2009-ICS-154 (9)
2009/3/2

知識型プロトタイプシステムの性能および再使用可能性の面の評価
－ 通信網への自動化された管理 －

ｱﾊﾞｰ サミラ†, 木下哲男‡

†東北大学情報科学研究科, ‡サイバーサイエンスセンター
〒980-8577 宮城県仙台市青葉区片平２－１－１

あらまし 本稿では、ネットワーク管理タスクの自律的な支援を行うための汎用的かつ再利用可能なドメイン知識の抽出と表現の手法を提案する。提案する知識に基づいたフレームワークにおいて，ネットワークドメインに関連した静的な概念情報は，ドメインオントロジーとして構成した。具体的には、管理対象に関する静的情報のオントロジ、および複数のエージェントに分散配置される管理知識のための表現モデルを提案し、実験用ネットワークシステムにおける試作システムの概要と実験例について議論する。実験結果は手動ネットワークの経営技術と比べて管理間接費のマーク付きの減少を確認する。再使用および修正のためのシステムの評価は、さまざまなテストケース内の知識の部品の使用を確認する。

キーワード 知識表現、領域オントロジ、ネットワーク管理、エージェント指向コンピューティング、故障検出、分散問題解決

Evaluating the Performance & Reusability Aspects of a Knowledge-based Prototype System:
— An Automated Management Support to Communication Networks —

Sameera ABAR†, and Tetsuo KINOSHITA‡

† Graduate School of Information Sciences ‡ Cyberscience Center
Tohoku University, 2-1-1 Katahira, Aoba-ku, Sendai, 980-8577, Japan
E-mail: † sameera@ka.riec.tohoku.ac.jp, ‡ kino@riec.tohoku.ac.jp

Abstract This paper proposes a reuse-based knowledge representation and acquisition strategy, for the automatic provision of just-in-time and just-enough, context-dependent resource knowledge, for actively managing the communication network systems. In the proposed knowledge model, the static domain content of a network system has been represented as the domain knowledge ontology, and the experiential management knowledge is encoded as the production-rule type representations of distributed multi-agent middleware architecture. For the proof of concept, an experimental network system has been set-up in the laboratory. A couple of test-cases have been designed for experimenting with the implemented prototype system. Experimental results confirm a marked reduction in the management-overhead as compared to the manual network management techniques, in terms of the time-taken and effort-done during a particular fault-diagnosis session. Validation of the reusability/modifiability aspects of our system, illustrates the flexible manipulation of the knowledge fragments within diverse application contexts.

Keyword Knowledge Representation, Domain Ontology, Agent-based Computing, Distributed Problem-solving, Failure Detection, Network Management

1. Overview

Growing complexity of the distributed communication network systems, and the associated voluminous information overhead have made the network management an increasingly troublesome task. For managing these huge distributed network systems, the manual procedures have become quite tedious. The effective management of huge and complex networks is possible only with the intelligent and automated network management tools. Key to the automation of network management functions is the detailed interpretation of the network-related knowledge resources. Hence, a knowledge acquisition, representation, and sharing technique, in the context of network management domain, for deployment with the multi-agent support mechanism, is highly desirable. The proposed knowledge-model has been constructed in-line with the CommonKADS [1], which is a comprehensive methodology for structuring the application intensive

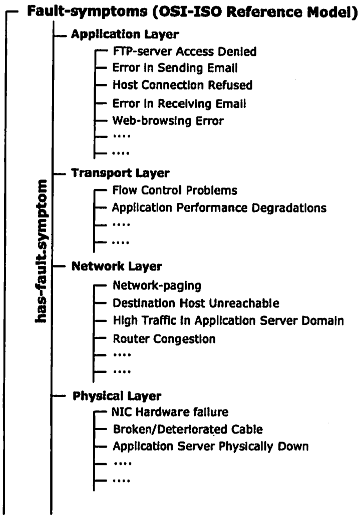
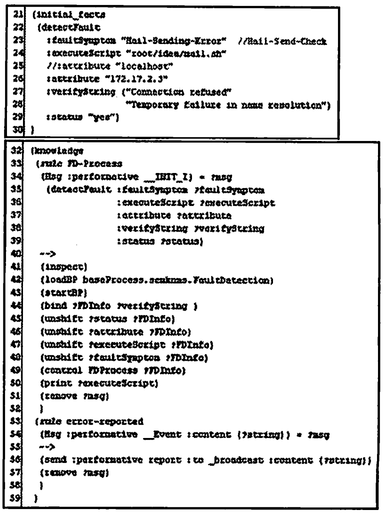


