SmartResource Project

Technical report

"*Deliverable 2.1, General Proactivity Framework (Pro-GAF)*"

1. Introduction

There is a huge amount of academic and industrial initiatives world-wide related to agent-oriented analysis. To organize those efforts a special Co-ordination Action for Agent Based Computing, AgentLink III[1], funded by the European Commission's 6th Framework Program was launched on 1st January, 2004 until December 2005. The AgentLink III initiative currently has registered 99 projects and 128 software products based on agent approach. Core technologies of several commercial organizations utilize different agent paradigms. For example, Whitestein Technologies[2] and Agent Oriented Software Pty Ltd[3] have provided advanced software agent technologies, products, solutions, and services for selected application domains and industries since 1999. Agent-based approach has been tried in a research of industrial automation systems domain [2, 7].

Modeling of multi-agent systems and behavior of concrete agents in it has been one of the interesting and significant topics in the concerned domain. Model-driven approach to design of agent behaviors emerged a long time ago and initially was based on UML modeling [9, 11]. Later this approach was extended to a level of meta-modeling [10]. As one of the mature UML-based methodologies for modeling multi-agent systems, Agent Modeling Language can be mentioned [12]. Currently, AML is used in commercial software projects, is supported by CASE tools and in the nearest future first version of its specification will be presented to public for its further development.

One of the famous formal theories about behavior in multi-agent systems [13] is developed and lectured in Free University of Amsterdam[4].

All the above efforts have elaborated in details a conceptual base of agent behavioral modeling and motivated its further development. There were even attempts to elaborate a conceptual convergence of an agent layer and Web Service Architecture [8]. However, academic efforts lack concrete details concerning methodology of modeling or have just very preliminary prototype implementations as e.g. the Agent Academy project [1].

Recently, ontology-driven approach is growing as an option to Model-driven one, while having several advantages:

- Possibility of reasoning on a level of a single model and inter-model relationships and mappings, supporting meta-model level as well.

[1] http://www.agentlink.org/
[2] http://www.whitestein.com/
[3] http://www.agent-software.com/
[4] http://www.vu.nl/

- Support of flexibility for tools (e.g. XSLT transformations) based on ontology during an evolution of the ontological model (see analysis of evolution of classes and properties and its impact on tools in [14]).
- More flexible modeling framework based on a graph [15].

DERI is among research centers that are very close to implementing really powerful prototypes of ontology-driven modeling for Web Services and Multi-Agent Systems. Significant efforts for development of agent goal-behavior frameworks based on WSMO standard (ontology-driven) have been conducted by a research group from DERI according to their vision of Semantic Web Fred [3].

As a possible option of implementation of agents behavior engine are Horn-like rules. W3C standardization efforts aimed at this direction, have recently resulted into a family of standards: RuleML[5] (Rule Markup Language), SWRL[6] (a Semantic Web Rule Language Combining OWL and RuleML), FOL RuleML[7] (First-Order-Logic RuleML), SWRL FOL[8] (SWRL extension to First-Order Logic). All these standards are tightly related to previous research carried out by IBM alphaWorks Labs in development of CommonRules[9] and BRML (Business Rule Markup Language). The initiative within CommonRules was aimed at a development of a framework for specification of executable business rules by non-programmer business domain experts. The final result represented a reusable technology of business rules and rule-based intelligent agents embodied as extensible Java library. Industrial Standards related to modeling and automation of business behavior are currently concentrated around BPEL4WS[10] and ebXML[11].

From the technological side, there are reliable options to be a basis for implementing frameworks for modeling behaviors in multi-agent systems: JADE-Jess-Protégé[12] and Aglets SDK[13]. In the JADE implementation several Java upper classes have been provided [4] and this promoted use of the JADE platform in implementation of tools for modeling complex agent behaviors. The JADE Framework has been extended by a BDI infrastructure within the Jadex[14] project [5] and its behavioral model was extended by Hewlett Packard Lab in their HP SmartAgent initiative [6].

The research presented in this document aims at development of a framework of ontology-driven proactive behavior of resources using Multi-Agent Systems.

2. RGBDF – Resource Goal and Behaviour Description Framework

Autonomous systems must be automatic and, in addition, they must have a capacity to form and adapt their behaviour while operating in the environment. Thus, traditional AI systems and most robots are automatic but not autonomous - they are not independent of the control of their designers. Autonomous systems are independent and are able to perform a self-control. As it is argued in this research, to do this, they must be motivated.

2.1 Resource Proactivity Approach

In Agent Environment as well as in real world the base for any interaction is behaviour of each individual. Further, integration of these individual behaviours may form behaviour of Agent Alliance.

[5] http://www.ruleml.org/
[6] http://www.daml.org/2003/11/swrl/
[7] http://www.daml.org/2004/11/fol/folruleml
[8] http://www.daml.org/2004/11/fol/
[9] http://www.research.ibm.com/rules/commonrules-overview.html
[10] http://www-128.ibm.com/developerworks/library/specification/ws-bpel/
[11] http://www.ebxml.org/
[12] http://jade.tilab.com/doc/examples/JadeJessProtege.html
[13] http://www.trl.ibm.com/aglets/
[14] http://vsis-www.informatik.uni-hamburg.de/projects/jadex/

In real world almost all of behaviours (actions) are goal-driven, but some of them are not. With software agents in mind we are focused just on the goal-driven behaviour. What is a goal-driven behaviour? Such behaviour means performing set of rules, which are aimed for achievement of certain goal. In return, goal is a fact which does not exist in a description of the environment, and an agent aims at appearance of the fact. As a result, we have a trio: behaviour which is driven by certain goal and which lies in performing actions following a set of behavioural rules. However, even having a rule base, which enables an agent to achieve a goal, still extra information (environmental facts) is needed. This is because each rule has to have a sufficient condition. In our case a sufficient condition is a presence of input data for action performing. Having the sufficient condition we should take into account necessary condition: presence of a goal along with a certain context (set of facts of the environment) for performing the goal. Not all goals assume execution of unambiguous rule(s). Some goals can be represented by aggregation of more specific goals.

Referring to the trios that were discussed above, each agent should have initial set of those trios (regulated by initial role). These trios represent expertise and experience of an agent. As well as in real world agents can exchange their expertise (rules for execution of actions depending on the goals and direct software modules for execution of actions). Availability of a wide spectrum of the trios gives a possibility for agent to automatically divide up goals (which cannot be achieved because of lack of information) to sub goals and to create a chain of nested trios.

One more thing from a modelling paradigm that can be applied to an agent is an agent role. Agent role means aggregate of goals corresponding to a specific purpose of the agent. Individual role does not assume a fixed set of activities, the set of the goals can be different even for the same role depending on the context. Such approach to the goal and behaviour description brings a possibility for agent to be more autonomous. Through utilization of this approach agent can change its role, set of the goals corresponding to its purpose depending on a condition of the environment. In other words, an agent can change its behaviour based on a context.

