UBIWARE Deliverable D1.1:

The Central Principles and Tools of UBIWARE

November, 2007

Date	Nov 6, 2007
Document type	Report
Dissemination Level	UBIWARE project consortium
Contact Author	Vagan Terziyan
Co-Authors	Artem Katasonov, Olena Kaykova, Oleksiy Khriyenko, Oleksiy Loboda, Anton Naumenko, Sergiy Nikitin
Work component	WP1-WP5
Deliverable Code	D1.1
Deliverable Owner	IOG, JYU
Deliverable Status	Mandatory, Internal
Intellectual Property Rights	Unaffected

Abbreviations

A2A – Agent to Agent (communication)
A2H – Agent to Human (communication)
ACL – Agent Communication Language
API – Application programming interface
APL – Agent Programming Language
COIN – Configurability and Integration
FIPA – Federation for Intelligent Physical Agents
GUI – Graphical User Interface
GUN – Global Understanding eNvironment
HRAP – Human-Resource AdaPter
HTML – Hypertext Markup Language
IOG – Industrial Ontologies Group
OWL – Ontology Web Language
R2R – Resource to Resource (communication)
RDF – Resource Description Framework
S-APL – Semantic Agent Programming Language
SSA – Semantic Search Assistant/Facilitator
SURPAS – Smart Ubiquitous Resource Privacy and Security
UI – User Interface
W3C – World Wide Web Consortium
XHTML – Extensible HyperText Markup Language
XML – eXtensible Markup Language

Table of Contents

Introduction

Recent advances in networking, sensor and RFID technologies allow connecting physical world objects to the IT infrastructure, which could enable realization of the "Internet of Things" vision. However, as the systems become increasingly complex, traditional solutions to manage and control them reach their limits and pose a need for self-manageability. Also, the heterogeneity of components, standards, data formats, etc, creates significant obstacles for interoperability in such complex systems. Achieving the interoperability by imposing some rigid standards and making everyone comply would not lead to an open ubiquitous environment. Therefore, there is a need for some middleware to act as the glue joining heterogeneous components together. The promising technologies to tackle these problems are the Software Agents for management of complex systems, and the Semantic Web, for interoperability, including dynamic discovery, data integration, and inter-agent behavioral coordination.

The project aims at a new generation middleware platform (UBIWARE) which will allow creation of self-managed complex industrial systems consisting of distributed, heterogeneous, shared and reusable components of different nature, e.g. smart machines and devices, sensors, actuators, RFIDs, web-services, software components and applications, humans, etc. The middleware will enable various components to automatically discover each other and to configure a system with complex functionality based on the atomic functionalities of the components.

This project builds on the foundation laid in the SmartResource project (2004-2006). SmartResource analyzed the central concepts related to our vision, and resulted in some pilot tools and solutions. In turn, UBIWARE will result in a complete and self-sufficient middleware platform. In addition to treating the central issues related to flexible semantic agent-based integration and coordination of heterogeneous components, it develops appropriate solutions in supporting but mandatory areas such as security, human interfaces and other.

The research efforts are combined with agile software development processes. Software prototypes will be iteratively developed during the whole project lifecycle based on real data, real needs and changing requirements of industrial partners. The result will be both the basic software tools for the UBIWARE platform and several industrial prototypes based on these tools.

Work in this project is divided into seven work-packages which are running in parallel:

1. Core Distributed AI platform design (UbiCore)
2. Managing Distributed Resource Histories (UbiBlog)
3. Smart Ubiquitous Resource Privacy and Security (SURPAS)
4. Self-Management, Configurability and Integration (COIN)
5. Smart Interfaces: Context-aware GUI for Integrated Data (4i technology)
6. Middleware for Peer-to-Peer Discovery (MP2P)
7. Industrial cases and prototypes.

Work-packages 1 through 6 include both research and development tasks. The research results from the work-packages are to be reported through the integrating deliverables, one per project year. This deliverable D1.1 integrates the research results of the first project year.

It was decided not to perform the work is the WP6 during the first project year (due to limitation in resources). Therefore, D1.1 integrates results from WP1 through WP5. This deliverable includes a separate chapter for the results of every work-package involved. Due to the nature of topics addressed, D1.1 is titled “Central principles and tools of UBIWARE”. The two following research deliverables, D2.1 and D3.1, will address “Individual resources and inter-resource communication in UBIWARE” and “Multi-resource orchestration in UBIWARE”, correspondingly.

UBIWARE Deliverable D1.1:
Workpackage WP1:
Task T1.1_w1:

1 UbiCore – Core Distributed AI platform design

SEMANTIC REPRESENTATION OF BEHAVIOR MODELS

Workpackage leader: ***Artem Katasonov***

The main objectives of the UbiCore are the following. It has to give every resource a possibility to be smart (by connecting a software agent to it), in a sense that it would be able to proactively sense, monitor and control own state, communicate with other components, compose and utilize own and external experiences and functionality for self-diagnostics and self-maintenance. It has to enable the resources to automatically discover each other and to configure a system with complex functionality based on the atomic functionalities of the resources. It has to ensure a predictable and systematic operation of the components and the system as a whole by enforcing that the smart resources act as prescribed by their organizational roles and by maintaining the "global" ontological understanding among the resources. The latter means that a resource A can understand all of (1) the properties and the state of a resource B, (2) the potential and actual behaviors of B, and (3) the business processes in which A and B, and maybe other resources, are jointly involved

The part of the work in this work-package, reported in this document, aimed at answering the following research questions:

- How the language for roles' scripts, developed in SmartResource project, has to evolve to enable the full spectrum of possibilities that is found in Agent Programming Languages (APLs) – to become, in addition to other benefits, a Semantic APL?
- How to implement the separation between a role's capabilities (individual functionality), and the business processes in which this role can be involved (complex functionality)?

1.1 Agent Programming Languages (APLs)

Several *Agent Programming Languages (APLs)* has been developed by researchers working in internal architectures and approaches to implementation of software agents. Examples of such languages are *AGENT-0* (Shoham, 1993), *AgentSpeak(L)* (Rao, 1996), *3APL* (Dastani et al., 2004) and *ALPHA* (Collier et al., 2005).

All of those are declarative rule-based languages and are based on the first-order logic of n-ary predicates. All of them are also inspired by the Beliefs-Desires-Intentions architecture (Rao and Georgeff, 1991). For example, an agent program in ALPHA consists of declarations of the beliefs and goals of that agent and declaration of a set of rules, including belief rules (generating new beliefs based on existing ones), reactive rules (invoking some actions immediately) and commitment rules (adopting a commitment to invoke an action). Sensors (perceiving environment and generating new beliefs) and actuators (implementing the actions to be invoked) are then pieces of external code, namely in Java.

Based on the review of the above-mentioned languages, we list the following important features of them, which should also be realized in UBIWARE's language for specification of behavior models:

- Ability to specify *beliefs* (something that the agent believes to be true) and *goals* (something that the agent does not believe to be true but wants to eventually become true).
- Ability to describe *behavior rules*, i.e. actions taken when a certain condition is met (can be either presence or absence of certain beliefs, or presence of certain goals). Ability to have as the result of firing rule all of following:
  - adding/removing beliefs
  - engaging sensors and actuators
  - creating commitments (actions to be executed later)
- Ability to describe *plans*, i.e. predefined sequences of actions.
- Ability to describe *commitments* that are:
  - executed when certain condition is met
  - dropped when certain condition is met (or is not met anymore)

Agent-oriented approach postpones the transition from the domain concepts to the machine concepts until the stage of the design and implementation of individual agents. The advantage of using an APL is that the transition is postponed even further, until the implementation of particular perceptors and actuators. This advantage seems to be, however, the only one that is considered. We did not encounter in literature approaches that would extend the role of APL code beyond the development stage. APL code is assumed to be written by the developer of an agent and either compiled into an executable program or interpreted in run-time but remaining an agent's intrinsic and static property. APL code is not assumed to ever come from outside of the agent in run-time, neither shared with other agents in any way.

Such export and sharing of APL code would, however, probably make sense. Methodologies for design of agent-based systems like OMNI (Vazquez-Salceda, 2005) describe an organizational role with a set of rules, and an APL is a rule-based language. So,

using an APL for specifying a role sounds as a natural way to proceed. The difference is that APL code corresponding to a role should naturally be a property of and controlled by the organization, and accessed by the agents' enacting the role potentially even in the run-time. Run-time access would also enable the organization to update the role code if needed.

The second natural idea is that the agents may access a role's APL code not only in order to enact that role, but also in order to coordinate with the agents playing that role. As one option, an agent can send to another agent a part of its APL code to communicate its intentions with respect to future activities (so there is no need for a separate content language). As another option, if a role's code is made public inside the organization, the agents may access it in order to understand how to interact with, or what to expect from, an agent playing that role.

However, when thinking about using the existing APLs in such a manner, there are at least two issues:

- The code in an APL is, roughly speaking, a text. However in complex systems, a description of a role may need to include a huge number of rules and also a great number of beliefs representing the knowledge needed for playing the role. Also, in a case of access of the code by agents that are not going to enact this role, it is likely that they may wish to receive only a relevant part of it, not the whole thing. Therefore, a more efficient, e.g. a database-centric, solution is probably required.
- When APL code is provided by an organization to an agent, or shared between agents, mutual understanding of the meaning of the code is obviously required. While using first-order logic as the basis for an APL assures understanding of the semantics of the rules, the meaning of predicates used in those rules still needs to be consistently understood by all the parties involved. On the other hand, we are unaware of tools allowing unambiguous description of the precise semantics of n-ary predicates.

As a solution to these two issues, we see creating an APL based on theW3C's Resource Description Framework (RDF). RDF uses binary predicates only, i.e. triples (n-ary predicates can be represented nevertheless, of course, using several approaches). For RDF, tools are available for efficient database storage and querying, and also for explicit description of semantics, e.g. using OWL. Our proposition for such an RDF-based APL is the *Semantic Agent Programming Language (S-APL).*

1.2 Semantic Agent Programming Language (S-APL)

1.2.1 S-APL Axioms

- Everything is a *belief*. All other mental attitudes such as desires, goals, commitments, behavioral rules are just complex beliefs.
- Every belief is either a semantic statement (subject-predicate-object triple) or a linked set of such statements.
- Every belief has the *context* that restricts the scope of validity of that belief. Beliefs have any meaning *only inside* their respective contexts.

- Statements can be made about contexts, i.e. contexts may appear as subjects or/and objects of triples. Such statements give meaning to contexts. This also leads to a *hierarchy* of contexts (not necessarily a tree structure though).
- There is the *general context G*, which is the root of the hierarchy. G is the context for global beliefs (as opposed to local ones). Nevertheless, every local belief, through the hierarchical chain of its contexts, is linked to G.

1.2.2 S-APL Notation

The notation that is selected for use in S-APL is a subset of Notation3 (http://www.w3.org/DesignIssues/Notation3.html). This notation was developed Tim Berners-Lee (inventor of WWW, founder of W3C, and the one who coined the Semantic Web concept). Notation3 was proposed by Berners-Lee as an alternative to the dominant notation for RDF which is RDF/XML. Notation 3 is a language which is more compact and probably better readable than RDF/XML, and is also extended to allow greater expressiveness.

One feature of Notation3, which in a sense goes beyond the standard RDF, is the concept of *formula* that allow Notation3 graphs to be quoted within Notation3 graphs using { and }. There is no definition what is the precise semantics of formulae. In S-APL, however, we avoid the issue by fixing that a formula is a *Container that holds a set of reified statements*. Under this convention, S-APL documents remain compliant to the standard RDF data model and can be translated into RDF/XML is a need would arise (this of course would lead to a significant increase in the document length).

The description of S-APL notation follows:

- A statement is a white-space-separated sequence of subject, predicate and object
- Dot (**.**) followed by a white space separates statements of the same level, i.e. *S P O . S P O*
- Semicolon (**;**) followed by a white space allows making several statements about the same subject, i.e. *S P O ; P O*
- Comma (**,**) followed by a white space allows making several statements having common subject and predicate, i.e. *S P O , O*
- **{ }** denotes reification, it may appear as the subject or the object of a statement and has to include inside itself one or more other statements, e.g. *S P { S P O }* or *{ S P O } P { S P O }*. Reification always implies a context; however, the relation is not necessarily 1-to-1. E.g. *{S P O} P O ; P O* implies that the statement in {} is linked to two different contexts defined as given.
- Colon (**:**) is used to specify an URI as a combination of the namespace and the local name, i.e. *ns:localname* There can be default namespace, the colon is used anyway, i.e. *:localname*.
- *@prefix prefix: namespace* links a prefix to a namespace.
- URIs given directly are to be inside < >, i.e. <*http://someaddress*>.
- Literals containing whitespaces are to be inside **“ “**, i.e. *“some literal”* .
- Comments are java-style, i.e. /* */.

1.2.3 Descriptive constructs

Note: "gb" namespace is used for the resources that are defined in the language's ontology. The default namespace is used for all the other resources, which are assumed to be defined somewhere else.

Simple belief:

```
:John :Loves :Mary
{:John :Loves :Mary} gb:is gb:truth
```

These two are equivalent and the language interpreter may transform the latter into the former to simplify the agent engine's job. The latter is introduced for syntactic purposes, to allow linking a statement to both the current context and some its sub-context (see below about gb:existsWhile).

Belief with a context:

```
{:John :Loves :Mary} :accordingTo :Bill
```

Goal / desire:

```
gb:I gb:want {:John :Loves :Mary}
```

Unconditional commitment to an action:

```
{gb:I gb:do java:ubiware.shared.RequestSenderBehavior}
     gb:configuredAs
          {x:receiver gb:is :John .
           x:content gb:is "bla bla" .
           gb:Success gb:add {:John :was :notified}
          }
```

When the agent's engine finds such a belief in G, it executes the specified action and removes the commitment. This is done only if the prefix is either java: or default (:).

If action has no parameters, it is to be either `{gb:I gb:do :Stop} gb:configuredAs {}` or `{gb:I gb:do :Stop} gb:configuredAs gb:null`.

In the configuration part, one may use special statements to add or remove beliefs. The subject can be `gb:Start`, `gb:End`, `gb:Success`, and `gb:Fail`. The predicate is either `gb:add` or `gb:remove`. See more on how beliefs' removal exactly works, see below in "Unconditional commitment to removing a belief".

Sequential actions / plan:

```
{gb:I gb:do ...} gb:configuredAs
     { ... gb:Success gb:add {gb:I gb:do ...} }
```

Unconditional commitment to removing a belief:

```
gb:I gb:remove {:John :Loves :Mary}
```

When the agent's engine finds such a belief in G, it removes the specified belief and then removes the commitment itself.

The object of such a condition, presents a *pattern* that is used for the removing beliefs through matching it with G. The details of this procedure follow (all applies also to removal through "gb:Success gb:remove {...}"):

- If several beliefs are given, e.g. {:John :Loves :Mary. :Mary :Loves :John}, and some of those match G while some other do not, those matching ones are removed. In other words, this case is equivalent to specifying several separate commitments gb:I gb:remove {:John :Loves :Mary}. gb:I gb:remove {:Mary :Loves :John }.
- The commitment is considered fulfilled and thus it therefore removed even if no matching belief was found and thus nothing was actually removed.
- If a hierarchical belief structure is given, e.g. {:John :Loves :Mary. :John gb:want { {:Mary gb:do …} gb:configuredAs {…} } } :accordingTo :Bob, the following statements will be physically removed: :John :Loves :Mary and {} gb:configuredAs {}. In other words, the statements are removed which do not refer to contexts at all or which have contexts as both the subject and the object.
- One can use *, e.g. :John :Loves *. In result, all the matching beliefs like :John :Loves :Mary, :John :Loves :Grandpa and even :John :Loves {…} will be removed.
- One can use variables, but only inside one statement, e.g. ?x :Loves ?x, or in a statement and its sub-statements, e.g. ?x gb:want {:Mary :Loves ?x}. Variables SHOULD not be used in statements of the same level, e.g. :John :Loves ?x. ?x gb:is :Girl, because reducing the set of matching values through following statements will not affect the set of beliefs removed in the preceding ones. E.g. :John :Loves :Grandpa will be removed anyway, even while there is no belief :Grandpa gb:is :Girl.

Note that contexts for rules, goals, and goals under work are protected against removal (both direct and through garbage collection). This means that e.g. `gb:I gb:remove {gb:I gb:want *}` will have no effect. If one wants to drop all the goals, one has to use `gb:I gb:remove {gb:I gb:want {* * *}}` instead.

Conditional commitment:

```
{:John :Loves :Mary} =>
     {gb:I :state :busy .
      {gb:I gb:do a:SendMail} gb:configuredAs {...}
     }
```

=> is the shorthand for `gb:implies`. When the agent's engine finds such a belief in G and finds out that the conditions (if several, seen as AND-connected) in the subject context are met, it adds to G all the beliefs specified in the object context. After that, the conditional commitment is removed.

Exclusive condition (gb:falseIf in the SmartResource's language):

```
{:John :Loves :Mary .
 gb:I gb:doNotBelieve {:John :hasBeen :notified}
} => {...}
```

The context defined as "gb:I gb:doNotBelieve {}" is a special "virtual" context which is the complement of G, i.e. it includes all the possible beliefs that are not part of G.

Goal as a condition:

```
{ gb:I gb:want {:John :Loves :Mary} } => {...}
```

Action that attempts achieving a goal (gb:achievesGoal in the SmartResource's language):

```
{ gb:I gb:want {:John :Loves :Mary} } >> {...}
```

>> is the shorthand for `gb:achievedBy.` When the agent's engine executes such a statement, it treats it the same way as `gb:implies (=>),` but in addition it moves the corresponding goal to some special context for goals under work (may be a backend context without a definition, thus not linked to G).

Prerequisites (gb:trueIfGoalAchieved achievesGoal in the SmartResource's language):

```
{ {...} => {...} } gb:requires {...}
```

When the agent's engine finds such a belief in G, it evaluates first the conditional commitment in the subject context. If it is to be executed, the engine checks the conditions in the object context. If they are all met, the commitment is executed. If some are not met, they are wrapped as "gb:I gb:want {}" (i.e. as goals) and added to G.

Commitment with a guard condition:

```
{ {...} => {...} } gb:is gb:truth ;
                   gb:existsWhile {...}
```

When the agent's engine finds an gb:existsWhile belief in G and finds out that a condition in the object context is not met, it removes from G all the beliefs specified in the subject context. Normally, this is to be used as mechanism for dropping unachievable or not-relevant-anymore commitments. However, this can also be used for specifying beliefs that depend on some other beliefs.

Behavioral rule:

```
{ {...} => {...} } gb:is gb:Rule
```

Unlike conditional commitments (those in G), rules belonging to the context defined as "{} gb:is gb:Rule" will not be removed after an execution.

Nothing prevents, however, use of gb:existWhile to define rules that are removed upon some condition.

Rule belonging to a role:

```
{ {...} => {...} } gb:is gb:Rule ;
                   gb:belongs :OperatorRole
```

Agent playing a role:

```
gb:I gb:haveRole :OperatorRole
gb:Nothing gb:haveRole :ExpertRole
```

Rule / conditional commitment that creates a rule / conditional commitment:

```
{...} => { {...} => {...} }
```

History of actions (informative)*:*

```
{
 { {gb:I gb:do :SendMail} gb:configuredAs {...} }
     gb:startTime "16.45"; gb:endTime "16.47";
     gb:result gb:Success
} gb:is :History
```

Quantified beliefs:

```
:John :Loves ?
```

```
{:John :Loves ?x} gb:forSome ?x
```

These two are equivalent and mean "John loves something". ? is the shorthand for `gb:Something`, while ?x is the shorthand for `http://..gb_uri../variables#x`.

```
{:John :Loves ?x. ?x :Is :Girl} gb:forSome ?x
```

This one means "John loves some girl".

```
{ {?x :Is :Man} => {?x :Is :Mortal} } gb:forAll ?x
```

This one means "Every man is mortal".

1.2.4 Querying constructs

The left side of => statements contains specification of beliefs to be matched against G.

Direct matching:

```
{:John :Loves :Mary} :accordingTo :Bill
```

Matching with variables:

```
{:John :Loves ?x} :accordingTo ?y
```

Such a query will be evaluated as true if there can be found some values of ?x and ?y so that the belief is present in G. In this case, the variables will be bind to the *first found* matching values, and, if the right side of => refers to these variables, these values will be used.

```
{{:John :Loves ?x} :accordingTo ?y. ?x gb:is :Girl} =>
                {gb:I gb:do java:SendMail} gb:configuredAs
                     {x:receiver gb:is ?x...}
```

The right side of => can also refer to three special variables, values for which are not taken from the beliefs but rather generated by the agent's engine: ?AgentName - the name of the agent, ?Today - the present date in the format "dd.mm.yyyy" and ?Now in the format "h:m:s".

Variables in the right side of => are substituted with their values as using String.replaceAll. Therefore, DB/?AgentName/received/?model.sapl is a legal expression using two variables: ?AgentName and ?model. This is in contrast to the left side of => where a whole resource (subject, predicate or object of a statement) can only be a variable.

Rules / conditional commitments that apply to all matching values:

```
{ ?x :Loves ?y } => { {...?x...?y...} gb:All ?x }
```

```
{ ?x :Loves ?y } =>
     { {{...?x...?y... } gb:All ?x} gb:All ?y } }
```

Normally, the rule / conditional commitment is executed for *the first found* matching combination of variables' values. The use of gb:All as shown in the first example above will lead that the rule will be executed for *every* matching value of ?x (taking first found value for ?y, if several values for ?y fit the same value for ?x) . In the second example, the rule is executed for every matching combination of the variables.

*Matching with *:*

```
:John :Loves *
```

This means "John loves something". In principle, this is almost the same as `:John :Loves ?x`, only that in saves the agent's engine from bother of recording the matching value for ?x. * is shorthand for `gb:Anything`.

Matching with variables standing for sets of statements:

```
?x :accordingTo :Bill
```

In this case, ?x is to be bind physically to the ID of the context and logically to the whole underlying graph. Therefore, the above means "all the information that Bill provided".

Special predicates (to realize e.g. FILTER of SPARQL):

Some predicates can be used, normally with variables as subjects, which are to be evaluated by the agent's engine rather than matched against G. Those include:

```
?x != 5
?x > 5
?x < 5
?x >= 5
?x <= 5
?x gb:regex "S.*h"
```

The first four are shorthands for `gb:gt, gb:lt, gb:gte, gb:lte.`

1.2.5 Agent's capabilities

In the SmartResource script language, it was only possible to use a RAB (a piece of Java code) in place of an action to be executed when a rule fires. Due to higher expressiveness of

this language, it is also possible to have some abstract *capabilities* as actions. If the object of `gb:do` has a predefined namespace (with e.g. prefix `java:`), it refers to a RAB:

```
{...} => { { gb:I gb:do java:ubiware.shared.SendEmail }
                                  gb:configuredAs {...} }
```

Otherwise (`http:` prefix), it refers to an abstract capability:

```
{...} => { { gb:I gb:do ex:SendEmail }
                       gb:configuredAs {...} }
```

A capability needs an *interface rule*, which will either directly prescribe the actions constituting the capability, or generate some beliefs or goals that would trigger the rules constituting the capability, e.g:

```
{ { gb:I gb:do ex:SendEmail } gb:configuredAs
  {:Receiver gb:is ?r . :Content gb:is ?c} } >>
{ gb:I gb:want {?c :SentTo ?r} }
```

Using >> instead of => has the same effect as for goals, i.e. the commitment is dropped if when the rule fires.

This elegantly solves the issue of separation between a role's capabilities (individual functionality), and the business processes in which this role can be involved (complex functionality). While one S-APL document may define a set of capabilities belonging to a rule, a completely different document may, if needed, describe the rules for engaging those capabilities and, in so, specify a business process in which the role may be involved.

1.2.6 S-APL run-time cycle

Figure 1.1 presents a simplified view of run-time cycle that the UBIWARE agent's behavior engine will need to implement to act based on S-APL behavior models.

In each iteration, the engine is to perform the following:

- First, check and remove all goals that match with G.
- Then, check all gb:existsWhile conditions found in G and remove the beliefs, whose existsWhile conditions are not met.
- Then, check all the gb:implies, gb:achievedBy, and gb:requires conditions found both is G (conditional commitments) and in the context {…} gb:is gb:Rule (behavior rules). If some conditional commitments or rules are executed, the result is that some new beliefs are added to G. In addition, executed conditional commitments are removed. At least at the current stage, the convention is that if there are several executable rules in one iteration, all are to be executed.
- Finally, all the gb:configuredAs (action commitments) and gb:I gb:remove {} (belief removal commitments) that are found in G are executed and then removed.

- If any changes to beliefs were made, start new iteration. Otherwise, collect garbage and block the run-time cycle until a new message arrives or some of running RABs makes some modification to the beliefs.

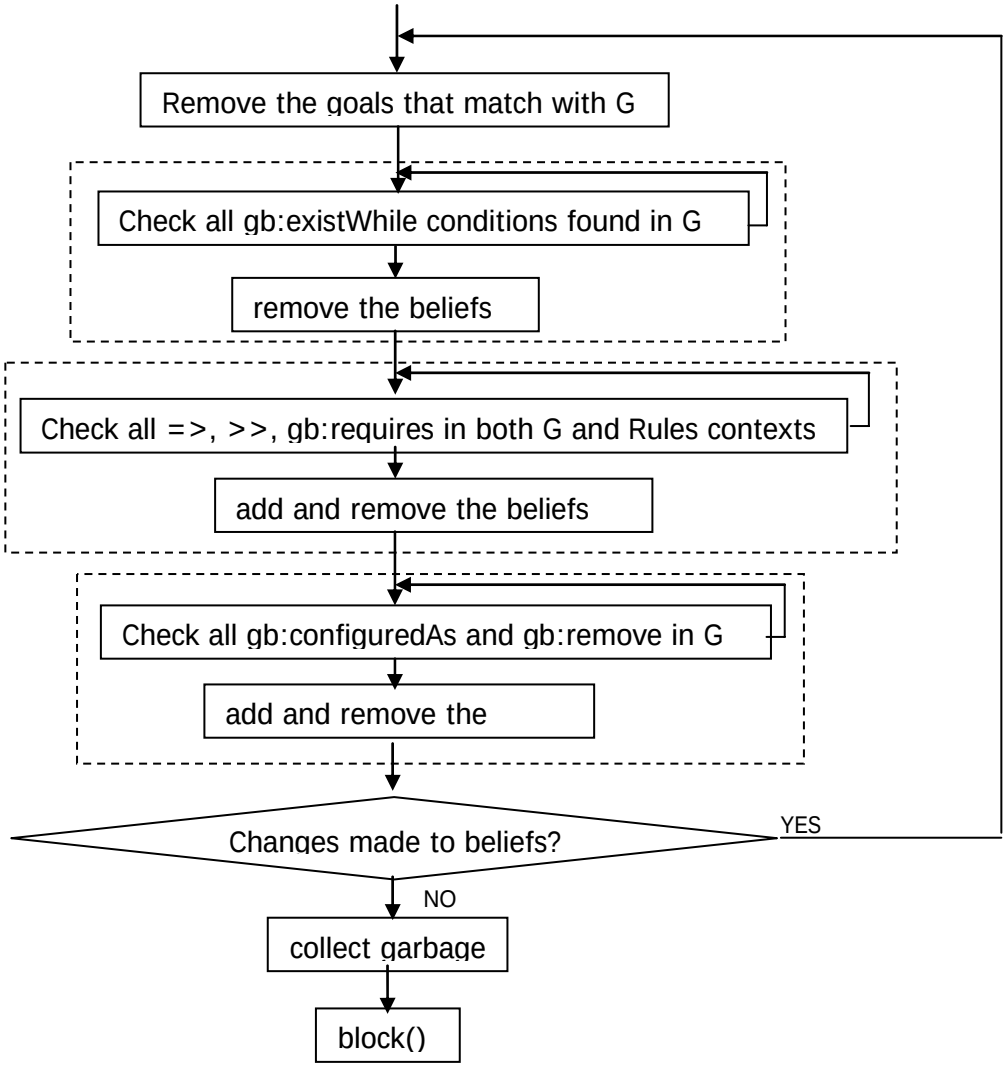

FIGURE 1.1 S-APL run-time cycle

As it stressed in the figure, actual modifications to the belief storage (adding and removing) are to be done after all the checks on each of the three stages are completed. In this way, e.g. the effect of execution of one rule could not prevent another executable rule from execution.

Garbage collection involves, in an iterative fashion:

- Removing all contexts that are not referenced, i.e. do not appear as the subject or the object of any statement.
- Removing all statements that are not referenced, i.e. are not linked to any context
- Removing all empty contexts, i.e. not having any member statements.

In result, any "hanging" sub-graph of beliefs will be removed as garbage. Therefore, it is always enough to explicitly remove just the top statement, e.g. {} gb:configuredAs {}, and let the garbage collector to remove the rest – the contexts appearing as the subject and the object.

1.2.7 Conclusions

The advantages of S-APL are the following:

- Simple model (triples in hierarchical contexts) that allows implementing any feature found in existing APLs.
- Expressive power is even greater than in existing APLs, because of full symmetry (everything is a belief): e.g. rules upon execution can add other rules of any complexity.
- Behavior specification is done using semantic predicates (e.g. implies, existsWhile)
  - Formally defined in an ontology.
  - Language is extensible with other such predicates.
- Reusable Atomic Behaviors and their parameters are also resources that can (and should) be ontologically modelled.
- So, there is *a basis* for sharing all 5 ontologies: External world, Mental states of agents, Properties of agents' bodies (available sensors and actuators), Input properties and Output properties (Bosse, T., Treur, J., 2000). Therefore, there is a basis for better understanding among agents with a goal of better coordination and collaboration among them.

To summarize the description of S-APL, below is the list of terms that are defined in the S-APL ontology (gb:You and gb:answer are introduced in the next chapter):

```
gb:is                 gb:remove             gb:lt (<)
gb:truth              gb:configuredAs       gb:gt (>)
gb:I                  gb:null               gb:lte (<=)
gb:doNotBelieve       gb:End                gb:gte (>=)
gb:want               gb:Start              gb:regex
gb:do                 gb:Success            gb:forSome
gb:implies (=>)       gb:Fail               gb:forAll
gb:achievedBy (>>)    gb:belongs            gb:All
gb:requires           gb:haveRole           gb:Anything
gb:existsWhile        gb:missingModel       gb:Something
gb:Rule               gb:missingRAB         gb:You
gb:add                gb:inNot (!=)         gb:answer
```

UBIWARE Deliverable D1.1:
Workpackage WP2:
Task T1.1_w2:

2 UbiBlog – Managing Distributed Resource Histories

INTER-AGENT INFORMATION SHARING

Workpackage leader: ***Artem Katasonov***

In UBIWARE, every resource is represented by a software agent. Among major responsibilities of such an agent is monitoring the condition of the resource and the resource's interactions with other components of the system and humans. The beliefs storage of the agent will, therefore, naturally include the history of the resource, in a sense "blogged" by the agent. Obviously, the value of such a resource history is not limited to that particular resource. A resource may benefit from the information collected with respect to other resources of the same (or similar) type, e.g. in a situation which it faces for the first time while other may have faced that situation before. Also, mining the data collected and integrated from many resources may result in discovery of some knowledge important at the level of the whole ubiquitous computing system. A scalable solution requires mechanisms for inter-agent information sharing and data mining on integrated information which would allow keeping the resource histories distributed without need to copy those histories to a central repository.

The part of the work in this work-package, reported in this document, aimed at answering the following research question:

- How to semantically markup the history of a resource in a system, in order to make it reusable for other resources and at the system-level?
- What mechanisms are needed for effective and efficient sharing of information between the agents representing different resources?

2.1 General approach

The general approach, which was selected in this work-task, is to avoid designing some special protocols and languages for inter-agent information sharing, but rather to reuse as much as possible the tools developed as part of the UbiCore (WP1).

Obviously, there is certain similarity in the following. On one hand, an agent always needs to query its own beliefs base in order to evaluate the left sides of its behavior rules in order to identify rules that are to be executed. On the other hand, when an agent asks another agent for some information, it, in a sense, queries the belief base of that other agent.