Figure-1: Hierarchical view of fault-symptom taxonomy



```
(initial_facts
 (detectFault
   :faultSymptom "Mail-Sending-Error"  //Mail-Send-Check
   :executeScript "root/idea/mail.sh"
   //:attribute "localhost"
   :attribute "172.17.2.3"
   :verifyString ("Connection refused"
                  "Temporary failure in name resolution")
   :status "yes")
)
```

```
(knowledge
 (rule FD-Process
  (Msg :performative __INIT_I) = ?msg
   (detectFault :faultSymptom ?faultSymptom
                :executeScript ?executeScript
                :attribute ?attribute
                :verifyString ?verifyString
                :status ?status)
  -->
  (inspect)
  (loadBP baseProcess.scukms.FaultDetection)
  (startBP)
  (bind ?FDInfo ?verifyString )
  (unshift ?status ?FDInfo)
  (unshift ?attribute ?FDInfo)
  (unshift ?executeScript ?FDInfo)
  (unshift ?faultSymptom ?FDInfo)
  (control FDProcess ?FDInfo)
  (print ?executeScript)
  (remove ?msg)
  )
 (rule error-reported
  (Msg :performative __Event :content (?string)) = ?msg
  -->
  (send :performative report :to _broadcast :content (?string))
  (remove ?msg)
  )
 )
```

Figure-2: Knowledge template of fault-detection agent

knowledge of the expert systems. This scheme aims to facilitate the reusability and shareability of the knowledge resources of a system.

2. Proposed Knowledge Model

Our modeling approach categorizes the network-related knowledge as the domain knowledge (static domain content, and dynamic status information), and the experiential management knowledge (inference strategy, and task structures). Ontologies are used to capture knowledge about some domain of interest; these describe the concepts in the domain and also the relationships that hold between those concepts. The hierarchy of concepts within our experimental network system, and their crucial properties are specified through the attribute-value mechanism, or more specifically, the domain knowledge ontology, which has been designed in Protégé-3.4 [2]. The management expertise prescribes the execution of various task relevant goals in the operational network system, and is incorporated as the fault-case reasoning models [Figure-4], where the nodes represent the events and whose directed edges represent causality. This experiential management knowledge is explicitly encoded as the production-rule type representations of the DASH-agents [3], which actualize the automated reasoning tasks in conjunction with the static domain-specific content. Moreover, the cooperative problem-solving is initiated when the agents invoke the underlying base-processes, which interact with the run-time dynamic status information during the diagnostic sessions. A base-process is in-fact an algorithm of run-time problem-solving, whereas agents act cooperatively on the front-end. The fault-symptom taxonomy as shown in Figure-1, contains the knowledge of nearly all the possible failures which could be encountered by the network user.

Task Knowledge or problem solving task structures are designed as the generic hierarchy of tasks which prescribe the activities to be performed in a domain of interest. The diagnosis is defined as the task of identifying the cause of a fault that is manifested by some observed behavior. Fault diagnosis process incorporates the following steps: observe the symptoms, develop a hypothesis about the cause of a symptom, test the hypothesis, and if the hypothesis fails, iterate these steps until a conclusion is reached.

3. Multi-agent Middleware System

The intelligence layer consists of multi-agents for handling the empirical task relevant knowledge to relieve the network operators from the tedious monitoring and control of the network system. The proposed system architecture is supported by the Agent-based Distributed Information Processing System (ADIPS) framework [4], which is a flexible computing environment for the implementation of multi-agent systems. This framework employs a repository-based system development methodology which allows autonomous adaptive actions

