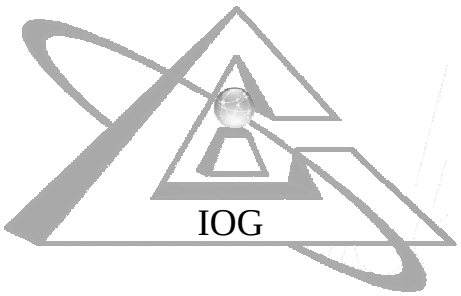


DESIGN OF THE SMARTRESOURCE PLATFORM

DELIVERABLE 2.2

Technical report

SmartResource: Proactive self-maintained resources in Semantic Web

10/11/2005

University of Jyväskylä

Agora Center

Author: Industrial Ontologies Group

Contact Information: e-mail: vagan@it.jyu.fi

Title: Design of the SmartResource Platform, Deliverable 2.2

Work: Technical report

Status of document: working draft

Number of Pages: 26

Abbreviations

RSCDF – Resource State/Condition Description Framework

RGBDF – Resource Goal/Behavior Description Framework

Contents

1 Introduction

Previous two works [RSCDF, RGBDF] present two frameworks as a logical evolution of the Resource Description Framework (RDF) for describing of proactive, context sensitive and goal-driven resources. There are Context Description Framework (CDF) and Resource Goal and Behaviour Description Framework (RG/BDF) correspondently.

One of the main features of the CDF is ability to describe context dependent facts (fact-statements) about resources. At the same time RG/BDF brings new (additional) vision in resource description. It is a description of a resource mental state. If we consider agent (software agent) as a resource in Global Understanding eNvironment (GUN) [GUN], then we face its believes and desires. Now we can describe not just statements that describe the facts, but also goals – statements, that describe wishful for resource (agent) state of environment, other resources and etc. Additionally RG/BDF gives us a possibility to describe the rules of resource behaviour with specifying the necessary and sufficient conditions, and rules of change for the environment.

Accordingly, now we directly come to behaviour annotation of resource (agent) and resource proactivity performance stages. Thus, we face two challenges: first of all we need handy and intelligence user interface for resource behaviour (rules) and resource mental states specifying, and then we need an engine to run these rules and perform actions.

Applying of new technologies should not to make life (business processes and etc.) complex, but vice versa should make it easy, flexible and scalable. It should be simple and attractive for users with a purpose to utilise such kind of Multi-Agent System. Of course we cannot equalize all users of the system.

On base of such platform of interactive proactive resources a lot of different processes can be modelled. It can be both complex processes (business processes, enterprise integration, distributed maintenance, distributed diagnostic and learning, supply chain management and etc.) and more primitive (personal agents' interaction, home devices interaction and etc.). Modelling of each process of those demands specific domain knowledge from system user (beginning from expert of a big corporation and ending with a housewife). But in the both cases user interface (whole module for interaction with user) should provide handy and intelligent functionality of the system.

2 Proactive Layer of the SmartResource Platform

Regarding to earlier works [OntoEnvironment], approach of heterogeneous resources integration in Semantic Web was presented. Accordingly to this approach, with a purpose to present resource (does not matter - it is a web resource or resource of the real world) as a SmartResource, we have to supply it with OntoShell. OntoShell – resource shell with two layers: Adaptation Layer and layer of resource proactivity (Proactivity Layer).

In this document we move accent to the resource proactivity, and give consideration of Resource Behaviour Engine. Before, we should define what kind of data we have as an input data for the engine.

2.1 Fact Statement

It is a Statement that describes facts of the whole system (environment) – states of the resources, their sub-histories and etc. RSCDF SR_Statement [RSCDF] (enhanced with context extension) fits for this purpose very well. For example, let's describe a state, that some device (Device #1) has some state description (State - description of the parameters values) at certain time (time of the Environment) (see Figure 1).

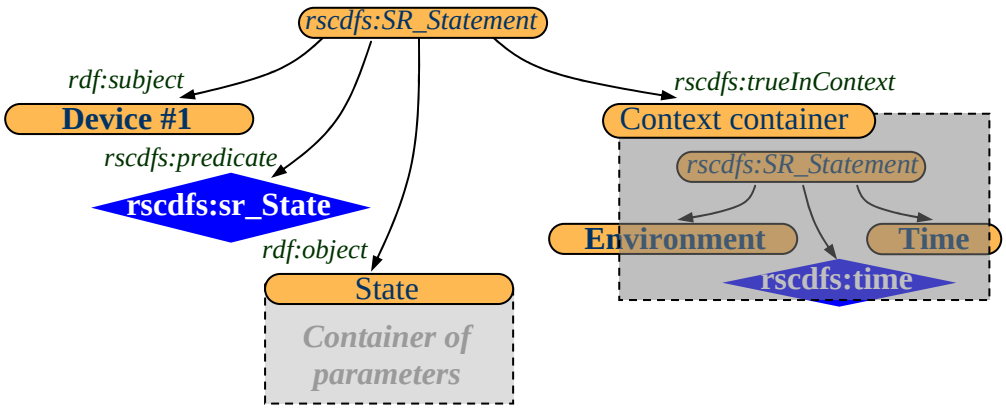


Figure 1 - Fact Statement

2.2 Non-Fact Statement (Mental Statement)

With this statement we have a possibility to describe not just a history as facts, but also describe mental states of a resource (accordingly to BDI (Beliefs-Desires-Intensions) Model). Regarding to [RGBDF] Goal_Statement describes goal statement which resource agent is aimed to make true. Let's further all Non-Fact Statements define through rgbdfs:NF_Statement (as a super-class of rgbdfs:Goal_Statement). As RSCDF Statement this NF_Statement also enhanced with context

extension, but takes on a value FALSE till the same RSCDF Statement appears as a Fact Statement in a resource (or environment) history. Then it takes on a value TRUE. Figure 2 shows us one of the NF Statements of SmartResource (its Agent). The goal is to have sent diagnostic request as a result of request sending action. Statement object is not defined because in this particular case it does not matter what diagnostic request is (it can be any diagnostic request).