Our approach is therefore to design the external querying process so it would be almost the same as if the agent itself would query its belief base to check the conditions for executing a rule. This also means that we plan to use the Semantic Agent Programming Language (S-APL, see the previous chapter) not only as the means for prescribing the agents' behaviors, but also as the inter-agents communication content language (to be used instead of FIPA-SL or other languages of this type). The advantages of this should be obvious as the symmetry and expressive power in the UBIWARE platform will be maximized. The agents will be able to query each other not only for some facts (present or historical) about the external world (the domain) but also, for example:

- Query if the other agent knows a plan for achieving a certain goal,
- Query if the other agent knows a rule that should be applied in a particular situation.

We will not repeat here the details about S-APL that have already been described in the previous chapter on WP1, and present only additional information.

2.2 External querying

The belief base of an agent can be queried externally, e.g. by other agents (of course subject to security and other policies). The core of a query is the same as normal, i.e. the same as if the agent itself would query its belief base to check the conditions for executing a rule. The core of the query has to be wrapped with:

```
gb:I gb:want { {gb:You gb:answer {..query..} } }
```

As the response, the agent is to send a part of its belief base. Some examples follow.

Direct matching query:

Request, e.g.:

```
gb:I gb:want {gb:You gb:answer { :John :Loves :Mary } }
```

Response should be either of type

```
:John :Loves :Mary or
gb:I gb:doNotBelieve {:John :Loves :Mary}
```

Matching with variables:

Request, e.g.:

```
gb:I gb:want { gb:You gb:answer { ?x :Loves ?y } }
```

Response should be either of type

`:John :Loves :Mary` (*the first found* matching combination of variables' values) or

```
gb:I gb:doNotBelieve {?x :Loves ?y}
```

Query for all matching values:

Request, e.g.:

```
gb:I gb:want { gb:You gb:answer {
                  {?x :Loves ?y} gb:All ?x
               }}
```

Response should be either of type

`:John :Loves :Mary. :Bill :Loves :Jane` (all matching values for ?x with some corresponding values for ?y) or

```
gb:I gb:doNotBelieve {?x :Loves ?y}
```

Matching with variables standing for sets of statements:

Request, e.g.:

```
gb:I gb:want { gb:You gb:answer {
                  {:John :Loves :Mary} => ?x
               }}
```

Response should be either of type

```
{:John :Loves :Mary} =>
     {gb:I :state :busy .
      {gb:I gb:do a:SendMail} gb:configuredAs {...}
} or
gb:I gb:doNotBelieve { {:John :Loves :Mary} => ?x }
```

In this example, an agent asks another agent for a rule of handling some situation. Other similar complex queries are also possible and may be useful.

Use of `gb:I gb:want {...}` in external querying may look unnecessary. However, this allows distinguishing between `gb:I gb:want {...}` and e.g. `:Boss gb:want {...}`, i.e. mediating a wish of another agent. Both cases may require exactly the same action to be taken, however, may affect differently on whether the agent will comply with the wish or not.

2.3 Other communicative acts

External request for action:

Agents can request other agents to perform some actions, corresponding either to abstract capabilities or RABs (of course subject to security and other policies).

The core of the request is almost same as if the agent itself would command itself to perform an action, only with `gb:You` in place of `gb:I`. In addition, the core is wrapped with `gb:I gb:want {}`:

```
gb:I gb:want { {gb:You gb:do ...}
                          gb:configuredAs {...} }
```

Informing:

Both external queries and external requests for action are requests by their nature and therefore should be contents of FIPA ACL messages with REQUEST performative.

On the other hand, an ACL message with S-APL contents and INFORM performative can be interpreted one agent informing the other one about something. An INFORM message

```
:John :Loves :Mary
```

informs that the speaker believes that John loves Mary. In principle, it also implies that speaker wants the listener to believe the same fact. The latter meaning can be emphasized by turning the message into a REQUEST with a wrapping

```
gb:I gb:want { gb:You gb:add {:John :Loves :Mary} }
```

UBIWARE Deliverable D1.1:
Workpackage WP3:
Task T1.1_w3:

3 SURPAS – Smart Ubiquitous Resource Privacy and Security

ACCESS CONTROL

Workpackage leader: ***Anton Naumenko***

Globalization of the economy, global and intercultural value chains, large-scale industrial environments, cooperative systems for the international production, logistic and marketing could hardly be imagined without the rapid evolution of information and communication technologies (ICTs). Moreover, continuous advances of ICTs and their adoption in the industrial world have been guaranteeing improvement and efficiency of industrial technologies in the last decades. Recent advances in networking, sensor and RFID technologies, etc allow connecting various physical world objects to the IT infrastructure, which could, ultimately, enable realization of the "Internet of Things" and the ubiquitous computing visions. However, the adoption of new ICTs in the traditional production industries, e.g. the process industry, the machinery industry, etc, is relatively slow. It is mainly because of the growing complexity of emerging ICTs, inadequate security infrastructures, and the fact that the research in ICTs usually focuses on the industries with a short cycle of innovations deployment, such as health care or banking, largely overlooking the needs of the production industries.

In response to these problems, this chapter focuses on the security challenges in UBIWARE. It presents the security threats, requirements, implications and access control measures needed for UBIWARE in the context of its industrial adoption. Industrial cases, outlined in the paper, align our research results on UBIWARE, as such, with the real world needs and serves as a trigger and source of requirements for the research on security,

particularly. We describe our long-term vision for the security and privacy management in emerging new types of environments, which we refer to as Smart Ubiquitous Resource Privacy and Security (SURPAS). SURPAS is mainly based on the advances in the Semantic Web, Multi-Agent Systems, and Ubiquitous Computing domains. Particularly, this chapter presents the SURPAS research framework which guides our research towards SURPAS. It is a consolidated formal system of research ideas and prototypes for the interoperable pro-active context-aware self-protecting security management. The main components of the SURPAS research framework are the conceptual semantics of security policies, functionality of security mechanisms, including functional semantics, algorithms, abstract architecture, and reference implementation, and adopting applications in different business domains (e.g. industrial maintenance, subcontracting management, smart house, etc).

The rest of the chapter is organized as follows. Section 1 briefly discusses the security implications of UBIWARE. Section 2 addresses the security threats in UBIWARE. Section 3 presents analysis of security concerns regarding industrial adoption of UBIWARE. Section 4 presents the SURPAS research framework. Sections 5 and 6 give a detailed description of the SURPAS conceptual and functional semantics, respectively. Section 8 exemplifies our research and development ideas using the industrial cases. Section 9 presents conclusions and future research directions.

3.1 UBIWARE security implications

UBIWARE advances existing technologies to a qualitatively new level and brings to life new complex industrial environments, where traditional approaches to manage security fall short. Also, existing security measures for the technologies on which UBIWARE relies, e.g. multi-agent, are not in a mature stage and still require significant elaboration to mitigate associated risks. The security cannot be added to the UBIWARE platform later but the design decisions regarding security have to be thoroughly correlated with the requirements, characteristics and design of the platform, due to mutual impact on resulting features of UBIWARE. Thus, the analysis of the major characteristics of UBIWARE and their implications on security is an important activity, and has to be conducted throughout the development of UBIWARE.

3.1.1 Openness

Openness of environments has several dimensions and refers to a range of features. The UBIWARE-supported industrial environments are open in a sense that they create, and are created, by business networks. Every partner of such a network can both use the environment (participate in different roles) and contribute some resources developing the environment further. UBIWARE is built on the top of open standards and technologies developed by open communities. Open industrial environments introduce more challenging security problems due to a greater amount of risks and threats. For example, open-source agent platforms and agents are easier to compromise and alternate for malicious behavior. Also, open environments require complex reputation-based trust management solutions.

3.1.2 Dynamics

Dynamics of the environments considered leads to unpredictable changes of their states, due to complexity and ad hoc nature of relations between entities in these environments. This challenges application of traditional techniques for achieving common security goals such as availability, reliability and integrity. In addition, the dynamics complicates introducing adequate techniques to ensure manageability and accountability.

3.1.3 Heterogeneity

Heterogeneity should be considered in the contexts of industrial environments and security infrastructure itself. As to industrial environments, the heterogeneity of resources poses a great variety of security requirements which UBIWARE has to meet. This variety of requirements and the variety of available security solutions related to the technologies, on which UBIWARE relies, complicate the construction of a consolidated security infrastructure. Therefore, the interoperability becomes one of the most important factors.

3.1.4 Decentralization

Distributed nature of UBIWARE-supported industrial environments reduces privacy concerns of partnering organizations because of local management of historical data associated with the owned resources. The distribution of control also enhances survivability of the whole system and is known to reduce network traffic and to overcome network latencies (Harrison et al., 1995). However, the distribution of components complicates the management of security, especially the logging for audit activities.

3.1.5 Collaborative

Collaborative or social nature of industrial environments correlates with several other characteristics of UBIWARE. The major impact on security is that the communication has to be secured for an efficient and trusted collaboration. This area of research in security has traditionally been addressed. However, the unique features of agent technologies and high demands of industrial applications still keep a place for elaboration.

Internationality of today's industrial world requires that the policy languages have to flexible and expressive enough to handle the diversity of cultures, legislations, and traditions in international cooperation.

3.1.6 Context-awareness

Context-awareness is the next important factor of UBIWARE and access control. Location, time, resource's activity, history, etc are typical contextual data. Authorization of incoming requests may rely only on these contextual factors. For example, an access control decision may deny new requests because of overloading a SmartResource agent without evaluation of other information. In this example, an enforcement mechanism ensures availability of the SmartResource agent, in contrast with the most common support of confidentiality. While we cannot envision all possible contextual factors in UBIWARE, it is hard to define the border of the access control system. Thus, the access control should be extensible enough to leave a space for user-driven personalization or intelligent self-configuration of access control policies and mechanisms. Context-awareness, extensibility, intelligence, and personalization demand a great expressiveness of underlying access control models and languages for policy specification.

3.1.7 Flexibility

Flexibility is vital for applicability and interoperability of access control solutions in the heterogeneous UBIWARE environments. This flexibility of access control functionality should reflect extensibility and expressiveness of access control models and languages. The architecture of access control must follow component-oriented style of design. This is needed in order to compose and to reconfigure required access control functionality for specific usage scenarios using modular and flexible access control mechanisms and tools.

3.1.8 Extensibility and expressiveness

There is an obvious need to provide proper security measures in general and access control in particular. The UBIWARE industrial cases demand to have expressive, flexible, pervasive and ease-to-use means for authorizing access to the sensitive information and services. For example, these industrial cases highlight an additional issue associated with UBIWARE In the cases of power-network management and paper machinery maintenance, electrical engineers and maintenance experts are not authorities that define access control policies. Businesses and organizations that hire these people and commission their work should have full or partial authority over access control process depending on concrete needs. Ultimately, UBIWARE provides access to the content with the great variety of media types. Therefore, access control solutions should support the major standardized formats of data representation. It is desirable that access control solutions could be easily extensible for new and emerging formats. Heterogeneity in UBIWARE has wider scope than just variety of data formats. There are different wireless and wired network protocols, resources with different sets of features and characteristics, different settings of computational environments, different available security measures, and other.

3.1.9 Self-management

The last important issue to consider is security of access control solutions. Access control infrastructure should obviously be secure by design. For example, the guards are typically designed in such a manner that requests cannot bypass them. The availability of security-related components should be high. Accountability of access control mechanism must be ensured by proper logging for audit practices and technologies. Manageability is also an issue for the access control administration. These generic security goals could be further elaborated under a concept of self-management of access control solutions. Self-management in terms of security is self-protection that is a vision of pro-active context-aware autonomic security mechanisms to detect, identify and protect against various types of threats (Kephart and Chess, 2003)**Error! Reference source not found.**, (Horn, 2001).

3.1.10 Human centricity

Access control guards for UBIWARE must be usable within limited capabilities of embedded systems. Access control solutions should particularly bear with limited connectivity and low transmission rates, battery power shortages, lower processing power and storage space, limited means for user input/output interactions. There are also other limitations which are not technical. Human users tend to simultaneously perform multiple tasks. Human users are also prone to errors. Due to technical limitations, access control solutions as well as other utility applications should introduce low overheads in terms of attention required, network traffic, performance, storage, power consumption, etc.

3.1.11 Mobility

Mobility of agents and resulting security implications are well addressed in the literature [4-6]. However, the UBIWARE security is impacted by the mobility of both resources and agents, and also by the limitations of mobile devices and technologies. The mobility directly affects the solutions for all the security goals and requires some tradeoffs between mobility, performance and security.

3.1.12 Ambient intelligence

Ambient intelligence, ubiquity, and pervasiveness of information technologies have tighten the digital and physical worlds to the extent when security becomes the ultimate issue. The major implication of penetrating ICTs on security is that the risks and negative consequences of security threats become higher than ever. On the other hand, the security infrastructure itself has to become pervasive, interoperable and intelligent enough to naturally fit UBIWARE.

3.2 Security threats in UBIWARE

This chapter presents our review of the security threats in UBIWARE from the conventional and architectural perspective.

3.2.1 Conventional perspective

The commonly followed categorization of security threats in computer science follows to the high level security goals, i.e. confidentiality, integrity, availability, etc. This categorization starts with the corresponding high level threats, i.e. disclosure of information or unauthorized engaging of the service, corruption of information or service, and denial of service. Other widely recognized generic security goals are confidentiality, availability, reliability, manageability, accountability, responsibility, integrity, non-repudiation, anonymity, and privacy.

MAS, as well as more traditional client-server information systems, has different types of security threats, e.g. spoofing, unauthorized access, tampering, network eavesdropping, denial-of-service attacks, man-in-the-middle attacks, intrusion and etc (Srirama and Naumenko, 2007).

– **Spoofing** or masquerading is using a false identity in order to hide the original source of an attack and to gain access as a legitimate entity.

– **Unauthorized access** "is gaining access into any computer, network, storage medium, system, program, file, user area, or other private repository, without the express permission of the owner. Unauthorized access is the same as theft."[1]

– **Tampering** is malicious modification of an agent message in the network.

– Network **eavesdropping** is to monitor traffic for sensitive data such as plaintext passwords by placing sniffers in the middle of the network.

– In the **Man-in-the-middle** attacks, the attacker captures the messages, changes the contents or keeps the contents unchanged and replays them to the original target.

– **Denial-of-service** is a process of making a system, server or application unavailable.

– **Intrusion** is an "unauthorized act of bypassing the security mechanisms of a system"[2].

– **Repudiation** is a denial of having processed the data or having engaged with the service by the entities in MAS.

3.2.2 Architectural perspective

For MAS like UBIWARE, there is a different classification of threats that is based on the origins and targets of attacks (Jansen, 2000), (Jansen and Karygiannis, 1999). Widely

[1] www.michigan.gov/cybersecurity/0,1607,7-217-34415---,00.html
[2] www.tecrime.com/0gloss.htm

accepted classes of attacks are agent to agent platform, agent to agent, agent platform to agent, and entity to agent platform. In addition to these classes we have to consider agent to resource and resource to agent attacks that are specific for UBIWARE and SmartResource platform particularly. We follow this categorization in our research because it is more oriented to the architecture of UBIWARE. This architectural categorization couples more tightly security and system designs. We consider what the meanings of conventional threats are in each of the architectural classes of attacks.

3.2.2.1 Agent to agent platform

We will refer to threats of this category as agent-to-platform threats and attacks (TA2P). These threats arise when a remote or local SmartResource agent performs malicious or unintentionally harmful behavior against UBIWARE.

Id/Name	Actors	Description
TA2P1 Masquerading	SR agent A SR agent B new SR agent SR platform	The agent A provides false identity of the Agent B or not-existing (newly created, cloned) agent to the agent platform. There are several reasons to do that: 1. Intention to acquire access rights of the agent B in order to perform unauthorized access. 2. Intention to perform harmful actions under false identity to be unaccounted for the consequences. 3. Intention to decrease the reputation of the agent B by compromising its identity. 4. Intention to get new identity in order to get rid of old identity that is associated with the previous malicious behavior. 5. Unintentional malfunctioning of the agent A.
TA2P2 Denial of Service	SR agent A SR platform	The agent A starts to consume resources of the agent platform by requesting regular services with the high rate of requests than allowed or possible. The agent A exploits vulnerabilities in the agent platform in order to limit its availability to other agents and agent platforms. There are several reasons for that: 1. Intention to terminate the agent platform or to degrade the performance of the agent platform. 2. Intention to get more resources than it is prescribed for this agent by quota. 3. Unintentional invocation of services of the agent platform with vulnerabilities.
TA2P3 Unauthorized	SR agent A SR platform	The agent A accesses services and data of the agent platform without proper authorization. The reasons

access		are following: 1. The agent gets unauthorized permissions because of the ambiguity in the security policy 2. Intention to access sensitive data exploiting vulnerabilities of agent platform, caches, communication channels, persistence storages, in memory storages, etc. 3. Intention to engage unauthorized services with faked proofs of privileges. 4. Unintentional unauthorized access

3.2.2.2 Agent to agent

We will refer to threats of this category as agent-to-agent threats and attacks (TA2A). These threats arise when a SmartResource agent performs malicious or unintentionally harmful behavior against other SmartResource agents.

Id/Name	Actors	Description
TA2A1 Masquerading	SR agent A SR agent B new SR agent SR agent C	The agent A provides false identity of the agent B or not-existing (newly created, cloned) agent to the agent C. The reasons for this threat are the same as for the corresponding TA2P1 threat. The additional reason might be to compromise the agent C by the fact of communication between the agents B and C.
TA2A2 Denial of Service	SR agent A SR agent B SR platform	The agent A starts to consume resources of the agent B by requesting services of agent B with the high rate of requests or by sending an excessive amount of messages of different type. The agent A might try to exploit vulnerabilities of the agent B. The agent A can also try to kill, suspend or move the agent B be invoking corresponding functionality of the agent platform. There are several reasons for that: 1. Intention to terminate the agent B or to degrade the performance of the agent B. 2. Unintentional invocation of services of the agent B with defects. 3. Unintentional malfunctioning of communication functionality of the agent A.
TA2A3 Unauthorized access	SR agent A SR agent B	The agent A accesses services and data of the agent B without proper authorization. The reasons are the same as for the threat TA2P3.
TA2A4	SR agent A	The agent A denies its participation to factually

Repudiation	SR agent B	occurred transaction or communication with the agent B. The reasons might be different for that: 1. The agent A deliberately claims false information to be unaccounted for the results of occurred transaction or communication with the agent B. 2. The agents A and B have different views on the events because of the ambiguous business processes or communication protocols. 3. Accidental repudiation when proper implementation of transaction management and communication protocols are not in place as well as logging for audit solutions.

3.2.2.3 Agent platform to agent

We will refer to threats of this category as agent platform-to-agent threats and attacks (TP2A). These threats arise when a SmartResource platform perform malicious or unintentionally harmful behavior against SmartResource agents.

Id/Name	Actors	Description
TP2A1 Masquerading	SR platform A SR platform B SR agent B	The platform A provides false identity of the platform B to the agent B. The reasons for this threat are following 1. Intention to acquire reputation or access rights of the platform B. 2. Intention to perform harmful actions against agent B under false identity to be unaccounted for the consequences. 3. Intention to decrease the reputation of the platform B by compromising its identity. 4. Intention to compromise the platform B by the fact of communication between the platform B and agent B. 5. Accidental malfunctioning of the platform A.
TP2A2 Denial of Service	SR platform A SR agent B	The platform A terminates, suspends or moves the agent B, unfair fully allocates resources to the agent B, creates burden of requests for the agent B, provides services with unacceptable delays, does not follow service level agreements, etc. There are several reasons for that: 1. Intention to terminate, suspend or move the agent B. 2. Intention to degrade the performance of the

		agent B without authorization for that from the agent B. 3. Intention to keep the agent B from fulfilling some critical tasks. 4. Unintentional invocation of services of the agent B with defects. 5. Unintentional malfunctioning of the platform A.
TP2A3 Unauthorized access	SR platform A SR agent B	The platform A accesses services, code, state and data of the agent B without proper authorization. The reasons are following. 1. Intention to get unauthorized permissions because of the ambiguity in the security policy. 2. Intention to access sensitive data exploiting vulnerabilities of the agent B or using direct access to agent's code and state. 3. Intention to engage unauthorized services with faked proofs of privileges. 4. Unintentional unauthorized access.
TP2A4 Eavesdropping	SR platform A SR agent B	The platform A intercepts and monitors confidential communication activities, code, state and data of the agent B. The reasons are following: 1. Intention to collect sensitive and confidential information of the agent B. 2. Intention to monitor relations of the agent B to other agents and platforms. 3. Intention to collect information about other entities that the agent B may have. 4. Intention to avoid repudiation by the agent B. 5. Unintentional retention (logging) of sensitive and confidential information
TP2A5 Alteration	SR platform A SR agent B	The platform A modifies the communicating messages, code, state and/or data of the agent B. 1. Intention to corrupt messages, code, state and/or data. 2. Intention to tamper messages, code, state and/or data. 3. Intention to correct malfunctioning code or inconsistent messages, data or state. 4. Unintentional alternation.

3.2.2.4 UBIWARE to resource and resource to UBIWARE

We will refer to threats of these categories as UBIWARE-to-resource (TU2R) and resource-to-UBIWARE (TR2U) threats and attacks. The TU2R threats arise when a SmartResource agent or other entities of the UBIWARE platform perform malicious or unintentionally harmful behavior against resources which should be served by this agent and UBIWARE instead. This kind of threats is the most dangerous because it threatens real-world resources. As UBIWARE serves and the agent represents its resource, the TU2R threats are similar to the TP2A threats. The TR2U threats arise when a resource intentionally or accidentally performs harmful behavior against its SmartResource agent or UBIWARE in general. These threats include masquerading, denial of service, unauthorized access, repudiation and alternation.

3.2.2.5 Repositories to agent

The possibility for SmartResource agents to load their roles and RABs from different external sources should be also taken into account. The agents may unintentionally provide threats playing some role or using some RAB from the malicious pool of roles or RABs. Repositories might be responsible only for the provisioning of malicious roles and RABs when they do not use sufficient mechanisms for testing and verification of roles and RABs upon their publishing to repositories by original attacker. Things become even more complicated when security policies and security mechanisms can be loaded from external sources by agents.

3.3 Security questions in industrial cases

This chapter reveals security concerns in the UBIWARE industrial cases: decentralized management of power networks and proactive machinery maintenance in paper industry.

3.3.1 Decentralized management of power networks

In this section, we exemplify industrial impact, business benefits and security issues of UBIWARE using a case study in the domain of distributed power network management (Naumenko et. al., 2007) that we performed in collaboration with ABB company (Distribution Automation unit). We present four scenarios of potential new applications that could be created based on UBIWARE and discuss the security implications. ABB is a global vendor of hardware and software for power networks. The power networks themselves are owned, controlled and maintained by some local companies. It is noticeable that the control systems of different companies are not integrated.

However, the information exchange between sub-networks may be very important for fault localization, network reconfiguration, and network restoration when a fault happens on the border of sub-networks. This is our first scenario: introducing an inter-organizational smart middleware solution like UBIWARE could solve this issue. Existence of adequate

security mechanisms are a prerequisite though. A challenging research question is how to elaborate flexible and expressive framework for the distributed, collaborative and policy-based management of security. FIGURE 3.1 shows this first scenario.

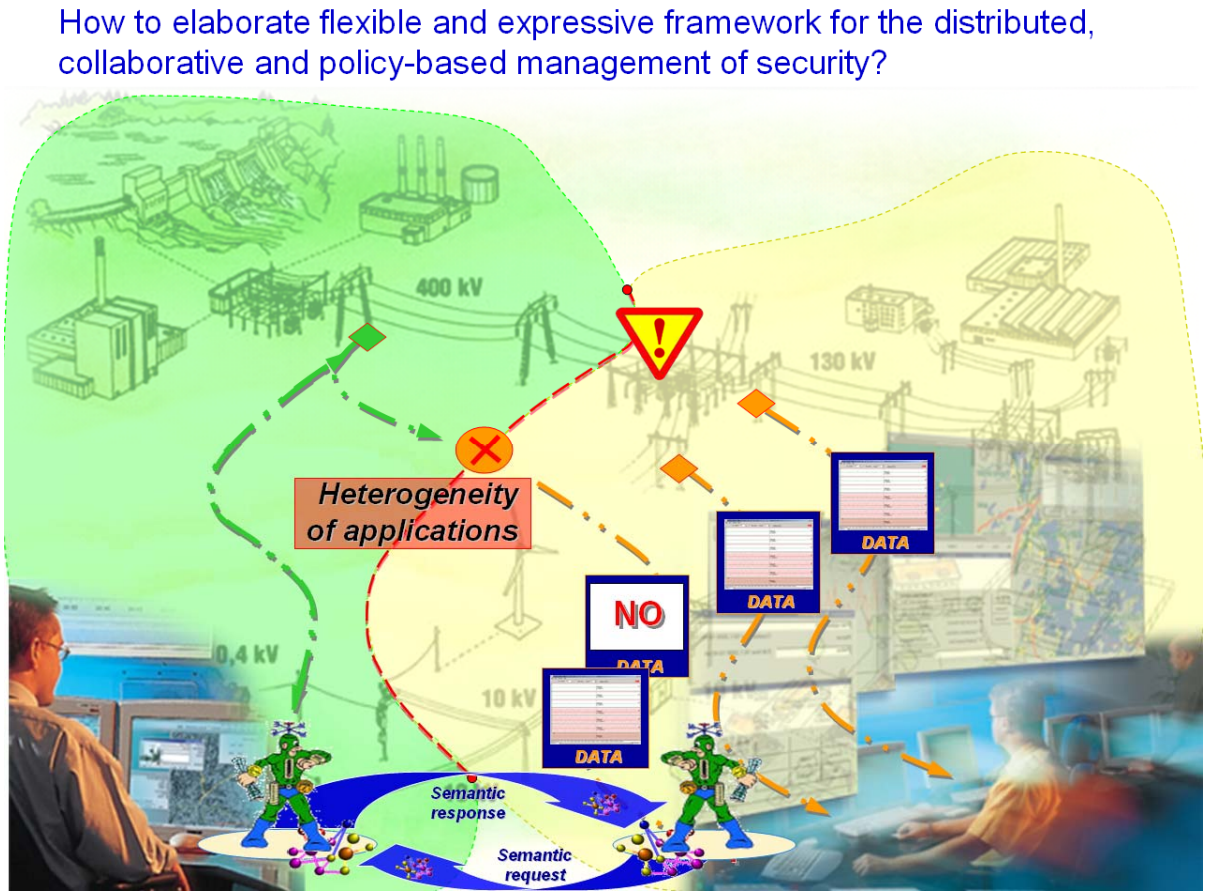

FIGURE 3.1 Security of information exchange between sub-networks

The second scenario in our vision is related to a new business model that ABB could implement. At present, all ABB expertise gets embedded into hardware or software systems and sold to the customers as it is. A new business model would be to start own Web-services providing implementation of certain algorithms, so the ABB customers will utilize those algorithms online when needed. ABB will be always able to update algorithms, add new, and so on. Noticeable that, if semantically defined, such Web-service can potentially be utilized across the globe even by the customers who never purchased any of ABB hardware or software. Regarding security, this means that UBIWARE must handle secure provisioning of (semantic) web services, which is still an open research question. We have already targeted issues related to the access control policies in (semantic) Service-Oriented Architecture (Naumenko and Luostarinen, 2006) and provisioning of Web services in mobile infrastructures (Srirama and Naumenko, 2007). However, there are still problems related to the secure communication, privacy and trust management. FIGURE 3.2 illustrates security questions and this scenario.

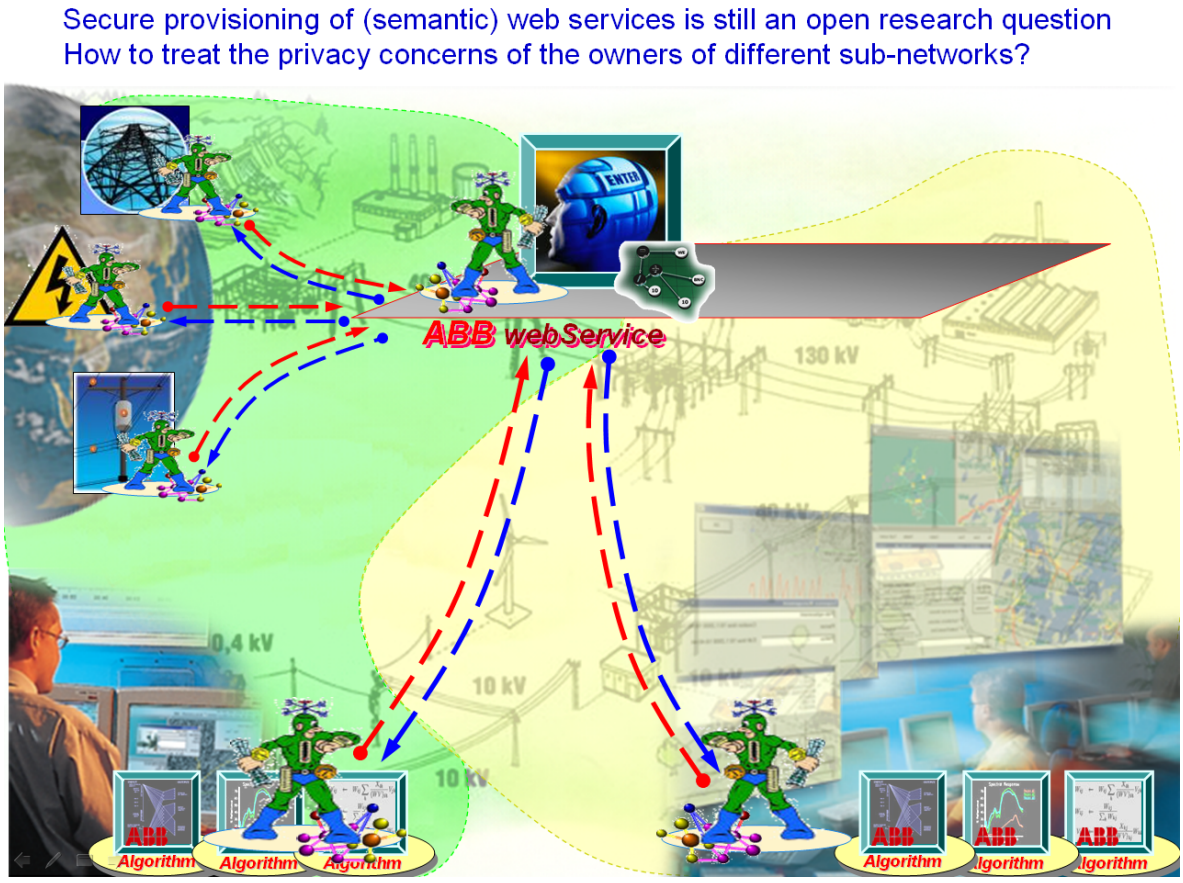

FIGURE 3.2 Security for web service-based business model

The third scenario in our vision is integration of contextual data with the currently used data such as the network structure and configuration, feeder relay readings, etc. Such integration can be used for evaluating existing threats for the power network, facilitation of fault localization, extending the operators' view of the power network, etc. Basically, integration of contextual information from external sources requires different approaches to trust management depending on the used techniques and purposes of integration. Thus, the major question related to security is how to formally compute reputation and trust for the external contextual services because these issues influence the confidence in predicted risks, fault locations, etc. FIGURE 3.3 illustrates security question in the third scenario.

The last scenario is transferring the knowledge of human experts to automated systems, by means of various data mining tools. For example, now it is always a decision of a human expert which of the existing fault localization algorithms will perform the best in the context of the current configuration of the power network and the nature of the fault. Such decisions made by an expert, along with the input data, could be forwarded to a learning Web-service. After a sufficient learning sample, this Web-service could start to be used in some situations instead of the human expert, e.g. in situations when a faster decision is needed or when the expert is unavailable. Considering humans as objects of access control in machine-to-human interactions is an interesting research question. The data mining algorithms perform better on

larger sample sets when they are collected for all equipment of power networks. A question is then how to treat the privacy concerns of the owners of different sub-networks.