Table-1: Test-cases for experimentation with the prototype system

Test Case	Network Node (Machine)	Initially Detected Fault-symptom	Root-cause of Failure
1	**Router_A-1**	**Network-paging [Slow-speed]**	**Excessive User-requests or Connectivity Failure**
	Performance-degradation occurs due to data bursts caused from the excessive number of user-requests, or equipment/link failure		
2	**Work-St_Aral**	**FTP-server Access Denied**	**Power Failure in Router-A_1**
	Power failure in Router_A-1 results in broken connectivity. Error messages from the data-transport layer are generated for all affected connections. As a consequence, disruption in the file access occurs		
3	**Note-Bk_Elm**	**Error in Sending Email**	**SMTP-server Process Down**
	SMTP-server process "sendmail" goes down in the application server [App-Sv_Maple], thereby causing an error while sending the email		
4	**Desk-Tp_Fawn**	**Host Connection Refused**	**HTTP-server Port Firewall-Blocked**
	Port of HTTP-server in Subnet-A is blocked by the firewall [ip-tables], causing the connection refused error or an error-page is displayed		
5	**Note-Bk_Pisces**	**Error in Receiving Email**	**POP-server Port-setting Failure**
	POP-server port has been configured incorrectly in the application server [App-Sv_Maple], thereby causing an error in receiving the email		
6	**Desk-Tp_Lynx**	**Destination Host Unreachable**	**Incorrect NIC-setting**
	Due to the NIC-setting problem of Desk-Tp_Lynx, connection is refused to HTTP-server in Subnet-A		
7	**Note-Bk_Kane**	**Web-browsing Error**	**DNS Configuration Problem**
	DNS misconfiguration results in the failed name resolution, hence Web-browsing is inaccessible		