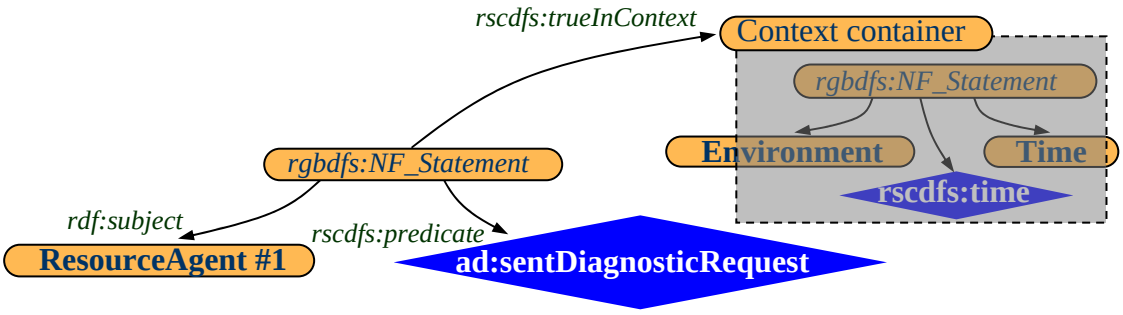


Figure 2 - Non-Fact (Mental) Statement

2.3 Rule Statement

One more Statement, which can be referred to Non-Fact Statements as a statement for rule description, is Rule Statement. This Statement allow us describe the rules of environment modification (changes in resources' states and descriptions). With this Statement we can define statement's truth dependence on other Statements (Non-Fact Statements), which play roles of necessary and sufficient conditions. Thus, the rule engine (Rule Engine), which follows these rules, can modify (add, delete and etc.) the content of the Environmental (contains Fact Statements) and Resource Mental (contains Non-Fact Statements) States Storages, and Rule (Behaviour) sets (see Figure 6). Sometimes, device can play not just role of Device (object of diagnostics), but also play role of diagnostic Service. During a long time, device can collect diagnoses of its states, and in future can provide diagnostic based on this labeled history. One of such Rule Statements can be, for example, rule that ResourceAgent should play role of diagnostic Service if it has received a diagnostic request, has been played role of Device and it is not playing role of diagnostic Service already (Figure 3). If preconditions will state that this Rule Statement is TRUE, then correspondent Fact Statement (that ResourceAgent plays role of diagnostic Service) will be added to the Environmental Storage.

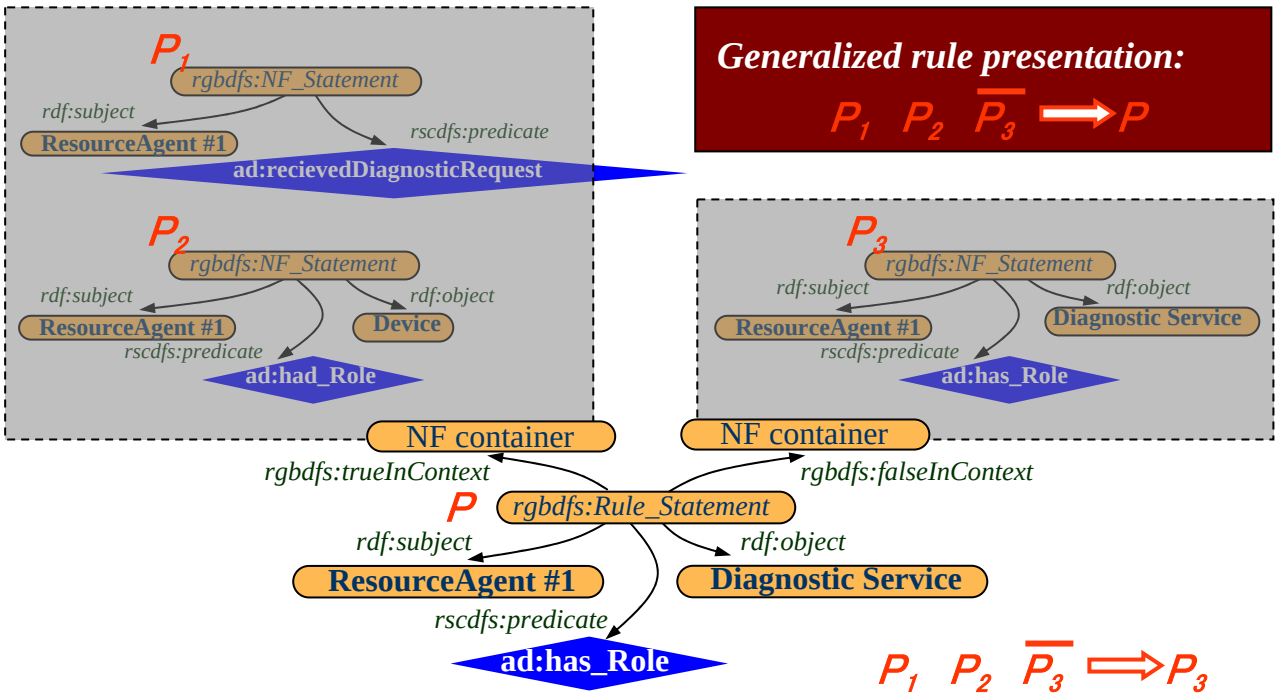


Figure 3 - Rule Statement

Such Rule Statement can be used for meta-rules definition as well. We can describe state of a rule via rgbdfs:ruleState_is property. The value for rule state is restricted by rgbdfs:Active and rgbdfs:Passive values. Thus, NF Statement (which defines state of a rule) can be activated (or deactivated) via Rule Statement (Figure 4) and can play role of a sufficient condition for the subject rule. Meta-rule definition gives us possibility to define context for rules and behaviours.

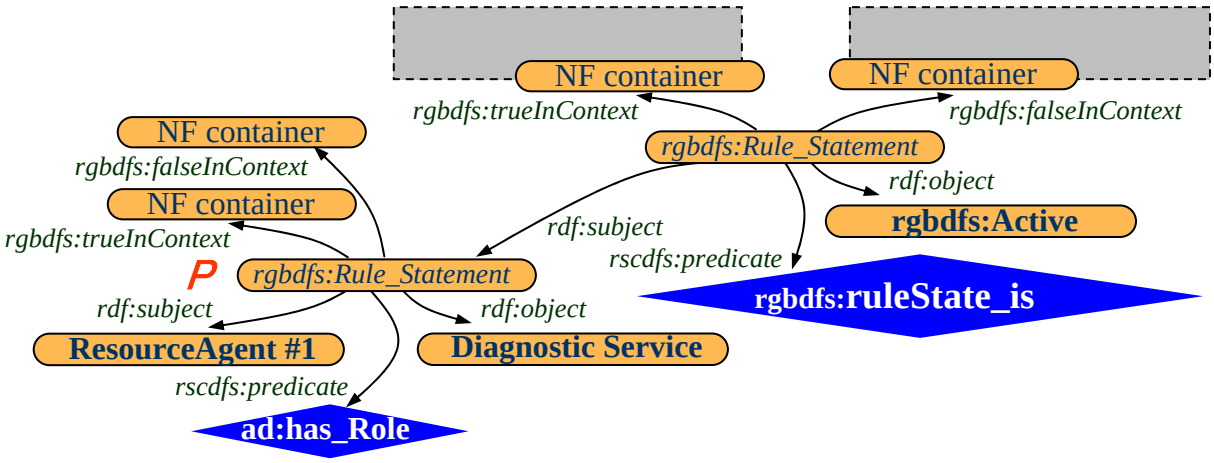


Figure 4 – Meta-Rule definition

Method of meta-rule execution all the time depends on Engine realization. It can be done in different ways. First way is when all rules are located in Engine Operational Rule Memory

(Storage) and contain NF Statements about their states (active or passive) as a sufficient condition. But in this case Operational Rule Memory will have huge amount of Rule Statements and it brings decreasing of rule engine performance. The next way is when platform has another rule engine for meta-rules tracking. Then all rules can be stored in some Rule Storage and will be added to the correspondent Engine Operational Rule Memory just when they are active (and removed from them otherwise).

