Contract Fulfilments Runtime Observations Analysis

Jiří HODÍK

Dept. of Cybernetics, Czech Technical University, Technická 2, 166 27 Praha, Czech Republic

hodik@agents.felk.cvut.cz

Abstract. *Contract Analyzer, which is the frontend component of toolkit for execution of observation, analysis and monitoring of electronic contract, is presented in this paper. The contracts, which are in scope, are described by a contract document in form of clauses describing duties of involved contract parties. Runtime analysis tools allow administrators and users to inspect the state of a running contract-based system, in particular contracts deployed in the system, their life-cycle status and fulfilment state.*

Keywords

Electronics contract, observation pipeline, analysis

1. Introduction

If the contract document (the document describing duties of all involved contract parties and confirmed by all of them) is created with respect to automatic observation and analysis, such processes may be automated and run without necessary interaction with human user. A key part of runtime observation is a process through which the actual behavior of individual contract parties is checked for conformance with respective governing contracts. Such checking, however, requires that relevant information on the behavior of the parties, both with respect to application processes they execute, and to managing their contractual relationships, is captured.

The concept of electronic contracts (in multi-agent system) has been introduced e.g. in [7]. The framework for on-demand services contracts with high frequency of occurrences is presented in [8]. This work also builds on the framework for the electronic contract management presented in [4]. Details of the observation process (in web-service domain) was presented in [1].

2. Contract

In a general sense, the contract identifies contract parties, defines quality of service, set of is a set of restrictions on the behavior of the contract parties etc. Contract structure and semantics is described e.g. in [6]. In this work we con-

centrate to the last point, which defines what has to be done, may be done, or is prohibited to be done, and under which conditions; i.e. behaviors of the contract parties resulting in a desired outcome of their cooperation. The desired behaviors 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 behaviors. A deviation from contracted behavior 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.

2.1. Contract Life Cycle and its Observation

There are three basic steps in the life-cycle of the contract. They are:

- *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 the value-adding due those the contract is concluded are performed. When all duties are fulfilled or exception prohibiting the contract conclusion fulfilment appears, this phase is terminated.

- *Contract dissolution* finishes the contract. 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, schedule negotiation) and second phase (concentration to modification of already concluded contract).

2.2. Contract Analysis

The first dimension (along which analysis in contract-based systems can be categorized) is the level of analy-
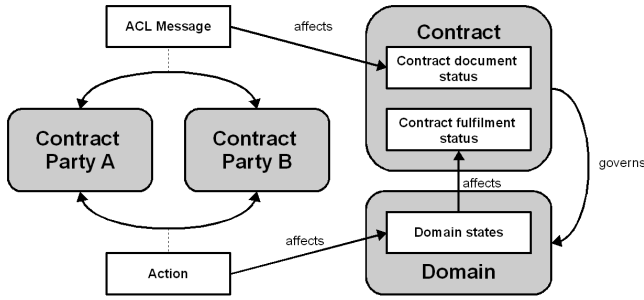
Fig.1. Relations between domains and observations.

sis. We distinguish three levels: (i) *contract level* concerning contracts and their instantiations; (ii) *domain level* concerning states and actions taking place in the domain; and (iii) *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: (i) *history* concerning the past events and their location in time; (ii) *current status* concerning the contract based systems as it is; and (iii) *trends* deducted from all accessible data. In parallel to time horizons there another two perspectives on visualization: (i) *state-concentrated* concerning world/things as they are, and (ii) *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.

The behavior of contract parties is modeled 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, and is therefore the key information observed at the domain level.

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

---

[1] Agent Communication Language

3. Observation Pipeline

The observation pipeline supports processes of collecting data, their fusion and processing of the resulting information (including the respective contract-compliance inference), and the final exploitation of such information by visualization tools. The pipeline consists of a set of separated components specialized to specifics task of the observation and analysis process. These constitute a distributed system implemented by multi-agent technology – each pipeline component is represented by an agent [1]. The concept of presented pipeline and the underlying middle-ware has been proved within several use-cases (for overview of these use-cases see [5]). Components of the pipeline are as follows:

- *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.

- *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 [4].

- *Contract Analyzer*, 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 an database.

