Using KQML as the Agent Message Content Language in an Electrical Power Network Simulation

Miroslav Prýmek, Aleš Horák

Faculty of Informatics, Masaryk University Brno,
Botanicka 68a, 602 00 Brno, Czech Republic
{xprymek, hales}@fi.muni.cz

Abstract. In the paper, we present the agent communication standards that are implemented in the Rice simulation system for management and control of processes conducted in an electrical power network. The system models the power network using the multi-agent approach. We summarize the specific requirements of the area of electrical power network simulation and specific adaptations of the Knowledge Query and Manipulation Language (KQML) directed to system optimizations are described.

1 Introduction

In the current research, we have focused on the exploration of the possibilities of the usage of KQML (see [1],[2]) as the knowledge distribution language in our power system simulator called Rice. The feature and architecture overview of the simulator prototype was published in [3] and [4]. The main purpose of the simulator is to model the costs of the maintenance of a power distribution network in comparison with the costs of the outage, failures and undelivered energy. [5]

The process of the simulator implementation has led to the discovery that the KQML features are versatile enough to build up an agent communication network suitable for the highly scalable, modular and distributed power system simulator. On the other hand, the universality of the KQML model limited the performance of the system and increased the complexity of the analysis.

To overcome these difficulties, we have proposed some knowledge distribution optimization which will be discussed in this paper.

2 Power System Simulation Requirements

The *Rice* system is developed as fully distributed and dynamical system consisting of the independent agents. The behavior of the system is defined strictly in the manner of defining the behavior of the agents of which it consists.

In our system we are simulating these processes:

1. *The energy flow and the transformation of its characteristics* – the topology of the distribution network vary by reorganizing load on the lines and switching the lines on and off. The lifetime of the particular power network components can be affected by their load.

© Václav Snášel (Ed.): Znalosti 2008, pp. 371–374, ISBN 978-80-227-2827-0.
FIIT STU Bratislava, Ústav informatiky a softvérového inžinierstva, 2008.

2. *Cost of the maintenance of the power distribution network components* – each component holds an information about its condition and simulates the process of the maintenance.
3. *Components failures and their consequences*. Component failures can affect only the component condition or function or can have global consequences (outage).
4. *External events* - simulation of the natural disasters affecting power distribution network and user intervention in the system functions.
5. *Dynamic agents' behavior changes* – the user can change the agents' behavior *on demand,* even if it requires agent's program code change. This feature also supports interchanging of the *strategies* between agents.
6. *System function visualization* – graphical representation of the status of every system component and energy flow

The problem of the implementation of a multi-agent system which can meet all given requirements is definitely a problem of developing of the agent's knowledge interchange network which will support all functions needed for the simulation of the given processes. Which KQML features are needed?

Energy characteristics are implemented as a set of the values in the agent's knowledge base. Being so, all *energy flow* can be implemented as a chain of the basic query-response messages (KQML performatives group 1 and 4 above).

Network topology is constituted by the query-response sender and receiver pairs. Topology can be dynamically changed then by changing these pairs. To establish this lasting connection between agents we will need performative *subscribe* (group 9).

Because of the independent agent concept of the multi-agent systems, *maintenance cost* and *component failures* are private processes of the agents. These processes result in the status change of the agent and eventually sending of the information about it to all the subscribing agents.

External events are implemented as an insertion of the particular knowledge into the agent's knowledge base. This requires the performative *tell*.

System function visualization is implemented in the stand-alone agent *Viewer*. This agent subscribes for the value changes of the knowledge which are to be visualized as the agent state (e.g. the actual power load of the line).

The last but not least function of the system is dynamically changeable agent behavior. This deserves *content language* in the KQML messages which is robust enough to transfer behavior code. It deservers the programming language which supports dynamic code compilation (in our system we use Python language).

3 Optimizations

From the needed-feature list above, you can see that only a small portion of the KQML possibilities are to be used for the implementation of our power distribution system simulator. Realizing this in the process of the software analysis, we have proposed these optimizations:

To use multiple content languages appropriate for the purpose of the particular message, i.e.:

- usage of the simple, easy-parseable content language for inserting and querying data of the agent knowledge base

- usage of the higher programming languages (e.g. Python) for other purposes where the above-mentioned language will not be sufficient. These languages are general enough for any data structure definition (including behavior definitions – i.e. function definitions).

