Contract Monitoring in Agent-Based Systems: Case Study

Jiří Hodík, Jiří Vokřínek, and Michal Jakob

Agent Technology Center
Gerstner Laboratory – Agent Technology Center
Department of Cybernetics
Faculty of Electrical Engineering
Czech Technical University in Prague
Technická 2, 166 27 Praha 6, Czech Republic
{hodik,vokrinek,jakob}@agents.felk.cvut.cz

Abstract. Monitoring of fulfilment of obligations defined by electronic contracts in distributed domains is presented in this paper. A two-level model of contract-based systems and the types of observations needed for contract monitoring are introduced. The observations (inter-agent communication and agents' actions) are collected and processed by the contract observation and analysis pipeline. The presented approach has been utilized in a multi-agent system for electronic contracting in a modular certification testing domain.

Keywords: Contract, Monitoring, Agent, Observation, Case Study.

1 Introduction

The electronic contracting support accelerates a 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 the electronic or real world under defined conditions. Since the business actors (contract parties) are naturally distributed, autonomous and self-interested, there is a great potential to model them as agents and implement the B2B applications as multi-agent systems.

The concept of electronic contracts in multi-agent systems has been introduced e.g. in [1]. The framework for on-demand service contracts with a high frequency of occurrences is presented in [2]. This work builds on the framework for the electronic contracting presented in [3]. Details of the observation process (in the web services domain) was presented by us in [4].

A key part of a contract-based system is the process through which the actual behaviour of individual agents is checked for conformance with respective governing contracts. Such checking requires that the relevant information on the behaviour of the parties, both with respect to the application processes they execute, and to managing their contractual relationships, is captured. The term *monitoring* has been traditionally used to refer both to the process through

V. Mařík, T. Strasser, and A. Zoitl (Eds.): HoloMAS 2009, LNAI 5696, pp. 295–304, 2009.

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.

2 Contract

In a general sense, the contract identifies the contract parties and defines the set of restrictions on the behaviour of the contract parties. Contract structure and semantics is described e.g. in [5]. In this work we focus on the desired behaviour defined by a set of clauses, which specify what is expected or allowed to be or not to be performed by the respective contract parties under specified conditions. The workflow derived from contract clauses (alternative paths are allowed) constitutes a possible implementation of the desired behaviour. A deviation from the contracted behaviour usually means a breach of contract clauses and possibly a breach of the contract. The contract may define procedures to solve minor breaches without terminating the contract.

The model of behaviour of contract parties is composed of individual actions, where each action corresponds to an elementary unit of activity. These actions are referenced from the 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.

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 [4]. 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 performing actions according to the contracted behaviour.

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 the related type of information, see Figure 1.

Monitoring of the contract execution consists in reasoning above both the data observed at the domain level and action results, which are declaratively described in the contract document by the clauses. For the mapping to the contract clauses the observations are required to hold at least the information about (i) *subjects* to which the observation relates (e.g., ID of agent invoking the action), and (ii) *objects* to which the observation relates (e.g. parameters with which the action is invoked).

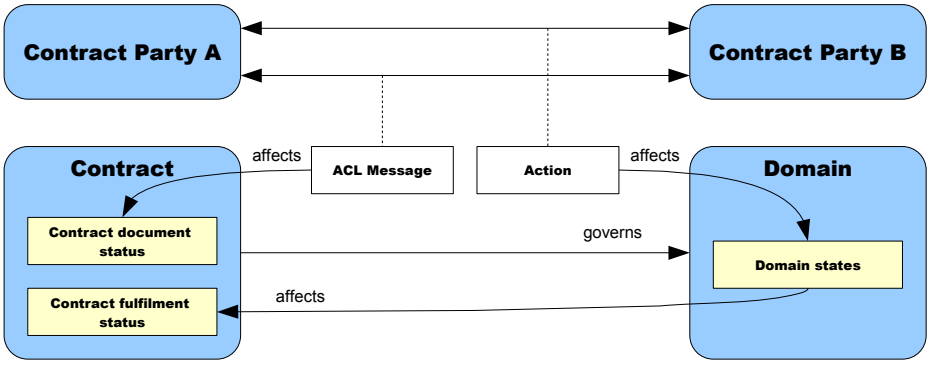

Fig. 1. Observation in contract-based systems

3 Observation and Analysis Pipeline

In the contract environment, data are gathered, collected, and processed by a set of specialized agents forming the *Observation and Analysis Pipeline*. The pipeline consists of (the interactions between the agents are depicted in the Figure 2.):

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

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 the data stored by the Observer. For the known contract documents and received reports of the actions performed, the Monitor is able to determine the fulfilment of contracts and their clauses. The outputs of the Monitor are provided back to the Observer and stored there. The Monitor contains an ATN (Augmented Transition Networks) based reasoner [3].