2.4 Behaviour Statement

It is also Rule Statement (namely subclass of Rule Statement). It describes a rule of behavior performance (fragmentation to more simple behaviours (via rgbdfs:has_Behaviour property) or performance of concrete executable modules (Action) (via rgbdfs:execution property). Analogically with Rule Statement, activation of the Behavior Statement depends on Environmental and Resource Mental States. Let's consider one behaviour (Figure 5) which is aimed to send diagnostic request in case if the device has a alarm situation and has got alarm statement. This behaviour means performance of some set of sub-behaviours or execution of correspondent executable module. Here statement, that device has an alarm, plays role of sufficient condition for behavior performance (via rscdfs:trueInContext property, which plays role of IF-THEN rule). From other side, statement, which states – ResourceAgent has sent diagnostic request, plays role of necessary condition (via rgbdfs:falseInContext property, which plays role IF NOT-THEN rule). And again, the rule engine (Behavior Rule Engine), which follows the rules and performs actions, can modify (add, delete and etc.) the content of the Rule sets, Environmental and Resource Mental States Storages (see Figure 6). As a result of the behaviour (described in Figure 5) goal will be reached and Fact Statement, which states that ResourceAgent has sent diagnostic request, will be added to the Environmental Storage.

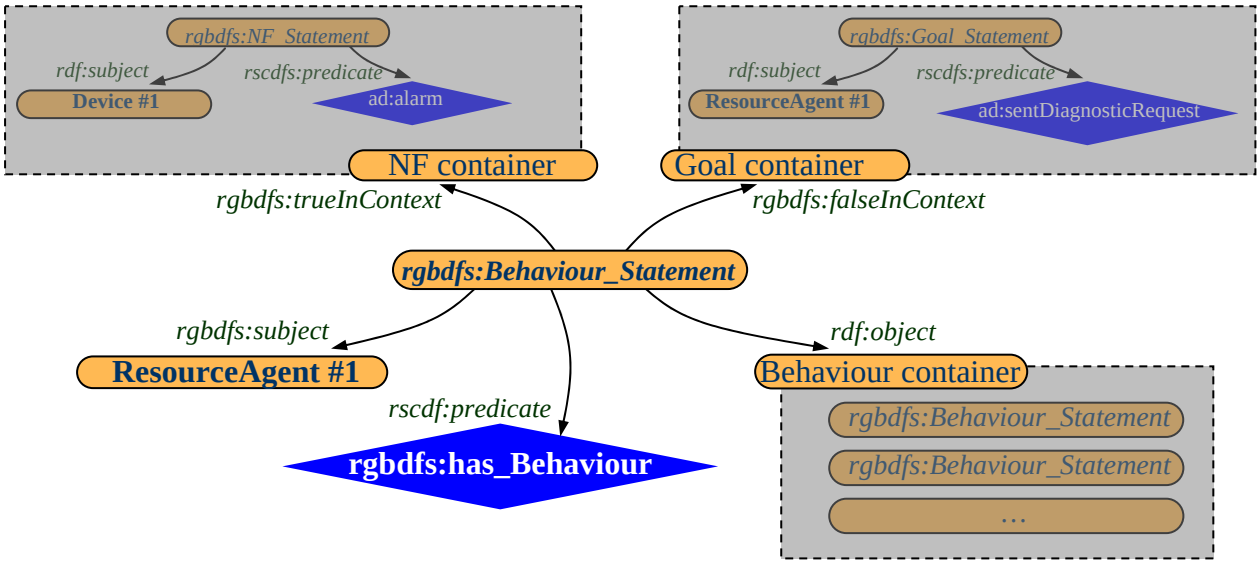


Figure 5 – Behaviour Statement

Thus, Figure 6 presents architecture of the proactive layer of the SmartResource Platform. Structure contains four storages: Environmental (contains Fact Statements) and Resource Non-Fact States (contains Non-Fact Statements and Rule Statements as well) Storages, storage where ontology and all instances (Resources: Devices, Services, Human Experts, Agents and etc.) are located, and storage of the programmable executable modules. Practically, Storage of the Fact Statements is presented by two storages: Operational Memory and Long Term History. Operational Memory contains updated relevant to performance data. For example if statement that ResourceAgent plays some new role goes to the Operational History, then statement about previous role should be removed to Long Term History; or irrelevant alarm statements should be removed. Such filter should not allow contradiction in operational data.

There are also two engines: Rule and Behaviour Engines, which iteratively check the rules, perform them and run actors (modules). As it has shown in example of Rule Statement description, mainly Rule Engine generates (changes) context for resource behaviour.