To develop a highly effective partial KQML parser with the best effectiveness in dealing with the most common messages, open for scalability and extendibility of the set of the KQML performatives understood.

3.1 Content languages

For interchanging complex data such as function definition we use Python programming language because of the possibility of ad hoc compilation of the given code. Usage of complex data structures is so rare that time-consuming code compilation process does not significantly affect general simulator performance.

However these higher-level languages are not suited for the exchange of the simple-structured data. With KQML we are not limited to use just one language for all purposes and rather we can use a language which is fittest for a given purpose. In our system, we divided message into two categories and use two expression languages for them.

The second language is used for setting and also querying agent's knowledge bases. We use simplest possible declarative language with expression in this form `label = value`, where label is a knowledge element label and value is a string representation of the given value. The special symbol '?' means that the value is not known or is to be asked.

The message content is constituted with one or more lines with the knowledge element/value pairs. This simple format can be parsed with regular expressions and all asked values ("?") can be automatically replaced with their respective values.

This declarative language in connection with the KQML concept of performatives is strong enough to express all operations on the knowledge base which are needed in our simulator:

`ask(voltage=?)`	which value is related with the knowledge labeled "voltage"? (which load is on the line)
`ask(power_free=10)`	is there (at least) 10 (kW) of a free distribution capacity?
`tell(voltage=10)`	load on the line is 10 (kV)
`untell(voltage=10)`or `tell(voltage=?)`	delete knowledge from the knowledge base
`tell(voltage=?)`	(as a reply to the *ask* message) the knowledge is not present in the agent knowledge base

3.2 Optimizing KQML parsing with an in-reply-to filtering

KQML messages semantic analysis is a difficult task. KQML messages can be nested constructing entities with a very complex meaning. To overcome this difficulty we used a concept of a partial KQML parser – there are defined canonical messages

which are most frequent and such must be handled most effectively. We use message filtering for it.

According to KQML specification, message should contain property *reply-with*. When answering to such a message, agent must send a message in which the property *in-reply-to* is set to a same value as the original message *reply-with* property.

By parsing the field *in-reply-to* first, we can identify the purpose of the message and we can suppose that the message is structured in accordance with the initial agreement (which should be part of the KQML *ontology* definition). Supposing this, the process of parsing and semantic analysis of the message can be simplified and significantly accelerated.

Using this concept we can construct KQML parser which handles most frequent messages much faster than a parser which will analyze the messages with no anticipatory knowledge. On the other hand, the universality of this parser is not affected – messages structured different than the ontology standard are analyzed with a full KQML analyzer.

4 Conclusions

The proposed optimizations have shown to be reasonable and have led to a significant speed up of the inter-agent communication. Even if our prototype is written in a slow scripting language, the performance achieved is promising for the future implementation of the system in more effective programming language.

The main advantage of the proposed optimizations is that they do not break KQML specification conformance and can be aimed specifically at the bottle-neck of the inter-agent communication and also can be fine-tuned with respect to the frequencies of a particular KQML message types and structures.

Acknowledgements

This work has been partly supported by the Academy of Sciences of the Czech Republic under the project T100300414.

References

[1] The DARPAKnowledge Sharing Initiative, *Specification of the KQML Agent-Communication Language.* 1993. http://www.cs.umbc.edu/kqml/papers/kqmlspec.pdf

[2] Labrou,Y.,Finin,T.,*A Proposal for a new KQML Specification.* 1997. http://www.cs.umbc.edu/kqml/papers/kqml97.pdf

[3] Prýmek, M., Horák, A., "Multi-Agent Framework for Power Systems Simulation and Monitoring." In *Proceedings of ICICT 2005*. Cairo, Egypt : Information Technology Institute, Giza, Egypt, 2005. pp. 525-538. ISBN 0-7803-9270-1.

[4] Prýmek, M., Horák, A., "Design and Implementation of Multi-Agent System for Analysis of Electrical Power Networks." In *Proceedings of ELNET 2005.* Ostrava, Czech republic: VŠB TU, Ostrava, 2006. pp. 525-538. ISBN 80-248-0975-3

[5] Prýmek, M., Horák, A., "Modelling of Economical Aspects of a Power System Failure." In *Proceedings of PEA 2006.* Anaheim, USA : International Association of Science and Technology for Development, 2006. s. 61-66. ISBN 0-88986-616-3.