Approach of RG/BDF assumes concentrating all the goals, roles descriptions and templates of behavioural rules in ontology. The templates of behavioural rules are described in a general way with a purpose to be applied to any particular agent. Such description requires utilization of a handy and flexible description schema (RG/BDFS-Lite), which will be presented later. Architecture of general Agent Platform is represented in Figure 1.

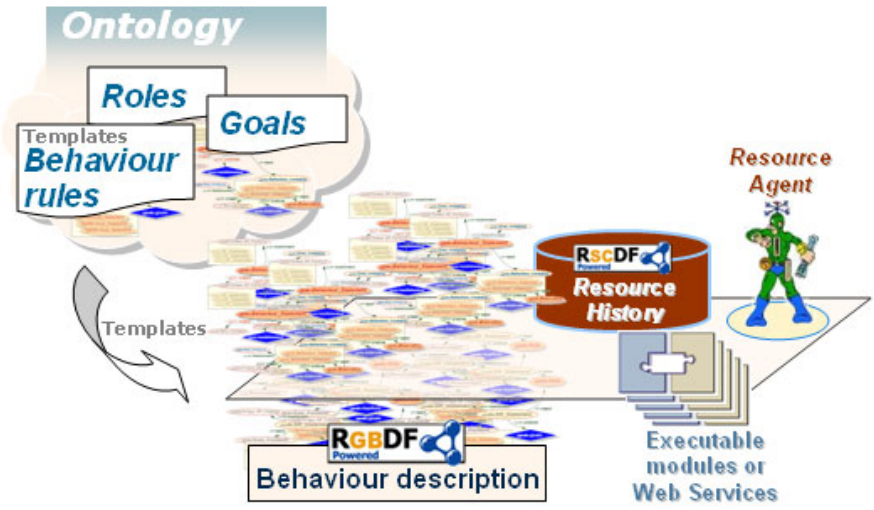

Figure 1. SmartResource platform architecture

On its own platform agent has Resource History, where it stores all statements about resource states, conditions and actions that have been performed by the resource agent and other information that can

be useful for it (statements about the environment). Some executable modules (code) that agent must perform also can be located there as an output of its behavioural rule chain. Otherwise, the agent has to utilize external web services. Agent always has to interact with ontology server to be able to download necessary role, goal description or behavioural template whenever the agent needs it.

Behavioural template represents a rule for behaviour in RDF serialization. The template is represented by a behavioural statement *rgbdfs:Behaviour_Statement* (it will be described in the next chapter) and contains necessary condition (goal) and sufficient condition (condition of the environment) as the contexts of rule execution and a set of the executive descriptions (specification of the executable modules that should be invoked) as an output of the rule.

We can divide the process of the resource goal and behaviour annotation to stages. The first one is a stage of goal instance definition that assumes creation (process of describing) of a statement to which an agent should strive. This goal can be specified directly by an expert or via specification of the agent goal. Based on the goal description appropriative behaviour template(s) have to be found in ontology, downloaded and transformed to the behavioural instance(s) on the resource platform. After this, the needed executable modules (if they are not located at the resource platform) also can be downloaded. As a final stage of goal/behaviour annotation process expert has to specify (add/modify) the context of the behaviour. Now the platform contains behavioural rule(s) in RDF/XML serialization form that can be performed by the agent engine. This engine follows the behavioural rules till the goal is achieved.

2.2 RG/BDF-Lite

In continuation to the idea of Context Description Framework (CDF), which was developed by Industrial Ontologies Group[15], such approach can be applied to context sensitive Resource Goal and Behaviour Description Framework (RG/BDF). RG/BDFS-Lite is an upper schema for description of resource goal and behaviour. It is based on the CDF schema and extends it as well as Resource State and Condition Description Framework Schema (RS/CDFS) does.

rgbdfs:Goal_Statement is a class of the goal instances. This class is similar to rscdfs:SR_Statement and is its subclass. Triple <SSS-PPP-OOO> describes a statement of a fact, which is currently absent in the resource history and which a resource aims to have (i. e. an agent must achieve this goal). Each goal is dynamic and can belong to a resource only in certain context (see Figure 2).

Figure 2. Goal Statement

rgbdfs:Behaviour_Statement - a class of the behavioural instances is represented in Figure 3. This class is a subclass of rscdfs:SR_Statement with extended properties. The rscdfs:ResourceAgent class plays a role of a range for the subject (***rgbdfs:subject***). Range of the statement's predicate (***rgbdfs:predicate***) is restricted by the ***rgbdfs:B_Property*** class (subclass of rscdfs:SR_Property). An object of the behavioural statement can be represented by rgbdfs:Behaviour_Container: a container for nested behavioural statements (if a root behaviour is complex) or for atomic execution(s). The ***rgbdfs:falseInContext*** property is a sub property of the rscdfs:falseInContext property. This property

[15] http://www.cs.jyu.fi/ai/OntoGroup/

has a little bit different meaning than its super property. It plays a role of a trigger, which switches on and switches off the execution of the rule (execution of the behavior, which is described via the behavioural container). Behavioural statement will be true, if in the resource history there is no a statement about the fact of the specified goal. The property makes a link to a goal container, which contains goal statement(s) (because it is reasonable to execute behaviour only when a goal has not been achieved). If presence of a goal is a necessary condition for the behaviour, then contextual statements (condition of the environment) are a sufficient condition (which is represented by the contextual container via the rscdfs:trueInContext property).

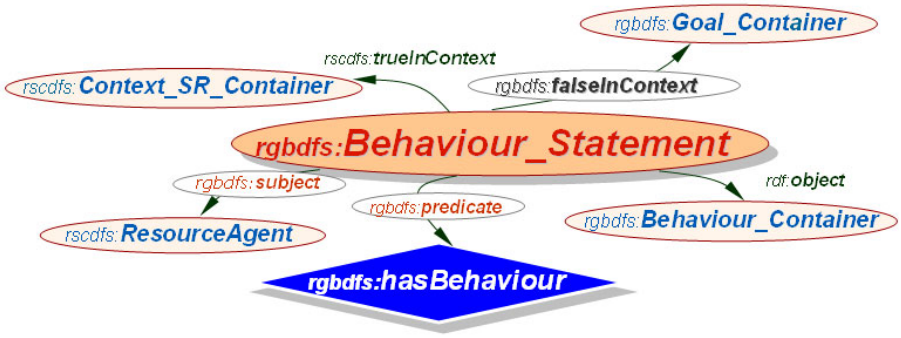

Figure 3. Behaviour Statement

rgbdfs:Goal_Container is a class of the instances of a goal container. This class is a subclass of rscdfs:SR_Container in general. It represents a container of goal statements, which define the goals. Such container plays a role of a context (using the rscdfs:falseInContext property) for a behavioural statement till the goal is achieved, and that is why it is a direct subclass of rscdfs:Context_SR_Container. The rgbdfs:gMember property is a property redefined from rscdfs:member and which defines instance of the rgbdfs:Goal_Statement class as a member of the container.