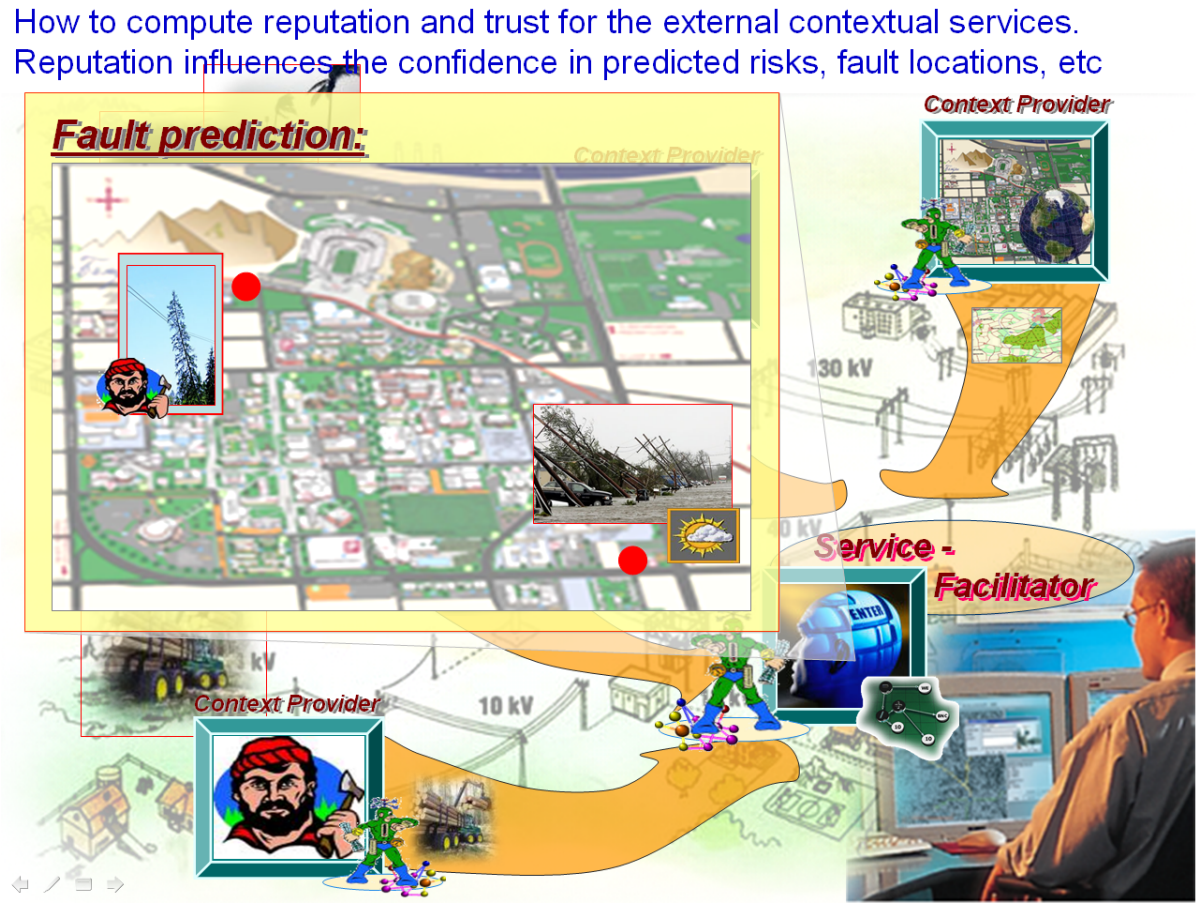

FIGURE 3.3 Security for the external contextual services

3.3.2 Proactive machinery maintenance services

This section describes the current situation with and business needs for security in the cluster of Metso's companies. The cluster of Metso's companies, Metso Paper Inc. and Metso Automation Inc., specializes in pulp and paper industry processes, machinery, equipment, control systems, related know-how and after sales services. The Metso Paper's offering extends over the entire life cycle of the process covering new lines, rebuilds and various services. Metso Automation supplies control systems and related ICTs for the products of Metso Paper. We were concentrating on two areas, namely Product Data Management (PDM) (Naumenko et al., 2005) and Proactive Machinery Maintenance Services (PMMS) (Luostarinen et al., 2006). The major findings are the following.

– The current level of security for PMMSs is not sufficient for the needs of managing cross-organizational processes. The elaboration of generic authorization enforcement mechanisms in the business network is crucial to handle the heterogeneity and to shift the

control over the authorization process from Metso to its customers. FIGURE 3. shows the current architecture for provisioning of PMMSs.

– The inter-organizational information exchange in the paper industry will extensively use the mill model. Currently, there are several research initiatives that try to use Semantic Web standards and technologies in order to develop appropriate solutions for the information exchange for the PDM process. When semantic standards come into use for PDM and PMMSs, then industrial resources for the access control will have semantic descriptions according to the mill ontology. FIGURE 3. illustrates the vision for a future collaborative platform for PDM based on the Semantic Web technologies.

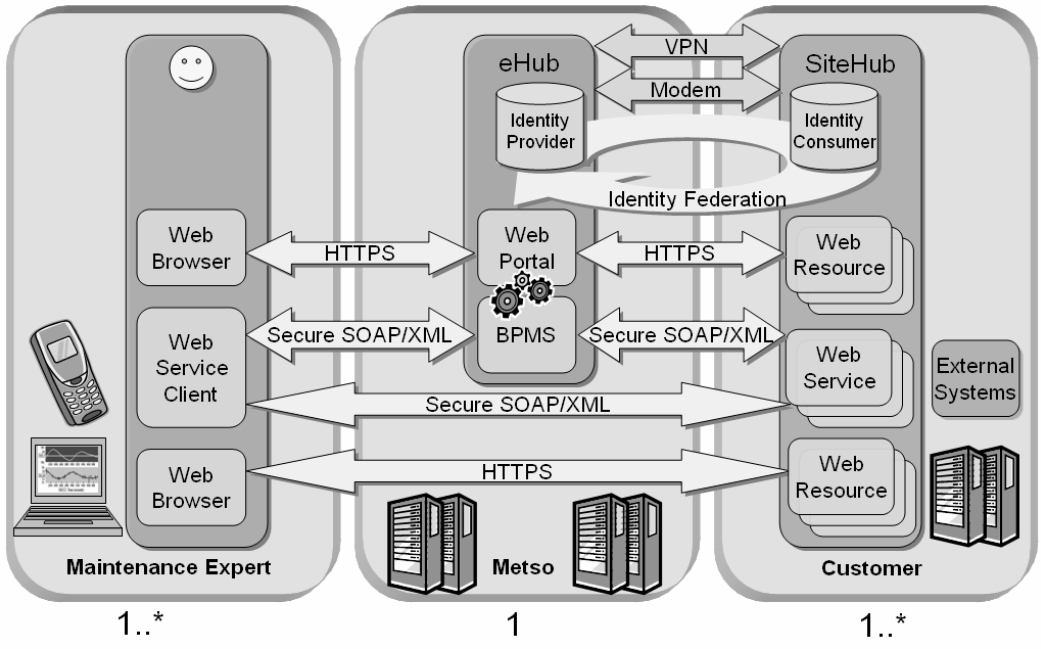

FIGURE 3.4 The architecture for remote machinery maintenance services

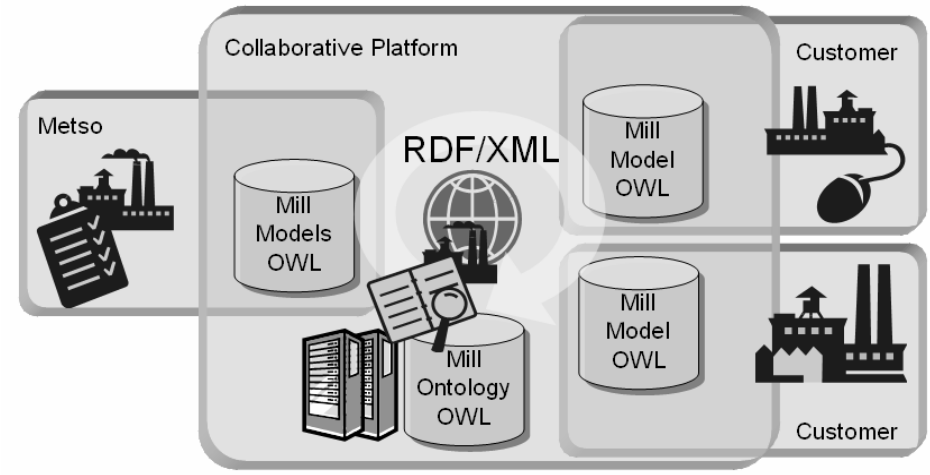

FIGURE 3.5 The collaborative platform for Product Data Management

3.4 SURPAS Research Framework

Traditional security goals like confidentiality, availability, reliability, integrity, manageability, accountability, responsibility etc, together with conventional measures and mechanisms that support security, do not cover all the needs and threats of new emerging computing environments. UBIWARE poses challenges for the research and development towards more pervasive and intelligent countermeasures. Such countermeasures have to provide the high level of user privacy, effective trust management, built-in self-security, context-awareness, and pro-activity. Moreover, protection of multi-agent systems like UBIWARE is still immature area, where the adoption of conventional security measures and elaboration of new techniques are promising (Jansen and Karygiannis, 1999), (Borselius, 2002).

The elaboration of a consolidated security infrastructure following the SURPAS research framework will lead to more innovative and intelligent industrial tools and will transform security from an obstacle to a driver of large-scale industrial collaboration. SURPAS follows the general UBIWARE vision – configuring and adding new functionality to the underlying industrial environment on-the-fly by changing high level declarative descriptions. Regarding security, this means that SURPAS is able of smoothly including new, and reconfiguring existing, security mechanisms, for the optimal and secure state of the UBIWARE-based system, in response to the dynamically changing environment.

FIGURE 3.6 illustrates the SURPAS research framework. The SURPAS semantics (Naumenko, 2006) is the main conceptual part of the research. The formal explicit specification of semantics is an input for the critical analysis of characteristics of suggested features and for further elaborations of other components of the framework. In a nutshell, the use of ontologies, instead of mathematical security and domain models, is the main characteristic of the SURPAS research.

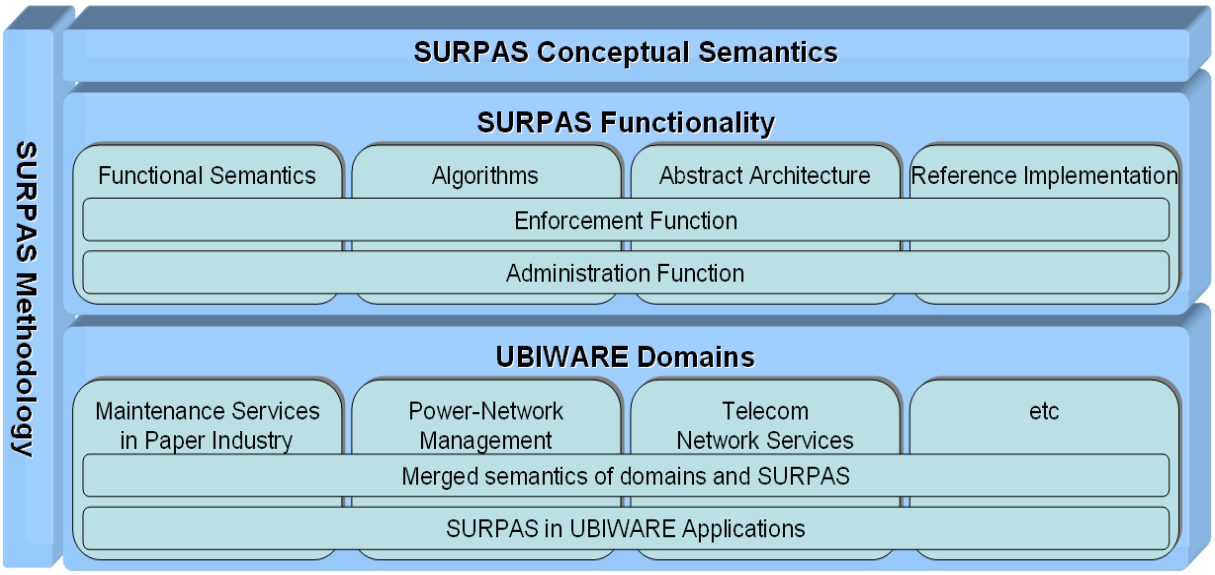

FIGURE 3.6 The SURPAS Research Framework

SURPAS functionality consists of two parts: the enforcement function and the administrative function. The SURPAS enforcement function defines SURPAS run-time policy-enforcement mechanisms. The SURPAS administration function defines mechanisms for managing SURPAS data like semantic annotations of resources and operations, domain ontologies, ontology-based policies, configuration settings for the enforcement function, etc. Each function is further decomposed into the functional semantics, algorithms for specific tasks, the abstract architecture based on ontologies and abstract use cases, and the reference implementation.

The functional semantics is an abstract specification of functionality. It precisely defines semantics of enforcement and administrative functions that change the state of the UBIWARE-based system to keep it consistent and secure.

The algorithms define explicit step-by-step procedures to perform or compute enforcement and administrative functions, mathematically specified in the functional specification, that are complex and for which the solutions are not obvious from their abstract specifications. Envisioned algorithms for rigorous study include: semantic annotating of requests; retrieving relevant domain ontologies; taxonomic and faceted classification of subjects, operations, and objects of access; retrieving relevant policy statements; resolving conflicts in relevant policies; and access control decision making.

The abstract architecture is an upper view on architectural components of SURPAS and interactions between them. This abstract description captures and reveals only fundamental elements and relations. Basically, the abstract architecture is a bridge between theoretical findings and adoption of SURPAS into practice.

The reference implementation takes the form of a set of software components. It serves for research and development purposes to prototype and test characteristics of proposed ideas, to generate feedback for upper components of SURPAS for further refinement, to make SURPAS tangible for better understanding and evaluating of proposed research ideas, and to implement components for possible reuse in industrial domains.

The UBIWARE adoption domains define industrial application areas of SURPAS. For example, a business of PMMS in the pulp&paper industry serves as a good domain for adopting UBIWARE. In this case, SURPAS will be important for managing access to control systems of maintained machinery equipment (Naumenko, 2007) in the business network of customers. The second promising application area is a business of decentralized management of power-networks owned by different business actors. Use of SURPAS in UBIWARE applications aligns SURPAS with the real world needs and issues. Merged semantics of domains and of SURPAS will be the result of merging or/and mapping of SURPAS ontologies with domain ontologies.

The SURPAS methodology is the formally described system of principles, practices and procedures that guides applying the SURPAS in concrete industrial cases of UBIWARE.

Another perspective on the SURPAS research framework would divide the research results according to the main areas of information and system security, namely access control, secure communication, privacy and trust. It is useful to consider along these dimensions all of the conceptual and functional semantics, algorithms, components of the abstract architecture and the reference implementation. In addition, architectural components of UBIWARE define the third, architectural, perspective on the SURPAS framework.

3.5 SURPAS Conceptual Semantics for access control

3.5.1 Model-theoretic semantics of OWL

The model-theoretic semantics of SURPAS is an extension of the semantics defined in the OWL standard. The OWL has two specifications of model-theoretic semantics: one is direct model-theoretic semantics and it should be referred by default as we do and the second is RDF-compatible model-theoretic semantics of OWL. Description of OWL semantics consists of four parts: formal specification of vocabularies and interpretations, interpreting embedded constructs, interpreting axioms and facts, and interpreting ontologies. Table 3.1 shortly presents the direct model theoretic semantics of vocabularies and interpretations of OWL (Patel-Schneider et al., 2004).

Table 3.1 – Direct model-theoretic semantics of vocabularies and interpretations of OWL

An OWL vocabulary and a datatype map	
$V \subseteq V_L \cup V_C \cup V_D \cup V_I \cup V_{DP} \cup V_{IP} \cup V_{AP} \cup V_O$ $V_C \supseteq$ {owl:Thing, owl:Nothing}	V is an OWL vocabulary. V_L is a set of the literals. V_C is a set of URI references of the classes. The class with qualified name owl:Thing is the class of all individuals. The class with qualified name owl:Nothing is the empty class. V_D is a set of URI references of the built-in OWL datatypes and rdfs:Literal. V_I is a set of URI references of the individuals. V_{DP} is a set of URI references of the data-valued properties. V_{IP} is a set of URI references of the individual-valued properties. V_{AP} is a set of URI references of the annotation properties. V_O is a set of URI references of the ontologies. V_{OP} is a set of URI references of the built-in OWL ontology properties.
D	A datatype.
L(d)	A lexical space, which is a set of Unicode strings.
V(d)	A value space.
L2V(d):L(d) $\rightarrow$ V(d)	A total mapping from the lexical space to the value space.
D:$V_D \rightarrow$ LV	A datatype map that is partial mapping from URI references to the corresponding XML Schema datatypes (may contain other datatypes as well).
An abstract OWL interpretation	
I=<R, EC, ER, L, S, LV>	An abstract OWL interpretation with respect to D with vocabulary V.
R	The domain of discourse or universe. It is nonempty set that contains denotations of URI references and literals in vocabulary V.

EC:$V_C \rightarrow P(O)$ EC:$V_D \rightarrow P(LV)$ EC(owl:Thing)=O EC(owl:Nothing)={} EC(rdfs:Literal)=LV $O \subseteq R$	A mapping from URI references of classes and datatypes to the corresponding class extensions and sets of literal values. O is the class extension of owl:Thing which comprises individuals of the R.
ER:$V_{DP} \rightarrow P(O \times LV)$ ER:$V_{IP} \rightarrow P(O \times O)$ ER:$V_{AP} \cup$ {rdf:type} $\rightarrow P(R \times R)$ ER:$V_{OP} \rightarrow P(R \times R)$	A mapping from URI references of data-valued, individual-valued, annotation, ontology properties and the rdf:type property to the corresponding class extensions.
L:TL $\rightarrow$ LV	A mapping from typed literals of V_L to literal values LV.
S:$V_C \cup V_D \cup V_I \cup V_{DP} \cup V_{IP} \cup V_{AP} \cup V_O \cup$ {owl:Ontology, owl:DeprecatedClass, owl:DeprecatedProperty} $\rightarrow$ R $S(V_I) \subseteq O$ S("I")=I, I a plain literal without language tag S("I"@t)=I, I a plain literal with language tag S("I")=L("I"), I a typed literal	A mapping from URI references in vocabulary V to their denotations in R. S is extended to plain literals in V_L by mapping them onto themselves
$LV \subseteq R$	Literal values of I. It contains the set of Unicode strings, the set of pairs of Unicode strings and language tags, and the value spaces for each datatype in D.

3.5.2 Model-theoretic semantics of SURPAS focusing Access Control

Use and compliance to the direct model-theoretic semantics of OWL allow relatively simple introducing of vocabularies and interpretations of specific concepts of SURPAS ontologies preserving and inheriting all features of OWL.

For the specification of the model-theoretic semantics we use URI references for names that are abbreviated to qualified names by using a namespace and a name of concept. The table 3.2 defines namespace names for used in the paper namespaces.

Table 3.2 – Namespace names

Namespace name	Namespace
surpas	http://www.cc.jyu.fi/~annaumen/surpas.owl#
sbac	http://www.cc.jyu.fi/~annaumen/sbac.owl#
sbacpriv	http://www.cc.jyu.fi/~annaumen/sbacpriv.owl#

sbacproh	http://www.cc.jyu.fi/~annaumen/sbacproh.owl#
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs	http://www.w3.org/2000/01/rdf-schema#
owl	http://www.w3.org/2002/07/owl#

Before defining concepts needed to specify access control policies in SURPAS, Semantics-Based Access Control (SBAC), we think it might be useful to define generic and abstract concepts on the upper level of SURPAS. This is needed to allow reusing of our research outcomes in other areas of security than access control. The table 3.3 introduces a stub of upper model for SURPAS.

Table 3.3 – Model-theoretic semantics of vocabularies and interpretations of SURPAS

A SURPAS extension to the OWL vocabulary	
$V_C \supseteq$ {surpas::SecurityStatement}	SURPAS class of security statements belongs to the OWL vocabulary of URI references of classes
$V_{IP} \supseteq$ {surpas:subject, surpas:predicate, surpas:object}	SURPAS properties belong to the vocabulary of URI references of individual-valued properties
$V_{SS} \subseteq V_I$	URI references of the security statements that are OWL individuals
A SURPAS extension to the OWL interpretation	
SS=EC(surpas:SecurityStatement)=S(V_{SS}), $SS \subseteq O$	A set of security statements.
subject(SS, O, ER(surpas:subject)) predicate(SS, O, ER(surpas:predicate)) object(SS, O, ER(surpas:object))	Binary relations subject, predicate, and object defined from the set of security statements to the set of discourse except literal values by the relation graphs that are class extensions of SBAC properties surpas:subject, surpas:predicate, and surpas:object respectively.

The set of security statements SS and three relations define a generic structure for specification of statements related to security like privileges, prohibitions, obligations for access control, trace statements for logging and audit, reputation statements and trust agreement statements for trust management, etc. The scope of this paper encompasses semantics of access control statements that is presented in the table 3.4.

Table 3.4 – Model-theoretic semantics of vocabularies and interpretations of SBAC

A SBAC extension to the OWL vocabulary	
$V_{CR} \subseteq V_C$	URI references of the resources classes.
$V_{CO} \subseteq V_C$	URI references of the operations classes.
$V_C \supseteq$ {sbac:ClassOfResources, sbac:ClassOfOperations, sbac:AccessControlStatement}	SBAC classes belong to the OWL vocabulary of URI references of classes

$V_{IP} \supseteq$ {sbac:subject, sbac:operation, sbac:object}	SBAC properties belong to the vocabulary of URI references of individual-valued porperties
$V_{ACR} \subseteq V_I$ $V_{ACO} \subseteq V_I$ $V_{ACS} \subseteq V_{SS} \subseteq V_I$	URI references of the access control resources, operations and statements that are OWL individuals
A SBAC extension to the OWL interpretation	
ACR=EC(V_{CR})=S(V_{ACR}) ACR $\subseteq$ O	A set of resources that could be subjects and objects of access control. ACR is union of class extensions of all classes of resources that correspond to URI references of the vocabulary V_{CR} and it is the set of denotations of all URI references of vocabulary V_{ACR}.
ACO=EC(V_{CO})=S(V_{ACO}) ACO $\subseteq$ O	A set of operations.
ACS=EC(sbac:AccessControlStatement)= =S(V_{ACS}), ACS $\subseteq$ SS $\subseteq$ O	A set of access control statements.
CR $\subseteq$ P(ACR) CR=EC(sbac:ClassOfResources)	A set of subsets of resources defined by URI references of vocabulary V_{CR}.
CO $\subseteq$ P(ACO) CO=EC(sbac:ClassOfOperations)	A set of subsets of operations defined by URI references of vocabulary V_{CO}.
acSubject(ACS, CR, ER(sbac:subject)) acOperation(ACS, CO, ER(sbac:operation)) acObject(ACS, CR, ER(sbac:object)) acSubject $\subseteq$ subject acOperation $\subseteq$ predicate acObject $\subseteq$ object	Binary relations access control subject, operation, and object defined from the set of access control statements to the set of classes of resources, operations, and resources by the relation graphs that are class extensions of SBAC properties sbac:subject, sbac:operation, and sbac:object respectively. Introduced relations are subsets of corresponding relations from the SURPAS extension.

The ACR is a set of individual resources. A resource is an entity of physical or digital world that is a subject or an object of access. Definition of the resource as a set for subjects and objects gives more flexibility in access control rights specification because it is hard to separate resources on passive and active in environments where artificial resources play active roles and their relations to human users are weak or are not present. This means that a resource may operate not on behalf of a user but because of its own initiative. Researches in agent technologies and machine to human interaction are good sources for more details on this topic. The ACO is a set of individual operations that could be actions, transactions, access modes, etc.

Resources and operations are classified and collected to named sets that form hierarchies of resources and operations. The CR is the set of subsets of resources and the CO is the set of subsets of operations. Those sets are partially ordered by the transitive subset relation that is subClass property in the OWL.

The ACS is a set of access control statements that denote a many-to-many abstract relation between subject, operation and object of access using three binary relations from access

control statements to subject resources, operations, and object resources. The ACS specializes semantics of the generic structure of different purpose security statements. The main feature of the access control statement semantics and the whole SBAC is that security-related statements mentioned above are specified between classes instead of individuals. In their turn, specializations of ACS define concrete semantics in the submodels, for example privilege, obligation and prohibition statements. The table 3.5 presents semantics of privilege concept and authorization rule for policies that use only privileges.

Table 3.5 – Model-theoretic semantics of interpretations of SBAC privileges

A SBAC Privilege extension to the SBAC interpretation	
PRIV $\subseteq$ ACS PRIV=EC(sbacpriv:Privilege)	A set of privilege statements that is a subset of the set of access control statements.
Access: ACR $\times$ ACO $\times$ ACR $\rightarrow$ Boolean Access(s, o, ob)=($\exists$ priv $\in$ PRIV, s $\in$ subject(priv), o $\in$ operation(priv), ob $\in$ object(priv))	An authorization rule for policies with privilege statements only.

The PRIV is a subset of ACS and denotes the set of individual privileges of access control. A privilege is an authorization of resources to access other resources using some operations. A decision of access granting or prohibiting depends on memberships of subject, operation and object elements in sets that are in definitions of privileges. The decision algorithm evaluates the containment relation between individual elements and sets taking into account partial order of sets. There is a possibility to define the membership relation of elements using only leaf sets from the taxonomy of sets while the decision algorithm can infer the membership of elements to other sets based on the subClass relation that forms hierarchy of sets.

Support of only positive authorizations in the form of privileges guaranties a conflicts free specification of access control policies. However, even in this case the model has an implicit prohibition that everything is prohibited unless it is privileged. Introducing means for the specification of prohibitions (Table 3.6) in the SBAC model enhances expressivity of the policy language to make negative authorizations explicit.

Table 3.6 – Model-theoretic semantics of interpretations of SBAC prohibitions

A SBAC Prohibition extension to the SBAC interpretation	
PROH $\subseteq$ ACS PROH=EC(sbacproh:Prohibition)	A set of prohibition statements that is a subset of the set of access control statements.
Access: ACR $\times$ ACO $\times$ ACR $\rightarrow$ Boolean Access(s, o, ob)= $\neg$ ($\exists$ proh $\in$ PROH, s $\in$ subject(proh), o $\in$ operation(proh), ob $\in$ object(proh))	An authorization rule for policies with prohibition statements only.

It is evident that policies with privileges and prohibitions are not free from conflicts in an arbitrary case. These policies require mechanisms to resolve conflicts and ambiguity for the guarantied decidability. Following the fundamental principle of access control for ensuring

confidentiality, prohibitions always precede privileges. For example, block lists in mobile phones prohibit accepting calls from given phone numbers while there is a general implicit privilege to accept calls from everybody. Note, for this example prohibitions are mostly used to specify policies in the form of block lists. Although in the most cases policies will follow the fundamental principle, there is a need to specify the precedence (Table 3.7) between privileges and prohibitions to facilitate at least the explicit specification of the fundamental prohibiting principle with the further precedence of privileges.

Table 3.7 – Model-theoretic semantics of interpretations of SBAC precedence

A SBAC Precedence extension to the SBAC interpretation	
precedes(P(ACS), P(ACS), ER(sbac:precedes))	A binary relation that denotes precedence in specifications of policies between sets of access control statements.
Access: ACR × ACO × ACR → Boolean Access(s, o, ob)= (∃ priv ∈ PRIV, s ∈ subject(priv), o ∈ operation(priv), ob ∈ object(priv), ((precedes(PRIV,PROH) ∨ precedes(PROH,PRIV), ¬ (∃ proh ∈ PROH, s ∈ subject(proh), o ∈ operation(proh), ob ∈ object(proh))))	An authorization rule for policies with privilege and prohibition statements.

The notion of obligations in the SBAC model supports a provisional authorization model where policies define provisional operations that must be executed to fulfill conditionally positive access control decisions or/and to supplement negative decisions. Obligations may conflict with privileges and prohibitions. There are conflicts when obligations dictate to take unprivileged or even prohibited actions. The above defined binary relation helps to configure the actual precedence between privileges, permissions, obligations and other possible access control statements for particular cases. However the provisional authorization and obligations are targets for our research during the next iterations. FIGURE 3.7 presents the core part of the SBAC model.

The SBAC interprets the facts, axioms and ontologies as defined by the OWL direct model-theoretic semantics. Notable and important interpretations of OWL for the SBAC are provided briefly below. The OWL provides a possibility to specify sets of individuals using descriptions. Descriptions are axioms and they include class identifiers, restrictions and boolean combinations of other descriptions. Boolean combinations are union, intersection, and complement. Restrictions are placed on properties and called also facets. For example, a description "intersectionOf({domain:Student, restriction(domain:status value(domain:visiting)})" denotes a set of individuals that have class "domain:Student" as their type and satisfy the restriction: property "domain:status" has value "domain:visisting" where "domain" is the namespace of domain ontology. Descriptions allow flexible specification of access control policies for further inferring access control statements applicable to individual resources and operations based on their taxonomic and faceted classifications.

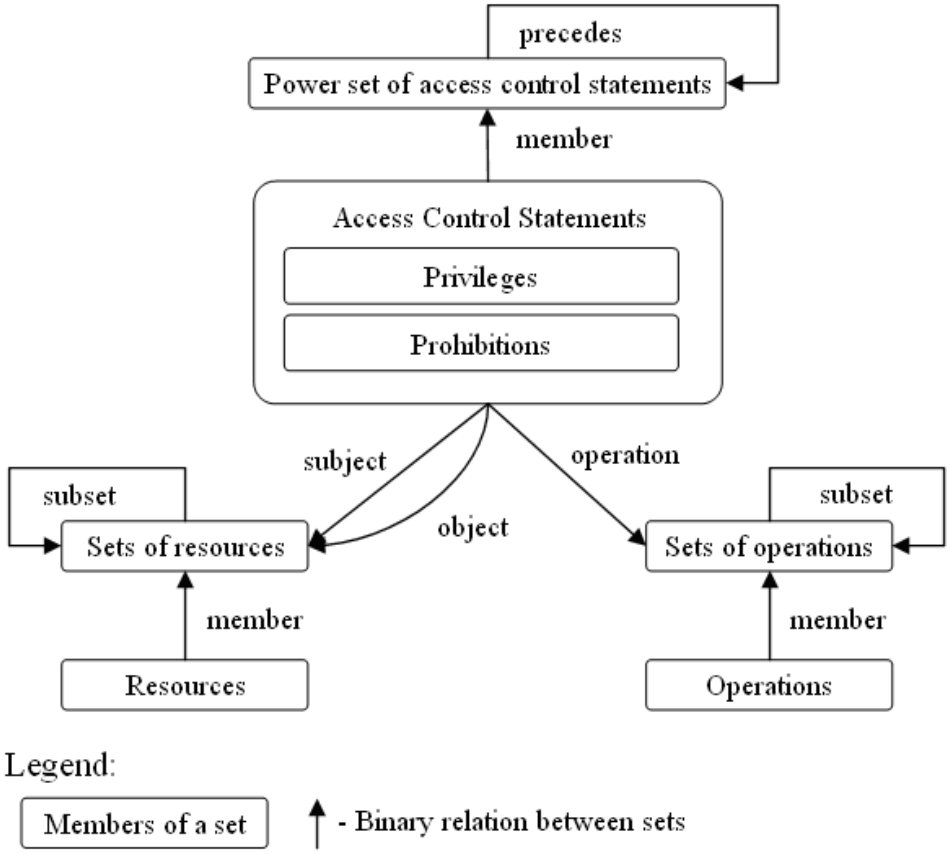

FIGURE 3.7 The SBAC model

Taxonomic classification relies on the class extensions of several relations: the sets containment relation rdf:type, the subset partial order relation rdfs:subClassOf and the subset partial order relation for binary relations rdfs:subPropertyOf. Last two relations form taxonomies of classes and properties. The faceted classification is a sophisticated alternative to the traditional classification schemes. In the OWL facets are sets of individuals defined by restrictions on their properties. Another useful OWL feature for organizing access control statements is specification of an enumerated class as subject, operation or object of access by the explicit specification of all individual members.

Interpretation of ontologies is the key issue for evolution, consistency, reasoning and organising features of SBAC, domain knowledge and concrete policies separately in different ontologies for flexible and joint further use with the high conceptual granularity. Annotation and ontology properties, namely owl:versionInfo, owl:priorVersion, owl:backwardCompatibleWith, and owl:incompatibleWith, help to record a history of evolution of SBAC and domain onotlogies, policies, trust agreements, etc. Ontology property owl:imports gives the extra effect of importing the contents of target ontology into the current ontology. The OWL standard (Patel-Schneider, 2004) defines conditions when an abstract OWL interpretation satisfies an OWL ontology. The definitions of when and how a collection of ontologies and axioms and facts is consistent and entails an ontology or axiom or fact provide background for reasoning and maintaining integrity of SBAC data.