Analyser, an agent dedicated to the evaluation and presentation of the contract-related information obtained from other pipeline components (e.g., Observer) as well as other data sources.

There may also be other agents, which are not components of the 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 the contract agreements and information about their life-cycle state. The Contract Storer agent is residing on top of a database.

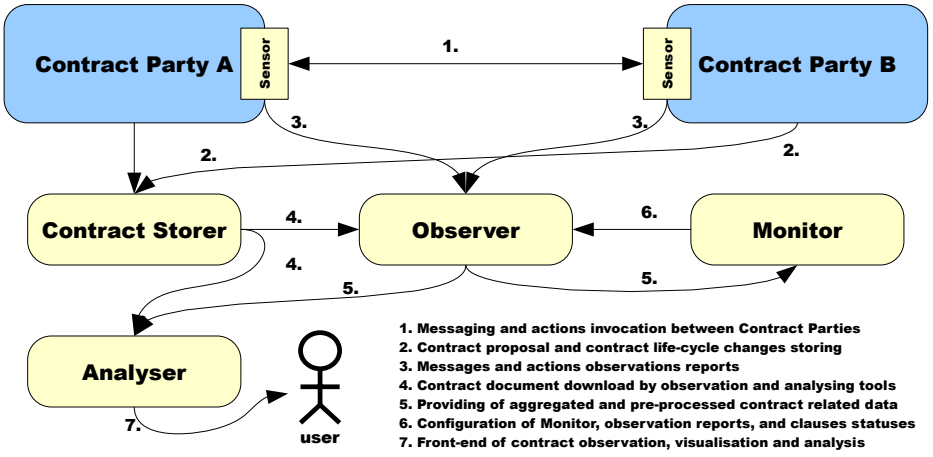

Fig. 2. Interactions between the observation pipe-line agents

4 Use Case

The concept of the presented pipeline and the underlying middleware has been validate on several use-cases [6]. 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 of modular certification testing 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.

In line with the use case description, we focus only on the core part of the overall business domain encompassing four types of contract parties (Candidates, Test Centres, Test Room Operators and Testers) and 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.

[1] `http:/www.certicon.cz/`
[2] `http://www.cski.cz/`
[3] The European Computer Driving Licence is the registered trade mark of The European Computer Driving Licence Foundation Limited in Ireland and other countries. `http://www.ecdl.org/`

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 (e.g., 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 the 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) which are mutually dependent and changes in one of them may affect the other contracts.

4.1 Usage Scenario

On 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 the 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 Center about parameters of requested test sessions; on the supplier side, this involves communication between the Test Center 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.

In the following, we focus on the contract between the Candidate and the Test Center (the contract parties) and we follow its evolution through the whole contract life-cycle. An example of the contract XML is given in Figure 3.

Contract Creation. The activities in this phase consist of messages exchanged between the Candidate and the Test Center. The flow of activities in this phase is:

1. Candidates send ACL messages to the Test Center with session requests containing a contract template and suggested values for contract variables (e.g., the type of the testing session).
2. The Test Center performs matchmaking between the assembled requests from various Candidates and available suppliers' resources. The best match, i.e. the configuration of suppliers and parameters of the test session meeting restrictions and optimizing preferences of all involved parties, is then translated into a set of bilateral Certification Test contract proposals and propagated back to each Candidate via ACL messages (zero, one or multiple proposals can be send to each Candidate). The contracts are also forwarded to the Contract Storer.