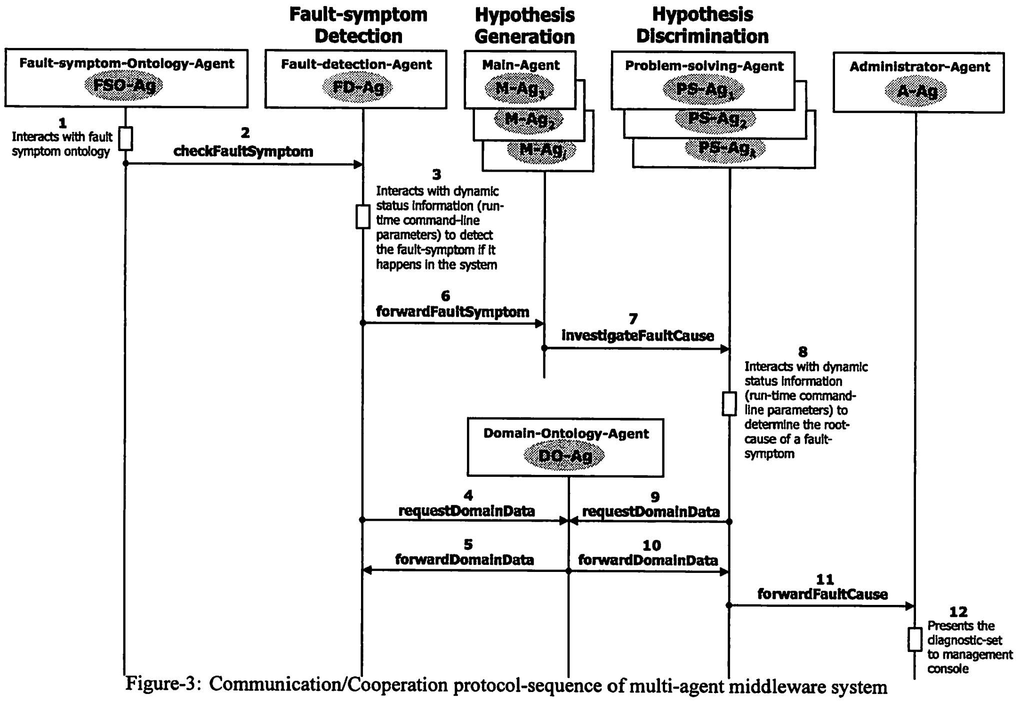
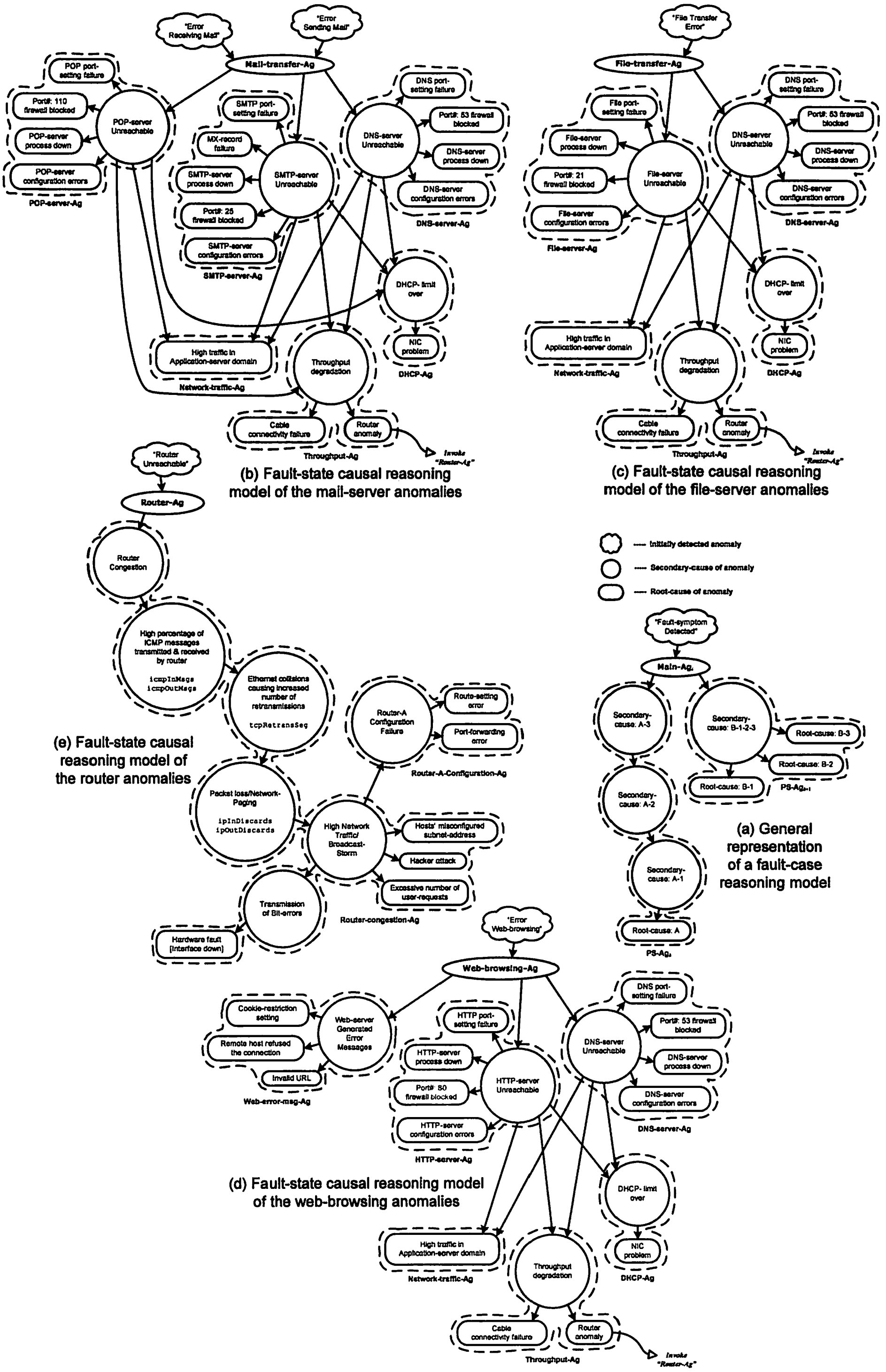

Fault-symptom Detection
Hypothesis Generation
Hypothesis Discrimination
Fault-symptom-Ontology-Agent
FSO-Ag
Fault-detection-Agent
FD-Ag
Main-Agent
M-Ag$_1$
M-Ag$_2$
M-Ag$_i$
Problem-solving-Agent
PS-Ag$_1$
PS-Ag$_2$
PS-Ag$_k$
Administrator-Agent
A-Ag
1
Interacts with fault symptom ontology
2
checkFaultSymptom
3
Interacts with dynamic status information (run-time command-line parameters) to detect the fault-symptom if it happens in the system
6
forwardFaultSymptom
7
investigateFaultCause
8
Interacts with dynamic status information (run-time command-line parameters) to determine the root-cause of a fault-symptom
Domain-Ontology-Agent
DO-Ag
4
requestDomainData
9
requestDomainData
5
forwardDomainData
10
forwardDomainData
11
forwardFaultCause
12
Presents the diagnostic-set to management console