4. 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
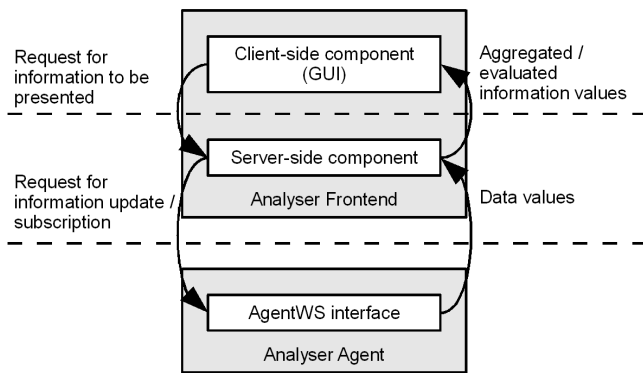
Fig.2. Internal architecture of Contract Analyzer.

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 [2].

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

4.1. Internal Architecture and Implementation of Contract Analyzer

Following the depiction of how Contract Analyzer fits within the rest of the contracting environment, we now describe the internal architecture the Contract Analyzer. The Contract Analyzer subsystem is composed of two components (see Figure 2 for a graphical scheme):

- *Analyzer Agent* provides a proxy towards the rest of the contracting environment and it is responsible for aggregating data from other agents and pre-processing for the presentation to the user.

- *Analyzer Frontend* manages interactions with the user. It consists of a server-side and client-side part. The server-side part access the Analyzer Agent via the agent platform interface.

CONTRACT framework Agent Shell (see [1, 2, 3]) is used as the basis for the implementation of the Analyzer Agent. J2EE servlets utilizing Google Web Toolkit[2] and

²http://code.google.com/webtoolkit/

gwt-ext[3] library are used for the implementation of the Analyzer Frontend.

4.2. Functionality and User Interface

Here we present the user's perspective of the Contract Analyzer. Its functionality is designed from the concept-centric perspective and it centers at instantiated contracts as the primary object of the analysis. By the Contract Analyzer the information is presented to the user through a number of different views described in following subsections.

System Overview

The system overview (see Figure 3) provides basic information about the contracting environment. Its contains fast overview of all deployed contracts and all participating parties. For each contract its name, ID, start date, end date, life-cycle status, and aggregation of clauses fulfilments are presented. For each contract party participating on any contract number of its roles in the contracts as well as the list of roles are presented. When any of the contract or contract party is selected, its view is open to present the related details.

Data about contract name, ID, dates, and life-cycle status as well as details of contract parties' participation on contract are acquired from the Contract Repository. Aggregations of clauses fulfilments are evaluated from clauses statuses reports provided by Observer, which receives them from the Monitor.

Contract Detail

The contract detail view (see Figure 3) provides detailed information related to a specific selected contract. The contract to be visualized may be selected either directly from the contract detail view, from the system overview, or from view of any on it participating party.

As the system overview contains very basic information about the contract, the contract view provides with the any information, which relates to the contract and is accessible to the Contract Analyzer. It contains contract name, its ID, start date, end date, life-cycle status, aggregation of clauses fulfilments, name and reference to contract template and parent contracts (if used), list of participating contract parties and their roles, and the table of clauses. The table of clauses provides with overview of all clauses defined in the contract. For each clause its ID, responsible party, modality, and fulfilment status are presented.

³http://www.gwt-ext.com/

Fig. 3.   Screenshot of a system overview and contract view

The contract detail view also provides (on request) XML of the contract document or any of selected clause. If any contract party is selected, its details are presented in the party view.

Information about clauses fulfilments are evaluated from clauses statuses reports provided by Observer; rest of information is acquired from the Contract Repository.

Party Detail

The party detail view provides detailed information on a selected Contract Party. As the Contract Analyzer is contract oriented, the party detail view contains information about the parties that are extracted from the information about contracts. I.e. only contract parties participating at any contract may be visualized here. The party to be visualized may be selected from the system view, contract view, or directly in the party view.