As it was mentioned in the previous chapter, goal can be divided into a set of sub goals. Thus, a goal container also plays a role of a set of goals, members of which are sub goals of a complex goal. With the purpose of defining a set of sub goals for a complex goal, the property ***rgbdfs:subGoal*** was defined in RG/BDFS-lite (see Figure 4). The domain and range for this property are rgbdfs:Goal_Statement and rgbdfs:Goal_Container classes correspondingly.

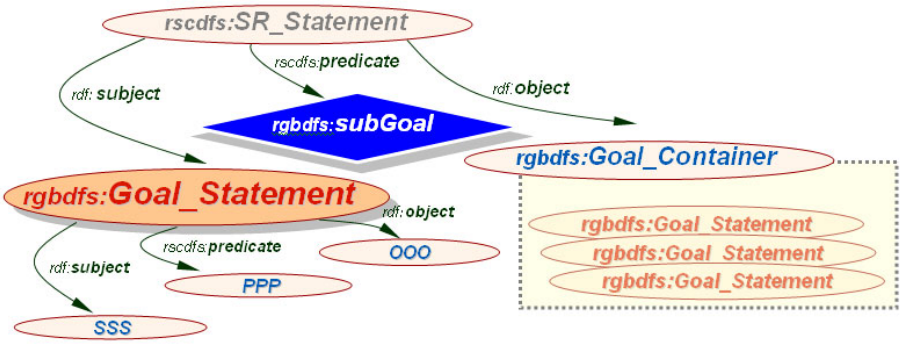

Figure 4. RGBDF Goal

rgbdfs:Behaviour_Container is a class of the instances of a behavioural container. As a subclass of rscdfs:SR_Statement, it has a redefined rgbdfs:bMember property. The main role of the behavioural container is collecting nested behaviours for a complex behaviour (represented by a behavioural statement) (see Figure 5).

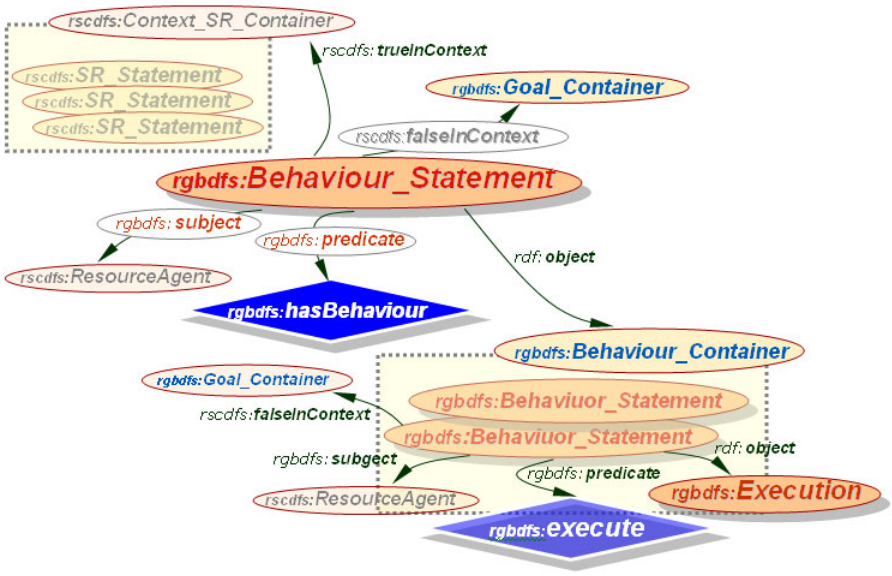

Figure 5. RGBDF Behaviour

Simple behaviour, that assumes execution of a certain action (invocation of certain method, code…), can be described via the ***rgbdfs:execute*** property (instance of the rgbdfs:B_Property class), which defines instance of ***rgbdfs:Execution*** class for a resource agent (see Figure 4). This instance describes the exact method (code, service, etc.), inputs, outputs and other features of an executive entity. With a purpose of defining a complex behaviour (which assumes execution of a set of nested behaviours) for an agent, RG/BDFS-lite has the ***rgbdfs:hasBehaviour*** property (instance of the rgbdfs:B_Property class). This property defines a set of behavioural statements for a resource agent via a behavioural container (see Figure 5).

Another important part of behavioural structuring is an agent role (see Figure 6). The ***rgbdfs:hasRole*** property defines a role (***rgbdfs:Role***) for a resource agent in certain context. ***rgbdfs:goals*** is another property related to an agent role and which defines a goal or a set of the goals that correspond to the subject role via a goal container.

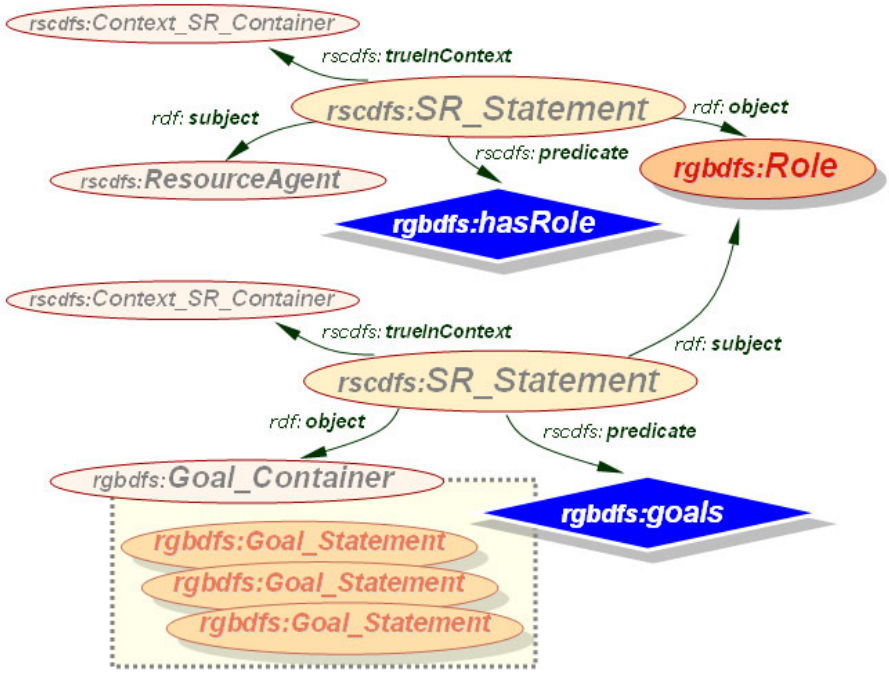

Figure 6. RGBDF Role

As it was mentioned previously, resource agent can have different roles, while a set of the goals can be different even for the same role depending on the context (environmental condition). Thus, with a purpose of having a possibility to define a context for them, these two properties are instances of the rscdfs:SR_Property class.