3.5.3 The SURPAS ontologies

There are several alternatives to define the SURPAS ontologies because the OWL has different syntaxes that are useful for different purposes. Abstract syntax of the OWL is more appropriate and useful for access to and evaluation of SURPAS comparing with the exchange syntax. Purpose of abstract syntax is informal specification of ontologies that facilitates analysis of concepts and relations.

A regular OWL ontology consists of annotations, axioms, and facts. Annotations carry information about authorship, versioning and other data associated with an ontology and concepts. Facts and axioms provide information about classes, individuals and properties that form main content of an ontology. An ontology can have name that is intended to be the address where it can be found, although this is out of formal semantics.

The SURPAS ontology defines three classes and three individual-valued properties with explicit definition of their names (note: OWL allows defining anonymous concepts). Specification of classes and properties consists of axioms that associate concepts' identifiers with the specification of their characteristics, for example that surpas:subject, surpas:predicate and surpas:object properties have surpas:SecurityStatement class as their domains.

```
Ontology(surpas:ontology
Class(surpas:SecurityStatement)
ObjectProperty(surpas:subject
domain(surpas:SecurityStatement))
ObjectProperty(surpas:predicate
domain(surpas:SecurityStatement))
ObjectProperty(surpas:object
domain(surpas:SecurityStatement)))
```

The OWL standard gives the formal semantics (Patel-Schneider et al., 2004) of operation for importing ontologies. The SBAC ontology imports the SURPAS ontology in order to specialize the security statement and three relations (Naumenko, 2007). The introduced class for access control statements is a sub class of security statements. Subject, operation and object relations of access control statements are sub properties of corresponding relations from the SURPAS ontology. The SBAC ontology defines also restrictions on these relations that their values must be classes of resources and operations respectively. For this purpose there are two sub classes of the owl:Class concept that denote the class of resources and class of operations. Finally, there is an axiom defining relation of precedence between privileges, prohibitions and obligations.

```
Ontology(sbac:ontology Annotation(owl:imports
surpas:ontology)
Class(sbac:ClassOfResources partial owl:Class)
Class(sbac:ClassOfOperations partial owl:Class)
Class(sbac:AccessControlStatement partial
surpas:SecurityStatement
restriction(sbac:subject
allValuesFrom(sbac:ClassOfResources))
restriction(sbac:operation
allValuesFrom(sbac:ClassOfOperations))
```

```
restriction(sbac:object
allValuesFrom(sbac:ClassOfResources)))
ObjectProperty(sbac:subject super(surpas:subject)
domain(sbac:AccessControlStatement)
range(sbac:ClassOfResources))
ObjectProperty(sbac:operation super(surpas:predicate)
domain(sbac:AccessControlStatement)
range(sbac:ClassOfOperations))
ObjectProperty(sbac:object super(surpas:object)
domain(sbac:AccessControlStatement)
range(sbac:ClassOfResources))
ObjectProperty(sbac:precedes))
```

The SBAC privilege and prohibition ontologies import the SBAC ontology in order to extend it with only one class axiom that defines the class of privilege statements or the class of prohibition statements. These classes are subclasses of the abstract class of access control statements. Thus the classes inherit all properties defined for the class of access control statements. The individuals of these classes are positive and negative authorizations that define classes of resources that may or cannot perform access using operations from specified classes of operations to resources from specified classes of resources.

```
Ontology(sbacpriv:ontology Annotation(owl:imports
sbac:ontology)
Class(sbacpriv:Privilege partial
sbac:AccessControlStatement))

Ontology(sbacproh:ontology Annotation(owl:imports
sbac:ontology)
Class(sbacproh:Prohibition partial
sbac:AccessControlStatement))
```

Other ontologies are to be developed to implement features of SURPAS like context-awareness, rules, trust management, exchange of access control assertions, semantic logging and audit, etc. However those are subjects for further research.

FIGURE 3.8 illustrates the concepts of SURPAS ontologies, importing mechanism amongst the ontologies and possible policies commitments to different SBAC features introduced in the paper. Ontologies for the policies A and D are able to define only privilege and only prohibition statements respectively. The policies B and C may contain both types of statements. Privileges have precedence over prohibitions in the policy B and prohibitions precede privileges in the policy C.

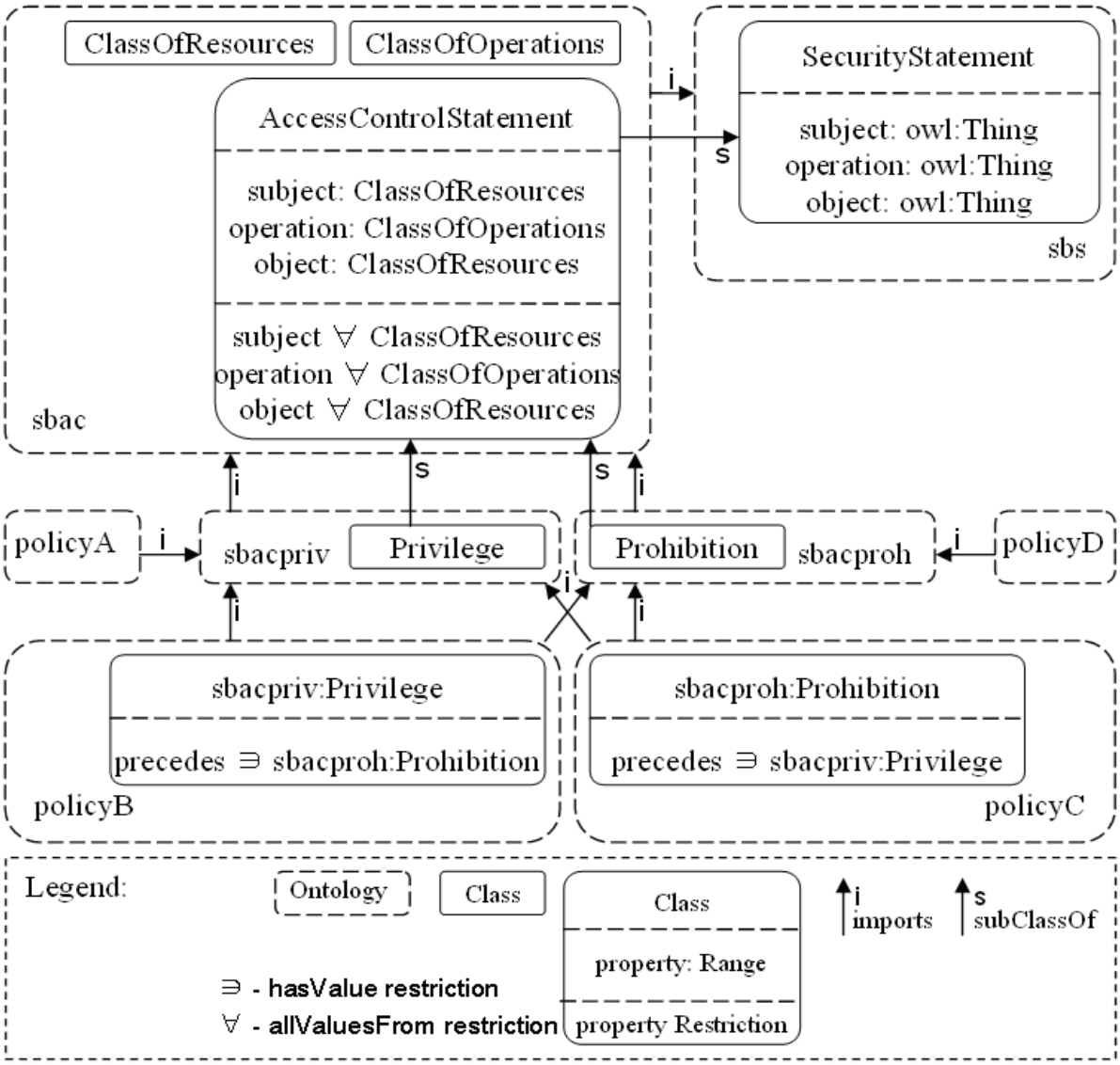

FIGURE 3.8 The SBAC Ontologies

3.5.4 The role of ontologies in SURPAS

The SBAC ontologies consolidate and formally specify knowledge of the access control research and development domain. This means that SBAC ontologies represent and organize mainly already formalized knowledge in existing access control models. Similarly to traditional access control models, SBAC ontologies aim to support the specification of policies accordingly to standards, legal regulations and commonly accepted in a domain of interest practices, agreements, approaches, etc. The whole framework generally and SBAC ontologies in particular reuse achievements in the Semantic Web research and development area. FIGURE 3.9 illustrates the place and role of SBAC ontologies.

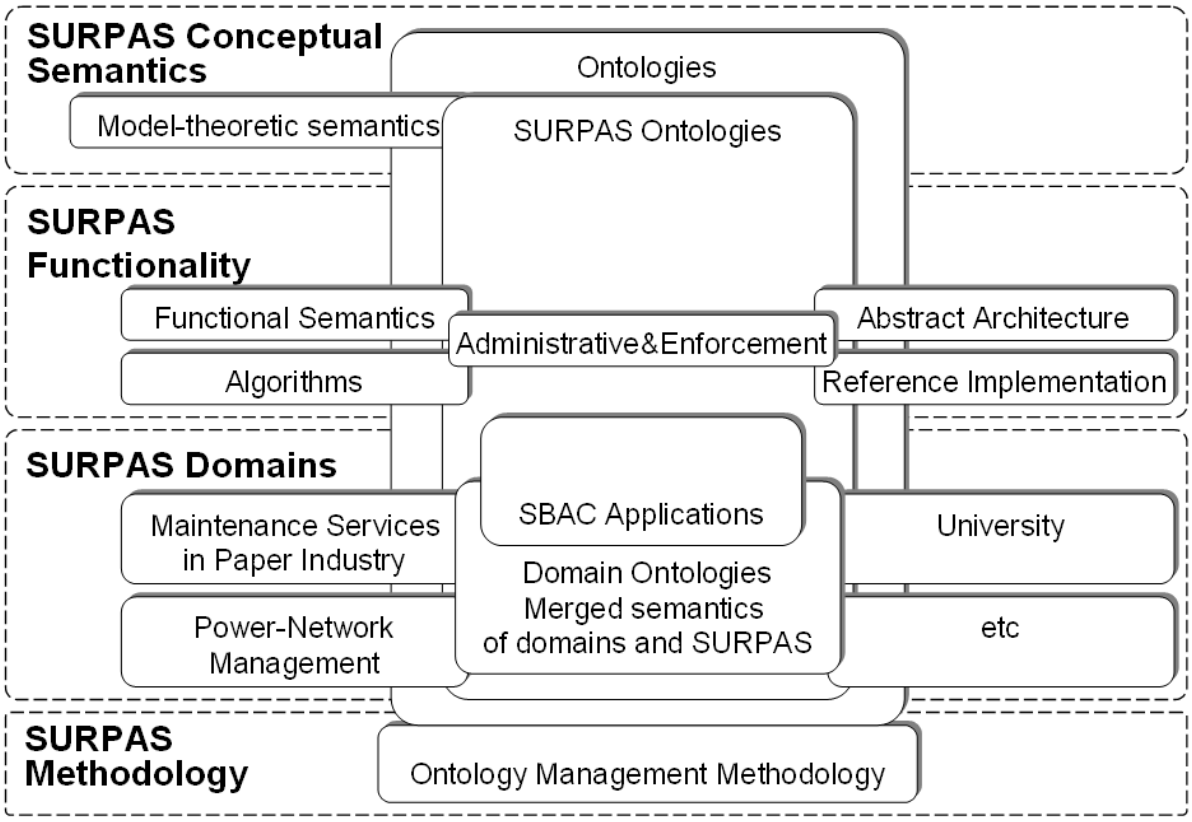

FIGURE 3.9 The place and role of the SURPAS ontologies

SURPAS ontologies and Semantic Web standards shape the abstract design of the SBAC mechanism that is specialized to concrete designs for particular technologies and paradigms of ICTs. The design of the SBAC mechanism follows the Ontology-Driven Architecture (ODA) (Tetlow et al., 2006) to enforce and administrate policies of organizations specified by ontologies.

The main role of SBAC ontologies is to provide means for the ontological domain modeling. The SBAC ontology specifies upper concepts and other SBAC and domain ontologies import it. Domain ontologies import the policy ontology because it has concept of privilege that is necessary to encode authorizations. Although it is out of scope of this paper, domain ontologies may import other SBAC ontologies that define for example how to specify trust statements, rules, contextual information, etc.

3.5.5 An example of the specification of domain ontology

The domain ontology formally specifies concepts of domain, resources, operations, privileges and their semantic relations. It imports the SBAC and policy ontologies. FIGURE 3.10 shows an example of specification of privileges for users to print on printers based on their relations to an organizational structure. The individual organizational units form the hierarchy by the partial order relation part-whole defined as the transitive property partOf.

Units are superior to their parts. The example does not have hierarchy of operations and it defines the only one individual operation that denotes printing.

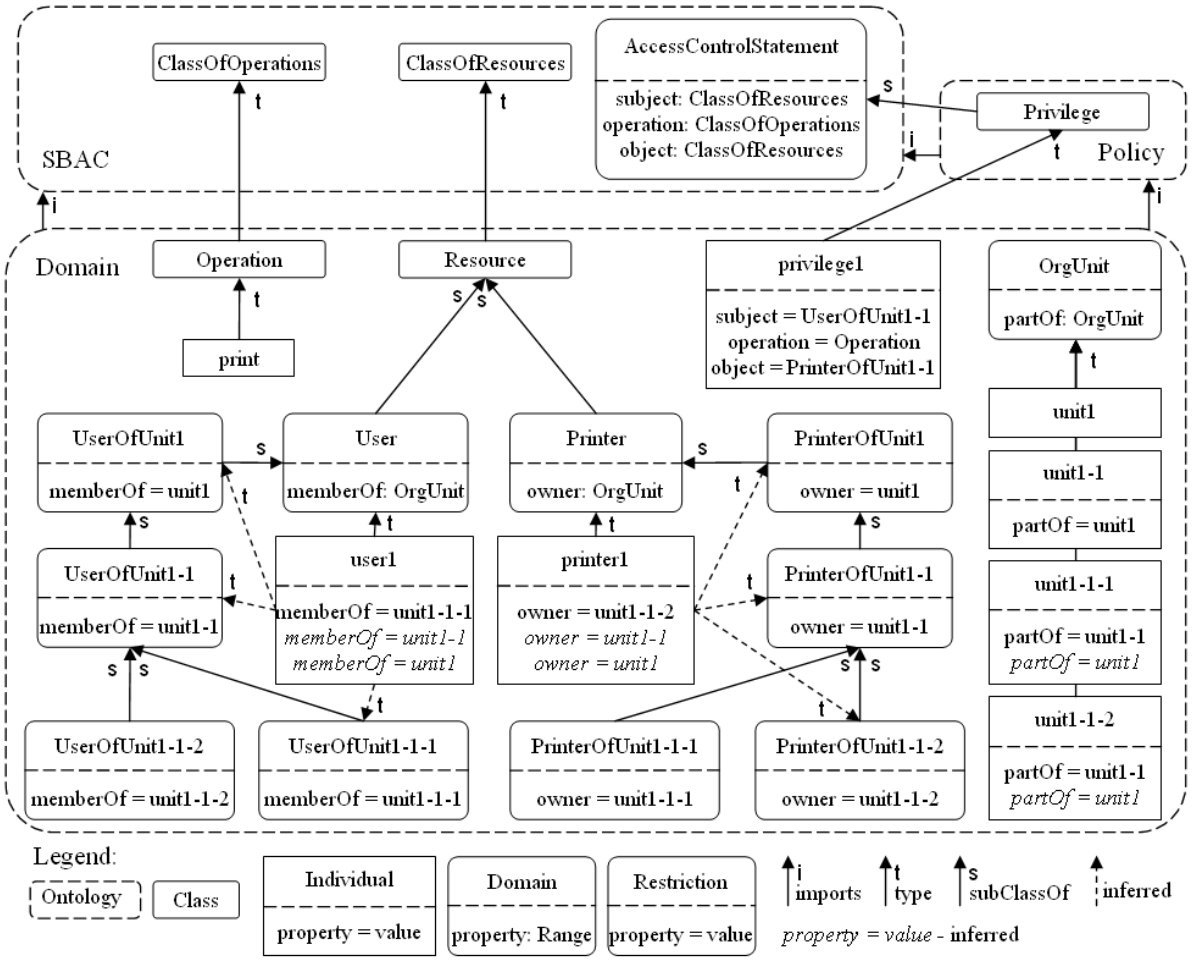

FIGURE 3.20 The example of the domain ontology

Hierarchies of classes of resources are more advanced. They are formed by the subClassOf transitive relation to reflect partially ordered sets of printers and users. An interesting thing is the combination of taxonomic and faceted classification of individual users and printers. The domain ontology has the class of all users and the property of users' membership in organizational units. Let in this domain a user be a member of all units which have as a part specified (direct) unit of a user. Thus as long as the range of the membership property is partially ordered to the hierarchy of organizational units, restrictions (facets) on this property must form the hierarchy of sets of users that is isomorphic to hierarchy of organizational units. The individual user of some unit belongs to the facet with this unit as fixed value for the membership property. This is the example of faceted classification. Other facets to which the user belongs are inferred based on the hierarchy of facets (taxonomic classification). The user also gets inferred assignments of units to the membership property. The same explanation holds for the specification and classification of individual printers.

The actual specification of the authorization for users of unit1-1 to print on corresponding printers looks simple comparing to the previous description of taxonomic and faceted classification of resources. The authorization rule for the tuple <user1, print, printer1> is true because privilege1 exists with subject as UserOfUnit1-1, operation as Operation, object as PrinterOfUnit1-1 and user1 belongs to the set denoted by the class UserOfUnit1-1, print belongs to the set denoted by the class Operation, printer1 belongs to the set denoted by the class PrinterOfUnit1-1.

3.6 SURPAS Architecture

This section presents the research results towards the abstract architecture of SURPAS, piloting and testing this abstract architecture, and designing secure SmartResource agent.

3.6.1 Abstract Architecture

The abstract architecture is an upper view on the components of SURPAS and interactions between them. This abstract design captures and reveals only fundamental elements and relations taking into account security patterns (Mazhelis and Naumenko, 2006). Basically, the abstract architecture is the main bridge between theoretical findings and adoption of SURPAS into practice because the abstract architecture integrates the research on theoretical issues with practical concerns and with the development of applications. Another role of the abstract architecture is to ensure interoperability and reusability for SURPAS. The abstract architecture consists of abstract design of common or shared characteristics of SURPAS that can be formally related to every valid SURPAS implementation. Possible concrete designs will be interoperable and will reuse reference implementations because of the shared abstract design.

The abstract architecture is a part of the SURPAS functional semantics. In addition to the abstract architecture, the SURAPS functional semantics includes other research components, i.e., formal specification of functionality, algorithms and reference implementations. The abstract architecture is closely related to these components. The formal specification of functionality and algorithms provide functional requirements for the design of abstract architecture. Then, the abstract architecture serves as an input to the process of piloting and testing research ideas. Due to the central role of ontologies in SURPAS, the abstract architecture should follow Ontology-Driven Architecture (ODA) paradigm (Tetlow et al., 2006) of software design. ODA is an emerging and immature research target. This is an additional challenge in tackling this research component.

The abstract architecture of access control in SURPAS, SBAC, reflects two main functions: the administrative function and the run-time authorization function, that is also called enforcement function or access control mechanism. The SBAC enforcement function defines an access control policy enforcement mechanism in SBAC. The enforcement function controls run-time access of requestors to protected resources according to ontology-based access control policies, credentials of requestors, attributes of objects and operations

using algorithms of SBAC. The SBAC administration function defines mechanisms of manipulation with the SBAC data including semantic annotations of resources and operations, domain ontologies, ontology-based policies, configuration settings for the enforcement function, and other. We concentrated on the SBAC enforcement function during our research.

The really first step towards the abstract architecture is identifying the common components and characteristics that any SBAC implementations would have. These generic components and characteristics form the scope of abstract architecture. The next step is to integrate revealed components and characteristics in architectural abstractions. After that some crucial components and features have to be prototyped for rationality and feasibility study. The list of some generic elements is created taking into account security patterns [17-20] and the proposal to use advances of Semantic Web.

- A subject of access is an active resource that requests some operation over other resources. It is an actor in any authorization pattern also known as the client or requestor. SBAC profiles and domains specialize the notion of entities that can be subjects of access while the abstract architecture deals with the notion of abstract subjects.
- An operation of access is an abstract concept that generalizes in the abstract architecture access types, operations, methods, procedures, transactions, etc used in profiles and domains.
- An object of access abstractly represents a protected object or passive resource in other words.
- A guard (also known as protected system, single access point, checkpoint, enclave, reference monitor, policy enforcement point) is a widely used component that mediates access to protected resources by enforcing rules of corresponding access control policies. The abstract architecture requires that the guard must evaluate all requests, correctly evaluate policies, be incorruptible, and not be bypassable. There is a single centralized guard that mediates access to all resources or a set of guards each for some distinct type of resource or even for individual resource. Guards may be proxies (Gamma et al., 1995) or embedded as part of protected resources. This impacts the performance and assurance of the system. The abstract architecture deals with guards as proxies. This is a more general case and if needed the functionality of guards can be integrated with resources. Peculiarity of profiles and domains determines concrete design solutions.
- A policy (access decision function, policy decision point) has a set of rules (called rights, permissions, prohibitions, authorizations, etc) that define which subjects may access which objects using which operations. It is advisable to use this pattern to isolate policy decision logic from resource and enforcement code. It might be infeasible to perform the decision making outside resources or enforcement components (guard). The abstract architecture defines that the policy is always an ontology or set of ontologies and it makes access decisions based on semantics of subjects, operations, objects, context, and policy rules of access.
- A rule is a part of policy and for the SBAC rules are represented by access control statements.
- A subject and object descriptors are well known patterns that provide access to relevant attributes of subject and objects of access respectively in situations when

checking attributes is independent from establishing them, there are different sources of attributes, different attributes are needed in different contexts, etc. The abstract architecture of SBAC deals with subject, operation, object, policy, and context descriptors in the form of semantic annotations. These annotations have to be protected because attributes may embody sensitive data too. We position the SBAC for open and dynamic environments thus the retrieval and creation of semantic annotations for involved entities is not a trivial task.

– Context is a container for the data that are relevant for the access control decision and enforcement relating to execution or administrative state of the environment, operation, session, etc.

We identified the common components and characteristics for the SBAC enforcement mechanism. These components are the proactive guard (ProGuard), the policy information retrieval component (PIR), the context information retrieval component (CIR), and the resource information retrieval component (RIR). ProGuard is the proxy and guard for protected resources and information retrieval components. ProGuard enforces an access decision based on the reasoning over the semantically encoded access control policy and the semantic annotations of a subject, an operation, an object and a context of access. ProGuard, driven by results of reasoning, collects all needed semantic annotations and policy rules to make an access decision for communicating with information retrieval components, thus acting proactively. The reasoner interactively provides instructions to get additional data for further reasoning or a decision about access finally. The PIR component provides semantic annotations of access control policies and of trust agreements between cooperative partners. The RIR and CIR components provide unified interfaces to access semantic annotations of resource's attributes and contextual data respectively. FIGURE 3.31 shows the SBAC abstract architecture of the SBAC enforcement mechanism. FIGURE 3.4 illustrates a control flow of ProGuard.

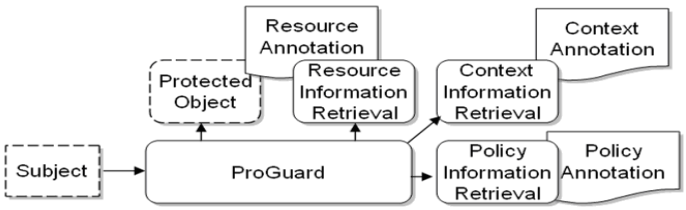

FIGURE 3.31 The abstract architecture of the SBAC enforcement mechanism

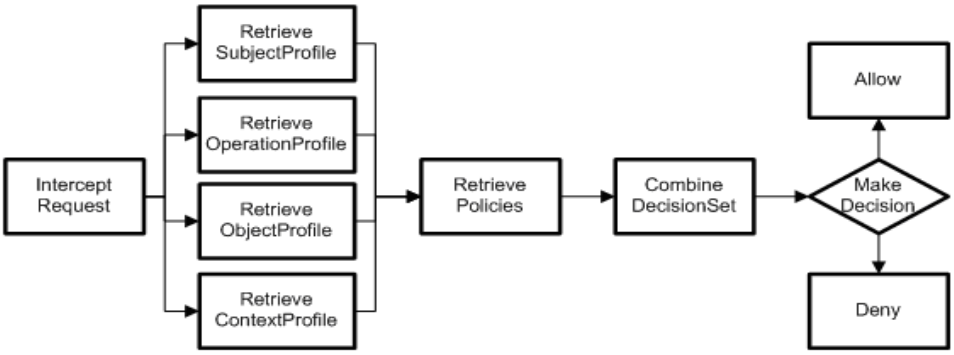

FIGURE 3.42 The SBAC enforcement procedure

3.6.2 Piloting the abstract design

The development environment consists of several interrelated elements shown in FIGURE 3.5. Java 2 standard edition development kit version 1.5 (http://java.sun.com/j2se/1.5.0/) is a programming language and platform that was chosen for the prototyping of research ideas. Jena (http://jena.sourceforge.net/) is a semantic web framework for java developed within the HP Labs Semantic Web Programme (http://www.hpl.hp.com/semweb/). ARQ (http://jena.sourceforge.net/ARQ/) is a SPARQL processor for Jena. SPARQL is a query language for the RDF developed by W3C (Prud'hommeaux and Seaborne, 2006). Eclipse (www.eclipse.org) is an open source community that produces extensible with huge amount of plugins integrated development environment (IDE). The last version of the IDE is 3.2. A UML modeling tool is an eclipse plugin EclipseUML 2.1 Free Edition (http://www.omondo.com/) produced by Omondo Inc. The Eclipse Test & Performance Tools Platform (TPTP, http://www.eclipse.org/tptp/) project consists of four subproject one of which provides tools for tracing and profiling java applications for further analysis of performance. The really first thing to do for piloting the abstract design is to create defined SBAC ontologies in the RDF/XML exchange syntax of OWL. For this purpose the protégé (www.protege.stanford.edu) is the most appropriate tool. The protégé is an open source and free ontology editor with the number of plugins for editing (Protege-OWL) and visualizing (Ontoviz, OWL Viz) OWL ontologies. Web server is a container for developed in the protégé SBAC, domain and policy ontologies that are accessible by the prototype from Internet through HTTP protocol.

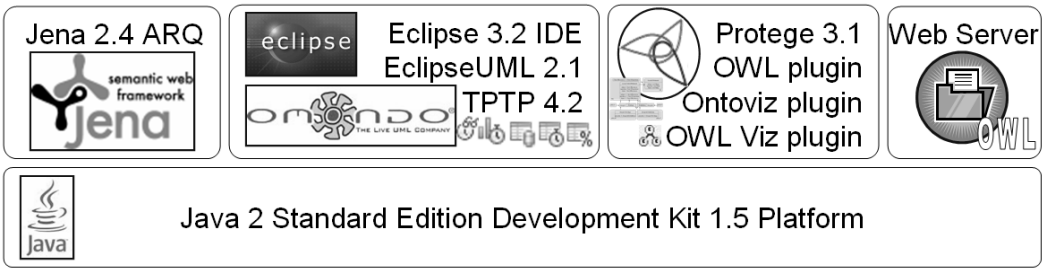

FIGURE 3.53 The development and testing environment

FIGURE 3.64 consists of two UML diagrams. The first diagram is a class diagram with upper classes of subjects, ProGuards and protected objects. Subjects can access other resources that are targets of access. Protected objects are protected by ProGuards that serve as proxies decorating (Gamma et al., 1995) operations provided by protected objects. ProGuards are resources and could be accessed by subjects the same way like protected objects. ProGuards are also protected objects in their turn for two reasons: they can be protected by other ProGuards like the second diagram shows and they have the same interfaces as objects they represent according to proxy and decorator design patterns. The second diagram is a collaboration diagram of objects that represent both mixed and nested structures of ProGuards. The hybrid structure means that subject first access an upper guard to get reference for access and lower level guards check request against policies. The nested structure of guards follows the secure proxy security design pattern (Blakley et al., 2004) and

is known under different names like defense in depth, single sign-on, delegation, security protocol encapsulation, tunneling, nested protected systems, etc.

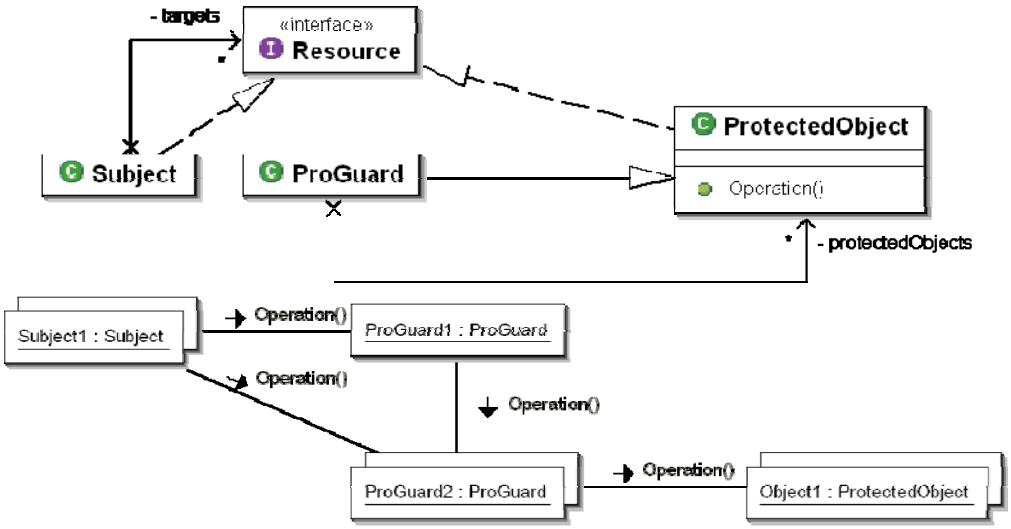

FIGURE 3.64 The upper class and collaboration UML diagrams.

ProGuard is a façade (Gamma et al., 1995)] for the whole complex systems that implements its functionality. The internal architecture of the ProGuard consists of the information retrieval components and decision maker. FIGURE 3.75 presents associations between the ProGuard and information retrieval components in the form of UML class diagram.

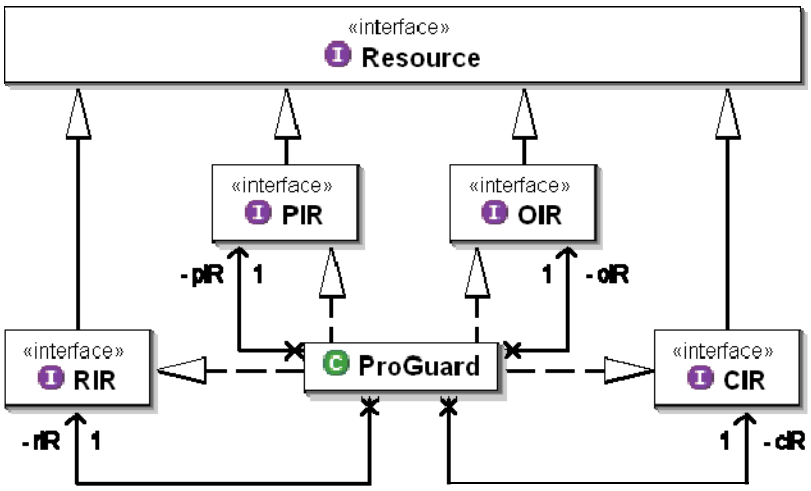

FIGURE 3.75 The ProGuard UML class diagram.

Being the facade for the whole subsystem the ProGuard provides interfaces of components and it delegates fulfillment of requests to the components that actually implement the interfaces. Moreover the ProGuard control access and protects the information retrieval components and itself as was above motivated during the description of abstract architecture. If the ProGuard is composed by the arbitrary number of information retrieval components of each type then it shall have additional means for coordination of delegation of calls. To make

this situation simpler at this level of design, the ProGuard can have only one information retrieval component for each type while they can be composed or linked to networks of information retrieval that is a subject of design on their levels.