As the system overview contains very basic information about the party, the party detail view provides with contract party name and identification details (description and reference), contracts in which the party is involved and its roles in contracts, and tale of clauses for their fulfilment the contract party is responsible. For each clause the table of clauses contains its ID and ID of the related contract, modality and fulfilment status. On request the XML of selected clause may be presented.

Domain

The domain detail view presents the domain-level information in form of the events records stored by the Observer. There are four types of events that Observer collects and so is able to provide to Contract Analyzer. They are reports of predicate value changes, actions performed, ACL messages sent, and clauses fulfilment changes. Each of the report contains a time-stamp that allows sorting of reports,

and understanding their mutual relations as they appear in time.

The predicate report contains, beside the time-stamp, ID of the predicate and parameters for those it is valid. The action report contains the action performing initiator and the action performer, the action ID and its parameters (in form of message). The ACL message report contains message sender and receiver, message ID and performative. The clause report contains ID of the clause, ID of the contract and new fulfilment status.

5. Conclusion

This paper presents Contract Analyzer, which is frontend component of the observation and analysis pipeline for monitoring fulfillments of obligations concluded by electronics contracts. The Contract Analyzer concentrates mainly to visualization of information (static as well as online observed), which relates to the contract. History of performed activities as well current state of the contract is presented together with aggregations of observed data.

Contract Analyzer is implemented as an agent based on a platform developed (as well as whole pipeline) within project CONTRACT[4]. Information to be presented to the human user is combined from various sources accessible via agent interface.

Potential improvements are in (i) extension of analysis and visualization of historical data as well as current status of the contract obligations fulfillments, and (ii) concentration to the future trends, prediction of future states, and modeling of the contract parties behaviors.

Acknowledgements

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.

References

[1] BÍBA, J., HODÍK, J., JAKOB, M., PĚCHOUČEK, M. Contract observation in web services environments. To appear in: *Proceedings of Internatonal Workshop on Service-Oriented Computing: Agents, Semantics, and Engineering (SOCASE 2009)*. London, UK, Springer-Verlag, 2009.

[2] D4.1 Web Services Framework for Contract Based Computing (Revision for Month 18). Technical report of IST CONTRACT PROJECT, 2008.

[3] D4.2 Core code libraries for Web Services framework for contract based computing (Revision for Month 24). Technical report of IST CONTRACT PROJECT, 2008.

[4] FACI, N., MODGIL, S., OREN, N., MENEGUZZI, F., MILES, S., LUCK, M. Towards a monitoring framework for agent-based contract systems. In: *CIA '08: Proceedings of the 12th international workshop on Cooperative Information Agents XII*. Berlin, Heidelberg, Springer-Verlag, 2008, p. 292 - 305.

[5] JAKOB, M., PĚCHOUČEK, M., MILES, S., LUCK, M.. Case Studies for Contract-based Systems. In: *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems*, 2008, p. 55 - 62.

[6] OREN, N., PANAGIOTIDI, S., VAZQUEZ-SALCEDA, J., MODGIL, S., LUCK, M., MILES, S. Towards a formalisation of electronic contracting environments. In: *COIN 2008: Proceedings of AAAI Workhop on Coordination, Organization, Institutions and Norms in Agent Systems*, 2008, p. 61 - 68.

[7] SALLÉ, M. Electronic Contract Framework for Contractual Agents. In: *AI '02: Proceedings of the 15th Conference of the Canadian Society for Computational Studies of Intelligence on Advances in Artificial Intelligence*, London, UK, Springer-Verlag, 2002, p. 349 - 353.

[8] STREITBERG, W. Framework for the Negotiation of Electronic Contracts in E-Business on Demand. In: *CEC '05: Proceedings of the Seventh IEEE International Conference on E-Commerce Technology*, IEEE Computer Society, 2005, p. 370 - 373.

About Authors. . .

Jiří Hodík works as the research fellow in the ATG, CTU. He graduated from Czech Technical University in Technical Cybernetics. His research interests include artificial intelligence, multi-agent systems, auctions, multi-criteria decision-making, virtual markets, trust and reputation building and managing, and virtual organizations. During the visiting scholarship at the State University of New York at Binghamton he worked on immune system methods for information security systems (project BASIS funded by U.S. Air Force Research Lab at Rome NY).

---

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