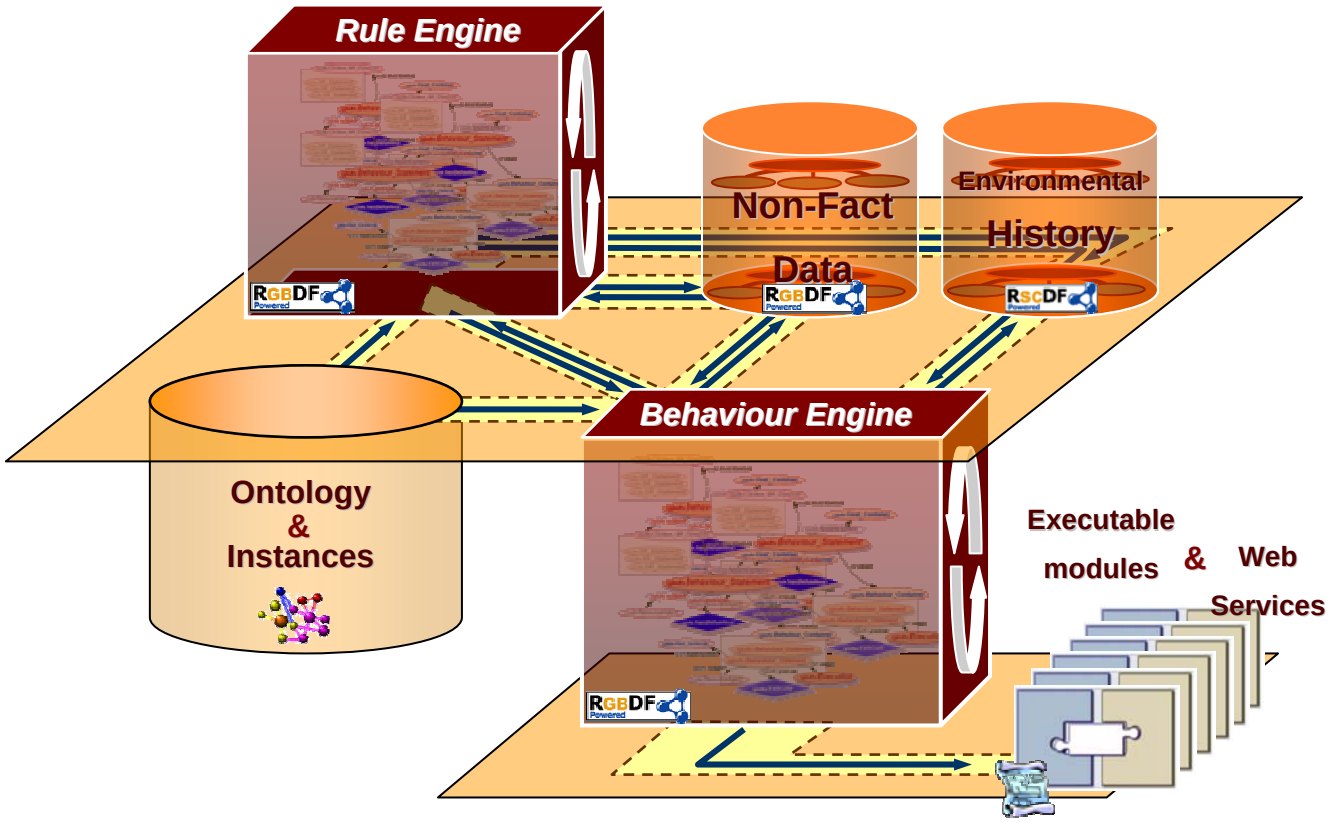


Figure 6 – Proactive Layer of the SmartResource Platform

From the system usability point of view, all these complex models of elements interaction and functionality of the engines should be hidden from end-user vie handy and intelligent SmartInterface. One of the necessary information that user should specify during ResourceAgent setting is a goal (Goal Statement that describe statement ResourceAgent aims which). At the same time user should specify input data (not necessary existing fact from the History (if there is not any statements yet), but specify template – statement without object). During all these activities, interface should provide user all available information from ontology and data, stored on the Platform: list of instances, list of intellectually filtered properties and etc. Thus, if there are semantic profiles of accessible executable modules and web services (with semantically annotated inputs and outputs), then "behaviour modelling module" on the Platform will generate behaviour rules automatically (will try to build process execution). Otherwise, we have a need to specify semantic profile for all available executable modules on the Platform and for web services that will be used. If there is no any executable module or web service which can exactly satisfy the goal, then goal can be divided to a set of sub-goal based on correspondent information in ontology or iterative process of automatic sub-goal generation (based on required inputs for modules, which can reach the goal but inputs are not provided). Thus, the goal will be reached by using set of interactive executable modules. Again, if such platform applies for new

infrastructure, it brings need to define not just rules of Agent behaviours but also rules of Environment (whole system) influence (rules of behaviour context). All these actions user should do in worst case, when we adapt platform for specific case (specific domain). But there is also easiest way to configure ResourceAgent, if it used for widespread, widely used and known process. This way is based just on Agent Role specification. It implies that ontology contains all relations between agent roles and goals with correspondent behaviour rules. But as you can see both these ways (especially second one) assume that huge amount of hard work has been done by ontology engineers beforehand, and ontology contains enough information (knowledge) to allow SmartInterface demand as les as possible from user.

3 Behaviour execution

Let's consider an example (described in [RGBDF]) and try to follow platform execution. It is 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 self-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. Nested hierarchy of agent behavioural rules presented in Appendix A.

These rules can be formalized via productions (Production System). Production systems are a very useful tool for modelling behaviour. They can model cognitive processes such as reasoning but the way that they do this is generally regarded to be different to the way that it occurs in humans. Hence the widespread use of production systems and other symbolic techniques for modelling. That is not to say that these types of models are not widely used still today. It is agreed, however, that the method underlying these models is different to the processes of the brain or mind. Production systems have three main components [McTear, 1988]. These are as follows: a Rule Base, a Working Memory, an Inference Engine or Interpreter.

Let's define all Non-Fact Statements as predicates [P_1, P_2, P_3, P_4, P_5, P_6, *and* P_7], sets of sub-behaviours via rule collections [B_0, B_2, B_3] and actions executable modules via actions [A_1, A_2, A_3, A_4] (see Appendix A).

***P*$_1$** – Device has a diagnosis (state about diagnosis);

***P*$_2$** – Device has some alarm statement;

P_3 – ResourceAgent has sent diagnostic request;

P_4 – Device has at least one state description;

P_5 – Device has formed sub-history of states;

P_6 – ResourceAgent has linked with appropriate service;

P_7 – ResourceAgent has received diagnostic response.

Now we can define simple Production System as a set of the rules accordingly to our example (Appendix A). Sub-behaviours are presented by sub-set of the rules and are performing by engine as a separate thread. Figure 7 shows us Production System of ResourceAgent (which represents a field device in device diagnostics case) behaviour.

Figure 7 – Production System of ResourceAgent behaviour

Actions A_1, A_2, A_3, A_4 (Figure 7) are internal actions, which are performed by subject ResourceAgent and affect Fact Statements (appearance of them in History Storage), which in their turn affect Non-Fact Statements' truth. At the same time, we have actions A_N *and* A_K – external actions. They are actions of other ResourceAgents, which also affect Non-Fact Statements through operation with Fact Statements.

A_1 – History generation based on existent device states. As a result new Fact Statement that Device has sub-history of states will be generated and located in Operational Memory. It makes correspondent Non-Fact Statement TRUE (P_5 is TRUE). At the same time it removes statements of the device states (which are collected to the history) from Operational Memory (then P_4 is FALSE);

A_2 – Search of relevant Diagnostic Service. In the end of the action, Fact Statement that ResourceAgent is linked with Diagnostic Service will be added to Operational Memory. It makes P_6 TRUE;

A_3 – Diagnostic request sending action. This action adds Fact Statement that ResourceAgent has sent diagnostic request (it makes P_3 TRUE), removes Statement that Device has sub-history and Statement about alarm state from the Operational Memory, because sent request has operated this issues (it makes P_5 and P_2 - FALSE). Finally this action remove statement about linking to the Service (P_6 is FALSE, it can be irrelevant link for the next aim);

A_4 – Receiving diagnostic response. It adds Fact Statement about Device diagnosis and makes P_1 TRUE;

A_N – External action, which is preformed by another ResourceAgent, adds new Fact Statement about alarm situation to the Operational Memory (that makes P_2 TRUE);

A_K – External action, which generates new Fact Statements about Device states. These statements are located in Operational Memory and it makes P_4 TRUE.