Figure-3: Communication/Cooperation protocol-sequence of multi-agent middleware system


(b) Fault-state causal reasoning model of the mail-server anomalies

(c) Fault-state causal reasoning model of the file-server anomalies

(e) Fault-state causal reasoning model of the router anomalies

(a) General representation of a fault-case reasoning model

(d) Fault-state causal reasoning model of the web-browsing anomalies

Figure-4: Fault-case reasoning models for the probable occurrence of various anomalies in the network system

Table-2: Quantitative evaluation criteria based on the time-taken, and the number-of-procedures adopted, to resolve a particular fault-case, in the proposed system

Initially Detected Fault-symptom			**Network Management Method**		
			Manual Procedure	**Proposed System**	**Reduction in T or S [%]**
1	**Network Paging**	T	349.6	52.7	84.93
		S	10.6	1	90.57
2	**FTP-server Access Denied**	T	655	58.3	91.1
		S	19	1	94.74
3	**Error in Sending Email**	T	821.8	74.4	90.95
		S	25.0	2	92.00
4	**Host Connection Refused**	T	482.5	80.7	83.28
		S	8.5	3	64.71
5	**Error in Receiving Email**	T	615	85.3	86.13
		S	9	4	55.55
6	**Destination Host Unreachable**	T	709.0	56.6	92.02
		S	24.4	1	95.91
7	**Web-browsing Inaccessible**	T	836	94.1	88.75
		S	23.5	4	82.98

T = Time-Taken, Average Reduction in T = **88.16%**
S = #-of-Steps, Average Reduction in S = **82.35%**

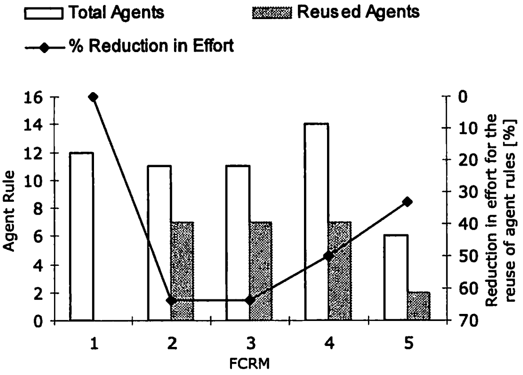


Figure-6: Reuse of agent-based fault-case reasoning models

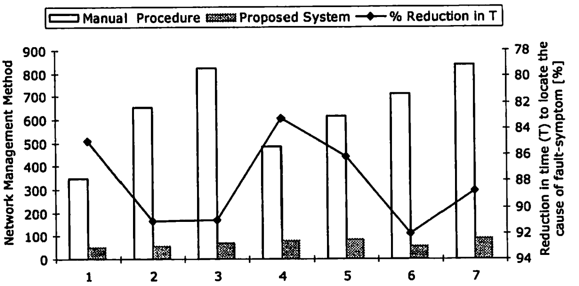


Figure-5 (a): Quantitative evaluation based on the time-taken, to resolve a particular fault-case

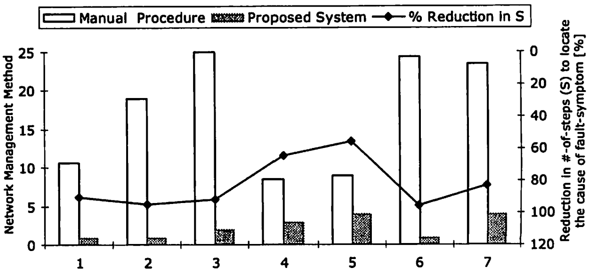


Figure-5 (b): Quantitative evaluation based on the number-of-procedures adopted, to resolve a particular fault-case

Table-3: Reuse of agent-based fault-case reasoning models in the proposed system

Fault-case Reasoning Model (FCRM)		**Agent**	**Base-process**	**Agent Rule**		**R [%]**
				Total	**Reused**	
1	**FCRM for SMTP-related Anomalies**	**5**	**2**	**12**	**0**	**0**
2	**FCRM for POP-related Anomalies**	**5**	**2**	**11**	**7**	**63.63**
3	**FCRM for FTP-related Anomalies**	**5**	**2**	**11**	**7**	**63.63**
4	**FCRM for HTTP-related Anomalies**	**6**	**2**	**14**	**7**	**50.00**
5	**FCRM for Router-related Anomalies**	**2**	**1**	**6**	**2**	**33.34**