The presented Resource Goal Behaviour Description Framework is fully compliant with the BDI (Belief-Desire-Intention) model well-known in the scientific world of Multi-Agent Systems [16]. By now, a quite big amount of research results is available in the domain of agents based on BDI model. Recent results report significant development of modelling frameworks for BDI agents [17] and different logic programming languages for it, such as AgentSpeak [18]. BDI model has been actively developed towards cooperative behaviour of agents [19], particularly with a purpose of exchanging executive plans [20]. Figure 7 explains a parallel between BDI and RGBDF models.

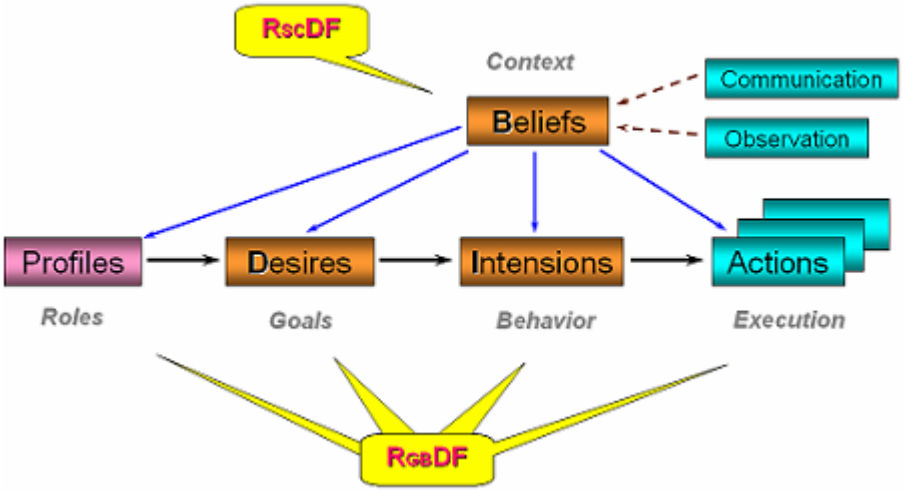

Figure 7. BDI: Underlying Model for RGBDF

2.3 Definition and configuration of resource's proactivity/behavior

Let us consider a case of agent behaviour. It will be a simple case of a device diagnostics performed by a web service. Actually, we have two smart resources: conventional resources (field device and web service) supplied with the agents that maintain them.

Agent, which represents a field device, plays a role of a patient that wants to take care (to know its own condition/diagnosis) of itself in case if certain alarm happens. Thus, the goal of this agent is to get a statement about a diagnosis from a diagnostic unit (in our case diagnostic web service) based on sub history of device states, if an emergency statement appears. It is a complex goal and contains nested sub goals. Agent has to send a diagnostic request to a web service that requires initial collecting of the set of device states and searching appropriate web service. After the request has been sent, the agent must get corresponding response with a statement about diagnosis from the web service. On the other hand, we have a web service agent, which plays a role of a therapist (diagnostic unit). The goal of this agent is to diagnose based on sub histories of device states. It is also a complex goal, which assumes receiving a diagnostic request, diagnosing and sending a response back to the field device agent.

As it was mentioned before, ontology contains templates of roles, goals and behaviours. Figure 8 represents two templates of roles and templates of goals that correspond to them.

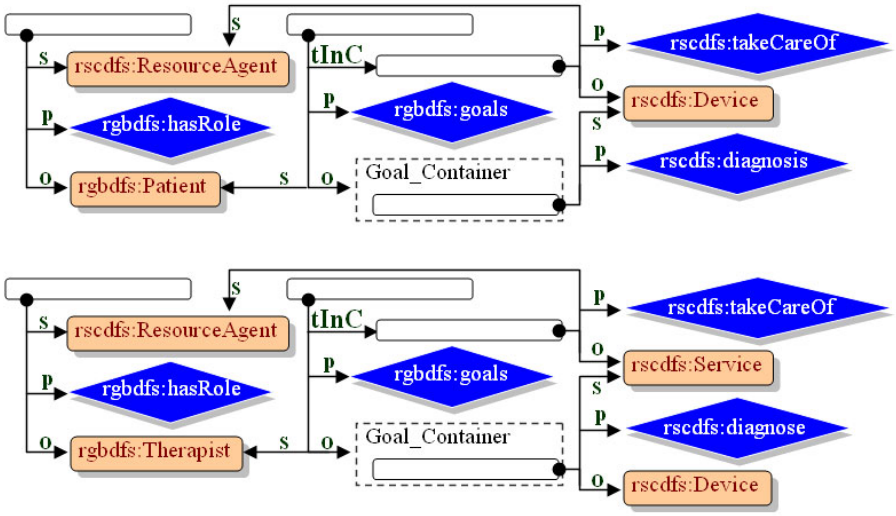

Figure8. Role and correspondent Goal templates

Further, we will concentrate on an example of the first mentioned agent, which maintains a field device. As you can see, the agent, which has a role of a patient, has a goal to get a diagnostic statement about certain device in a context that the agent takes care of this device. However, it is not an atomic goal, and therefore it has a set of nested sub goals. Next figure demonstrates, how nested sub goals can be described in an ontology (Figure 9.).

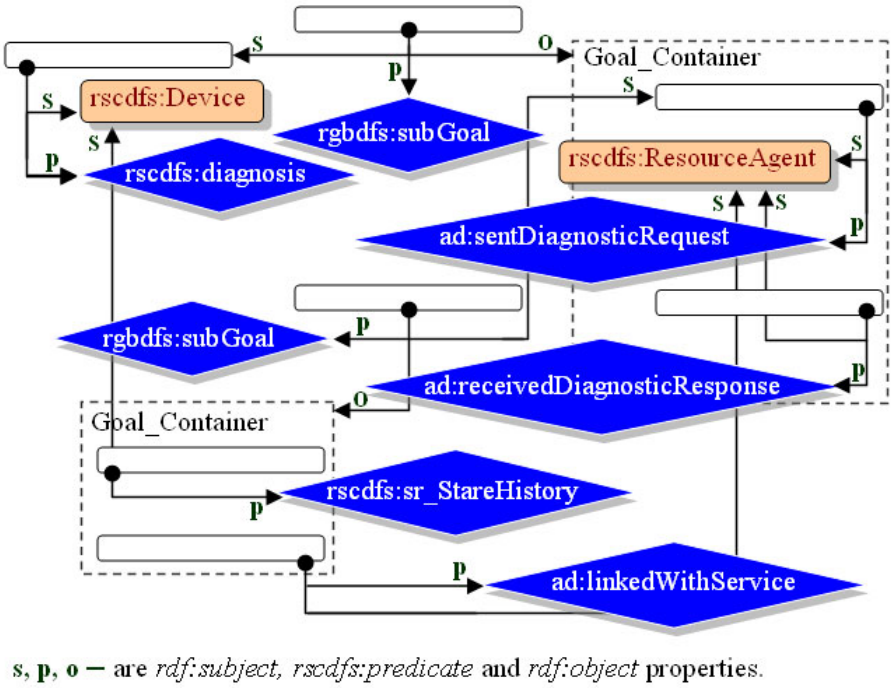

Figure 9. Nested Goal representation

In a similar way template of agent behaviour can be described via rgbdfs:Behaviour_Statement. Such statement links certain statement about execution (which defines executive module for a certain action through rdf:object property using rgbdfs:Behaviour_Container) with a goal (described through the rscdfs:falseInContext property) and with a context (which specifies a condition, when the action has to be performed through the rscdfs:trueInContext property).