3.6.3 Testing the pilot

The piloting of the abstract architecture was conducted with the main purpose to test performance of the SBAC enforcement mechanism and gather information for rationality and feasibility study of the whole vision. FIGURE 3.86 is a UML sequence diagram generated after profiling of the testing prototype run. This diagram shows the test sequence aggregated to the upper level objects that are the subject, protected object and ProGuard. The test application firstly creates three elements and then starts the sequence of one request from the subject to the ProGuard with the following evaluation of the request.

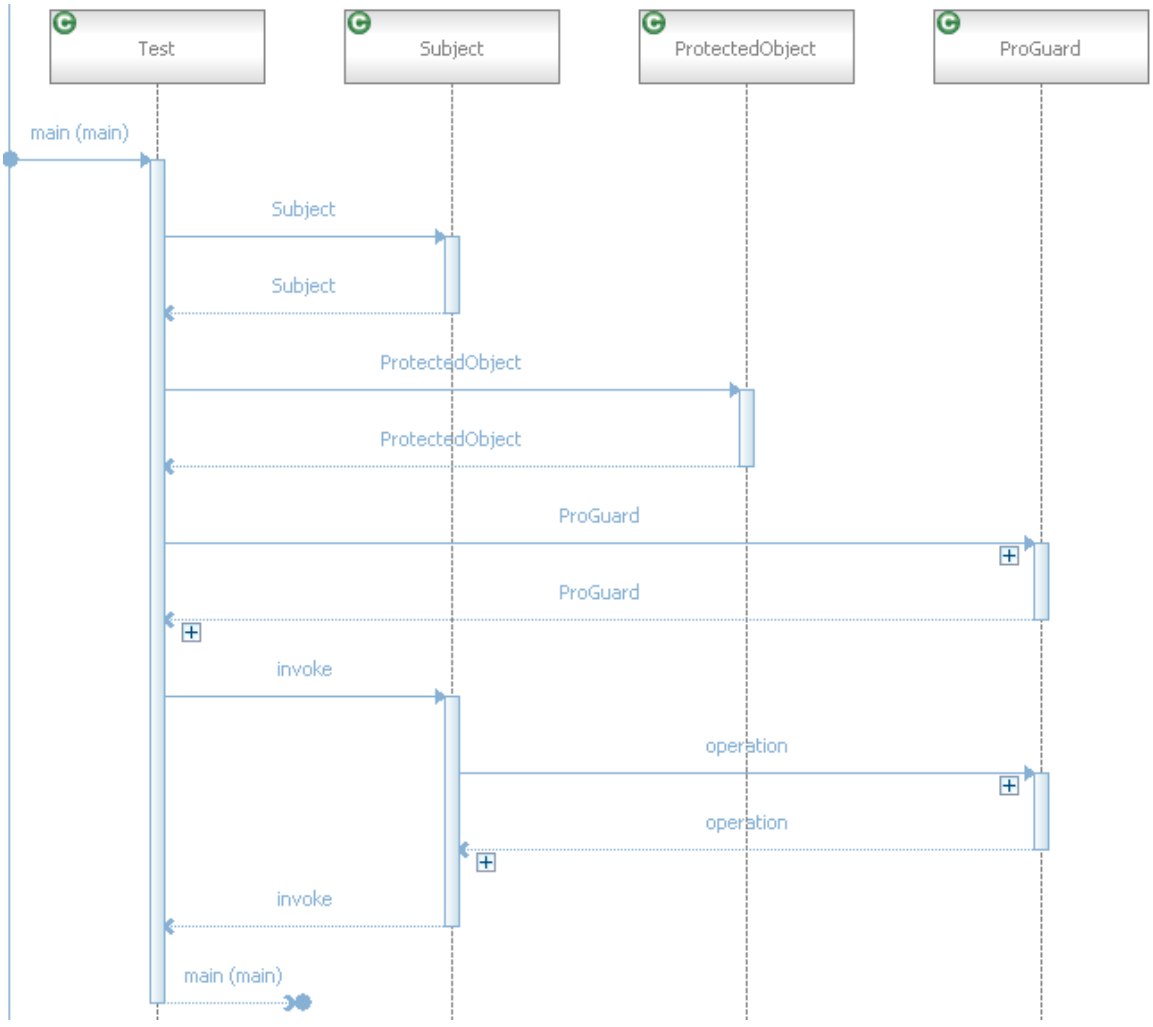

FIGURE 3.86 The testing run (aggregated upper sequence).

There are basically two control flows with distinct characteristics and impacts on the overall performance where monitoring and measuring is needed. The first is the process of starting up the ProGuard when it creates and initializes all internal components. The performance of this process is crucial for the fast restarting of the ProGuard in the case of restoration after fault or something else. Obviously the time at which the ProGuard starts is not as important and critical as the response time of run-time evaluating of requests. This is the second process and it occurs each time when subjects try to run operations over protected objects.
There are different factors that impact the performance of the SBAC enforcement mechanism. We have already identified above that the overall performance accumulates both performances of the start-up and evaluating processes:

- The performance of the start-up process is determined by the time of start-ups of ProGuard components. The information retrieval components have to initialize their sets of semantic annotations and information retrieval networks. The decision maker has to initialize the knowledge base with at least SURPAS ontologies in highly dynamic environments and can initialize the knowledge base with all ontologies and semantic annotations in the case of closed environments.
- The performance of the evaluating process is determined by the retrieving of semantic annotations in the case of dynamic environments and actual decision making. The decision making process is broken down to three activities. The first is combining the decision making knowledge base based on retrieved semantic annotations, applicable SURPAS, domain and policy ontologies that might be loaded partially during the start-up process. The second is preparing the query according to the request and SBAC authorization rules that correspond to applicable policies. The third activity is executing of the query against the prepared knowledge base using the query engine. FIGURE 3.97 shows the UML sequence diagram of the SBAC decision making control flow upon the request arrival from the subject.

The above breakdown of performance reveals major factors that influence performance of the SBAC enforcement mechanism:

- The size of knowledge base that is compounded of the number of instances, classes, properties and access control statements in the SBAC, domain and policy ontologies.
- Activities of initialization and preparation of the knowledge base for the decision making process can be allocated to the start-up and evaluating processes based on the availability of semantic annotations and ontologies in different environments. The allocation of these activities shapes the compromising balance between performances of the both processes.
- The complexity of the querying the knowledge base differs because of different complexity of the authorization rules for policies that commit to different features of the SBAC illustrated on the figure 4. Complexities of policies A and D are equal and correspond to the case when only either privileges or prohibitions are used. Complexities of policies B and C are equal and correspond to the case when both privileges and prohibitions are used.
- The performance of implementation of the query engine is the most crucial for the evaluating process as shown below during description of testing.
- The performance of implementation of the knowledge base impacts performances of the both processes.

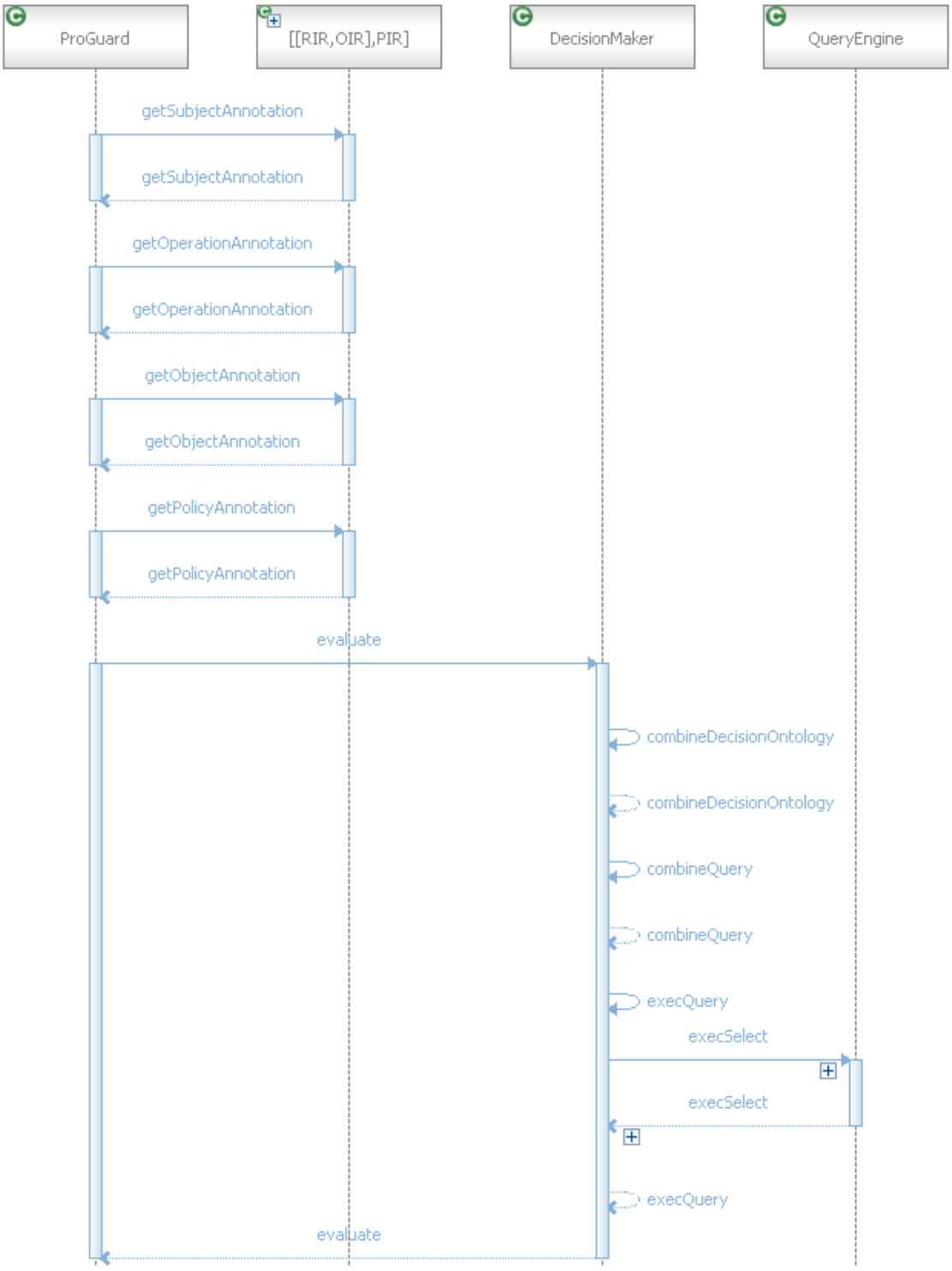

FIGURE 3.97 The testing run (SBAC decision making sequence).

The UML component diagram (FIGURE 3.108) depicts the concrete architecture of the SBAC enforcement mechanism for the measuring of minimal response time of the SBAC decision making process in the settings that support the maximal performance. The internal structure of the ProGuard consists of the decision maker that has in-memory knowledge base (decision set) in the form of ontology model provided by the Jena framework and the query engine (query processor) provided by the ARQ processor of SPARQL queries to RDF data. All ontologies are placed into the web server and accessible via HTTP protocol.

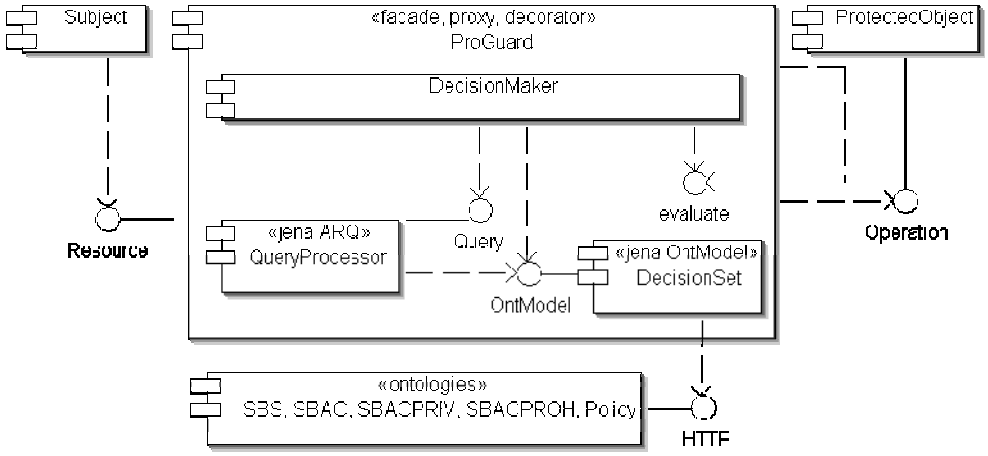

FIGURE 3.108 The concrete architecture with the fastest possible response time.

The fastest response time of the evaluating process corresponds to the simplest policy and domain ontologies. The policy ontology consists of one class of active resources with one individual, one class of passive resources with one individual and one class of operations with one operation. The policy has the only one privilege statement defined using the above described classes. All RDF statements of SBAC, domain and policy ontologies are loaded into the decision set during the start-up process. The SPARQL query corresponds to the authorization rule from the table 4 for policies defined using only privilege statements (the query for policies with only prohibitions is analogical).

```
PREFIX example: <http://www.cc.jyu.fi/~annaumen/example.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX sbacpriv: <http://www.cc.jyu.fi/~annaumen/sbacpriv.owl#>
PREFIX sbac: <http://www.cc.jyu.fi/~annaumen/sbac.owl#>
PREFIX surpas: <http://www.cc.jyu.fi/~annaumen/surpas.owl#>
SELECT ?x
WHERE {
?x rdf:type sbacpriv:Privilege .
?x surpas:subject ?subject .
?x sbac:operation ?operation .
?x surpas:object ?object .
example:resource_1 rdf:type ?subject .
example:operation_1 rdf:type ?operation .
example:resource_2 rdf:type ?object .}
```

For the execution time analysis the TPTP platform provides tools by monitoring method time data. The cumulative CPU time of the ProGuard start-up process is 12,256 seconds which are caused mainly by the start-up (12,141 seconds) of the decision making component and by initializing the in-memory decision set more specifically. The cumulative CPU time of the

evaluating process is 0,813 seconds which are fully caused by the query execution over the decision set.

This cumulative CPU time does not take into account time of I/O operations with memory thus it is smaller than the real invocation time of both processes (14,559 and 2,05 seconds) but fairer for the comparison with fixed type of CPU because the real cumulative time depends from bigger number of characteristics of the hardware. The personal computer that was used for testing is IBM PC with the CPU AMD Athlon XP 3000+, 1 GB of RAM, and OS Microsoft Windows XP Professional version 2002 with Service Pack 2.

3.6.4 Architecture of the secure SmartResource agent

The central in UBIWARE is the architecture of a secure SmartResource agent depicted in FIGURE 3.19. This architecture of an agent extends the one from (Terziyan and Katasonov, 2007) by adding the security components. It can be seen as consisting of four layers: reusable atomic behaviors (RABs), behavior models corresponding to different roles the agent plays, SURPAS security policies, and the behavior engine.

A reusable atomic behavior (RAB) is a piece of code implementing a reasonably atomic function. As the name implies, RABs are assumed to be reusable across different applications, different agents, different roles and different interaction scenarios.

The behavior of an agent is defined by the roles it plays in one or several organizations. Some examples of the possible roles for the power-networks domain: operator's agent, feeder agent, agent of the feeder N3056, fault localization service agent, ABB fault localization service agent, etc. Obviously, a general role can be played by several agents. On the other hand, one agent can (and usually does) play several roles, potentially coming from different organizations. A role consists of a set of beliefs representing the knowledge needed for playing the role and a set of behavior rules. Roughly speaking, a behavior rule specifies conditions of (and parameters for) execution of various RABs. Obviously, RABs need to be parameterizable. Notice that, in UBIWARE, if a role specifies the need of interaction with another agent, that agent is always specified by its role, not name or another unique identifier of a particular agent.

The behavior engine is the same for all the SmartResource agents. The behavior engine consists of the agent core, and the two core activities that we named "assign role" and "live". The AssignRole activity is responsible for parsing roles into the beliefs and behavior rules storages. The Live activity implements the run-time loop of an agent. Introducing SURPAS embeds the policy enforcement mechanism (see details below) into it. The Live activity has also to be protected by some built-in security measures. The Live activity iterates through all the behavior rules, checks them against current beliefs, goals and security policy constraints. After that, it executes RABs together with security mechanisms corresponding to roles and policies, respectively.

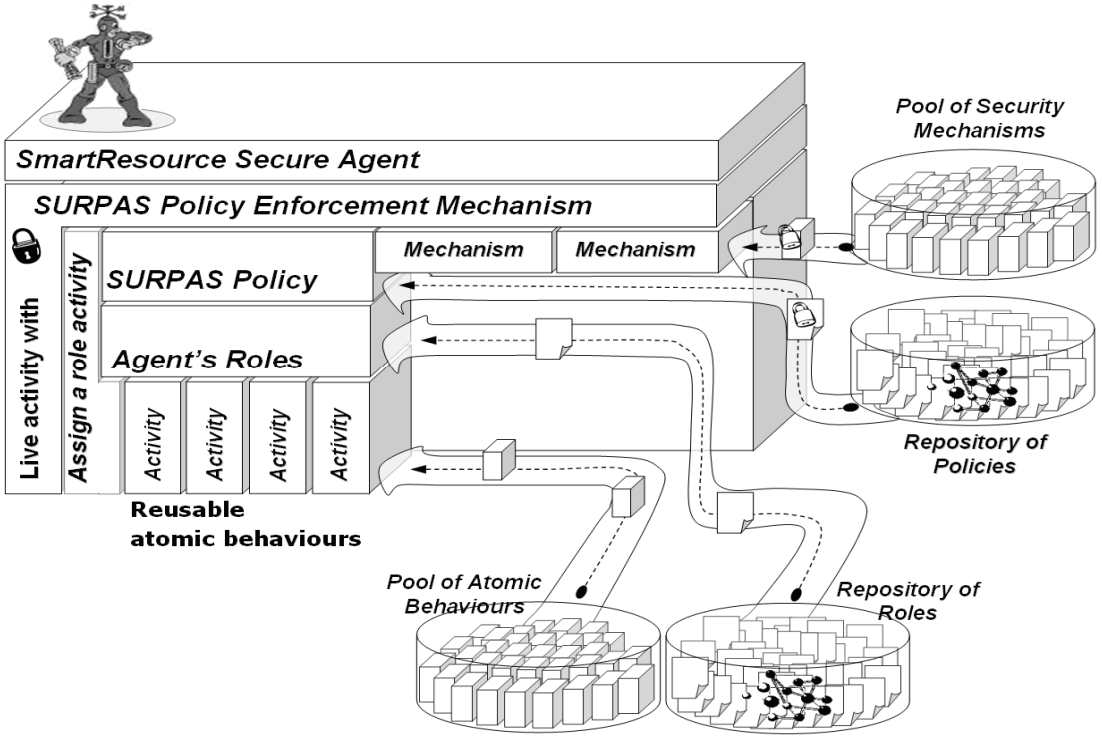

FIGURE 3.19 The architecture of the secure SmartResource agents

The SURPAS policy enforcement mechanism manages security policies and security mechanisms. Its main task is to enforce security policies by interweaving with the Live activity. SURPAS policies are declarative descriptions using expressive and machine-interpretable data formats of Semantic Web. They are reusable over different agents, processes and organizations. Usually, SURPAS policies restrict actions prescribed by roles and enforce use of security mechanisms in addition to normal activities.

Agents access the roles, policies, security mechanisms, and RABs from external repositories, which are assumed to be managed by the organizations which own or hire the agents, or trusted authorities. It is done either upon startup of an agent, or if the organization requests an update to be made. Externalization of roles, policies, security mechanisms and RABs has several advantages:

– Increased flexibility for control and coordination. Namely, the organization can remotely affect the behavior of the agents through modifying the behavior roles and security policies.
– The roles, policies, security mechanisms, and RABs can always be kept up-to-date.
– Possibility to create self-configuring and self-protecting agents.
– Agents may 'learn' in run-time how to play a new role and how to follow a new security policy.
– Organizations are able to provide not only instructions what to do (declarative descriptions of roles and policies), but also the tools enabling doing that (RABs and security mechanisms).
– Agents may have a "light start" with on-demand extension of functionality.
– Inter-agent behavior and security awareness. The agents can make some use of the information about some roles and policies, even if they do not follow them. One reason is to

understand how to interact with, or what to expect from, an agent playing those roles and following those policies.

In summary, the security components, which SURPAS introduces into the architecture of the SmartResource agent, are the policy enforcement mechanism that is built-in into the behavior engine, and security measures and security policies which can be either provided upon agent's startup or retrieved on demand.

In UBIWARE, we also envision some additional security related services, e.g. verifying and signing of roles, policies, security mechanisms and RABs by external trusted authorities to guarantee defect-free and proper behavior of agents. Regarding security, the beliefs storage of an agent has to supports following important activities: semantics-based logging and audit for proactive context-aware intrusion detection and non-repudiation, computing reputation for the management of trust relations between agents and security services, persistent storing of security contexts, and other.

3.7 SURPAS in industrial use cases

This Section exemplifies the above described conceptual semantics (Section 3.5) and architecture (Section 3.6) of SURPAS using the scenarios of the collaborative fault detection and localization in the distributed power network (Section 3.3.1) and the distributed proactive monitoring of paper production machinery (Section 3.3.2).

3.7.1 Secure decentralized management of power networks

FIGURE 3.143 illustrates the adoption of SURPAS for the decentralized management of power networks. The scenario starts when the operators 1 and 2, which belong to different companies A and B, notice abnormal behavior in their subnetworks. The company A owns the core distribution subnetwork that includes the substations 1 and 2. The company B manages the local power distribution starting from the substation 2 and including the substation 3. Both companies send maintenance workers in order to collect information on-site because initial remote fault detection and localization have not produced precise results. The workers 1, 2 and 3 belong to the maintenance crew of the company A. The worker 4 is send by the operator 2 from the company B. In UBIWARE, that automate cross-organizational processes between these two companies, secure SmartResource agents represent the maintenance workers, operators, companies and the equipment of this power network

The workers 1 and 2 observe substation 1 and adjacent network equipment. The worker 3 checks the substation 2. The worker 4 checks the substation 3 and a physical condition of the feeder between the substations 2 and 3. The workers write textual notes, take photographs, go through checklists, check measurements and other.

Before hand, operators and workers got the security assertions from their organizational SmartResource agents with permissions to access information of co-employees. Now, the operators monitor on-site collected material using Internet and GPRS connection to the

mobile terminals of corresponding workers (FIGURE 3.110 cases 1 and 2). The workers 1, 2, and 3 jointly browse collected material using a virtual P2P JXTA network extended to the mobile network (FIGURE 3.110 case 3) and a pure P2P using WLAN and Bluetooth connections when possible (FIGURE 3.110 cases 4 and 5). In this situation, the organizational SmartResource agents act as the authorization authorities according to the sequence diagram in FIGURE 3.12.

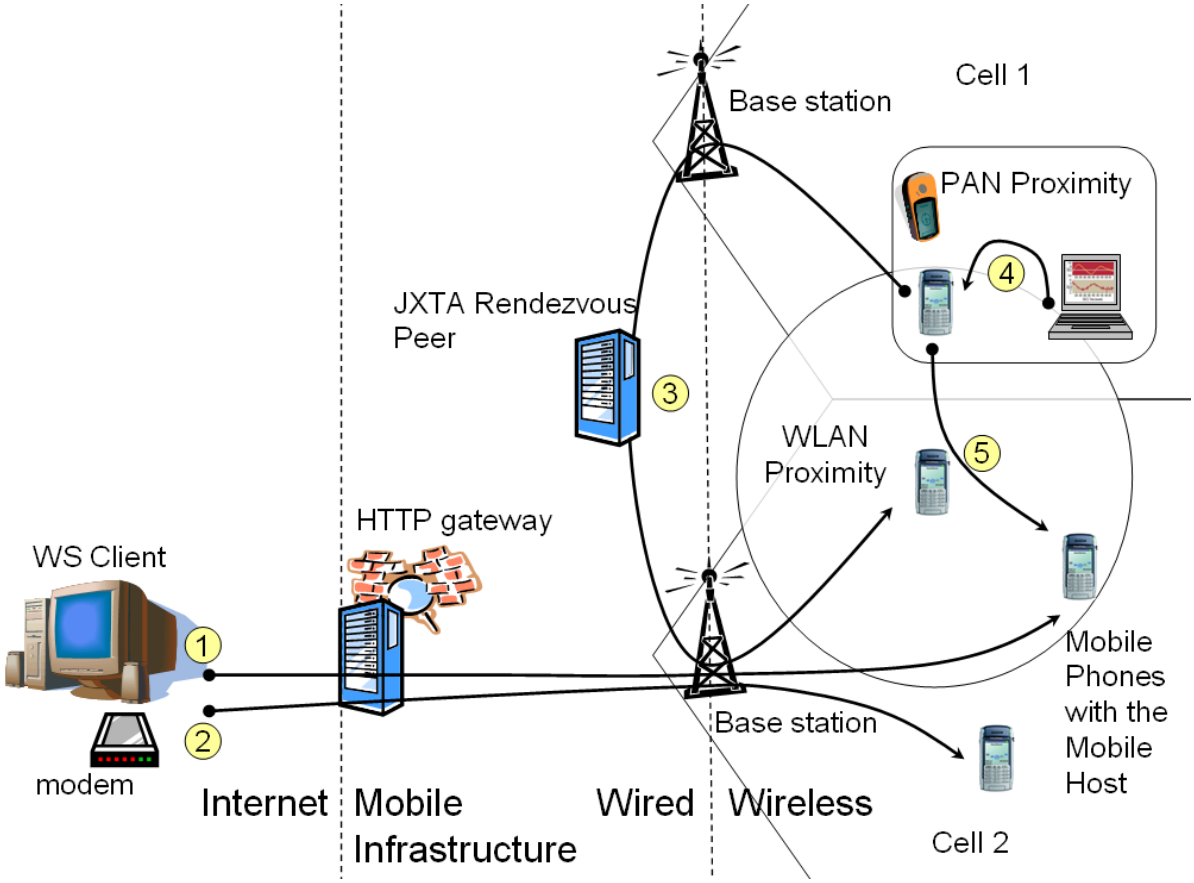

FIGURE 3.110 Communication between SmartResource agents in mobile environments

While checking the substations 2 and 3 and the feeder in between, the workers 3 and 4 try to access the collected material of each other. Their SmartResource agents delegate the authorization of incoming requests to the organizational SmartResource agents according to the sequence diagram of FIGURE 3.132.

The access is granted because both companies have an agreement to share maintenance information about all network equipment on the border of their subnetworks. Finally, the worker 4 discovers the tree that felt to the wires and is the most obvious cause of the abnormal behavior of the network. The photo with the GPS coordinates goes into the maintenance histories of both companies together with all the measurements related to the abnormal behavior for further reuse during detection and localization of similar faults.

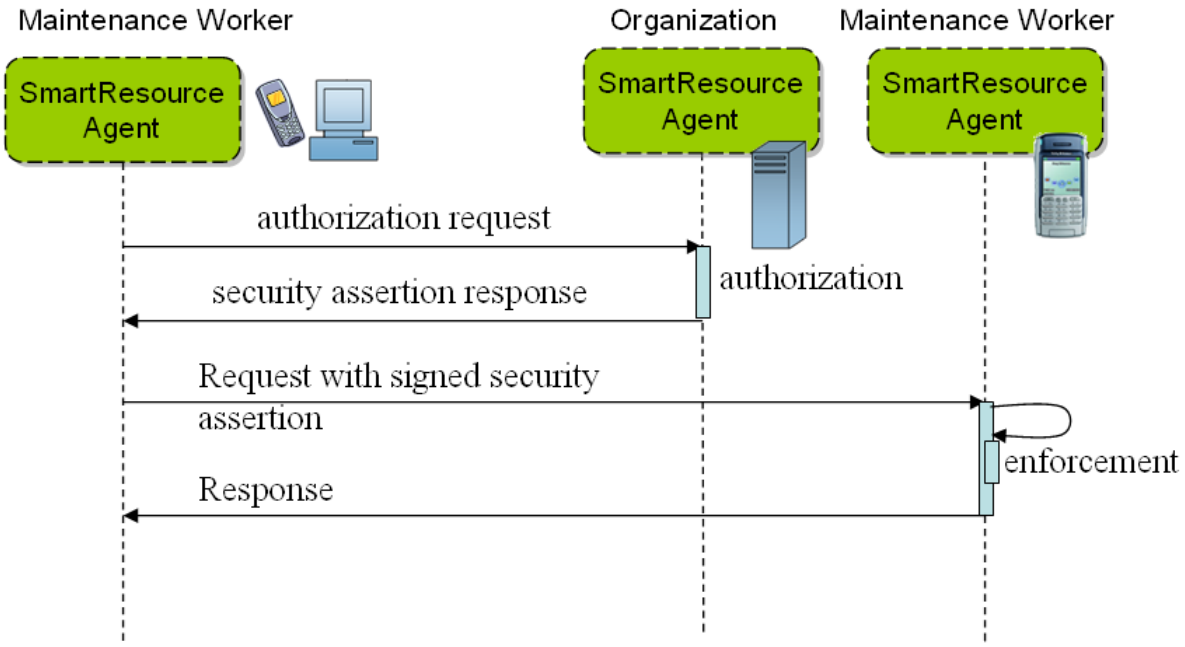

FIGURE 3.121 Organizational SmartResource agent as third-party authority

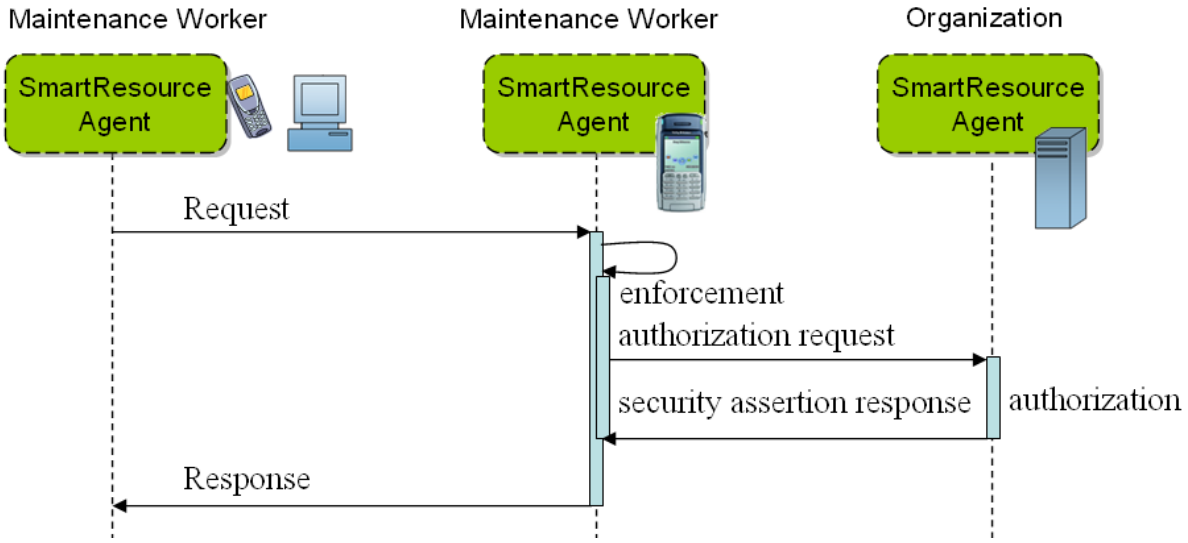

FIGURE 3.132 Delegation of authorization to the organizational SmartResource agent