3.1 Utilization of other execution engines

In some cases, it is better to utilize other (well used for specific domain) execution engines. For example, in case of process performed by web services, it makes a sense to use Business Process Execution Language (BPEL) Workflow Engine [BPEL]. With this purpose we have a need to enhance the Platform with Transformation module that transforms RGBDF behavior description to BPEL description of a process. Advantage of RGBDF process representation is that web services can be described through semantic profiles instead of exact web service description. Thus, we make a step from individual WS binding to Semantic Scenarios Specification. It gives a possibility to select suitable web services from the available set, and just after that to make a transformation to BPEL scenario. Such approach brings possibility to share and utilize knowledge about process without dependence on available services (Figure 8).

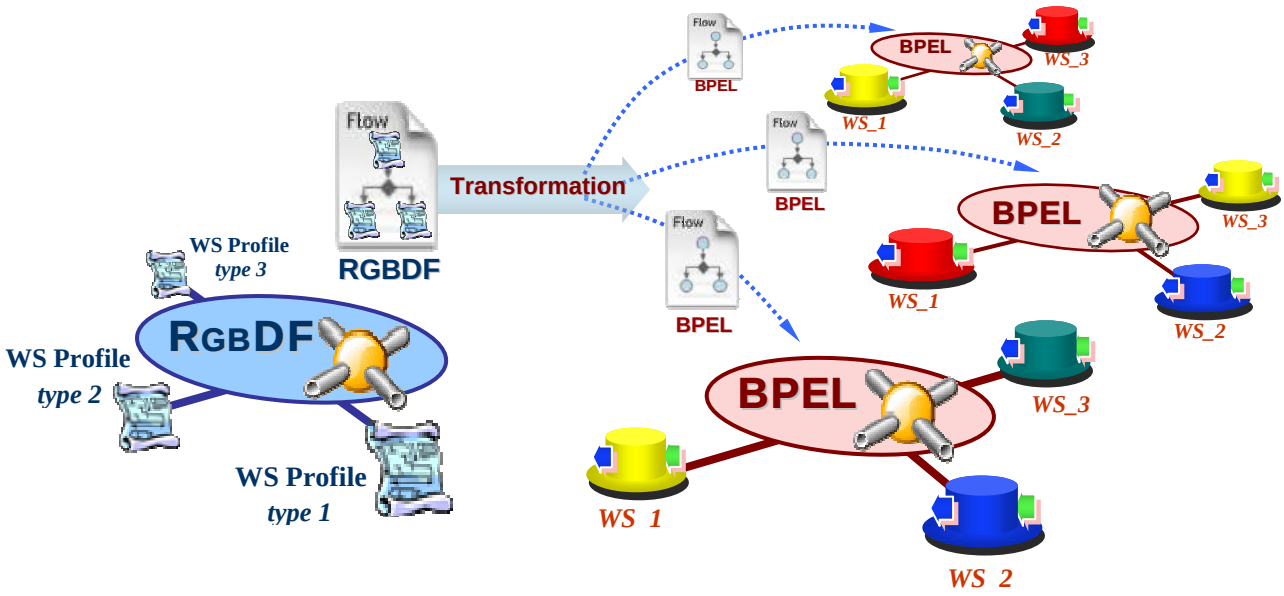


Figure 8 – Resource independent process knowledge sharing

Domain ontology provides common shared understanding of the domain representation. Each web service is semantically annotated according to this ontology by means of a WS Profile. Business Processes are modeled in a implementation-independent way (e.g. without hard binding to concrete service implementations) and can be stored in RGBDF or in another suitable process modeling language which allows for decoupling of business process logic from concrete activity implementation. Business process in this case represents rather logic of semantic data flow between semantically described service profiles. Real world web services can be considered as instances of the corresponding web service profiles (e.g. objects of a class in OOP). The flow enactment can be done dynamically by selecting Semantic Scenario Specification and automatically transforming it to ready-to-execute BPEL file. The transformation procedure lies in the selection of instances of appropriate web service profiles involved in a particular scenario (in MDA terms: Platform Independent to Platform Specific Model transformation).

4 Use Case: Instant Messaging

This section contains analysis of architecture of the Resource Agent Engine and requirements to it. The analysis is based on a use case of communication software applications (e.g. Instant Messaging) and related applications (e.g. Internet Calendaring and Scheduling) to be in the field of interest of TeliaSonera Oyj: one of the sponsors of the SmartResource project.

4.1 Use case description

The use case which will be described below does not pretend to be a basis for a competitive real-world application, but rather to reveal architectural features of the Resource Agent Engine and requirements to it.

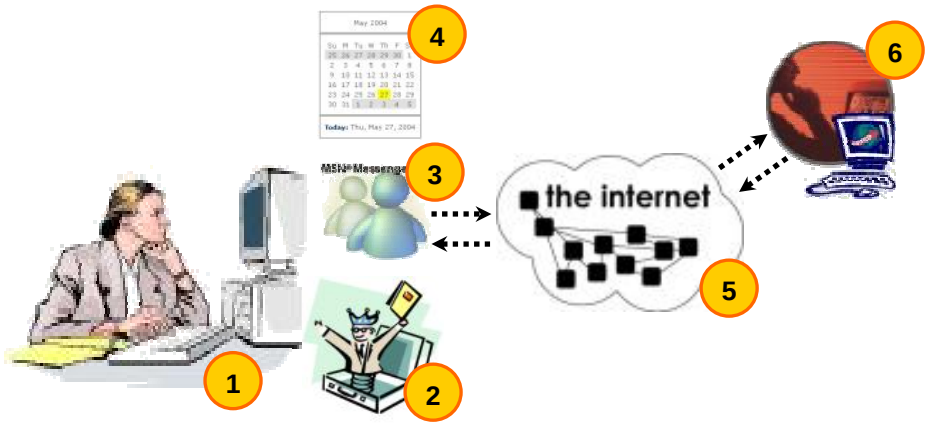


Figure 9 - Use case illustrated

The objects/actors that take part in the use case (in the Figure 9 the objects are labeled by numbers exactly as they are listed below):

1. *Workstation of a user*: conventional personal computer that is used nowadays by people without specialized software installed.

2. *SmartResource platform* installed on the workstation of the user. In this case we are mainly focused on Resource Agent Engine component of the platform. The purpose of the platform in this use case is to automate a working environment of the user.