Thus, let us consider a process of goal and behavior specification for an agent. At the initial stage an expert, which performs linking of the agent to a resource (field device), has to specify from an ontology certain role or a goal or even a set of them for current agent. If certain role was specified, then a set of corresponding goals or one goal can be retrieved automatically from the ontology. Then, based on goals corresponding to the agent, appropriate behavioural templates also can be retrieved from the ontology. After all necessary templates have been collected, their corresponding instances (with links to concrete instances of the resource agent, resources, etc.) have to be put to the agent storage on the resource platform. Depending on a complexity of the goals, an engine of an agent shell (see the example in Figure 10 and Figure 11) will compose nested hierarchy of agent behavioural rules automatically.

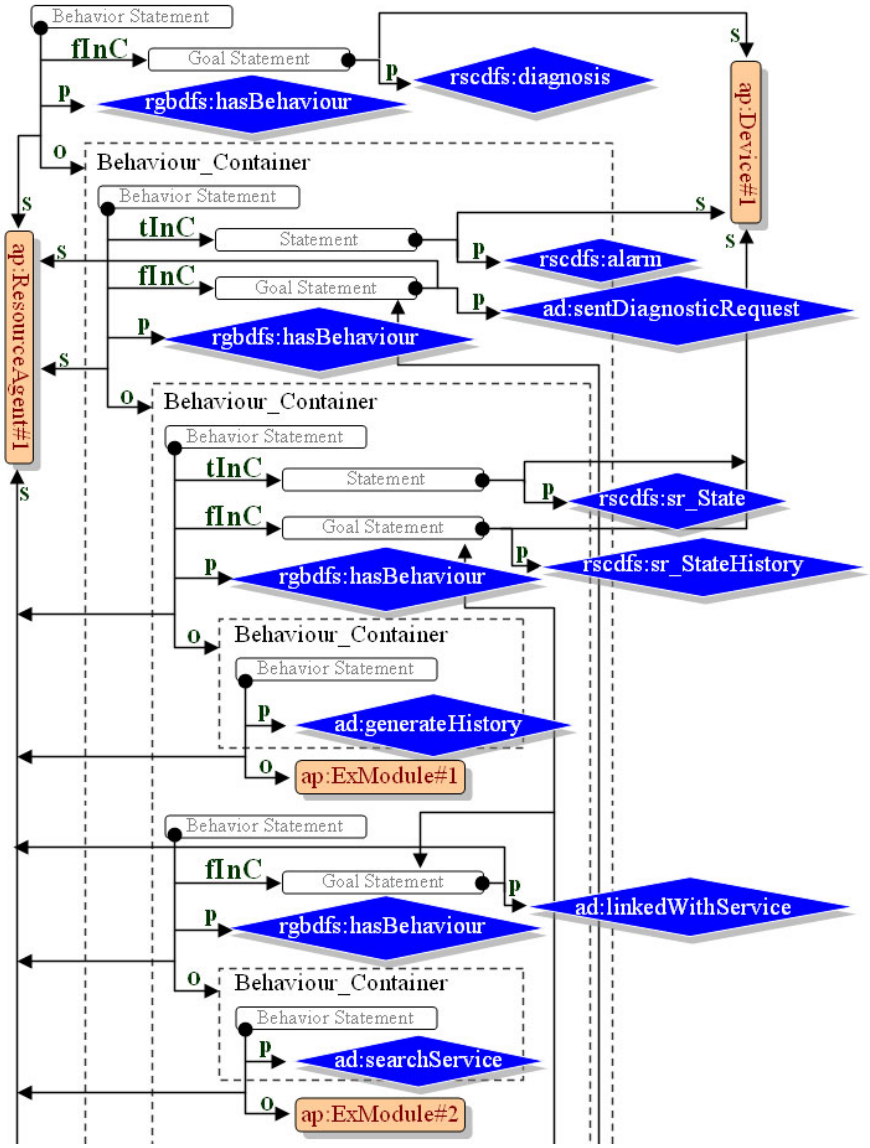

Figure 10. Nested hierarchy of agent behaviour rules

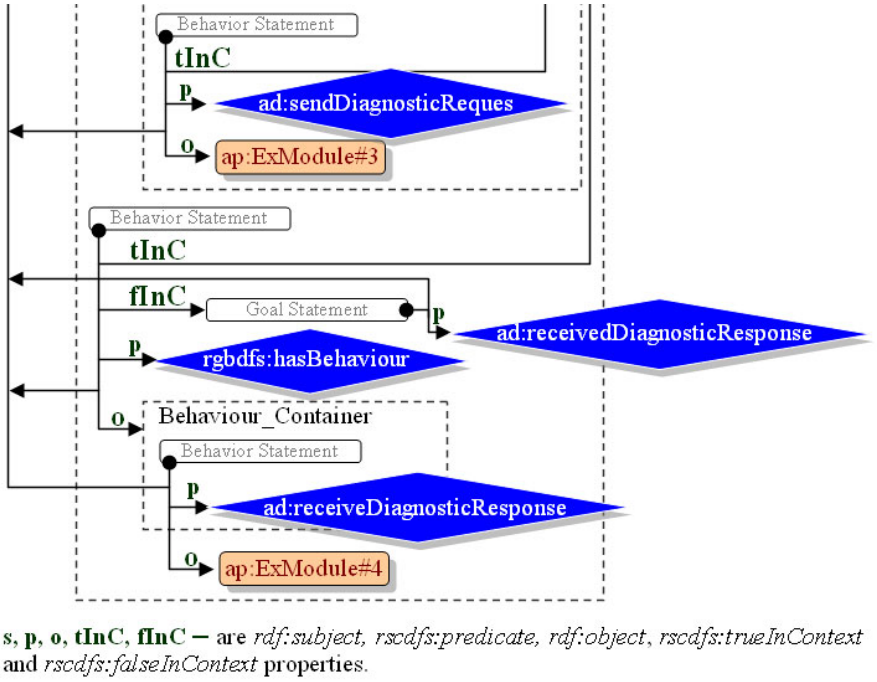

Figure 11. Nested hierarchy of agent behaviour rules (continuation)