In order to exemplify the ontology-based specification of access control policies according to the SURPAS ontologies, here we provide simplified specification of two privileges and related classes. The first privilege authorizes access to the power network equipment of the core distribution subnetwork by the maintenance personnel of the company A using roles of maintenance. The second privilege authorizes access to the power network equipment of the substation 3 by the maintenance personnel of the company A using the same roles. The first privilege belongs to the access control policy of the company A. This privilege is needed by the workers 1, 2, and 3, and operator 1 in order to exchange on-site information. The second

privilege supports the agreement to share maintenance information about all network equipment on the border of companies' subnetworks. In our use case, this second privilege is needed to retrieve the photo with the tree by the worker 3 from the mobile phone of the worker 4. The specifications for these privileges and related classes refer to the shared domain ontology "pwont". This domain ontology contains semantic annotations about the employees, companies, equipment and structure of power network, SmartResource roles and other.

```
<sbacpriv:Privilege rdf:ID="Privilege_1">
   <sbs:subject rdf:resource="#MaintenancePersonnelOfCompanyA"/>
   <sbac:operation rdf:resource="&pwont;#MaintenanceRole"/>
   <sbs:object rdf:resource="#PartOfCoreDistributionNetwork"/>
</sbacpriv:Privilege>
<sbacpriv:Privilege rdf:ID="Privilege_2">
   <sbs:subject rdf:resource="#MaintenancePersonnelOfCompanyA"/>
   <sbac:operation rdf:resource="&pwont;# MaintenanceRole"/>
   <sbs:object rdf:resource="#PartOfSubstation_3"/>
</sbacpriv:Privilege>
<owl:Class rdf:ID="MaintenancePersonnelOfCompanyA">
   <owl:intersectionOf rdf:parseType="Collection">
      <owl:Class rdf:about="&pwont;#MaintenancePersonnel"/>
      <owl:Restriction>
         <owl:hasValue rdf:resource="&pwont;#Company_A"/>
         <owl:onProperty rdf:resource="&pwont;#belongTo"/>
      </owl:Restriction>
   </owl:intersectionOf>
</owl:Class>
<owl:Class rdf:ID="PartOfCoreDistributionNetwork">
   <owl:equivalentClass>
      <owl:Restriction>
         <owl:hasValue rdf:resource="&pwont;#CoreDistributionNetwork"/>
         <owl:onProperty rdf:resource="&pwont;#partOf"/>
      </owl:Restriction>
   </owl:equivalentClass>
</owl:Class>
<owl:Class rdf:ID="PartOfSubstation_3">
   <owl:equivalentClass>
      <owl:Restriction>
         <owl:hasValue rdf:resource="&pwont;#Substation_3"/>
         <owl:onProperty rdf:resource="&pwont;#partOf"/>
      </owl:Restriction>
   </owl:equivalentClass>
</owl:Class>
```

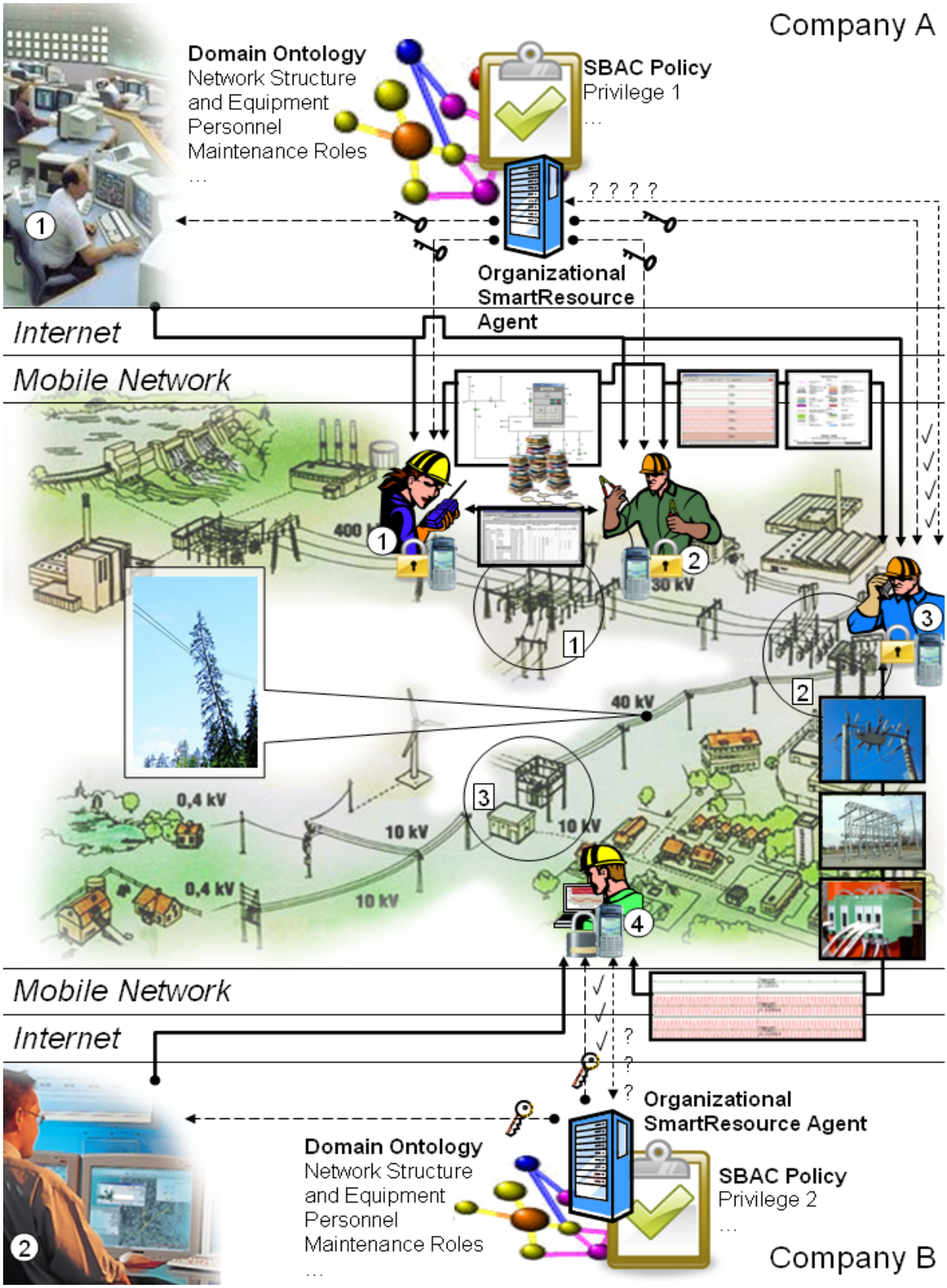

FIGURE 3.143 The SURPAS use case for decentralized management of power networks

3.7.2 Secure Proactive Machinery Maintenance Services

After deriving the real-world security questions for SUPRAS, we used the case of PMMSs in order to exemplify the adoption of SURPAS. We considered an example of specification of hierarchy of resources, hierarchy of operations, and access control privileges in the industrial maintenance domain.

FIGURE 3.154 partially illustrates ontologies and concepts used for the specification of a privilege that experts with an expertise in paper lines of series 480 are authorized for the condition monitoring of all components of third paper line of mill A.

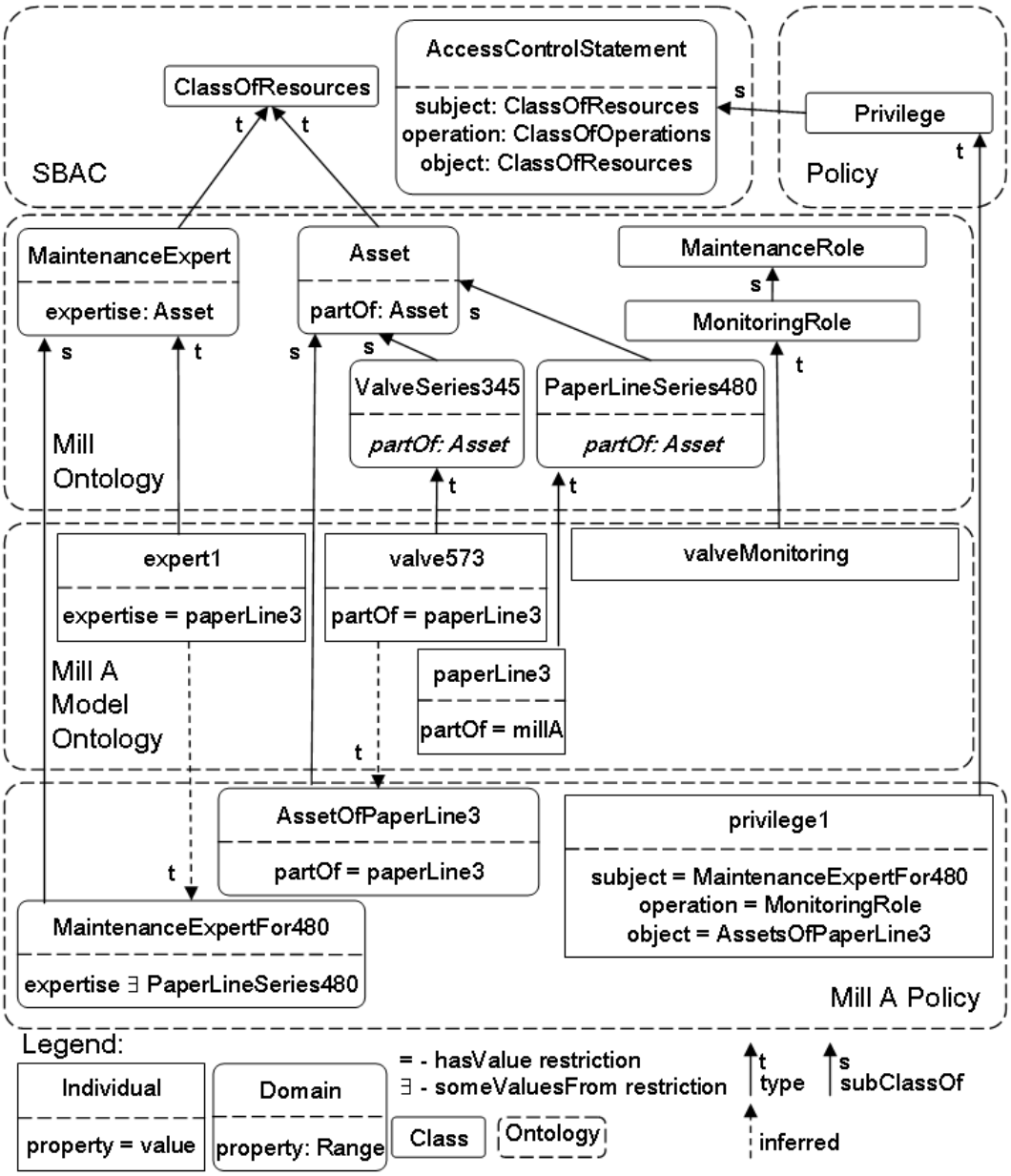

FIGURE 3.154 The SURPAS and domain ontologies for maintenance services

Three ontologies comprise the domain ontology for this case. The mill ontology is an upper level domain ontology commonly shared between collaborating partners in the paper industry that defines meta model for paper mills. All partners keep confidentially their own ontologies. The mill A model ontology consists of semantic annotations of concepts related to the mill A. Partners define access control policies in the form of ontologies based on the SBAC, Policy, Mill and their mill models ontologies.

The mill ontology contains hierarchies of classes that reflect classifications of human resources, industrial resources, information assets, etc. To classify the resources the mill ontology specifies the class of maintenance experts and the class of assets. The class of assets has an object property to define partially ordered sets of assets by the "part of" transitive relation. The class of assets is specialized further to two classes of industrial assets that are class of valves and class of paper lines of specific series. The mill ontology introduces the class of monitoring roles that are kind of maintenance roles.

The mill A model ontology has semantic annotations for a paper line number 3 of series 480, a valve of series 345 that is a part of this paper line, an expert with the expertise in this paper line, a valve monitoring role that provides means for the condition monitoring of valves and is modelled by an instance of the monitoring role.

The mill A policy ontology contains a privilege statement and supporting classes. The privilege defines class of maintenance experts with the restriction that the expertise property has at least some values from the class of paper lines of series 480. The expert defined in the mill A ontology satisfies the restriction and thus belongs to the class of authorized subjects of access. Note that the expert may be defined in the confidential ontology of provider of machinery maintenance services. However the specification of privilege by the owner of maintained assets can still be based on the shared mill ontology while the extraction of semantic annotations for accessing individual subjects is performed based on requests to the SmartResource agents and the classification to classes of authorized subjects is based on requestors' attributes. This is an important feature that ensures the feasibility of specifications of access control statements in dynamic, distributed and collaborative environments with incomplete data and meta data due to different reasons.

The privilege defines authorized operations (behaviours) modelled with monitoring roles. The objects of access are restricted to assets that are part of third paper line. The valve defined in the mill A model ontology is classified to authorized objects because it is an asset and a part of the third paper line. Thus an access control decision in the case of the expert 1 monitoring the valve with the valve 573 as the object of access will be positive.

We adopted the SURPAS abstract architecture, where all four components of the abstract architecture of the SBAC enforcement mechanism become SmartResource agents. FIGURE 3.165 shows the top level architecture and indicates the steps of a possible use case for the SBAC enforcement mechanism.

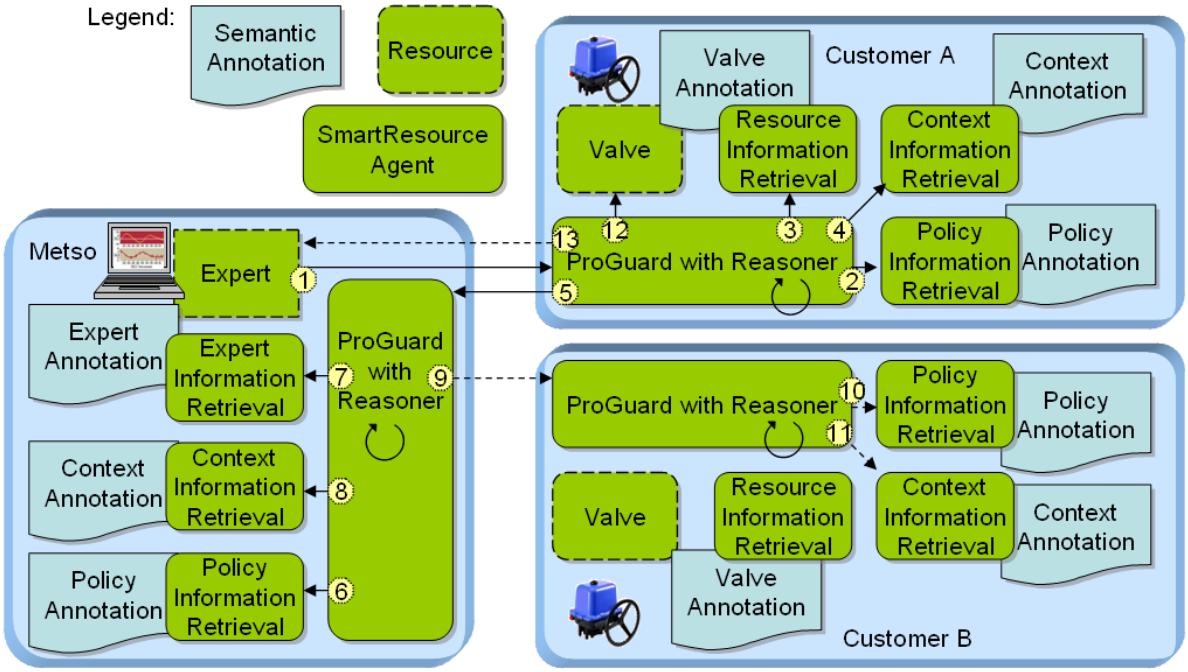

FIGURE 3.165 The SURPAS use case for maintenance services

1. Metso's maintenance expert requests status of valve of series 345 that is part of the running third production machine of customer A as a countercheck for a predicted fault.
2. The ProGuard intercepts the request and retrieves corresponding policy rules based on a policy annotation.
3. Policy rules require some additional information about the expert, valve and context of access for this kind of request. The ProGuard retrieves an annotation of the valve that is the object of access based on valve ID from the mill A model ontology.
4. The ProGuard retrieves contextual information that is available from a local context annotation.
5. The ProGuard of customer A does not have enough information internally, thus it forwards a request for the information about the context and the expert that is the subject of access.
6. Metso's ProGuard intercepts the request 5 and retrieves rules from a trust agreement through PIR of Metso. According to these rules, Metso can provide information about the resource and the context.
7. Metso's ProGuard retrieves the semantic annotation of the expert.
8. Metso's ProGuard retrieves the semantic annotation of relevant context.
9. Based on the annotation of context the reasoner implies that contextual information contains sensitive data because the maintenance expert predicts the fault based on a history of faults during an operation of paper production machine of similar type owned by the customer B and the distribution of this information can violate a trust agreement between Metso and the customer B. Thus Metso's ProGuard delegates the request to the customer B for a decision about possibility to share sensitive contextual data with the customer A.

10. The guard of customer B retrieves rules from its trust agreements with Metso and the customer A.
11. The ProGuard of customer B retrieves requested contextual information and reasons that customers share information about the condition monitoring and the diagnostics for this type of paper machine. The customer B forwards the decision to Metso Paper, which forwards semantic data about the subject and the context of access to the customer A.
12. The ProGuard of customer A based on all collected data makes positive decision granting access to valve's status.
13. Alternatively, the ProGuard denies access and replies with rejection of request.

3.8 Conclusions

Conventional approaches to manage and control security seem to have reached their limits in new complex environments. These environments are open, dynamic, heterogeneous, distributed, self-managing, collaborative, international, nomadic, ambient, and ubiquitous. New generation middleware such as UBIWARE will significantly advance the industrial automation towards automatic discovery, composition, orchestration, integration, invocation, execution monitoring, and coordination of industrial resources. These advanced automation techniques target physical world objects and thus put security as the core need-to-be-addressed issue. We described our long-term vision for the security and privacy management in such complex environments, SURPAS. It aims at policy-based optimal collecting, composing, configuring and provisioning of security measures in multi-agent systems like UBIWARE. This section concentrates on the access control issues in SURPAS. Particularly, we analyzed the security implications of UBIWARE, presented the SURPAS research framework which guides our research towards SURPAS, the SURPAS conceptual semantics and the SURPAS abstract architecture.

There are an enormous number of targets for further work. They include ontology engineering for fundamental elements of security, elaborating architectures, designing new specific algorithms for the intelligent security policy management, developing reference implementations and, finally, adopting research ideas into practice in real-world industrial settings.

UBIWARE Deliverable D1.1:
Workpackage WP4:
Task T1.1_w4:

4 Principles of the Configurability

Evaluating current approaches towards software configuration, configuration schemas and patterns. Defining the principles of the configurability in UBIWARE and applying them to the development of configurable resource adapters.

Workpackage leader: ***Sergiy Nikitin***

The main objective of the UBIWARE project is to develop an open and generic middleware platform which will allow creation of self-managed complex industrial systems consisting of distributed, heterogeneous, shared and reusable components of different nature, e.g. smart machines and devices, sensors, actuators, RFIDs, web-services, software components and applications, humans, etc. The middleware will enable various components to automatically discover each other and to configure a system with complex functionality based on the atomic functionalities of the components.

Autonomic components must be given certain degree of flexibility in order to make them more reusable and increase adaptability to new or changing environment. One of the key issues is to keep connectivity to the resources from the outer world (with respect to UBIWARE) connected via adapters. To keep resources always connected, we need to elaborate a framework for configurable adaptation, which will define the principles of the adapter configuration and change handling rules.

The notion of configurability and configuration is nowadays discussed in different application areas ranging from the Reconfigurable Computing[3] to Software Configuration

[3] Reconfigurable Computing - http://en.wikipedia.org/wiki/Reconfigurable_computing

Management[4]. Although, these topics seem to be relevant, they are quite different from the problem domain of the UBIWARE. In this paper we will narrow our scope to the configurability of software components, meaning that components have interfaces allowing changes to the components' behaviour. Furthermore, in order to illustrate the configurability in action, we select a specific component of the UBIWARE platform – an adapter, as an object of configuration.

The need for configurable software is dictated by the market demanding more and more adjustable and flexible solutions. There is a number of open source solutions available, such as Obix framework (OBIX). The framework offers an API to incorporate software configuration mechanisms into your application. It works with the XML-based portable configuration data, allows definitions of relationships amongst components, detects automatically changes to the configuration files and loads new settings. These solutions aim at the configurable initialization and deployment of applications and offer rather sophisticated APIs. The UBIWARE platform provides initial configurations as well, so these frameworks can be useful, however, due to autonomous nature of UBIWARE architecture, we need an instrument, that allows on-the-fly component (re-) configuration. For more sophisticated self-aware entities such as software agents, we need a mechanism for self-configuration.

The generic problem of configurability is a hot topic in a number of today's ICT areas. Reconfigurable hardware elements have brought new tasks to architects both on hardware and software levels. There are a number of on-going research activities in a field of reconfigurable and/or self-aware systems, for example SELFMAN (SELFMAN), CASCADAS (CASCADAS) and Deliver (Deliver).

4.1 Configurability in UBIWARE

We will start with the definitions and terminology and determine what the configurability in UBIWARE is and what it is not. Configurability can be of two types:

a) *Structural* – Instead of fixing some block, we just put new one. There can be dummy components, which can be configured to play any kind of functionality,

b) *Parametric* – Functional components allowing tuning of their functionality.

Of course, there can be a combination of these two.

In UBIWARE structural configurability (type a) is used for business process (re-)planning, whereas parametric configurability (type b) is applied for adjustment of individual components and atomic behaviors. The reasonable question here would be: "What can one configure in atomic behaviour/component?" The answer is: we use the notion of "Reasonably Atomic Behavior" to model building blocks of the system. "Reasonably atomic" means here, that there is no need to further decompose the functionality of the component, because it makes the modeling too complex. We claim that it is more convenient for a designer to define fewer components with configurable elements, rather than atomizing them up to the primitives Endless decomposition will bring us to the programming language primitives.

[4] SCM - http://en.wikipedia.org/wiki/Software_configuration_management

Furthermore, the composition of configurable components should be less computationally expensive. For example, we take an atomic function *transform(X,Y)* where X is an input document and Y is an output document. Suppose, we have a static source of input, that slightly changes once a year. The transformation logic is quite simple and fixed. In this case there is no need to make a third parameter which would be a transformation script (i.e. *transform(X,T,Y)*). The function does not change its output even when there is a small change in the input and transformation script. It still produces the same Y. Hence, there is no need to change the behavior as a whole and declare one more class. What would be reasonable is to configure it by calling separate *setConfiguration()* function to provide minor changes in the transformation logic. This approach would allow on the fly reconfiguration of the component, thus there is no need for redeployment and breakages in operation.

The above mentioned types of configurability (structural and parametric) correspond to the configuration of composite and atomic components respectively; however, composite component may have configurable parameters.

According to the project plan, this deliverable will focus on the application of the configurability to adaptation. We separate three major subjects of investigation: Configurability Framework, Configurable Adaptation and Configurable Transformation. The dependencies among them are shown on Figure 4.1.

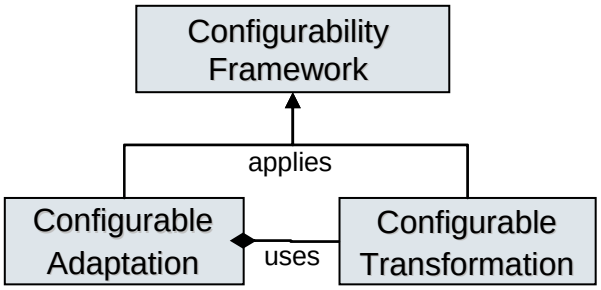

Figure 4.1 – Configurability Framework applied to adaptation

The Configurable Adaptation process involves Configurable Transformation, but both apply the Configurability Framework. The Transformation is only a part of Adaptation which also includes Connectivity and Self-Awareness modules.

4.1.1 Configurability framework

This section discusses the generic domain-independent understanding of the configurability. We need a formal framework for configuration, because modifications in the runtime need additional patterns and functionality to keep the system consistent, up and running.

4.1.1.1 Configuration in software lifecycle

Configurability influences all the stages in software lifecycle starting from the domain analysis and ending with the maintenance. Early, on the domain analysis stage the analysts should identify areas, where configuration will probably appear and state what kind of changes in the domain are expected or possible. Requirements analysis should further specify the expected flexibility of the software system. The Software Architecture will define interfaces of the components and the configurable elements will be modeled and handled properly to ensure secure and predicable configuration. The implementation will have its

peculiarities because configurable elements will have special methods and variables, thus influence the coding as such. The testing phase should be handled with special care; unit tests should include reconfiguration procedures and analyze the results taking into account security of reconfiguration and interdependencies. The deployment of the software will have an initial configuration either hardcoded or gathered in an initialization script. During the maintenance process, the reconfigurations should preserve system integrity and consistency, thus implement transaction logic for vertical (top-down) reconfiguration and negotiation protocols for horizontal (peer-to-peer) reconfiguration.

4.1.1.2 Programming the Configurability

The configuration of any entity in UBIWARE is performed using a Configuration Script, which reaches the target object either in form of an ACL message (if the object is an agent), or as an input parameter of the *Configurable* interface, and particularly *setConfiguation()* method.

Every component featuring the configurable characteristics, must implement *Configurable* behavior or *Configurable* interface. The former one is used for agent-level behavior configuration and the latter one configures atomic software primitives. Both the behaviour and the interface allow setting the configuration and getting the configuration description. The configuration script is processed by the agent, and if the agent decides to start configuration, it launches the *ConfigurationBehavior*, which encapsulates the script processing and configuration process logic. The *ConfigurationBehavior* generates the *ConfigurationDescription* objects, which represent either current or desired state of the agent's components (see Figure 4.2).

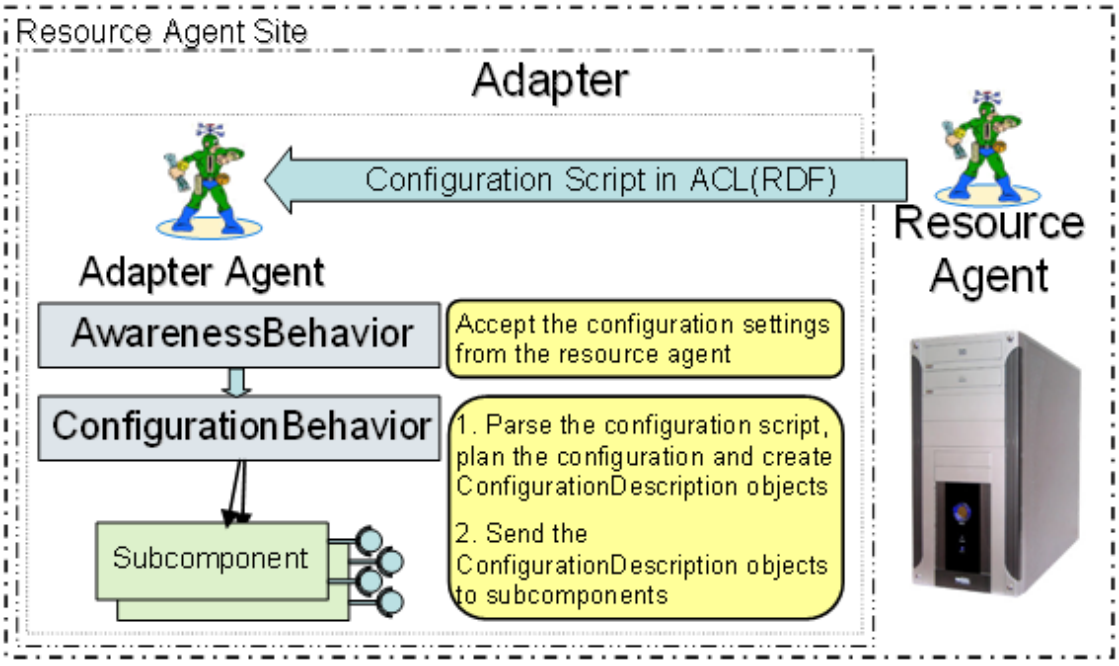

Figure 4.2 – Processing the configuration script

The propagation to subcomponents is organized via *Configurable* interface. Every configurable software component has a precise specification of configurable parameters and

only these parameters can be changed. Changes to the Atomic Behaviors are done via *setConfiguration()* method of the interface, which may contain consistency checking procedures. The configuration process uses transaction mechanism for rolling back the whole operation if at least one of the subcomponents was not configured properly. The Configuration Transaction Algorithm is shown on Figure 4.3.

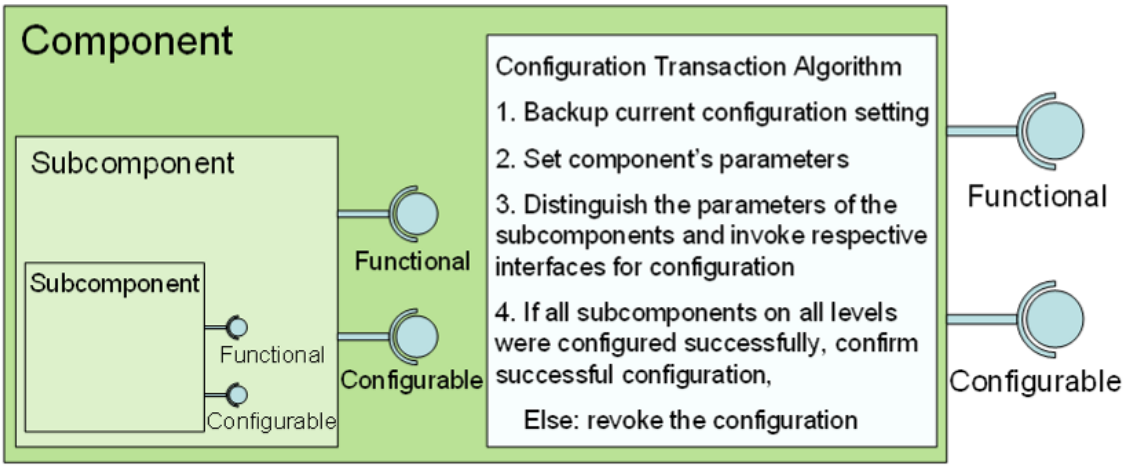

Figure 4.3 – Configuration Transaction Algorithm of Configurable Component

The compulsory preliminary activities for configuration are defined as follows:

- Check security and policy permissions for performing the configuration
- Check if the configuration is safe (if an agent performs some action at the time of configuration, it may lead to an unpredictable behavior). The safest is to perform configuration when an agent is idle.
- Suspend agent's activities for the period of configuration in order to avoid unsynchronized execution

When the configurable component is running, it has a configured state, which is described by the *ConfigurationDescription* that corresponds to the instance of the *ConfigurationDescription* class in the Transformation Ontology (see Figure 4.4).

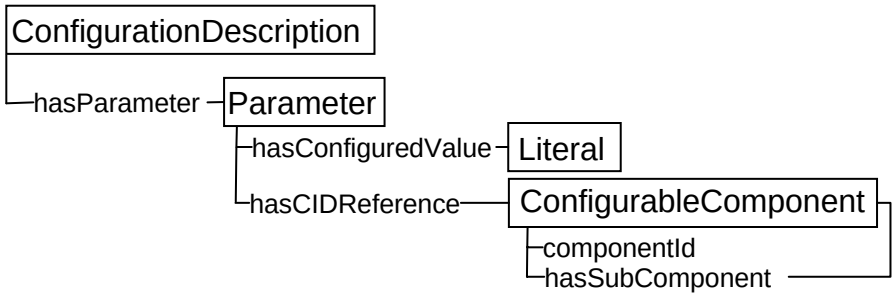

Figure 4.4 – ConfigurationDescription class

The ConfigurationDescription refers to the instances of the Parameter class, which is an upper class for the Parameter ontology. The parameter has a literal value and a reference to the Component ID. This reference is needed, because the component may have a subcomponent hierarchy with the same parameter types (see Figure 4.5).

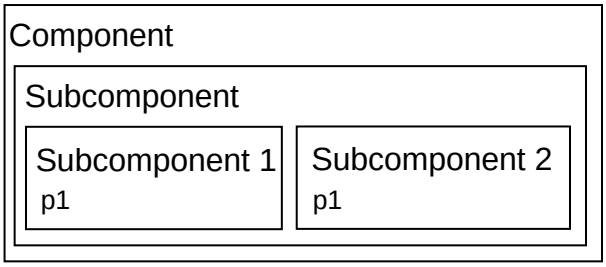

Figure 4.5 – Component hierarchy

The Subcomponent 1 and 2 can have the same parameter or even be of the same type, but configured differently. Therefore, we need an explicit definition and identification of the component structure.

