer**, the central component of the pipeline. The Observer collects data and provides them to the other agents of the pipeline.
- **Monitor**, an optional agent evaluating data stored by the Observer. For the known contract documents and received reports of the actions performed the Monitor is able to deduct contracts and their components fulfillment. The output of the Monitor are provided back to the Observer and stored there. The Monitor contains an ATN (Augmented Transition Networks) based reasoner [6].
- **Analyser**, the agent dedicated to evaluation and presentation contract related information from the other pipeline components (e.g. Observer) as well as other data sources.

There may be also other agents, which are not components of pipeline, participating on the pipeline task indirectly. At least one of them (which is crucial for a full and correct operation of the pipeline) is:

- **Contract Storer**, the agent storing of the contract agreements and information about their life-cycle state. The Contract Storer agent is residing on top of a database.

3.1 Runtime Analysis Process

In the observation pipeline, runtime analysis functionality is supported by the Contract Analyzer. Contract Analyzer interacts with other agents in the pipeline in order to collect the information required for the analysis and presentation to the human user. Technically, all such interactions are handled by the Contract Analyzer agent, a component that acts as an agent proxy between the Contract Analyzer and other agents in the pipeline. The other agents act roles of Contract Repository and Observer.

Contract Analyzer accesses the Contract Repository for the contract-level information about the contracts

deployed in the contracting environment, including their life-cycle status and other contract meta data (e.g. parent contracts). Observer (or possibly multiple Observers) is accessed to obtain domain-level information (actions performed, domain predicate values) and fulfilment-level information (fulfilment state of contracts and contract clauses, danger of clauses being breached etc.). Fulfilment-level information is provided only if the Observer is collaborating with Monitor. Screenshot of the user interface of the Analyzer is on Figure 3. Detailed information of contract monitoring in agent based system can be found in [8].

4 USE CASE: MODULAR CERTIFICATION TESTING

The concept of presented pipeline and the underlying middleware has been proved within several use-cases [9]. This section presents an implementation of the modular certification testing use-case, where the contract defines the testing session and related duties.

The business model used by information system for modular certification testing (see Figure 4) has been developed by Certicon a.s.¹ in cooperation with the Czech Society for Cybernetics and Informatics (CSKI)². Modular certification testing allows a large number of heterogeneous and independent businesses to flexibly collaborate on the provision of certification services. The model has been applied to computer literacy testing using the European Computer Driving Licence (ECDL)³ concept, first in the Czech Republic and later in Slovakia, and can be equally well applied to other certification programmes of the similar structure.

¹ Certicon a.s.: <http://www.certicon.cz/>

² CSKI: <http://www.cski.cz/>

³ ECDL: <http://www.ecdl.org/>

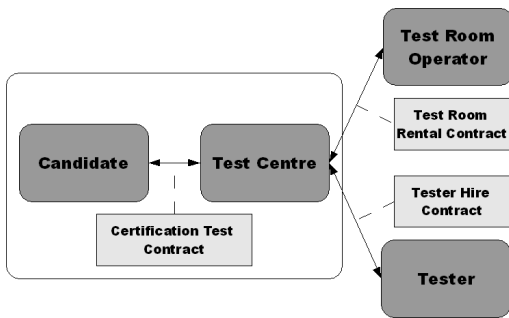


Figure 4: Business model of the use case

4.1 Use Case Contract Parties

The full business domain of the presented use case contains four types of contract parties:

- **Candidate**, the final consumer of the service (Modular Certification Testing), there are from tens to thousands of Candidates expected in the domain.
- **Test Centre**, the central unit and coordinator of the service administration, face to the customer. As the Test Centres do not collaborate, only one agent acting this role is expected in the community.
- **Test Room Operator**, owner of the facility necessary for providing of the service. Several Test Room Operators are expected in the domain.
- **Tester**, person competent (and necessary) to provide the service to the Candidate. Several Testers are expected in the domain.