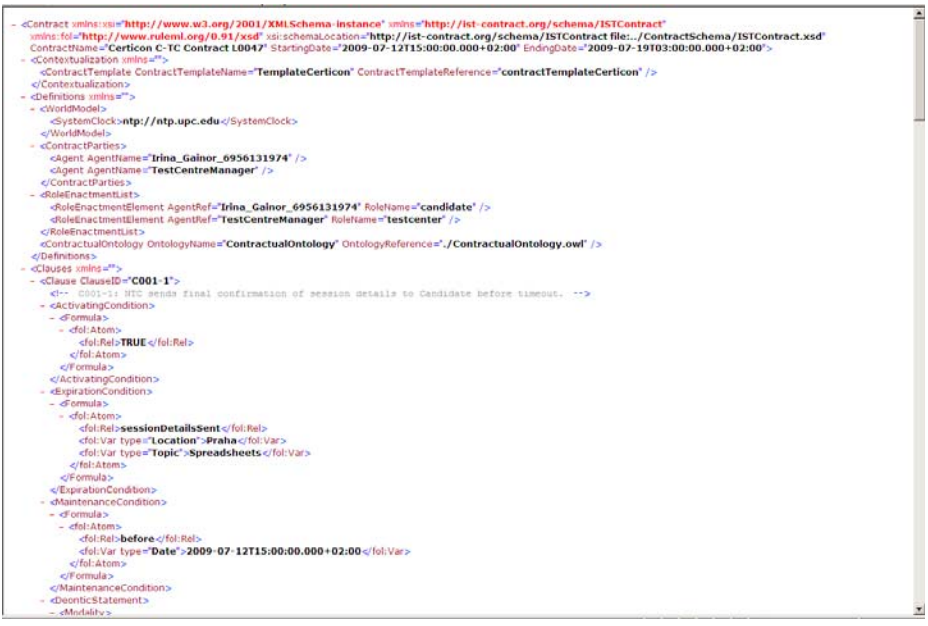

Fig. 3. Example of the contract XML

3. Each Candidate evaluates obtained proposals and responds to the Test Center whether the obtained contract proposals are acceptable or not. The contract can be either signed (the contract is concluded and its life-cycle continues by its execution) or rejected (the contract is not accepted and the process terminates). The state of each contract is updated in the Contract Storer.

Contract Execution. The activities in this phase consist of actions invoked by the contract parties. The activities and relations among them are defined in the contract. The flow of states and activities in this phase is:

1. Before session: this state is the initial state of the contract. To move to the next state, following two mutually independent actions must be performed:
   (a) Candidate: pay session fee
   (b) Test Center: provide session information
2. Session, performing test: this state contains activities related to the process of testing. The activities have to be invoked in the following sequence:
   (a) Candidate: register for the session
   (b) Test Center: Provide test to be filled out
   (c) Candidate: Submit test
3. Test evaluation: this state contains activities related to the process of test evaluation and providing the results to Candidate.
   (a) Test Center: notify Candidate about test result.

The actions performed are monitored by the observer (as described in section 3) and reflected in the Contract Storer by update of the contract state. If the flow is successfully finished, the contract life-cycle continues by dissolution phase during which the contract is dissolved. Violations that may occur during the contract execution are:

1. Before session
   (a) Candidate does not pay
   (b) Test Center does not provide session information
2. Session, performing test
   (a) Candidate does not register for the session
   (b) Test Center does not provide a test to be filled out
   (c) Candidate does not submit the test
3. Test evaluation:
   (a) Test Center does not notify the Candidate about the test result

Any of the contract violations terminates the execution phase and start of the dissolution phase with dissolution of the contract. Every action in the contract is safe-guarded by a dead-line whose expiration automatically violates the contract.