3. *Instant Messaging Client*: software for instant messaging through Internet, e.g. Jabber[1], ICQ, AOL IM, Yahoo IM, MSN IM, Skype, etc. See comparative analysis of existing Instant Messaging systems in [Wang&Manoharan, 2004].

4. *Internet Calendaring and Scheduling software*, e.g. Outlook Calendar[2], Mozilla Calendar[3], etc.

5. *Internet environment or WWW*: global networked environment, which is used in our case for communication of the software installed on the workstation with remote applications (e.g. Web Services, e-mail servers, Instant Messaging Servers).

6. *Web Service*: remote application that can be accessed via WWW and which provides a service based on corresponding W3C standards[4]. In our case it is a service of text content analysis (natural language processing, data mining) against certain patterns (to capture, for instance, greetings coming from one person to another).

The proactivity, which is intended to be implemented by the *SmartResource platform* in the use case, will be the following. The goal of the Resource Agent will be to track a history of user's actions, in particular, his/her calendar and instant messaging events. As soon as the user will create in his/her calendar a new entity with birthday event of a person, the Resource Agent has to start collecting a history of instant messaging performed by the user and analyze, whether the user has sent greetings to that person. For the text analysis, external Web Service has to be used. If the user is suspected not to congratulate the concerned person and the deadline date is approaching (the date by default can be two days), the agent will remind user of the preparations necessary for the event based on the available preferences of the concerned person. If the user will be ready for making appropriate arrangements, the agent has to launch a wizard to assist that process and to automate routine tasks as much as possible.

4.2 Advanced use case

For greater maturity of the use case described above, the following features will be added and elaborated in it:

---

[1] http://www.jabber.org/

[2] http://office.microsoft.com/en-us/assistance/HA011959531033.aspx

[3] http://www.mozilla.org/projects/calendar/

[4] http://www.w3.org/2002/ws/

- The agent has to be able to prepare and execute basic transactions with the bank account of the user (e.g. similar to Solo Service of Nordea bank[5] or operations with credit card of the user). In real-world the account made accessible for the agent could have a limited sum available for better security. The basic transactions can be awareness about the balance of the bank account, supplementing the account, payments.

- Besides the analysis of the textual content of instant messaging of the user, the agent could analyze content of the e-mails from the user to the concerned person, his/her SMS flow, postcards sent (recipients), history of calls in a mobile (receivers of calls), history of electronic purchases (comparison of gifts bought with preferences of the concerned person).

- Actions of the agent in response to an agreement of the user to find and buy a gift for his/her friend can include provision of a catalog of products based on preferences of the friend, execution of search queries for the products chosen (to Google, Froogle, commercial electronic catalogs). As search parameters type of a celebration, age of the friend, gift preferences can be used. Preparation of a delivery (transactions with post services) can be then automated, too. Wizard that would help to compose accompanying greeting text and a mean for its communication to the target person is an option.

4.3 Use case from the perspective of the SmartResource approach

To get better understanding of the use case from the technological (architectural) perspective, we will use the SmartResource maintenance life-cycle [Terziyan, 2005], see Figure 10.

Given that SmartResource in our case is a human user along with his/her workstation, then the history will contain data based on interactions of the user with *Instant Messaging Client* and *Internet Calendaring and Scheduling software.* Let the latter be Jabber and Microsoft Outlook Calendar respectively. We can consider those two as suppliers of sensor data about actions of the user: what days were marked as important and what messages were sent. Therefore, to be used by Resource Agent, Jabber and Outlook Calendar have to be adapted.

[5] https://www1.nordea.fi/solo/3/emarketing/ng/netbank_demo.ASP

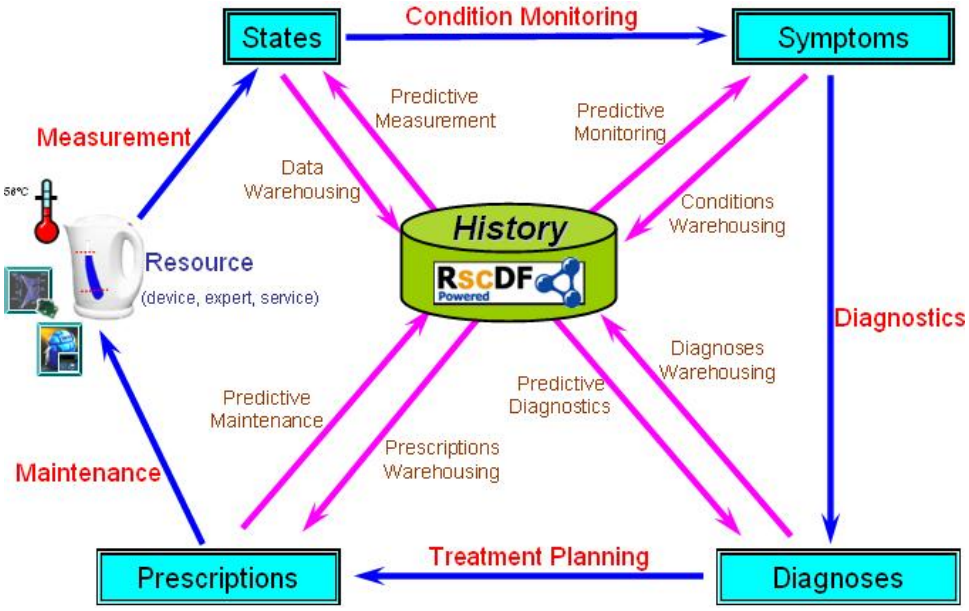


Figure 10 - Resource maintenance lifecycle in SmartResource

Now it is time to determine semantics that will be put into the history. As it was stated above in the description of the use case, "…as soon as the user will create in his/her calendar a new entity with birthday event of a person, the Resource Agent has to start collecting a history of instant messaging performed by the user". From the phrase we can infer that the process of "collecting the history of instant messaging" is a new behavior of the agent, which is activated only when the user creates a new entry in the calendar and that is the birthday entry with a person specified as having that birthday. At this point we can distinguish a complex behavior that assumes monitoring the appearance of a new birthday record in the e-Calendar, then collecting a history of the user's instant messaging with further analysis of the content using Web Service and preparing appropriate electronic transactions in response. For this complex behavior a separate RGBDF-role must be assigned, say "Manager of important dates". The specific steps or stages of this behavior described above can be included into the default release of the Resource Agent. The user will have a possibility to customize the behavior of the agent modifying the underlying instances of the RGBDF through handy User Interface (UI).

Considering the default behavior of the agent we can reveal some requirements to the SmartResource platform:

- Based on the RGBDF rules provided along with the SmartResource platform classes of sources of sensor data for the history can be known (e.g. e-Calendar, e-Scheduler, Instant Messenger, etc.). To enable the behavior of the agent, those sources have to be instantiated, i.e. instances of the classes have to be detected on

the workstation of the user (Jabber and Outlook Calendar installed) and appropriate adapters to them have to be uploaded and activated.