In the presented scenario, negotiations between Test Centre and Test Room Operator, as well as Test Centre and Tester, are omitted; Test Centre has full information about the available resources.

4.2 Use Case Contracts

In line with the use case description, we focus only on the core part of the overall business domain encompassing the following contracts:

- **The Certification Test Contract** between the Test Centre and the Candidate (C-TC) gives the Candidate right to attend the test session organized by the Test Centre, and specifies terms and conditions for participation of the Candidate on a specific test session organized by the Test Centre and related organization interactions like payment and attendance dues, refunding options and certification granting.
- **The Test Room Rental Contract** between the Test Centre and the Test Room Operator (TC-TRO) gives the Test Centre right to use a test room for the purpose of certification testing and specifies terms and conditions under which the test room is operated and used like software equipment, number of seats, payment and refunding terms, etc.
- **The Tester Hire Contract** between the Test Centre and the Tester (TC-T) commits the Tester to provide supervision or marking service to the Test Centre and specifies terms and conditions of this temporary employment relationship like attendance or marking dues, payments, etc.

For a test session to take place the system requires an instantiation of all the above-mentioned contracts (usually tens of C-TC contracts and several TC-TRO as well as several TC-T contracts, at least one of each type),

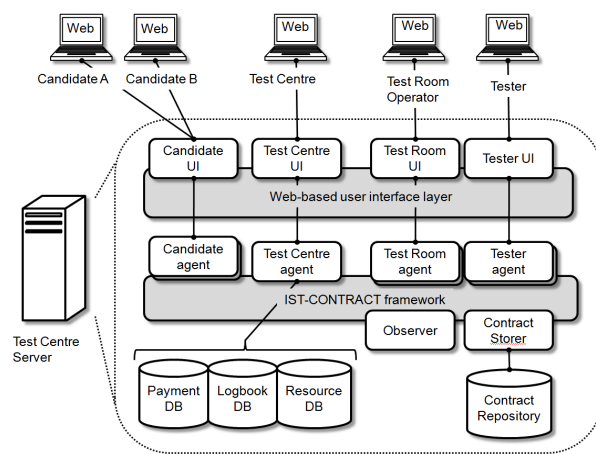


Figure 5: Information system architecture

which are mutually dependent and changes in one of them may affect the other contracts.

4.3 Scenario

At the top level, the overall scenario can be split into the *contract set-up phase* and the *contract execution phase*. During the contract set-up phase, all contract parties in the use case interact in order to agree on one or more test sessions. On the consumer side, this involves communication between the Candidates and the Test Centre about parameters of requested test sessions; on the supplier side, this involves communication between the Test Centre and its suppliers (Test Room Operators and Testers) in order to secure resource required for the intended test sessions. Once the details of the planned test sessions are agreed, the agreement is translated into a set of bilateral contracts which are then signed by the respective parties and enter into force.

After the contracts corresponding to a test session are signed, the contract execution phase commences. The execution phase itself can be divided into three sub-phases (i) pre-session, (ii) session, and (iii) post-session – depending on whether the respective activities take place before, during, or after the test session.

In addition to the above given chronological split of the global scenario, we can also differentiate based on which contract parties are involved in the respective transactions. Consumer-side scenarios only cover interactions between the Candidates and the Test Centre; supplier-side scenarios cover interactions between the Test Centre and Test Room Operators and/or Testers.

The described scenario models contract-related interactions between the Test Centre and Candidates. In the full real deployment, the interactions on the supplier-side between the Test Centre and Test Room Operators and Testers are also included.

4.4 Implementation

The information system for modular certification testing is composed of two main components: (i) server side responsible for data management, and (ii) client side running on user's computer and responsible for data presentation and interaction with the user. The system implementation is based on J2EE servlets, Google Web Toolkit⁴, and gwt-ext⁵ library for the implementation of the frontend.