Now, when the rules of agent behavior have been specified, it is a time to run agent engine for behaviour. Working space (storage) of SmartResource Platform (combination of a device and an agent, which maintains it) should be divided into two parts: a temporal storage and a long term one. Initially all information (all statements that concern the states and conditions of resources) are saved to the temporal storage and play a role of a behavioral context and input data for executive modules. As it was mentioned before, the goal statements (linked via the rscdfs:falseInContext property) play a role of a trigger to run certain behavioural rule. If there is no a statement in the temporal storage similar to the goal statement, then the agent engine executes a rule. Let us consider our example. Agent starts to perform root behaviour all the time, when a statement about device diagnosis does not exist in the temporal storage. Otherwise engine skips behavioral rule and passes to the next one on the same level of nesting. Each level can contain both types of behavioural statements: complex behavioural statements and atomic execution statements (which specify executive module via an instance of the rgbdfs:Execution class). In fact, these executive modules generate (add to the temporal storage a statement, which is required by the goal of the behavioural rule). For example, executive module, which is described by the instance *ap:ExModule#3*, generates a statement which asserts that the agent (ap:ResourceAgent#1) has sent a diagnostic request to certain diagnostic service. Thus, agent has achieved one of the sub goals. However, two sub goals (collecting of a history of the devise states and retrieval of suitable service for the diagnostics) have been achieved before, because they were needed for performing the executive module #3.When the agent has achieved the upper goal, the statement of the achieved goal is moved to the long term storage to be kept in the history of the SmartResource. At the same time all statements of the achieved nested goals have to be removed, too. Some of the contextual statements, which played a role of input data also must be removed (for example, the statement about an alarm, which plays a role of a context for a process pf sending a diagnostic request; and statements about the states of the device, which were used for the diagnostics).

3. RGBDF-Schema and Goal/Behaviour Ontology

RGBDF-Schema and goal/behaviour ontology are given in Appendix A.

4. Conclusions

As it has been presented, ontology-driven approach in modeling agent behavior is anticipated to become a powerful solution providing more benefits than conventional model-driven approaches. RGBDF that was described in this document has been designed within the second stage of the SmartResource project (Proactivity Stage). Resource Goal/Behavior Description Framework continues development of a modeling basis for the overall SmartResource platform. Further tools and use cases that will be developed within the Proactivity Stage based on RGBDF, will form a solid ground in favor of ontology-driven approach to modeling agent behavior.

5. References

[1] G. Laleci, Y. Kabak, A. Dogac, I. Cingil, S. Kirbas, A. Yildiz, S. Sinir, O. Ozdikis, O. Ozturk: A Platform for Agent Behavior Design and Multi Agent Orchestration. AOSE 2004: pp. 205-220.

[2] T. Pirttioja, I. Seilonen, P. Appelqvist, A. Halme, and K. Koskinen, Agent-based architecture for information handling in automation systems, 6th IFIP International Conference on Information Technology for Balanced Automation Systems in Manufacturing and Services (BASYS 2004), Vienna, Austria, September, 27-29, 2004.

[3] Stollberg, M.; Keller, U.; Zugmann, P.; Herzog, R., Semantic Web Fred - Agent Cooperation on the Semantic Web, 3rd International Semantic Web Conference, Hiroshima, Japan, 7 - 11 November 2004.

[4] Specification of Java library for programming agent behaviors, http://jade.tilab.com/doc/api/jade/core/behaviours/Behaviour.html.

[5] L. Braubach, A. Pokahr, W. Lamersdorf, Jadex: A Short Overview, in: Main Conference Net.ObjectDays 2004, AgentExpo.

[6] M. Griss, S. Fonseca, D. Cowan and R. Kessler, SmartAgent: Extending the JADE agent behavior model, HPL 2002-18, Jan 2002 (AOSE workshop).

[7] I. Seilonen, K. Koskinen, T. Pirttioja, P. Appelqvist, and A. Halme, Agent-based approach to enhanced flexibility in process automation systems, 3rd International Symposium on Open Control Systems, Helsinki, Finland, September 9-10, 2003.

[8] Zhao, L. and Mehandjiev, N. and Macaulay, L. (2004) *Agent Roles and patterns for supporting Dynamic behavior of Web Service Applications.* In: Proceedings of AAMAS04 Workshop on Web Services and Agent-Based Engineering (WSABE).

[9] A. Chella, M. Cossentino, and L. Sabatucci. Tools and patterns in designing multi-agent systems with passi. In *6th WSEAS Int.Conf. on Telecommunications and Informatics (TELE-INFO 04)*, Cancun, Mexico, May 2004.

[10] Djuric, D., Gaševic, D., Damjanovic, V., "AIR A Platform for Intelligent Systems," In Proceedings of the 1st IFIP Conference on Artificial Intelligence Applications and Innovations, Toulouse, France, August 2004.

[11] Viviane Torres da Silva, Ricardo Choren, Carlos José Pereira de Lucena: A UML Based Approach for Modeling and Implementing Multi-Agent Systems. AAMAS 2004: 914-921.

[12] R. Cervenka, I. Trencansky, M. Calisti, D. Greenwood. AML: Agent Modeling Language. Toward Industry-Grade Agent-Based Modeling. Lecture Notes in Computer Science, Springer-Verlag, Volume 3382/2005: Agent-Oriented Software Engineering V: 5th International Workshop, AOSE 2004, New York, NY, USA, July 19, 2004. Revised Selected Papers.

[13] Dastani, M., Jonker, C.M., and Treur, J., A Requirement Specification Language for Configuration Dynamics of Multi-Agent Systems. *International Journal of Intelligent Systems*, vol. 19, 2004, pp. 277-300.

[14] Naumenko A., Nikitin S., Terziyan V., Zharko A., Strategic Industrial Alliances in Paper Industry: XML- vs. Ontology-Based Integration Platforms, In: The Learning Organization, An International Journal, Special Issue on Ubiquitous Business Intelligence, Emerald Publishers, ISSN: 0969-6474, 16 pp., (to appear).

[15] Mazzocchi S., It's All about Graphs, White Paper, February 11, 2004, available at: http://www.betaversion.org/~stefano/linotype/news/46/, last accessed 17th April 2005.

[16] Rao, Anand S. and Georgeff, Michael P., BDI Agents: From Theory to Practice, in Proc. of the 1st International Conference on Multi-Agent Systems (ICMAS-95), San Francisco, USA, June, 1995.

[17] Mascardi, V., M. Martelli, F. Zini, and R. Degl'Innocenti: 'CaseLP: a Prototyping Environment for Heterogeneous Multi-Agent Systems'. Journal of Autonomous Agents and Multi-Agent Systems, 2004.

[18] Rao, A. S. 1996. AgentSpeak(L): BDI Agents Speak Out in a Logical Computable Language. In Proc. MAAMAW-96. Springer Verlag, LNAI 1038, pp. 42–55.