– Taking our use case, after the role “Manager of important dates” has been activated by the user (say, via launching a utility application) the SmartResource platform has to detect e-Calendars and Instant Messengers installed on the workstation of the user, e.g. by means of Windows System Registry. Then the platform must visualize detected software and request an approval of the user about their usage. From the technical side this procedure corresponds to the question “Can the detected software be treated as instances of the class e-Calendar/Instant Messenger”? In such dialogs with the user, the definitions of the classes used have to be provided in an unambiguous way, in order the user could understand the semantics behind them. If the platform was not able to detect some relevant software that the user utilizes, appropriate dialogs will make identification of other software as sensors possible.

For better understanding, the process of identifying the Instant Messengers to be monitored is illustrated in Figure 11. Standard Microsoft dialogs are taken as a basis. It would be reasonable to implement this functionality in a separate component to provide a possibility in its custom implementation by other vendors. In the Figure 11 red arrows indicate a sequence of the dialog with the user.

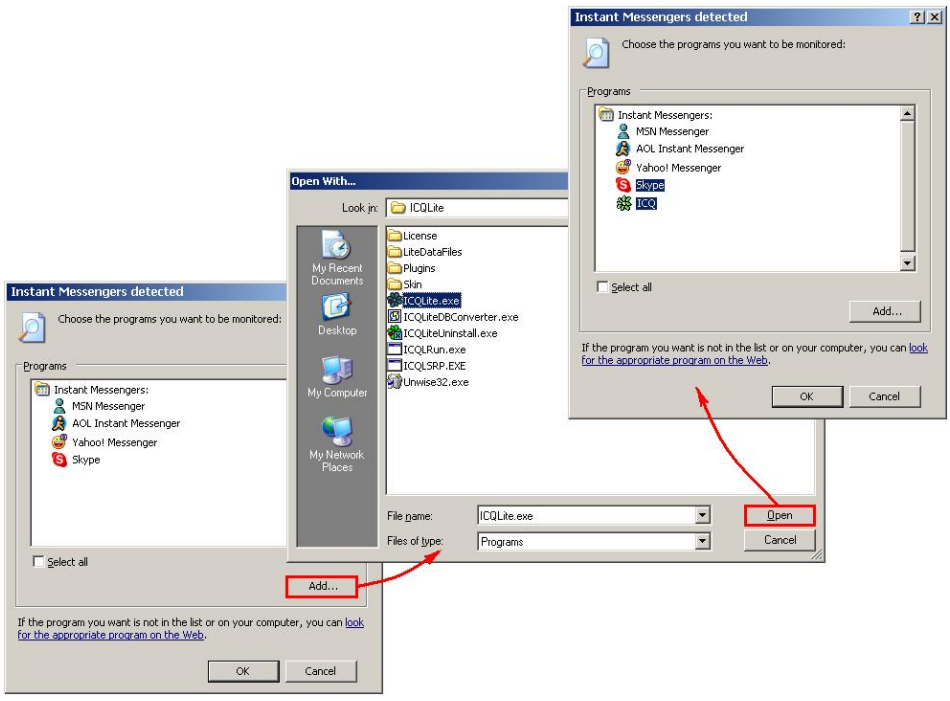


Figure 11 - Example of how a SmartResource platform could detect data sources

Further requirements are:

- As soon as the electronic calendars that will be monitored by the Resource Agent have been chosen by the user (we take Outlook Calendar as an example), the platform starts adapting those software. Based on the RGBDF description of the behavior of the agent, metadata that will be needed are identified. In case with Outlook Calendar the metadata comprises *type of an event*, *date of the event* and *person related to the event*.

- An adapter is not aware of the moment in time when the necessary metadata will be created by the user. Therefore, the implementation of the adapter must provide a mechanism of subscription for a retrieval of the necessary semantics.

5 Related works

There are some activities in Agent Behaviour area. One of those is an initiative of France Telecom Research & Development (FTR&D). They provided JADE Semantics Add-on as a framework based upon JADE [JADE] to interpret the meaning of the exchanged speech acts, according to their formal semantics as specified by FIPA-ACL; to make agents more flexible, in order to better interact in open environments; to simplify the coding of JADE agents. Figure 12 shows us a FTR&D's vision of a semantic agent.

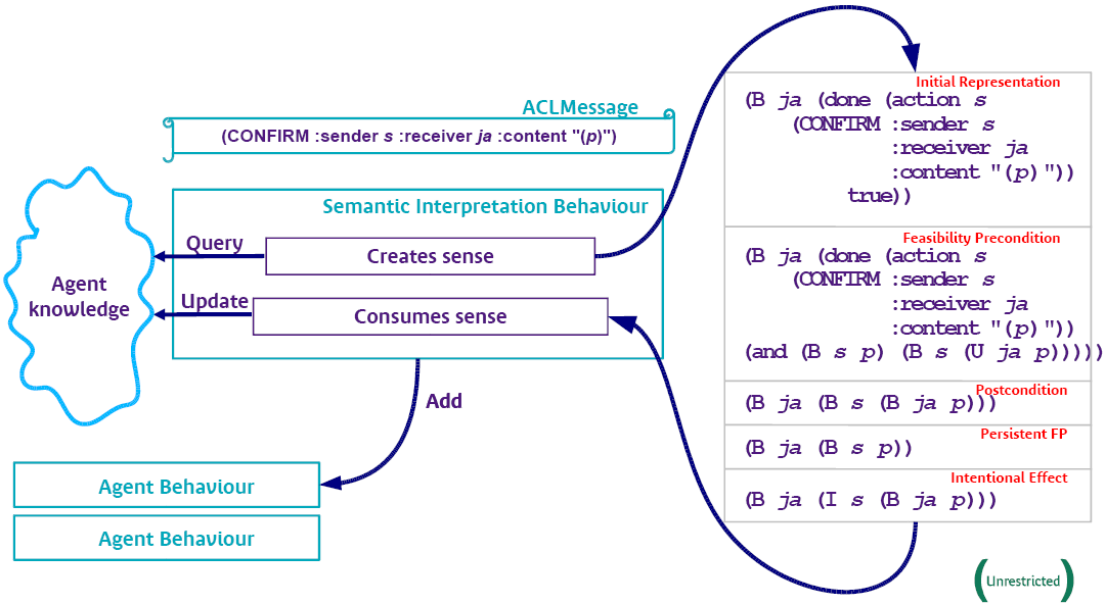


Figure 12 – JADE Semantic Agent (JSA)

6 References

[RSCDF] - Kaykova O., Khriyenko O., Naumenko A., Terziyan V., Zharko A., RSCDF: A Dynamic and Context Sensitive Metadata Description Framework , In: Eastern-European Journal of Enterprise Technologies, Vol. 3, No. 15, June 2005, ISSN: 1729-3774.