⁴ GWT: <http://code.google.com/webtoolkit/>

⁵ GWT-EXT: <http://www.gwt-ext.com/>



Figure 6: Screenshots of the information system GUI

In the core of the server side part of the application there is an instance of the IST-CONTRACT⁶ framework in which respective contract party agents are run. Individual actors are represented by agents – set of Candidate Agents, Test Center Agent, set of Test Room Agents and set of Tester Agents (see Figure 5 for an overview of the information system architecture). Users related to individual contract parties access the server from other machines via web-based thin clients. For each type of party, there is a separate GUI module supporting the right type of interaction between the entity and the Test Center.

In addition, the observation pipeline components run on the server, in particular a Contract Storer and an Observer. The Observer is accompanied (not depicted on Figure 5) with the Monitor and Analyser. Those components enable advanced contract monitoring and evaluation. Details about the implementation of the observation pipeline are presented e.g. in [5].

4.5 Functionality and User Interfaces

Here we present the user's perspective of the information system. Its functionality is designed from the concept-centric perspective and it centers at instantiated contracts as the primary object of the analysis. By the information system the information is presented to the user through a number of different views described in following subsections. In this section we describe the interfaces for the Candidate and Test Center (see Screenshot of the information system user interface of on Figure 6).

Test Candidate Interfaces

The upper side of the Candidate interface contains Candidate details and high-level controls (the session request preparation button and the logout button). The rest of the page is composed of tabs for (i) contract preparation phase (Received Proposals tab), (ii) history reviewing (History tab), and (iii) contract execution phase (Contract execution tab).

Session Request Submission. Test session request screen allows the Candidate to submit a request for testing. The Candidate is allowed to state his/her restrictions and preferences on the test session (session type, date etc.). Most entries in the entry form are strongly typed and for some of them the Candidate can only select from a set of options provided reflecting the restrictions on the Test Centre side. When the Candidate presses the Send Request button, the request is sent the Test Centre for processing.

Test Session Proposal Selection and Confirmation. After the Candidate's test session proposal is processed by the Test Centre's matchmaking process, the resulting set of session proposals is presented back to the Candidate (normally there will be a period of up to several days between request submission and session proposal selection). The set can include zero, one or more proposals, out of which the Candidate can choose any number. However, as selecting and confirming a proposal equals signing a Certification Test contract with the Test Centre (with the follow-up obligation to pay the session fee), the Candidate in most circumstances selects and

⁶ IST-CONTRACT: <http://www.ist-contract.org/>

confirms at most one proposal. All the requests and corresponding proposals are presented in the Received Proposals tab. The Candidate can sign a proposal or cancel proposals.

Contract Execution. The Test Contract Execution tab allows the Candidate to inspect detailed information about all its contracts. The Candidate can invoke actions (pay fee, register to the session, etc.) according to the contract life-cycle.

Contracts History Monitoring. This interface allows the Candidate to monitor all contracts and requests that exist in the system and in which the particular Candidate is/was involved.

Additional Candidate Interfaces. In addition to the above, there are a number of additional Candidate interfaces that are part of the information system and are associated with performing the transactions between the Candidate and the Test Centre: Test session fee payment, Registration for a test session, Test submission, and Test result browsing.

Test Centre Interfaces

The upper part of the Test Center interface contains Test Centre information and a logout button. The rest of the page is composed of tabs for (i) requests, proposals and available sessions overview (Active Proposals tab), (ii) history reviewing (History tab), and (iii) contract execution phase (Session execution tab).

Active Proposals Overview. This screen presents the test centre manager with an overview of active proposals generated according to the received test session requests. The proposals are generated by a matchmaking algorithm from all available sessions.

Session Execution Screen. Test Session Execution tab allows the Test Centre manager to inspect detailed information about a particular test session. The Test Centre can invoke actions regarding the sessions (start the session, close the session, etc.) or corresponding contracts (provide information, evaluate test, etc.) according to the contract life-cycle.