Any configurable component has binding of its parameters to the *ConfigurationDescription*. The way how binding is implemented is not specified, it can be, for example, an if-else code (see Figure 4.6). The processing logic of the component ensures the validity of the ConfigurationDescription objects and checks if all of them have reached the target.

```
public void setConfiguration(ConfigurationDescription cd)
                                    throws IllegalConfigurationException {
    //Get the parameters from the new ConfigurationDescription
    Enumeration<String> cparams=cd.getParameterList();
    /*Check if the parameters are valid and iterate over the
     * parameters to assign values. The validity is checked by
     * matching the new java object class with the current one.
     * The validity check may vary from implementation to
     * implementation and is not obligatory
    */
    while(cparams.hasMoreElements()){
        String paramname=cparams.nextElement();
        if(paramname=="http://iog.jyu.fi/paramidentifiers/ontoparameter"&&
                conf.getParameterObject(paramname).getClass()==
                    cd.getParameterObject(paramname).getClass()){
            ontoparam=(String)cd.getParameterObject(paramname);
        }
        else if(paramname=="http://iog.jyu.fi/paramidentifiers/coefficient"&&
                conf.getParameterObject(paramname).getClass()==
                    cd.getParameterObject(paramname).getClass()){
            ontocoefficient=(Integer)cd.getParameterObject(paramname);
        }
        /* The ConfigurationParameter does not exist in the local
         * parameter table. Revert to the previous configuration
         * and throw the exception
        */
        else{
            this.setConfiguration(conf);
            throw new IllegalConfigurationException();
        }
    }
    //Assign new configuration
    conf=cd;
}
```

Figure 4.6 – Configuration setting function

4.1.1.2 Structural Configurability (Composition)

Complex processes and scenarios in UBIWARE are represented as a set of inter-agent activities, or, in other words, parallel and sequential behavior executions. The sequential execution of behaviors within one agent entity is planned and initiated by the set of rules, which define agent's actions and plan new ones. The rule-based process planning and execution has been researched in the SmartResource project and has appeared as a General Networking Framework (Kaykova et al., 2006).

4.1.2 Configurable Adaptation

Configurable adaptation is considered to be a key success factor for UBIWARE. To make adaptation of external resources simple and transparent is quite challenging task, especially for a dynamic and changing environment. The adapter, therefore, in addition to transformation capabilities embeds more sophisticated features and becomes an autonomous configurable entity (see Figure 4.7).

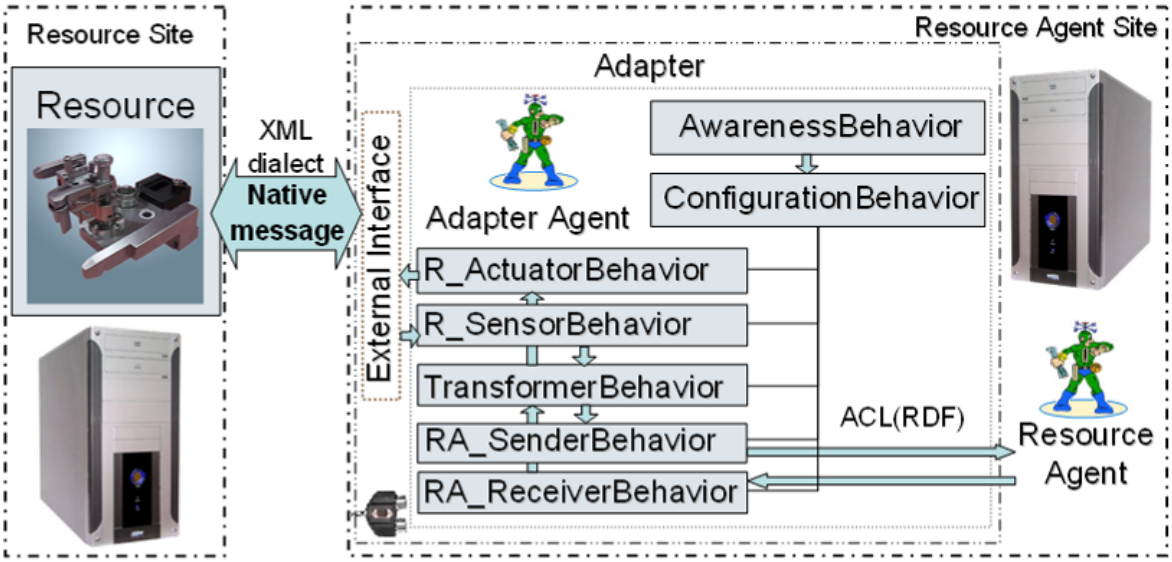

Figure 4.7 – The architecture of an agent-driven adapter

The behaviors of the adapter agent are divided into four logical groups: Self-management, Resource interaction, Transformation and Resource Agent Interaction.

AwarenessBehavior and *ConfigurationBehavior* belong to the Self-management group. They represent the intelligent part of the agent.

R_ActuatorBehavior and *R_SensorBehavior* represent Resource Interaction group. They deal with the connections (e.g. database connections) and message reception. However, both behaviors can be active from the software point of view and this separation is rather logical. The *R_SensorBehavior* may open database connection to retrieve some data, whereas *R_ActuatorBehavior* will write data to the Resource database or send control signals. The *TransformerBehavior* represents the transformation logic between the native resource language and UBIWARE platform language.

The *RA_ReceiverBehavior* and *RA_SenderBehavior* are responsible for interaction with the Resource Agent. We will describe different scenarios of interaction in the next section

4.1.2.1 Interaction scenarios

The interaction between three main entities of the adaptation process (Resource, Adapter Agent and Resource Agent) can be rather complex and involve negotiation protocols (so-called choreography). Different interaction cases determine the *TransformerBehavior* operation. We have distinguished key interaction scenarios for the above mentioned entities (see Figure 4.8).

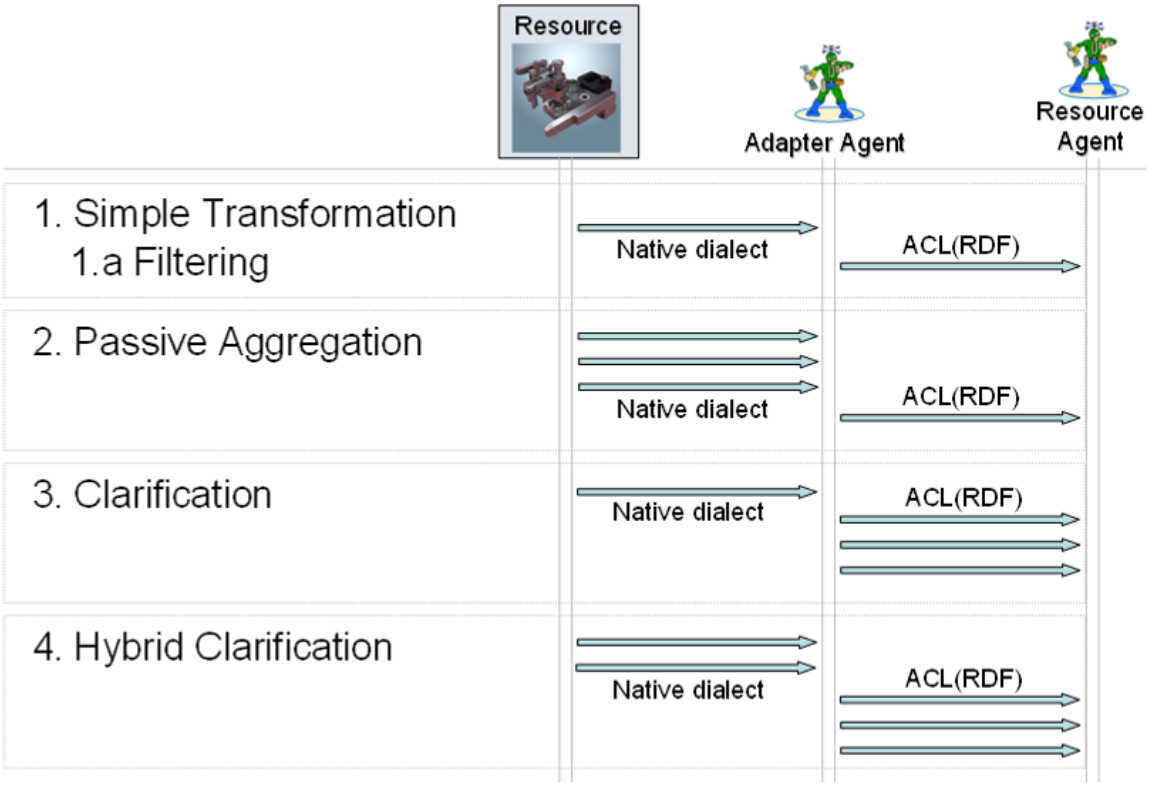

Figure 4.8 – Passive Adapter Scenarios

The first scenario describes trivial transformation case, when one message in a native dialect is transformed into the ACL message. *Filtering* is a particular case of the transformation, when only part of the input data is processed. It can be present in all the scenarios, regardless of the direction of the message flow or messages number.

In the *Passive Aggregation* scenario the Adapter Agent collects incoming messages and then sends the aggregated data in the ACL message to the Resource Agent.

The *Clarification* scenario parses complex message from the Resource and sends a sequence of messages to the Resource Agent. This scenario is reasonable, for example, when a service returns three alternative results. There is no need to create additional semantic entity to wrap two or three service results, but these results are sent separately instead. This may simplify processing logic of the Resource Agent.

The *Hybrid Clarification* combines features of both the Aggregation and Clarification scenarios. It represents a many-to-many relationship of the incoming and outgoing messages. The Adapter Agent receives a number of messages from the resource and then generates a number of messages to the Resource Agent.

In case of the active adapter operation, we have distinguished the main types of scenarios (see Figure 4.9).
In the *Simple Active Request* the Adapter Agent periodically queries the Resource and sends query results as an ACL message.
The *Resource Agent Request* differs from the latter one only by the query initiator. Whereas in the *Simple Active Request* the query is initiated by the logic of the *Adapter Agent*, in the second case, the *Resource Agent* acts as an initiator.

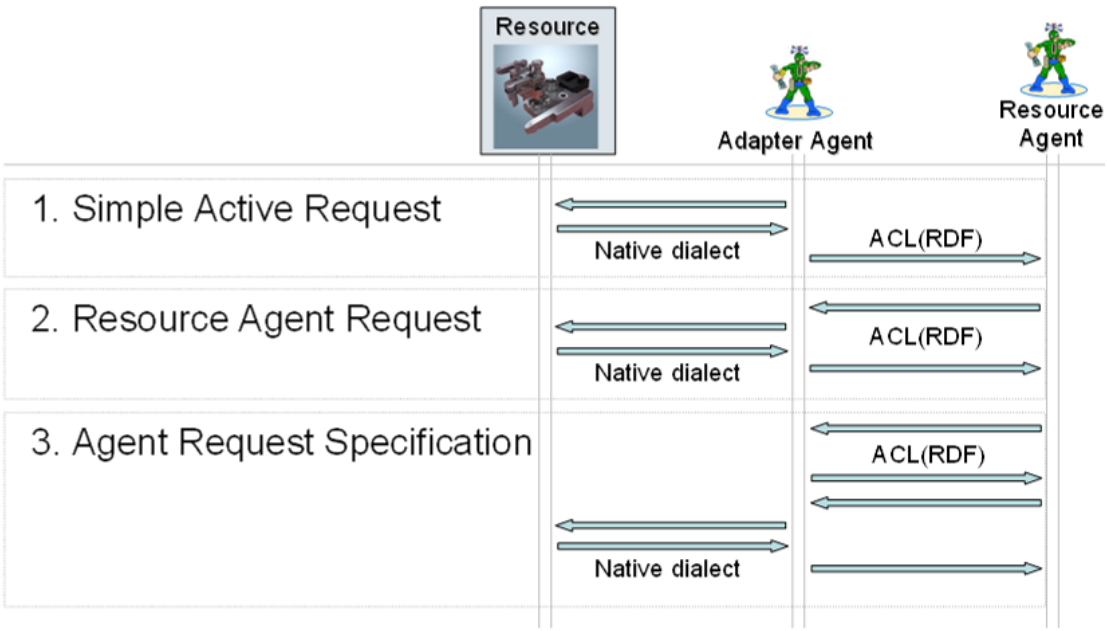

Figure 4.9 – Active Adapter Scenarios

The Agent Request Specification is a scenario, where the *Adapter Agent* clarifies what is the exact need of the *Resource Agent* before querying the *Resource*.

4.1.2.2 Communication with sources of adaptation

The adaptation in UBIWARE has rather broad application. The sources of adaptation may vary from RFID sensors up to databases and web services. In this section we will discuss the differences among sources and design key software components of the adapter agent behaviors.
We distinguish sources by the connectivity, data formats and interaction methods. The connectivity can vary by methods of access to the resource. For example, RPC call, JDBC connection, message-based service invocation or local driver or API based connection to any software.
The data formats extracted from the source can also have different nature and format. It can be a bitmap image or comma-separated file, or XML document. It can also be a Java object.
The interaction methods distinguish the way, how resource data is obtained. It can be a listener that waits for incoming data from the resource, or a time-, or event-based scheduled requestor. Although the actuator and sensor behaviors are logically separated (see Figure x), they may use the same software components in their implementation. Figure 4.10 shows the parameters for atomic components differentiation.

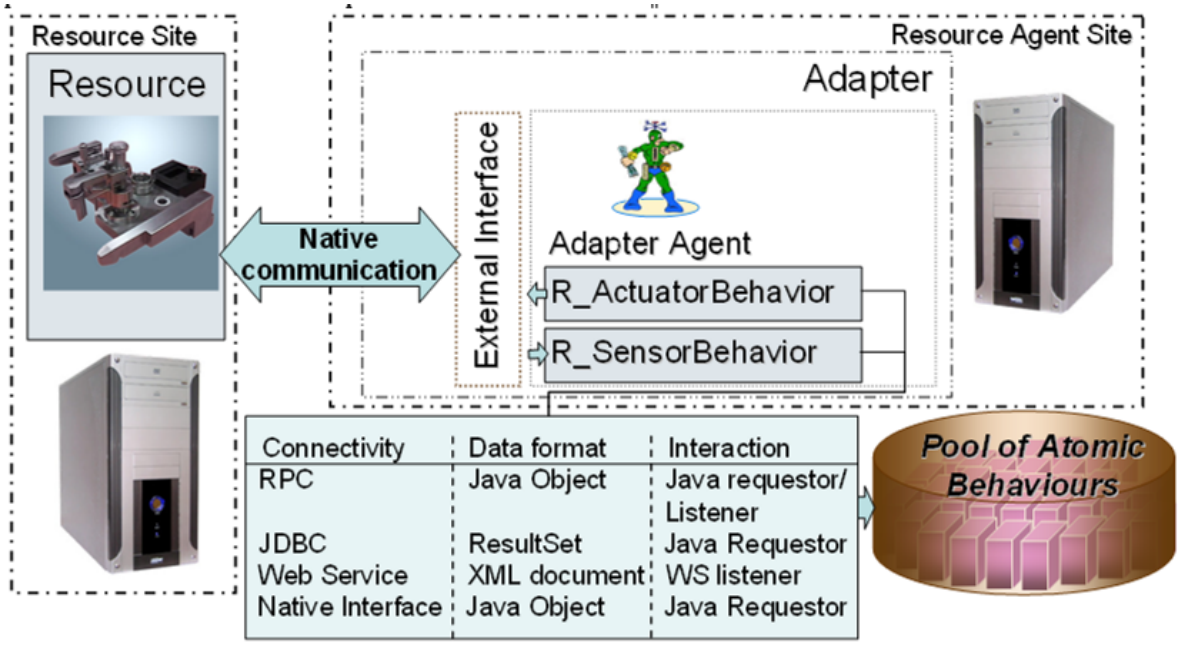

Figure 4.10 – Parameterization of Atomic Behaviors for interaction with Resource

Although, these criteria allow further division of the functionality into the atomic elements, we stop here the division of the behaviors and leave further differentiation to the configurability. For example, the hardcoded Sensor atomic behavior can implement *JDBC-ResultSet-JavaRequestor* combination and produce a *ResultSet Java object* as an output. However, it can also be a *Hashtable Java object* as an output. The data format of the retrieved object determines the input format for the Transformer. On the Transformer side the syntactic preprocessing can be configured to match the semantic transformer, e.g. XML as a string can be parsed to the DOM document and vice versa. The interaction specifies also the how the states are handled in both Actuator and Sensor behaviors. The aggregation (when an agent is waiting for a certain sequence of messages for further processing) is defined by further specialization of the Interaction type, for example, Passive Aggregation is a Listener type with the specified finite state algorithm (e.g. invoke transformation when five messages have been collected).

4.1.3 Configurable Transformation

The Configurable Transformation module represents the transformation logic of the adapter. In context of adaptation, it can be considered as a subcomponent of the adapter, but we have separated it into the independent component, because the transformation, as such, is not only a part of adaptation. It can also be a part of service functionality. In fact, any activity of a service can be interpreted as a transformation depending on the add value of the output compared to the input. The key element of the transformation module is a *TransformerBehavior*. It is a domain independent entity that encapsulates the transformation logic. This logic varies and can be implemented using different technologies: It can be a Java code performing image analysis and text extraction or XSLT script for XML documents. We,

therefore, introduce a *Transformation Ontology* – a common model for describing transformations. The transformations are *Reusable Atomic Behaviors* which require descriptive specification basis for classification of the conceptualizations. The *Transformation Ontology* consists of the core part and the pattern part. The core part defines the generic objects and methods of transformation, whereas the pattern part conceptualizes structural patterns of the objects being transformed and their relationships.
The core ontology comprises the following elements:

- TransformerBehavior – a root element representing the transformation entity. Its properties are shown on Figure 4.11.

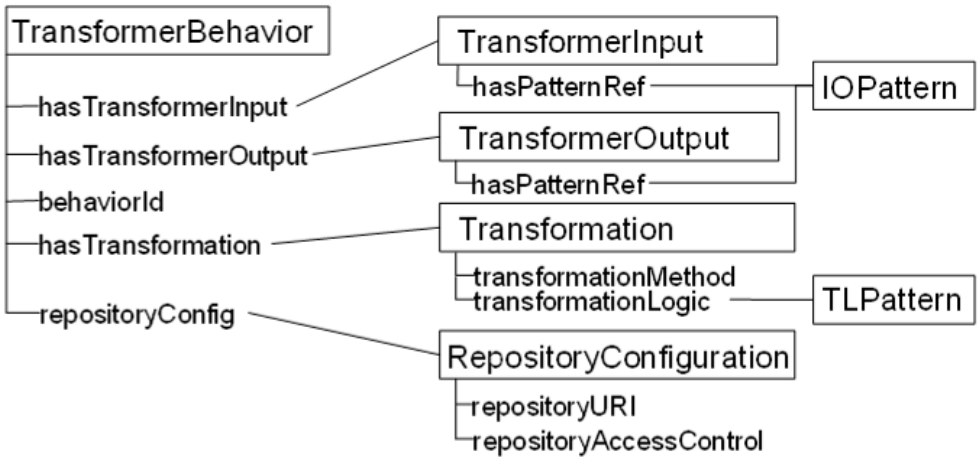

Figure 4.11 – Transformation Ontology Core

The core elements describe the structure of the Atomic Behavior, i.e. real software code that performs transformation. The TransformerBehavior class has a URI identifier of the Behavior and reference to the online repository, where binary implementation of the Behavior can be obtained from. Both TransformerInput and TransformerOutput refer to the descriptive pattern of the object being transformed and the object being generated. The TransformationMethod is a reference to the ontology of transformation techniques like pure Java code transformation, XSLT-transformation, service-based transformation (e.g. we may need a pdf-file generation), etc.
The pattern part defines the exact format of the input or output. It describes the Java type of the object and its structure up to the reasonable extent. It is somewhat similar, to the JDBC ResultSet object's metadata and can be linked to the appropriate ontology to simplify the specification of the transformation logic defined in the *TLPattern*.

4.1.3.1 Configurable Transformation in action

The above mentioned ontology describes the static structure of the *TransformerBehavior*, whereas enactment of the configurable transformation requires additional description of the actions in runtime (see Figure 4.12). The configuration message in S-APL language passes the policy checking procedure and if it is successful, the message is processed by the *ConfigurationBehavior* that, extracts the XSLT-script and forms the *ConfigurationDsecription* object. The script is assigned as a parameter to the *ConfigurationDescription*, which is passed to the *configure()* method of the *TransformerBehavior*. In particular case the *TransformerBehavior* is quite simple and really

atomic, meaning that it does not contain configurable subcomponents. During the configuration, the transformation activities are suspended by the *AwarenessBehavior* to avoid unsynchronized access to the code.

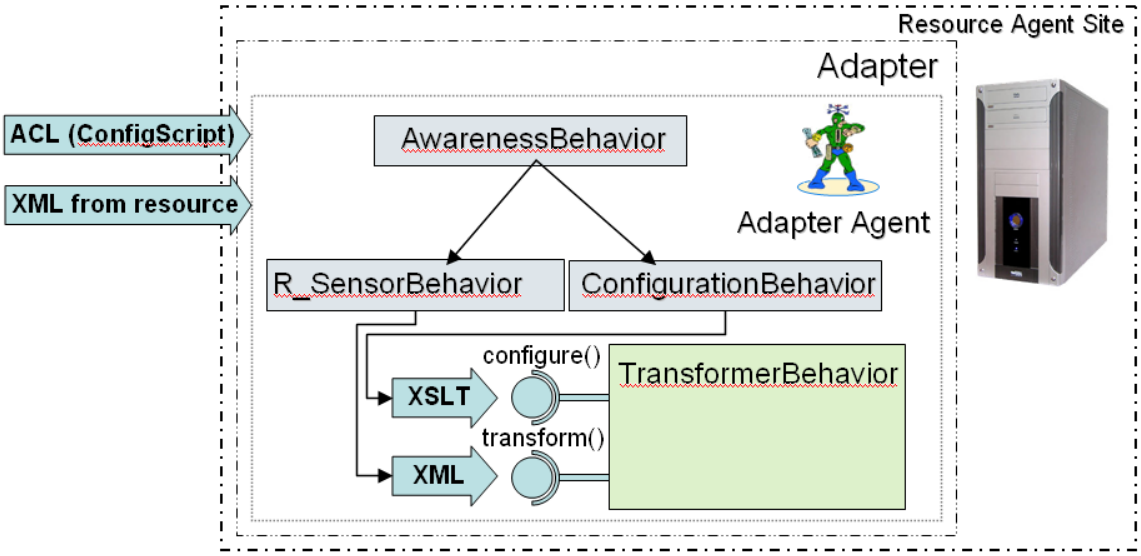

Figure 4.12 – Example adapter with TransformerBehavior

4.2 Conclusions and Outlook

The need for configurable adaptation has appeared as a result of industrial cases analysis, which disclosed the problems of the static code in process industry. Minor changes in the business logic of information systems lead to maintenance breaks and involve a lot of human resources, particularly programmers for code maintenance, testers, system administrators for giving the access to the running platforms, and deployment, who launch the updated version as a product. The configurability changes the process with the anticipation of possible changes and making them a part of the functionality in the very beginning, thus giving more flexibility to the business process management. Most of the changes now go from the programmer's level to the level of business process configuration – a descriptive script-based business logic, which can be updated dynamically in a runtime. This allows the business process manager to handle a vast number of situations without annoying the programmers and provide fast changes in response to customers needs. The configurability can be used not only for a product customization, but also for a dynamic, on the fly support and maintenance.

UBIWARE Deliverable D1.1:
Workpackage WP5:
Task T1.1_w5:

5 General Vision of 4I Technology and Its Application in UBIWARE

REQUIREMENTS FOR HUMAN ADAPTATION

Workpackage leader: ***Oleksiy Khriyenko***

Next-generation integration systems will utilize different methods and techniques (Semantic Web and Web Services, Agent Technologies and Mobility) to achieve the vision of ubiquitous knowledge. Unlimited interoperability and collaboration are important values for a multitude of areas in daily life. Integration of heterogeneous applications and data sources into an interoperable system is one of the most relevant challenges for many knowledge-based corporations nowadays. Thus, the development of Global Understanding eNvironment (GUN) (Kaykova et al, 2005), which would support interoperation between all the resources and exchange of shared information, is challenging, but potentially profitable task.

The challenge of enabling computer systems to make better use of Web data by making that data machine-processable has been taken up by the Semantic Web effort, which proposes formal knowledge structures to represent concepts and their relations in a domain. These structures are known as ontologies and the World Wide Web Consortium (W3C)[5] has recommended two standards, the simpler Resource Description Framework (RDF)[6] and the more expressive Web Ontology Language (OWL)[7].

[5] http://www.w3c.org
[6] http://www.w3.org/RDF
[7] http://www.w3.org/TR/owl-absyn

In recent years, the discovery of "Web as platform", termed in some quarters as Web 2.0 (O'Reilly, 2005), and innovative websites like Flickr[8], Wikipedia[9], Google Map[10], Wikimapia[11] and Yahoo Maps[12] encourage social networking. Regarding to the core characteristics of Web 2.0, a website is no longer a static page to be viewed in a browser, but is a dynamic platform upon which users can generate their own experience. The richness of this experience is powered by the implicit threads of knowledge that can be derived from the content supplied by users and how they interact with the site. Another aspect of this Web as platform is sites which provide users with access to their data through well defined APIs and hence encourage new uses of that data, e.g. through its integration with other data sources.

Accordingly to Lyndon J.B. Nixon work (Nixon, 2006), as the current trends develop we expect to experience a future Web which will be media rich, highly interactive and user oriented. The value of this Web will lie not only in the massive amount of information that will be stored within it, but the ability of Web technologies to organize, interpret and bring this information to the user. And as usually, a graphical user interface is one of the important parts in performing of these processes. Media presentation is a key challenge for the emerging media-rich Web platforms. Several information visualization techniques have been developed in the last years due to the need of representing and analyzing the huge amount of data generated by several applications or made available through the World Wide Web. Previously, we had a deal with data visualization, precisely with a data format representation. Depending on a data format, whether it is a text, an image or a video, graphical user interface presents the data in certain way. On the next stage, small step has been done in visualization of object part-of relations, namely as a tree visualization. The second step was a step when we came to semantic definition of the objects, and have found a need to represent ontology concept tree and semantic graph (Yuxin et al., 2005). But it was just a step to semantics and ontology representation.

In following new technological trends, it is time to initiate a new stage of multidimensional resource visualization (visualization of resource properties, contexts of inter-resource communication and interaction) and a stage of semantic metadata-based visual browsing across resources.

5.1 Intelligent Resource Visualization

5.1.1 Motivation for intelligent resource visualization

One of the reasons for intelligent resource visualization springs from a necessity of the resource search and browsing processes enhancement.

With the reference to the research (Marcos et al., 2005), there are a number of important criticisms that can be made of Classical Model of information search. On the one

[8] http://www.flickr.com
[9] http://www.wikipedia.org
[10] http://maps.google.com
[11] http://www.wikimapia.org
[12] http://maps.yahoo.com

hand, this model does not adequately distinguish between the needs of a user and what a user must specify to get it. Very often, users may not know how to specify a good search query, even in Natural Language terms. Analyzing what is retrieved from the first attempt is used not so much to select useful results, as to find out what is there to be search over. A second important criticism of the Classical Model is that any knowledge generated during the process of formulation a query is not used later on in the sequence of search process steps, to influence the filtering step and presenting step of the search results, or to select the results. Finally, Classical Model provides an essentially context-free process. There is no proper way in which knowledge of the task context and situation, and user profile can have an influence on the information search process.

To address these criticisms, the WIDE Model of information retrieval (Marcos et al., 2005) treats the general task of information finding as a kind of design task, and not as a kind of search specification and results selection tasks. Information retrieval is understood as a kind of design task by first recognizing the difference between users stating needs and forming well specified requirements, and then properly supporting the incremental development of a complete and consistent requirements, and the re-use of the knowledge generated in this (sub) process to effectively support the subsequent steps in the process that concludes in a useful set of search results. Also, there are several projects that are aimed to somehow enhance the Classical Model of information retrieval. For example, a problem of search query uncertainty has been faced in one of the projects of Industrial Ontologies Group (IOG):"*Semantic Facilitators for Web Information Retrieval"*[13]. The main idea of the project is that Semantic Search Assistant/Facilitator (SSA) uses ontologically defined knowledge (WordNet[14]) about words from Google search request and provides possibility for user to specify right meaning of the words from available set of them. Further, based on the description of selected word meaning, SSA uses embedded support of advanced Google-search query features in order to construct more efficient queries from the formal textual description of searched information (Kaykova et al., 2004).

Thus, we can see how much work is doing in the area of the classical information retrieving model enhancement by adding some new useful features. Even more, nowadays there are a lot of efforts aimed at the creation of a fully ontology-based semantic query and search mechanisms, where search query is created based on ontological concepts specification. But it is complicate and challenging task. And still, it is not evident how to detect the user needs and to provide only relevant ontological concepts for the user during the query specification.

With growing interest to the usage of Wiki-based systems, in near feature we will face a huge mass of information. But, fully unintelligible for machines/software, and very problematically and unhandy searched/browsed by human, this information will be totally useless. Nowadays, interest of experts starts to turn to the side of Web 3.0. Web 3.0 is a term that has been coined to describe the evolution of Web usage and interaction that includes transforming the Web into a database, a move towards making content accessible by multiple non-browser applications, the leveraging of artificial intelligence technologies and the Semantic web and three dimensional interaction and collaboration (Web 3.0). According to Wikipedia, Web 3.0 is the final step in the decomposition of monolithic Web Pages into

[13] SemanticFacilitator (project report) - www.cs.jyu.fi/ai/OntoGroup/SemanticFacilitator.htm
[14] WordNet - http://wordnet.princeton.edu

discrete components that include: Presentation (HTML and (X)HTML), Logic (Web Services APIs), and Data (Data Models) trinity that transitions Web Data containment from Web Pages to Web Data. Its emergence simplifies the development and deployment of Data Model driven composite applications that provide easy, transparent and organized access to "the world's data, information, and knowledge". Web 3.0 thus promises to be much more useful than 2.0 and to render today's search engines more or less obsolete. To fit the vision of Web 3.0, there is a possibility to annotate Wiki-system content (using Semantic MediaWiki) and apply knowledge of a given subject (or domain) to build intelligence into Wiki. It adds semantics to the Wiki content, but search and browsing of annotated information are left as nonreinforced issues.

Thus, when we come to the vision of GUN (where all the resources of the virtual and the real world are connected and interoperate with each other) and an amount of information becomes so huge, we have to elaborate new visualization techniques that simplify the search and browsing processes through reducing amount of queries via context-dependent resource visualization. Following this approach, we have a need somehow to visualize the resource properties (in different from "directed arc between objects" representation way), the various relations between resources and the inter-resource communication process. And even more, we have a need to make this visualization more context-dependent, to be able to represent information in handy and adequate to a certain case (context) way, to reach a plasticity of UIs (Thevenin and Coutaz, 1999). Thus, the main focus in GUI development will be concentrated on the resource visualization aspects and we have a challenging task of *semantically enhanced context-dependent multidimensional resource visualization*.

5.1.2 Visualization of a resource

Several information visualization techniques have been developed in the last years due to the need of representing and analyzing a huge amount of data generated by several applications or made available through the World Wide Web. In the beginning, information systems had a deal with data visualization, precisely with a data format representation. Depending on a data format, whether it is a text, an image or a video, graphical user interface presents a data in certain way. On the next stage, small step has been done in visualization of object part-of relations, namely in a form of a tree. Then, there was a step when we came to semantic definition of the objects, and have found a need to represent ontology concept tree and semantic graph (Yuxin et al., 2005). Recent expectations regarding a new generation of the Web strongly depend on a success of Semantic Web technology. But it was just a small step to the semantics and ontology representation.

Information visualization aims to provide compact graphical presentations and user interfaces for interactively manipulated large numbers of items. Information visualization is the study of how to effectively present information visually. A lot of work in this field focuses on creating innovative graphical displays for complicated datasets. Now it has become evident that we cannot separate visual aspects of both data representation and graphical interface from interaction mechanisms that help a user to browse and query a data set through its visual representation.

The problem of manipulation with a huge amount of information is a complexity of the search query specification and provisioning of the relevant links for the through content browsing. The idea of intelligent resource visualization is to simplify the search and browsing processes via associative resource visualization. Multidimensional associative resource visualization means visualization of a resource depending on a context, via association with various aspects of resource being (relations with the other resources, domains, areas of interest, etc.). Some times we cannot specify exactly what we are looking for, but we feel that it is somehow related to certain stuff, certain situation, certain context. Such visualization can give us a hint, turn to the right direction, show us related objects and provide links to them. In other words, visualization will utilize context-based filtering and enrichment of the visualized scene with the relevant links.