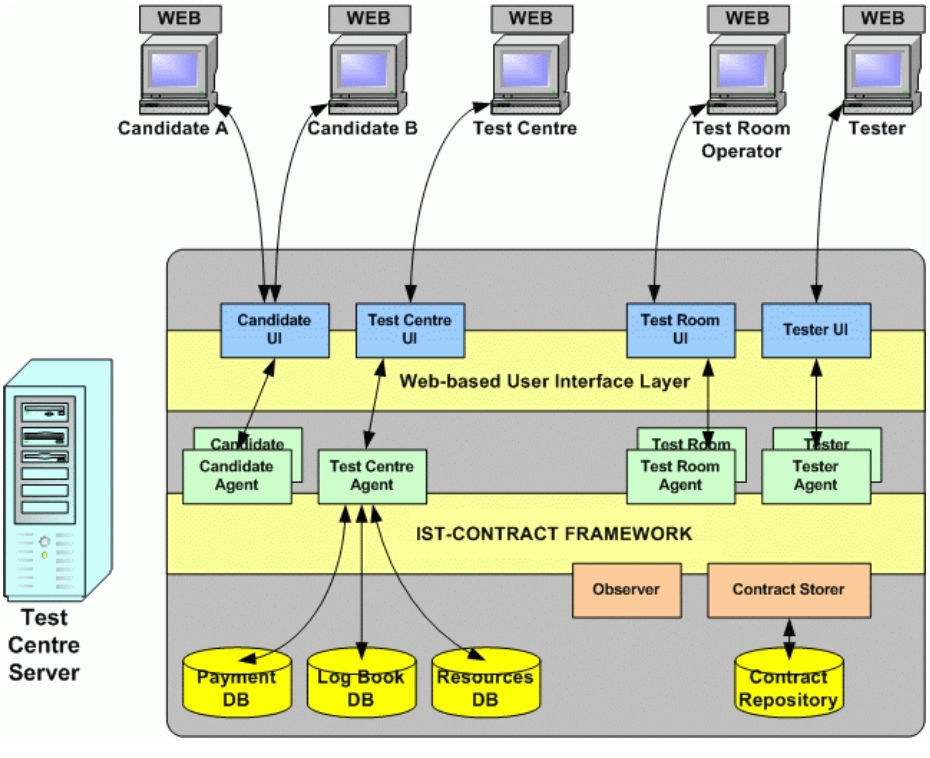

Fig. 4. Prototype architecture

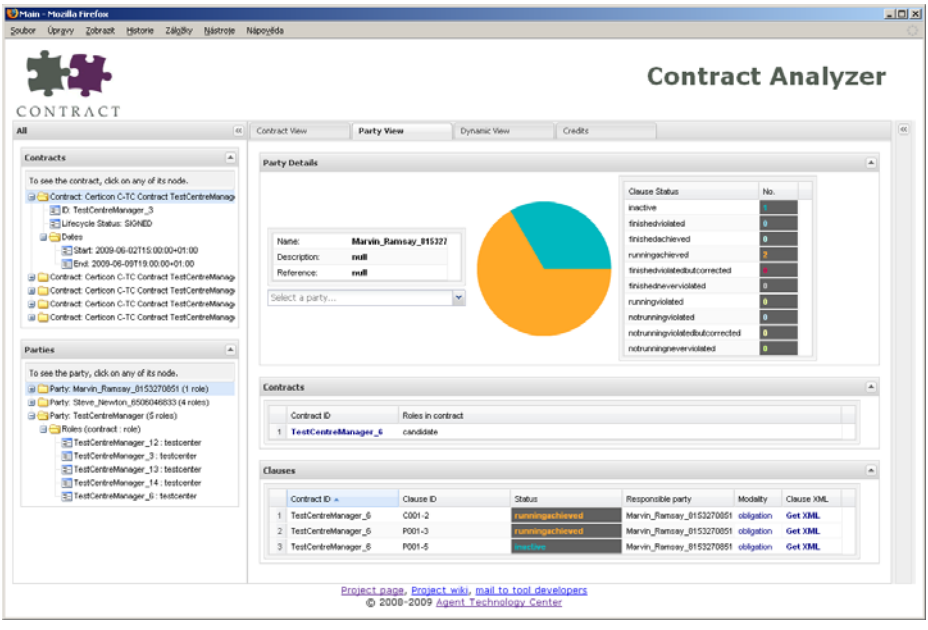

Fig. 5. Analyser GUI providing the frontend to the Observation pipeline

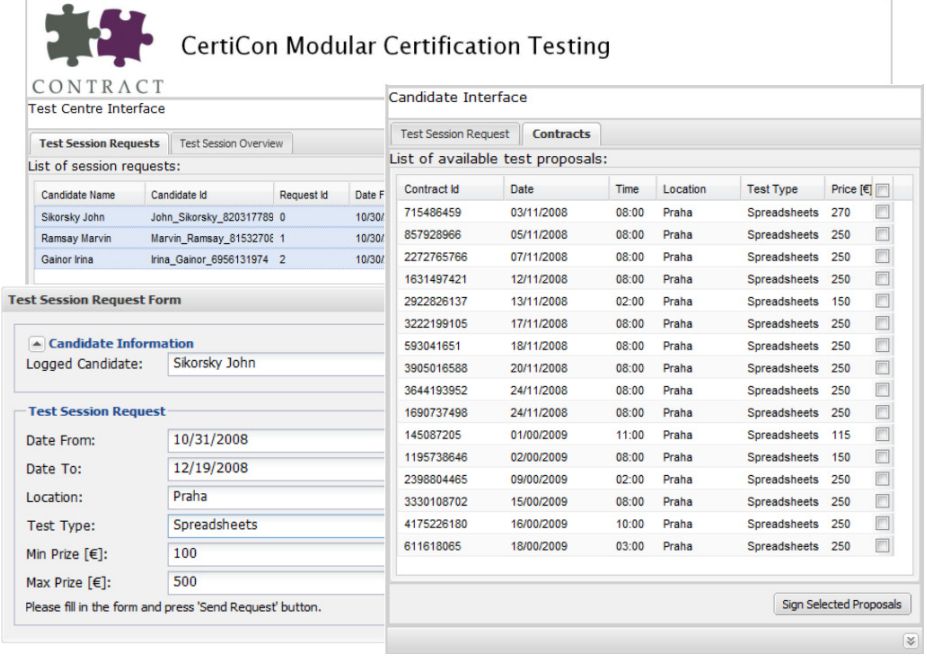

Fig. 6. GUI of the running prototype

Contract Dissolution. In the described case, there are no activities needed for contract dissolution. In the end of the contract life-cycle, the contract status is updated in the Contract Storer and its observation is finished.

4.2 Implementation

At the core of the application is an instance of the IST-CONTRACT[4] framework in which respective contract party agents, one for each contract party in the scenario, are run. Individual actors are represented by agents – a set of Candidate Agents, Test Center Agent, a set of Test Room Agents and a set of Tester Agents (see Figure 4 for an overview of the prototype architecture). Users corresponding to the 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 (see Figure 6).

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 4) by the Monitor and Analyser. Those components enable advanced contract monitoring and evaluation. Screenshot of the user interface of the Analyser is on Figure 5. Details about the implementation are presented e.g. in [4].

5 Conclusion

In this paper, we have introduced the implementation of electronic contract-based multi-agent system for the support of modular certification testing. 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 speeds up the negotiation of testing sessions.

Acknowledgement

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.