R = Reduction in effort for the reuse of agent rules on each designed distributed system. The internal state knowledge of the agents comprises of: Identification Knowledge specifies the environment or the system where the workplace agents are instantiated. Fault-Detection Knowledge [the code-snippet in the Figure-2], and Problem-Solving Knowledge inevitability constitutes the core knowledge for detecting the anomalies, and has been expressed as the production-rule type instructions in an agent. For each initial fault-symptom detected, various run-time checks are performed for all possible occurrences of the fault-symptom, and the information obtained (about the root causes of the obstacle), is compared with the pre-defined one, and then the problem-solving findings are forwarded to the management console. Figure-3 shows the communication and cooperation protocol-sequence of the multi-agent middleware system.

4. System's Evaluation

Our experimental network system, comprises of the 100BASE-TX Ethernet, firewall configured as NAT (Network Address Translation), routers, bridges, various personal computers arranged in three sub-networks, and Fedora-Core:ver-2.0 has been installed on all machines.

4.1 Validating the Efficiency of Prototype System

The performance of the proposed knowledge-model method is determined in comparison to the OS-default network management tools. Several failures [Table-1] obstructing the normal operation of network system are generated, and accordingly and it is required to detect these failures with the manual network management tools and mechanisms, as well as with the implemented prototype system. Also, the time elapsed between the notification of the failure to its root-cause determination has been measured, plus the number of procedures executed. These criteria serve as the index for measuring the practical worth and applicability of the automated network management notion, thereby determining the extent to which the burden of the network administrator has been reduced. The results indicated in the Table-2 determine an average reduction of 88.16% in time, and 82.35% in the number of steps performed, in comparison to the manual method, while resolving a particular fault-case.

Results of automation procedure are affected by the number of nodes of the network system, the instantiation time of agents onto various network nodes, the size of the log-file interpreted by the java base-process to locate the failure, and the time required to get the collective result indications from the network nodes under investigation, on the management console.

4.2 Validating the Reusability of Prototype System

Table-3 demonstrates a considerable reduction in the effort required while designing the agent-based fault-case reasoning models, for various failure scenarios. The fault-case reasoning model of SMTP-related failures has been created initially as shown in the Figure-4 (b), and it has been shown that when these small grain agents were reused for POP, FTP, HTTP and router-related anomalies, a considerable reduction in the design-effort has been reported. Similarly, the network domain ontology can also be modified efficiently on the centralized management console. More specifically, it concludes that some alteration of the network do not reflect the changes in knowledge models embedded in the system. This makes the knowledge acquisition activity less complex. It has been assumed that an equal amount of effort is required to alter/modify/design each knowledge component.

5. Concluding Remarks

We have presented a knowledge representation, acquisition, and utilization approach for the automated management support to communication network systems. Validation of the reusability aspects of the system, as well as its performance measurements has been done. The ontology design and implementation in this project, in the practical network management domain, is the most novel concept which has never been reported in the literature.

Acknowledgements:

This research project has been partially supported by the Japan's MEXT "21st-Century COE (Center Of Excellence) Program [System Construction of Global-Network-Oriented Information Electronics]" and the "Global-COE Program [CERIES]" of Tohoku University (2005 ~ 2009), and the "Tohoku Kaihatsu Kinen Foundation", Japan (2006 ~ 2008). Authors are thankful to Dr. Yukio Iwaya, Dr. Takahiro Uchiya, Dr. Susumu Konno, Mr. Takahide Maemura, and Mr. Satoshi Yoshimura for their worthful technical assistance.

References

[1] G. Schreiber, B. Wielinga, R. de Hoog, H. Akkermans, and W. Van de Velda, CommonKADS: A Comprehensive Methodology for KBS Development, IEEE Expert, Vol. 9, pp. 28-37, Dec. 1994.

[2] Protégé — An ontology editing tool: http://protege.stanford.edu/

[3] DASH — Distributed Agent System based on Hybrid Architecture: http://www.agent-town.com/dash/index.html

[4] T. Kinoshita, and K. Sugawara, ADIPS Framework for Flexible Distributed Systems, Lecture Notes in AI, Springer-Verlag, Vol. 1599, pp. 18-32, 1999.