[RGBDF] - Kaykova O., Khriyenko O., Terziyan V., Zharko A., RGBDF: Resource Goal and Behaviour Description Framework, In: Proceedings of the 1st International IFIP/WG12.5 Working Conference on Industrial Applications of Semantic Web (IASW-2005) , August 25-27, 2005, Jyvaskyla, Finland, Springer, IFIP, pp. 83-99.

[GUN] - Kaykova O., Khriyenko O., Kovtun D., Naumenko A., Terziyan V., Zharko A., General Adaption Framework: Enabling Interoperability for Industrial Web Resources , In: International Journal on Semantic Web and Information Systems, Idea Group, ISSN: 1552-6283, Vol. 1, No. 3, July-September 2005, pp.31-63.

[OntoEnvironment] - Khriyenko O., Kononenko O., Terziyan V., "OntoEnvironment: an Integration Infrastructure for Distributed Heterogeneous Resources", In: "IASTED International Conference on Parallel and Distributed Computing and Networks (PDCN 2004)", Innsbruck, Austria, 17-19 February, 2004.

[McTear, 1988] - Turner, R., de Roeck, A., Understanding Cognitive Science, ed. Michael F. McTear, (University of Ulster at Jordanstown) Chichester: Ellis Horwood, 1988, 264 pp. (Ellis Horwood series in cognitive science) ISBN 0-7458-0161-7 (hb)

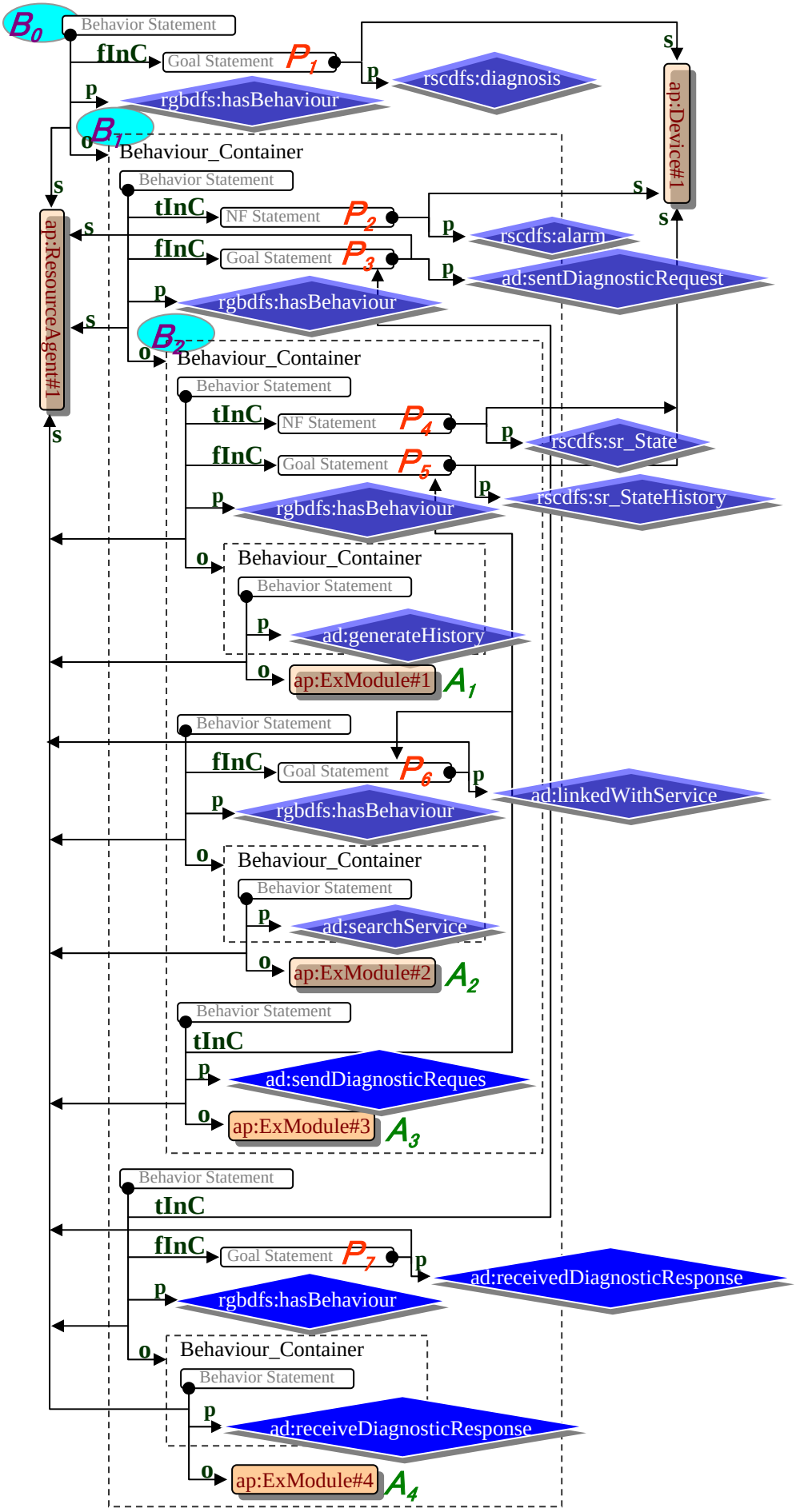
[BPEL] - Business Process Execution Language for Web Services, http://www-128.ibm.com/developerworks/library/specification/ws-bpel/

[JADE] – Java Agent DEvelopment Framework, http://jade.tilab.com/

[Wang&Manoharan, 2004] - X. Wang and S. Manoharan, "A Comparative Analysis of Instant Messaging", *IASTED Int. Conf. on Advances in Computer Science and Technology (ACST)*, S. Sahni (eds.), published by ACTA Press, Calgary, Canada, held in St. Thomas, Virgin Islands, USA, Nov., 2004.

[Terziyan, 2005] Terziyan V., SmartResource: Utilizing Semantic Web Services to Monitor Industrial Devices, In: A. Heimburger, T. Kauppinen, H. Lehtinen, J. Multisilta, M. Paajanen and K. Rytkonen (eds.): *XML - the Enabling Technology for Integrating Business Processes*, 2005, Tampere University of Technology, Pori, Finland, ISBN 952-15-1312-8, ISSN 1795-2166, pp. 89-103.

APPENDIX A - Nested hierarchy of agent behavioural rules [RGBDF]



s, p, o, tInC, fInC – are *rdf:subject, rscdfs:predicate, rdf:object, rgbdfs:trueInContext* and *rgbdfs:falseInContext* properties.