[4] `http:/www.ist-contract.org/`

References

1. Sallé, M.: Electronic contract framework for contractual agents. In: AI 2002: Proceedings of the 15th Conference of the Canadian Society for Computational Studies of Intelligence on Advances in Artificial Intelligence, London, UK, pp. 349–353. Springer, Heidelberg (2002)
2. Streitberger, W.: Framework for the negotiation of electronic contracts in e-business on demand. In: CEC 2005: Proceedings of the Seventh IEEE International Conference on E-Commerce Technology, Washington, DC, USA, pp. 370–373. IEEE Computer Society Press, Los Alamitos (2005)
3. Faci, N., Modgil, S., Oren, N., Meneguzzi, F., Miles, S., Luck, M.: Towards a monitoring framework for agent-based contract systems. In: Klusch, M., Pěchouček, M., Polleres, A. (eds.) CIA 2008. LNCS, vol. 5180, pp. 292–305. Springer, Heidelberg (2008)
4. Bíba, J., Hodík, J., Jakob, M.: Contract observation in web services environments. In: Proceedings of Internatonal Workshop on Service-Oriented Computing: Agents, Semantics, and Engineering (SOCASE 2009), London, UK. Springer, Heidelberg (to appear, 2009)
5. Oren, N., Panagiotidi, S., Vazquez-Salceda, J., Modgil, S., Luck, M., Miles, S.: Towards a formalisation of electronic contracting environments. In: Hübner, J.F., Matson, E., Boissier, O., Dignum, V. (eds.) COIN 2008. LNCS (LNAI), vol. 5428, pp. 156–171. Springer, Heidelberg (2008)
6. Jakob, M., Miles, S., Luck, M., Oren, N., Kollingbaum, M., Holt, C., Vazquez, J., Storms, P., Dehn, M.: Case Studies for Contract-based Systems. In: Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems (2008)