Contract History Monitoring. Similarly to Candidates, the Test Centre can monitor all the contracts, requests and sessions status that exists in the system and the Test Centre is/was involved into.

5 CONCLUSION

In this paper we have introduced the information system for modular certification testing implemented as multi-agent system utilizing electronic contracts. The observation and analysis pipeline has been utilized to enhance the automated contract monitoring and analysis. The system enables fast and efficient monitoring of obligations and their fulfilment that reduces the administrative overhead of the process and speed-ups the testing session negotiation.

The information system is contract oriented i.e. the contract negotiation, execution support, and fulfilling monitoring are in focus. When contract is signed, actions of contract parties (candidate and test centre) are monitored for possible perturbations.

The prototype of the information system is presented as well. It employs several mutually independent tools developed within the IST project CONTRACT. The system is supported by the following tools: Contract Observer (collecting of observed data), Monitor (pre-processing of observed data), Analyser (visual analysis of contracts and observations), and Contract Repository (contract documents data-store). For integration of the

tools service oriented architecture is employed, and for user interfaces the light clients are implemented.

6 ACKNOWLEDGEMENTS

The research described is part-funded by the EC FP6 projects CONTRACT (contract No. 034418), I*PROMS Network of Excellence, and also by the Ministry of Education, Youth and Sports of the Czech Republic grant No. MSM 6840770038.

The authors' organizations and research sponsors are authorized to reproduce and distribute reprints and on-line copies for their purposes notwithstanding any copyright annotation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of other parties.

7 REFERENCES

- [1] Sallé, M., 2002, Electronic Contract Framework for Contractual Agents, 15th Conference of the Canadian Society for Computational Studies of Intelligence on Advances in Artificial Intelligence – AI '02: 349-353.
- [2] Streitberger, W., 2005, Framework for the Negotiation of Electronic Contracts in E-Business on Demand, 7th IEEE International Conference on E-Commerce Technology – CEC '05: 370-373.
- [3] Oren, N., Panagiotidi, S., Vazquez-Salceda, J., Modgil, S., Luck, M., Miles, S., 2008, Towards a Formalisation of Electronic Contracting Environments, AAAI Workshop on Coordination, Organization, Institutions and Norms in Agent Systems – COIN '08: 61-68.
- [4] Confalonieri, R., Álvarez-Napagao, S., Panagiotidi, S., Vázquez-Salceda, J., Willmott, S., 2008, A Middleware Architecture for Building Contract-Aware Agent-Based Services, SOCASE International Workshop on Service-Oriented Computing: Agents, Semantics, and Engineering – SOCASE '08, 1-14.
- [5] Bíba, J., Hodík, J., Jakob, J., 2009, Contract Observation in Web Services Environments, SOCASE International Workshop on Service-Oriented Computing: Agents, Semantics, and Engineering – SOCASE '09, [in print].
- [6] Faci, N., Modgil, S., Oren, N., Meneguzzi, F., Miles, S., Luck, M., 2008, Towards a Monitoring Framework for Agent-Based Contract Systems, 12th International Workshop on Cooperative Information Agents – CIA '08: 292-305.
- [7] Hodík, J., Vokřínek, J., Bečvář, P., 2009, Support for Virtual Organisation Creation – Partners' Profiles and Competency Management, IJAOS, 2/3: 230-251
- [8] Hodík, J., Vokřínek, J., Jakob, M., 2009, Contract Monitoring in Agent-based Systems: Case Study, 4th International Conference on Industrial Applications of Holonic and Multi-Agent Systems – HOLOMAS '09, [in print].
- [9] Jakob, M., Pěchouček, M., Miles, S., Luck, M., Chábera, J., Dehn, M., Holt, C., Kollingbaum, M., Oren, N., Storms, P., Vazquez, J., 2008, Case Studies for Contract-based Systems, 7th International Conference on Autonomous Agents and Multiagent Systems – AAMAS '08 – Industry and Applications Track, 55-62.