[19] Ancona D. and Mascardi V., Coo-BDI: Extending the BDI Model with Cooperativity, In Proc. of the First Declarative Agent Languages and Technologies Workshop (DALT'03), J. A. Leite, A. Omicini, L. Sterling and P. Torroni (Eds.), Springer Verlag, LNAI 2990, 2004, pp. 109-134.

[20] Ancona D., Mascardi V., Hübner J. F., Bordini R. H., Coo-AgentSpeak: Cooperation in AgentSpeak through Plan Exchange, In Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2004), N. R. Jennings and C. Sierra and L. Sonenberg and M. Tambe (Eds.), pp. 698-705, ACM press, 2004.

Appendix A: RGBDF-Schema and Goal/Behaviour Ontology

RGBDF Schema as RDF/XML

```
<?xml version='1.0' ?>

<!DOCTYPE rdf:RDF [
        <!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
        <!ENTITY rdfs 'http://www.w3.org/2000/01/rdf-schema#'>
        <!ENTITY rscdfs 'http://www.cc.jyu.fi/~olkhriye/rscdfs/0.3/rscdfs#'>
        <!ENTITY rgbdfs 'http://www.cc.jyu.fi/~olkhriye/rgbdfs/0.1/rgbdfs#'>
]>

<rdf:RDF
    xmlns:rdf="&rdf;"
    xmlns:rdfs="&rdfs;"
    xmlns:rscdfs="&rscdfs;"
    xmlns:rgbdfs="&rgbdfs;"
>
<rdfs:Class rdf:about="&rgbdfs;Role">
      <rdfs:comment></rdfs:comment>
      <rdfs:label>Role</rdfs:label>
      <rdfs:subClassOf rdf:resource="&rdfs;Class"/>
</rdfs:Class>

<rdfs:Class rdf:about="&rgbdfs;Execution">
      <rdfs:comment></rdfs:comment>
      <rdfs:label>Execution</rdfs:label>
      <rdfs:subClassOf rdf:resource="&rdfs;Class"/>
</rdfs:Class>

<rdfs:Class rdf:about="&rgbdfs;Goal_Statement">
      <rdfs:comment></rdfs:comment>
      <rdfs:label>Goal_Statement</rdfs:label>
      <rdfs:subClassOf rdf:resource="&rscdfs;SR_Statement"/>
</rdfs:Class>

<rdfs:Class rdf:about="&rgbdfs;Behaviour_Statement">
      <rdfs:comment></rdfs:comment>
      <rdfs:label>Behaviour_Statement</rdfs:label>
      <rdfs:subClassOf rdf:resource="&rscdfs;SR_Statement"/>
</rdfs:Class>

<rdfs:Class rdf:about="&rgbdfs;Goal_Container">
      <rdfs:comment></rdfs:comment>
      <rdfs:label>Goal_Container</rdfs:label>
      <rdfs:subClassOf rdf:resource="&rscdfs;Context_SR_Container"/>
</rdfs:Class>

<rdfs:Class rdf:about="&rgbdfs;Behaviour_Container">
      <rdfs:comment></rdfs:comment>
      <rdfs:label>Behaviour_Container</rdfs:label>
      <rdfs:subClassOf rdf:resource="&rscdfs;SR_Container"/>
</rdfs:Class>

<rgbdfs:B_Property rdf:about="&rgbdfs;B_Property">
      <rdfs:comment>Type of itself</rdfs:comment>
      <rdfs:label>B_Property</rdfs:label>
      <rdfs:subClassOf rdf:resource="&rdfs;Class"/>
      <rdfs:subClassOf rdf:resource="&rscdfs;SR_Property"/>
```

```
</rgbdfs:B_Property>

<rdf:Property rdf:about="&rgbdfs;gMember">
      <rdfs:label>member</rdfs:label>
      <rdfs:domain rdf:resource="&rscdfs;Goal_Container"/>
      <rdfs:range rdf:resource="&rscdfs;Goal_Statement"/>
      <rdfs:subPropertyOf rdf:resource="&rscdfs;member"/>
</rdf:Property>

<rdf:Property rdf:about="&rgbdfs;bMember">
      <rdfs:label>member</rdfs:label>
      <rdfs:domain rdf:resource="&rscdfs;Behaviour_Container"/>
      <rdfs:range rdf:resource="&rscdfs;Behaviour_Statement"/>
      <rdfs:subPropertyOf rdf:resource="&rscdfs;member"/>
</rdf:Property>

<rdf:Property rdf:about="&rgbdfs;falseInContext">
      <rdfs:label>falseInContext</rdfs:label>
      <rdfs:domain rdf:resource="&rgbdfs;Behaviour_Statement"/>
      <rdfs:range rdf:resource="&rgbdfs;Goal_Container"/>
      <rdfs:subPropertyOf rdf:resource="&rscdfs;falseInContext"/>
</rdf:Property>

<rgbdfs:B_Property rdf:about="&rgbdfs;hasBehaviour">
      <rdfs:label>hasBehaviour</rdfs:label>
      <rdfs:domain rdf:resource="&rscdfs;ResourceAgent"/>
      <rdfs:range rdf:resource="&rgbdfs;Behaviour_Container"/>
</rgbdfs:B_Property>

<rgbdfs:B_Property rdf:about="&rgbdfs;execute">
      <rdfs:label>execute</rdfs:label>
      <rdfs:domain rdf:resource="&rscdfs;ResourceAgent"/>
      <rdfs:range rdf:resource="&rgbdfs;Execution"/>
</rgbdfs:B_Property>

<rscdfs:SR_Property rdf:about="&rgbdfs;subGoal">
      <rdfs:label>subGoal</rdfs:label>
      <rdfs:domain rdf:resource="&rgbdfs;Goal_Statement"/>
      <rdfs:range rdf:resource="&rgbdfs;Goal_Container"/>
</rscdfs:SR_Property>

<rscdfs:SR_Property rdf:about="&rgbdfs;hasRole">
      <rdfs:label>hasRole</rdfs:label>
      <rdfs:domain rdf:resource="&rscdfs;ResourceAgent"/>
      <rdfs:range rdf:resource="&rgbdfs;Role"/>
</rscdfs:SR_Property>

<rscdfs:SR_Property rdf:about="&rgbdfs;goals">
      <rdfs:label>goals</rdfs:label>
      <rdfs:domain rdf:resource="&rgbdfs;Role"/>
      <rdfs:range rdf:resource="&rgbdfs;Goal_Container"/>
</rscdfs:SR_Property>

<rdf:Property rdf:about="&rgbdfs;subject">
      <rdfs:label>subject</rdfs:label>
      <rdfs:domain rdf:resource="&rgbdfs;Behaviour_Statement"/>
      <rdfs:range rdf:resource="&rscdfs;ResourceAgent"/>
      <rdfs:subPropertyOf rdf:resource="&rdf;subject"/>
</rdf:Property>

<rdf:Property rdf:about="&rgbdfs;predicate">
```

```
      <rdfs:label>predicate</rdfs:label>
      <rdfs:domain rdf:resource="&rgbdfs;Behaviour_Statement"/>
      <rdfs:range rdf:resource="&rgbdfs;B_Property"/>
      <rdfs:subPropertyOf rdf:resource="&rscdfs;predicate"/>
</rdf:Property>

</rdf:RDF>
```

Goal/Behaviour Ontology

```
<?xml version='1.0' ?>

<!DOCTYPE rdf:RDF [
<!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
<!ENTITY rdfs 'http://www.w3.org/2000/01/rdf-schema#'>
<!ENTITY rscdfs 'http://www.cc.jyu.fi/~olkhriye/rscdfs/0.3/rscdfs#'>
<!ENTITY rgbdfs 'http://www.cc.jyu.fi/~olkhriye/rgbdfs/0.1/rgbdfs#'>
<!ENTITY gbOntology 'http://www.cc.jyu.fi/~olkhriye/rgbdfs/ontology/gbOntology#'>
]>

<rdf:RDF
    xmlns:rdf="&rdf;"
    xmlns:rdfs="&rdfs;"
    xmlns:rscdfs="&rscdfs;"
    xmlns:rgbdfs="&rgbdfs;"
    xmlns:gbOntology="&gbOntology;"
>

<gbOntology:Patient rdf:about="&gbOntology;Patient">
      <rdfs:comment></rdfs:comment>
      <rdfs:label>Patient</rdfs:label>
      <rdfs:subClassOf rdf:resource="&rgbdfs;Role"/>
</gbOntology:Patient>

<gbOntology:Therapist rdf:about="&gbOntology;Therapist">
      <rdfs:comment></rdfs:comment>
      <rdfs:label>Therapist</rdfs:label>
      <rdfs:subClassOf rdf:resource="&rgbdfs;Role"/>
</gbOntology:Therapist>

<rscdfs:SR_Property rdf:about="&gbOntology;sentDiagnosticRequest">
      <rdfs:label>sentDiagnosticRequest</rdfs:label>
      <rdfs:domain rdf:resource="&rscdfs;ResourceAgent"/>
      <rdfs:range rdf:resource="&rscdfs;DiagnosticRequest"/>
</rscdfs:SR_Property>

<rscdfs:SR_Property rdf:about="&gbOntology;receivedDiagnosticRequest">
      <rdfs:label>receivedDiagnosticRequest</rdfs:label>
      <rdfs:domain rdf:resource="&rscdfs;ResourceAgent"/>
      <rdfs:range rdf:resource="&rscdfs;DiagnosticRequest"/>
</rscdfs:SR_Property>

<rscdfs:SR_Property rdf:about="&gbOntology;receivedDiagnosticResponse">
      <rdfs:label>sentDiagnosticRequest</rdfs:label>
      <rdfs:domain rdf:resource="&rscdfs;ResourceAgent"/>
      <rdfs:range rdf:resource="&rscdfs;DiagnosticResponse"/>
</rscdfs:SR_Property>

<rscdfs:SR_Property rdf:about="&gbOntology;sentDiagnosticResponse">
      <rdfs:label>sentDiagnosticRequest</rdfs:label>
```

```
      <rdfs:domain rdf:resource="&rscdfs;ResourceAgent"/>
      <rdfs:range rdf:resource="&rscdfs;DiagnosticResponse"/>
</rscdfs:SR_Property>

<rscdfs:SR_Property rdf:about="&gbOntology;linkedWithServise">
      <rdfs:label>linkedWithServise</rdfs:label>
      <rdfs:domain rdf:resource="&rscdfs;ResourceAgent"/>
      <rdfs:range rdf:resource="&rscdfs;Service"/>
</rscdfs:SR_Property>

<rgbdfs:B_Property rdf:about="&gbOntology;generateHistory">
      <rdfs:label>generateHistory</rdfs:label>
      <rdfs:domain rdf:resource="&rscdfs;ResourceAgent"/>
      <rdfs:range rdf:resource="&rgbdfs;Execution"/>
      <rdfs:subPropertyOf rdf:resource="&rgbdfs;execute"/>
</rgbdfs:B_Property>

<rgbdfs:B_Property rdf:about="&gbOntology;searchService">
      <rdfs:label>searchService</rdfs:label>
      <rdfs:domain rdf:resource="&rscdfs;ResourceAgent"/>
      <rdfs:range rdf:resource="&rgbdfs;Execution"/>
      <rdfs:subPropertyOf rdf:resource="&rgbdfs;execute"/>
</rgbdfs:B_Property>

<rgbdfs:B_Property rdf:about="&gbOntology;sendDiagnosticRequest">
      <rdfs:label>sendDiagnosticRequest</rdfs:label>
      <rdfs:domain rdf:resource="&rscdfs;ResourceAgent"/>
      <rdfs:range rdf:resource="&rgbdfs;Execution"/>
      <rdfs:subPropertyOf rdf:resource="&rgbdfs;execute"/>
</rgbdfs:B_Property>

<rgbdfs:B_Property rdf:about="&gbOntology;receiveDiagnosticResponce">
      <rdfs:label>receiveDiagnosticResponce</rdfs:label>
      <rdfs:domain rdf:resource="&rscdfs;ResourceAgent"/>
      <rdfs:range rdf:resource="&rgbdfs;Execution"/>
      <rdfs:subPropertyOf rdf:resource="&rgbdfs;execute"/>
</rgbdfs:B_Property>

<rgbdfs:B_Property rdf:about="&gbOntology;sendDiagnosticResponce">
      <rdfs:label>sendDiagnosticResponce</rdfs:label>
      <rdfs:domain rdf:resource="&rscdfs;ResourceAgent"/>
      <rdfs:range rdf:resource="&rgbdfs;Execution"/>
      <rdfs:subPropertyOf rdf:resource="&rgbdfs;execute"/>
</rgbdfs:B_Property>

<rgbdfs:B_Property rdf:about="&gbOntology;receiveDiagnosticRequest">
      <rdfs:label>receiveDiagnosticRequest</rdfs:label>
      <rdfs:domain rdf:resource="&rscdfs;ResourceAgent"/>
      <rdfs:range rdf:resource="&rgbdfs;Execution"/>
      <rdfs:subPropertyOf rdf:resource="&rgbdfs;execute"/>
</rgbdfs:B_Property>

<rgbdfs:B_Property rdf:about="&gbOntology;makeDiagnostic">
      <rdfs:label>makeDiagnostic</rdfs:label>
      <rdfs:domain rdf:resource="&rscdfs;ResourceAgent"/>
      <rdfs:range rdf:resource="&rgbdfs;Execution"/>
      <rdfs:subPropertyOf rdf:resource="&rgbdfs;execute"/>
</rgbdfs:B_Property>
</rdf:RDF>
```