All the resources have a set of properties, and if consider that all the resources are parts of the World and all of them are related and somehow linked to each other, then we have a very huge amount of resource properties. It seams like it is impossible to elaborate handy query system that will operate with all these properties. Thus, context-based approach is a grate solution to resolve this problem. In certain sense, context is an extraction of the resources, some of their properties, relations that are relevant to certain situation, action or other aspect of resource being (see FIGURE 5.1). Applying context-dependent visualization we reduce amount of the "steps" on a path leading to the final destination. Thus, a context can be a basis for specific visualization view of a resource and other related to it resources. For example, "part-of" relation (if it concerns physical relation of resources) can be visualized as a 3D model of nested or somehow connected resources; or 2D model, if third dimension is not valuable (for example to present the resources on a map, if they are part-of the World). From the other side, "part-of" relation of a resource can be abstract and cannot mean any physical contact with other resource. Resource can be a part of some (business) process. In this case, there is no physical contact between the parts and such relation should be represented in different way (for example relation graph).

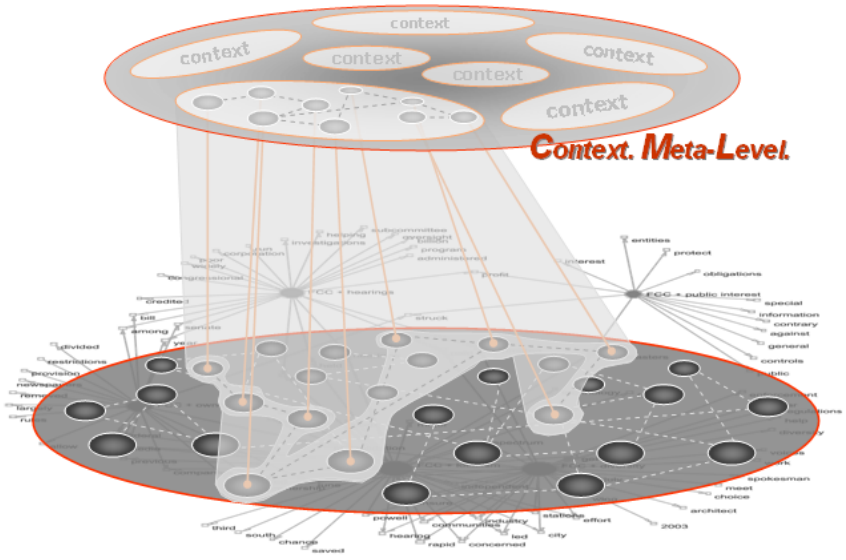

FIGURE 5.1 Context. Meta-level extraction

To show the different dimensions of a resource, let us consider a “person” resource (FIGURE 5.2). The visualization of a “person” in a context of healthcare & condition of one’s organs can be performed in a way of human body diagram (with a view of the organs). This can allow expert/doctor to check the organs’ condition (based on visualized properties), the influence of the organs on each other and the body systems; to switch the view to the internal view of necessary organ and manipulate with related information. At the same time, ”person” resource in a context of healthcare & location of a healthcare organization can be visualized in a form of a map with a highlight location, whether it is an organization, which belongs to the person employer, or just the nearest organization to the person location. Another visualization dimension of a “person” resource can be occupation/profession. It can be performed via visualization of a working area/place with the relevant work-related links: duties, area of interests, professional related resources, contacts, etc. For example, if the “person” is a goalkeeper of a football team, then its visualization in a context of profession can be displayed in a form of a team on a football field. Then we have an access to the other team members (know their roles in the team), have a link to a stadium (“stadium” resource, which provides facilities for the training) and to the home “team” resource. One more context of “person” resource visualization can be a family relation of a person. In this case, visualization can be performed in a form of a genealogical (family) tree. Several other examples of context-dependent resource visualization can be founded in the figure below (FIGURE 5.2).

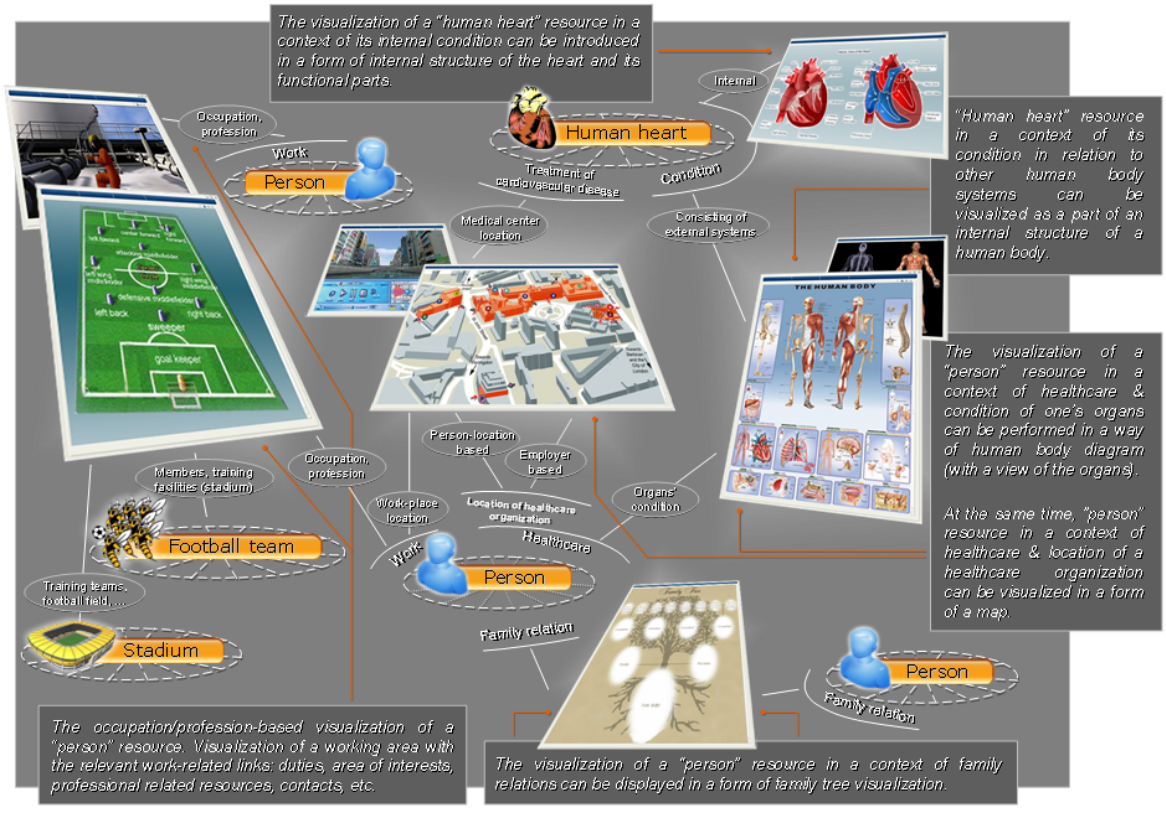

FIGURE 5.2 Resource visualization

Depending on a context, human/expert needs information (information related to subject resource) to be visualized in certain way. This gives us one of the requirements for visual

interfaces – ability to represent information regarding to chosen contextual property of a resource or contextual situation. Such interface should allow user to simply choose a context for the data representation, and should even support cases of multiple contextual property selection for the complex views and filtering purposes. Choosing the appropriate representation(s) is challenging and research is needed to evaluate and compare different approaches. At the same time, different visualization modules are specialised and present resources in certain domain. Such domains can target restricted set of resources as well as wide. For example, one of the restricted domains can be anatomy and processes between the parts of the human body. In this case, a visualization module can present the information based on 3D human body visualization and should be able to visualize the properties from anatomy domain. This task becomes more challenging in case of a wide target domain. For example, we need to represent information vie spatial resource location view and cover all of the resources, and then interface should somehow visualize all the properties. Thus, the next requirement for visual interface is to be able to visualize the properties of the resources that are belonging to the covered domain ontology.

5.2 4I (FOR EYE) TECHNOLOGY

5.2.1 Utilization of Intelligent Resource Visualization in next-generation systems

There are already some developed domain-oriented software applications, which try to visualize data in domain specific and suitable for human way (graphics software from SmartDraw®[15], concept-browser Conzilla[16] and Human Semantic Web browser Conzilla2, Google Maps, etc.). But still, they are developed for specific standalone domain-oriented applications. And when we face real need in an open unlimited collaboration environment (Web 3.0), we will have to develop mach more visualization tools and modules that are aimed to visualize various resource properties, contexts, situations and associations to provide human flexible and handy Human-Machine interaction interface. Thus, semantic-based context-dependent multidimensional resource visualization approach can be a basis for the development of such interface. It is well known that the 3rd dimension gives new possibilities for organising, presenting and interacting with content. Ideally 3D will make computer interaction easier, more intuitive and more natural. 3D is also very useful for presenting great amounts of data in easily understood 3D visualisations or graphs. That is why Web3D Consortium [17] aims to promote the open standards for Real-Time 3D Communication. Concerning the 3D software producers we can highlight Octaga AS[18] as a producer of world class real-time 3D software products, which creates outstanding content for real-time 3D; and Media Machines Inc.[19] - a leading provider of 3D virtual worlds

[15] SmartDraw® - www.smartdraw.com
[16] Conzilla concept-browser – www.conzilla.org
[17] Web3D Consortium - www.web3d.org
[18] Octaga AS - www.octaga.com
[19] Media Machines Inc. - http://william.mediamachines.com

software and services for the Web. Some analysts remain divided over whether 3D on the Web will be of much interest to a general audience. At the same time, marketers have reported success with 3D retail environments in which products can be rotated and manipulated in three dimensions. Other recently grooving areas of use include multi-user games, e-learning applications, data visualization and warehousing, and collaborative design and engineering. Presented resource visualization approach can be utilised in the development of Web 3D application to enhance them with more natural and associated resource visualization and context-dependent browsing technique.

Now it has become evident that we cannot separate visual aspects of both data representation and graphical interface from interaction mechanisms that help a user to browse and query a data set through its visual representation. Following GUN-Resource centric approach, let us consider user interfaces for context-based resource access and contextually related information retrieving. The challenging task is to create a visual interface that provides integrated information from variety of information providers in context-dependent way. And here, Intelligent Resource Visualization is a basis for 4i (FOR EYE) technology, that can be considered as a valuable extension of the text-based Semantic MediaWiki to Context-based Visual Semantic MediaWiki, a new generation of resource collaboration environments that follow the vision of Web 3.0.

5.2.2 4i Infrastructure

As was mentioned in previous section, one of the requirements for visual interfaces is an ability to represent information regarding to chosen contextual property of a resource, an interface should allow user to simply choose a context for data representation, and should even support cases of multiple contextual property selection for complex views and filtering purposes. Such requirements can be met by MetaProviders - sui generis portals of Resources with specific visualization view. It is named by MetaProvider in a sense that it provides an access and presents other resources, which in turn are providers of own information (data). All GUN-Resources have certain own location (physical and digital). But it does not mean that they should have just one way to get an access to them. MetaProvider is an interface-mediator that gives a possibility to mark/select a resource (object) on its interface and provide the link to original resource location. In other words, it allows resource registration for further access to its data. At the same time, any resource can be registered on variety of different MetaProviders in different views. The main feature of MetaProviders is each party that takes care of some GUN-Resource registers the resource itself. It causes fast filling of information accessible through MetaProvider. And each user/resource in one moment has an access to related information of amount of others. But such interoperability brings a new requirement for the MetaProviders and users. They should share common ontology to be interoperable on semantic level. Additionally to semantic interoperability, GUN-Resources are proactive/goal-driven resources and supplied with a Resource Agent for resource-to-resource (R2R)/ agent-to-agent (A2A) communication.

4i (FOR EYE) is an ensemble of GUN Resource Platform Intelligent GUI Shell (smart middleware for context dependent use and combination of a variety of different MetaProviders depending on user needs) and MetaProviders, visualization

modules/platforms that provide context-dependent filtered representation of resource data and integration on two levels (information/data integration of the resources to be visualized and integration of resource representation views with a handy resource browsing) (see FIGURE 5.3). Context-awareness and intelligence of such interface brings a new feature that gives a possibility for user to get not just raw data, but required integrated information based on a specified context. GUI Shell allows user dynamic switching between MetaProviders for more suitable information representation depending on a context or resource nature. From other side, MetaProvider plays fore main roles:

- Context-aware resource visualization module that presents information regarding to specified context in more suitable and personalized for user form;
- Interface for integrated information visualization with intelligent context-aware filtering mechanism to present only relevant information, avoiding a glut of unnecessary information;
- Visual Resource Platform that allows resource registration, search, access and modification of needed information/data in a space of registered resources;
- Mediator that facilitates resource to resource (R2R) communication.

Such switching and filtering process is a kind of visual semantic browsing based on semantic description of the context and resource properties.

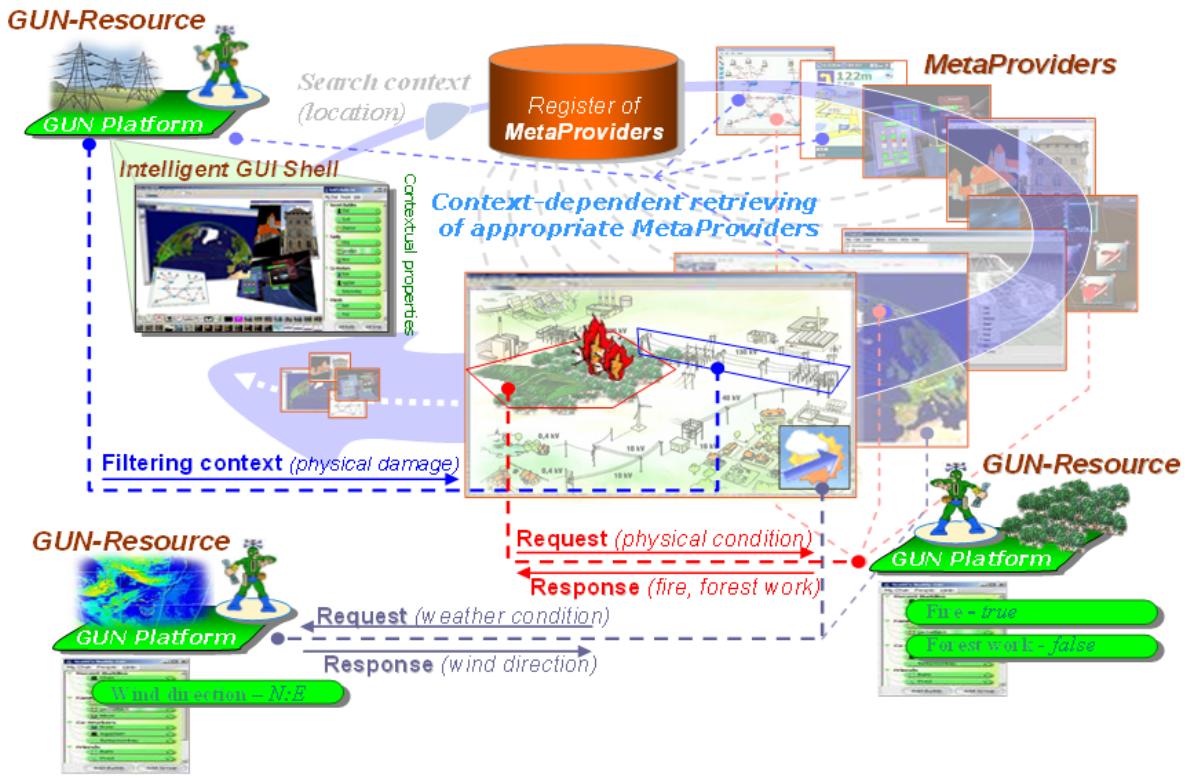

FIGURE 5.3 Intelligent Interface for Integrated Information (4i technology)

Let us consider GUN-Resource that presents a power line as a main initiator of a visualization process (FIGURE 5.3). Intelligent GUI, as a part of GUN Platform, provides an

opportunity for user to initiate context-based search process that returns appropriate MetaProvider or a set of them. Search process can be performed via centralized or decentralized system of MetaProviders registration. Depending on a contextual property, Intelligent GUI Shell provides an access to filtered set of retrieved MetaProviders. Then user can register the resource (if it is not registered yet) or/and get related to the resource integrated information on MetaProvider interface. It is not necessary that search result will contain just one instance of a fit class of MetaProviders. This is an open environment and there is can be a set of various realizations of MetaProviders from different producers. Also, GUI Shell allows dynamic switching between MetaProviders for more suitable information representation, depending on a context (a set of contextual resource properties). From other side, MetaProvider provides API to specify information filtering context – a context for visualization of appropriate resources and their necessary properties. In this scenario user asks the MetaProvider to show resources in certain area around the subject resource (power line) in the context of physical damaging and relevant to this resource properties. Thus, physical conditions of other two resources (forest and weather that are shown in the figure) have been requested and the values of correspondent properties have shown on the interface.

Now, when expert has recognized alarm situation, he/she need, for example, to change the architecture of the electrical chain (electricity supplying) and for this purpose can easily switch to another MetaProvider with more appropriate internal view of power line architecture. Another valuable benefit of such smart ensemble architecture is possibility to perform autonomous agent-based resource communication via MetaProvider's and GUI Shell APIs. It is an open environment for MetaProviders and it is a good base for different business models that can be built on it. Thus, with a purpose to be an interoperable part of open environment, each player has to be supplied with API and should be semantically adapted to understand the requests and to provide understandable response.

5.2.3 Requirements for the technology components

Intelligent GUI Shell:

- ***context-based selectiveness and filtering***

One of the main features of the GUI Shell is an ability to represent the information regarding to chosen contextual property of the resource. There are two aspects: relevance of presented information to the context and dependence of information representation view on the data representation context. Interface should allow user to simply choose a context for data representation, and support even multiple contextual property selection for complex views and filtering purposes. It is not necessary to define all of the contexts manually, the system should be supplied with a module for intelligent construction of the complex context from the initial one based on domain ontology and knowledge bases. Concerning the example from the previous section (FIGURE 5.3), a context of needed data is a physical damaging of the resource (power line).It is quite natural that physical damaging can be caused directly or indirectly by other resources that are located near by. From this we can get a context for better data representation view. This context is location (spatial resource representation). From the other hand, it is evident that it is much better to have as many as possible resources

(allocated near by subject resource) registered on certain instance of appropriate MetaProvider. It gives us additional parameter for search query or search result filtering such as an amount of the relevant registered resources. Concerning an intelligent filtering of data to be presented, system should select and visualize just relevant to the context resource properties. Regarding to our case, it is not necessary to visualize all of the properties of resource "Forest" if just fire condition and forest works can cause a physical damage of the resource "Power Line". And of course, to build such intelligent filtering mechanism, we need to have knowledge, ontology of resource relations and different processes descriptions.

Thus, the GUI-Shell should provide handy interface for resource properties browsing and context generation. And if we are talking about an interface that provides dynamic switching between different views (MetaProviders), then Shell should provide a possibility to generate context from the visualization module (MetaProvider) interface. It brings a need to organise a flexible mechanism for interoperation between Shell and MetaProviders' interfaces.

- ***personalization***

Concerning the personalization issues, GUI-Shell should provide user personalization features. System should try to remember user preferences regarding to visualization views, more popular MetaProviders and etc. At the same time system should support alliance approach, when resources are bound in some business, production or another process. In this case system should propose the MetaProviders which have registered the necessary parts (partners). Another personalization issue concerns the preferred information visualization (visualisation of the resource properties). Because there are can be many MetaProviders, which are belonged to the same class of MetaProviders, but the methods of visualization certain properties may be different. That is way system should detect the preferred for user visualization methods and provide user personalization on that level.

MetaProvider:

- ***integrated data representation***

As a portal for registered resources, MetaProvider should support a simple resource registration process (depending on a specific of presentation view), store resource semantic profiles for further resource discovery and inter-resource communication. Inter-resource communication implies a following the same ontology by resources. Following the common ontology brings a possibility to create a correct request to the resources, and get and present appropriate data from them. As was mentioned before, integrated data representation means representation of the data from different resources in certain context related to a subject resource. It is a representation of filtered relevant to the certain case data. There is a sense to perform a filtering process on the user side (i.e. on the side of GUI-Shell), but an input data for the request creation and filtering process should come from the resources that are accessible via MetaProvider API.

- ***resource property visualization***

Information Visualization aims to provide compact graphical presentations and user interfaces for interactively manipulating large numbers of items. Information visualization is the study of how to effectively present information visually. Much of the work in this field focuses on creating innovative graphical displays for complicated datasets.

Information Visualization allows designers to present a large amount of information using abstract representations. Geographic and scientific visualization applications usually use representations determined by the nature of the data being displayed. Location within the graphic is usually used to represent location in space. For example in 2D maps, a projection of space structures the representation, whilst in 3D models of the body or of physical processes such as meteorological predictions graphics are constrained by a 3D spatial framework. On the other hand, Information Visualization allows designers to choose among a palette of possible representations that fill space in a variety of ways, such as hierarchies, time lines, networks, tabular displays and the like, to produce information abundant displays. Choosing the appropriate representation(s) is challenging and research is needed to evaluate and compare different approaches.

Visualization techniques include selective hiding of data, layering data, taking advantage of 3-dimensional space, using scaling techniques to provide more space for more important information (e.g. fisheye views), and taking advantage of psychological principles of layout, such as proximity, alignment, and shared visual properties (e.g. color). Advanced interfaces also need to address the longer term process of analysis that may require annotation, history keeping, collaboration with peers, and the dissemination of results and procedures used. Faster rendering algorithms, sophisticated aggregations techniques to deal with large datasets, and novel labelling techniques are also needed, and along with careful studies of users and their needs will lead to successful visualization applications.

Different MetaProviders are specialised and present the resources in certain domain. Such domains can target restricted set of the resources as well as wide. For example, one of the restricted domains can be anatomy and processes between the parts of the human body. In this case, MetaProvider can present the information based on 3D human body visualization and should be able to visualize the properties from anatomy domain. This task becomes more challenged in case of wide target domain. For example, if MetaProvider represents information vie spatial resource location view and covers all of the resources, then it should somehow visualize all the properties. Thus, the main requirement in that part is to be able to visualize the properties of the resources that are belonging to the covered domain ontology.

- ***dynamic interoperability***

Thus, interoperability among the components of the environment will be provided via metadata and ontologies. Additionally, if we consider the open environment, members of which are agent-driven (i.e. proactive, autonomous, goal-driven, intelligent and etc.) to enable communications, coordination and negotiations between them, then MetaProvider should support inter-agent communication via API.

- ***usability***

Here we come to usability issues that play important role in user interface development. If we use the term data usability to describe quality of information or quality of data in the context

of information visualization applications, then, regarding to (Freitas et al., 2002), data usability is associated to three principles:

a) data reliability, which describes the feasibility of the gathering data process as well as the confidence level, including interval for errors, etc. that can cause distortion between reality and model (reality represented by the system);

b) minimal impact on data changing, i.e., the system must avoid changing the information and it must allow recovering original information whenever it is needed. However, in practice, this data stability is not feasible because frequently data must have to be adapted to visualization constraints such as the reduction of dimension when presenting n-dimensional data in a 2D or 3D visualization, for example. This 2D or 3D representation breaks down the usability of original data. It is clear that we cannot avoid some changes during the visualization process but we can try to reduce their impact; and

c) support decision-making, which means that data representation should be understandable by end-users and help them to make decisions.

Common usability suggestions:

Accordingly to W3C Working Group Note: "Common Sense Suggestions for Developing Multimodal User Interfaces" (Larson, 2006), the suggestions are organized into four major principles of user interface design. The following four principles determine how quickly users are able to learn and how effectively they are able to perform desired tasks with the user interface:

- Satisfy real-world constraints
- Communicate clearly, concisely, and consistently with users
- Help users recover quickly and efficiently from errors
- Make users comfortable

There is a set of the different aspects suggestions: task-oriented, physical, environmental, consistency, organizational, conversational, reliability, social, advertising, ambience, accessibility, suggestions that concern listening mode, system status and human-memory constraints. Multimodal user interface developers should follow the above four principles and apply the following suggestions to avoid many of the potential usability problems caused by using modes incorrectly.

5.3 4i (FOR EYE) Technology in UBIWARE

In UBIWARE, humans are important resources, which can play several distinct roles: a resource under care, a service provider, a user, and an administrator. Obviously, the humans need some graphical interfaces to interact with the rest of the system. The same person can play several roles, switch between them depending on the context, and, in result, require different interfaces at different times. In addition, a UBIWARE-based system presents a large integration environment with potentially huge amounts of heterogeneous data. Therefore, there is a need for tools facilitating information access and manipulation by humans. A semantic context-aware multimodal visualization approach would provide an opportunity for

creating smart visual interfaces able of presenting relevant information in a more suitable and more personalized form.

From the GUN point of view, a human interface is a special case of a resource adapter (Human-Resource AdaPter - HRAP). We believe, however, that it is unreasonable to *embed* all the data acquisition, filtering and visualization logic into such an adapter. Instead, *external* services and application should be effectively utilized. Therefore, the intelligence of a smart interface (HRAP) will be a result of collaboration of multiple agents: the human's agent, the agents representing resources of interest (those to be monitored or/and controlled), and the agents of various visualization services – MetaProviders via an agent of HRAP (see FIGURE 5.4). This approach makes human interfaces different from other resource adapters and indicates a need for devoted research.

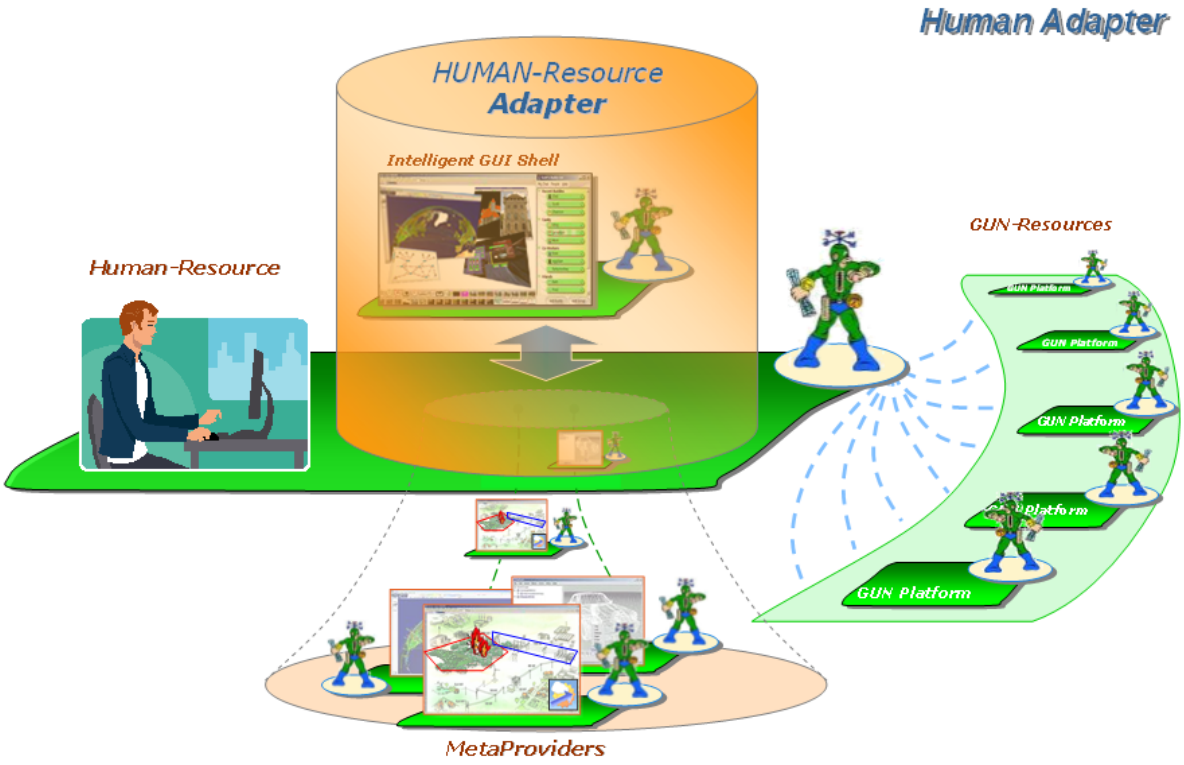

FIGURE 5.4 Human-Resource Adapter based on 4i (FOR EYE) technology

Workpackage 5 - "Smart Interfaces: Context-aware GUI for Integrated Data (4i technology)" will study dynamic context-aware A2H (Agent-to-Human) interaction in UBIWARE, and will elaborate on a technology which we refer to as 4i (FOR EYE) technology. 4i (FIGURE 5.5) enables creation of a smart human interface through flexible collaboration of an Intelligent GUI Shell, various visualization modules, which we refer to as MetaProvider-services, and the resources of interest.

Based on 4i technology, an infrastructure will be embedded into UBIWARE enabling effective realization of the following system functions:

- visualization of data provided by a service in response to a request;

- search, retrieving and visualization of data required by a human expert,
- providing access to contextual information, and visualization of it;
- visualization of resource registration, configuration, and security policy establishment processes;
- resource discovery via MetaProviders (because they act as thematic portals).

This workpackage plays a significant role in UBIWARE – it connects human eyes and human hands to it. The results from this workpackage will enable UBIWARE to provide flexible and context-aware interfaces to humans participating in activities of a UBIWARE-based system.

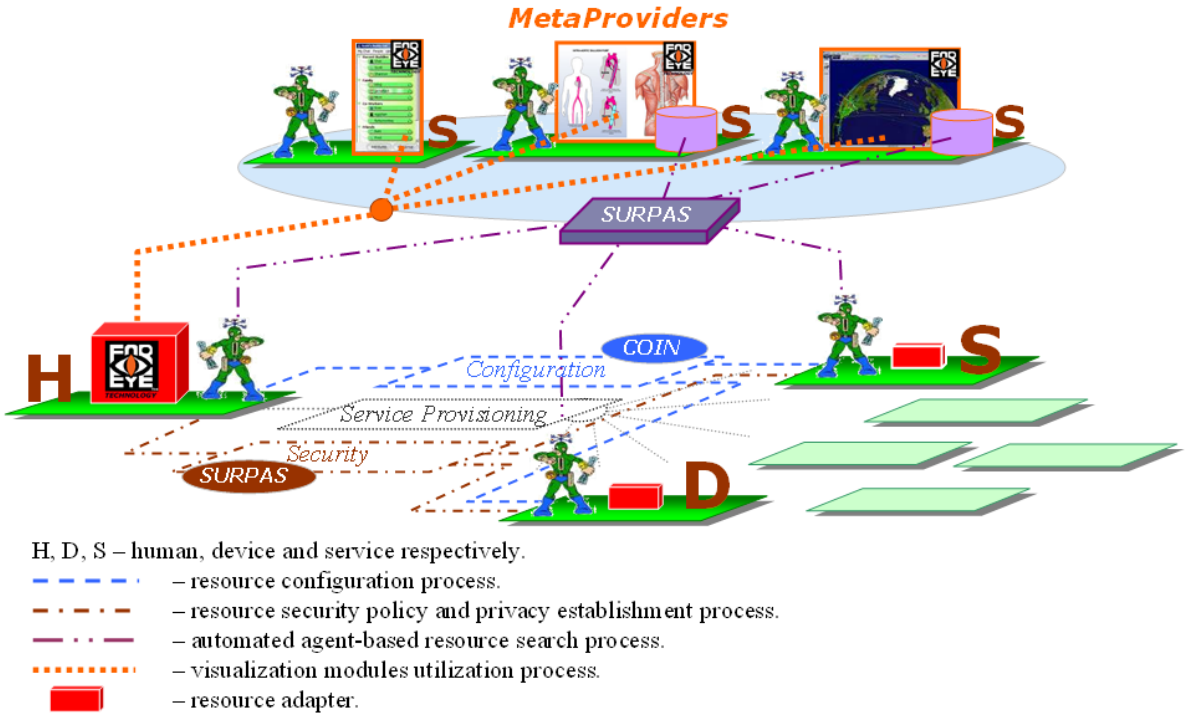

FIGURE 5.5 Processes related to UBIWARE Platform and 4i (FOR EYE)

Assuming that a UBIWARE-based system will have some human administrators, there is a need for a GUI through which the administrators will be able to manually configure the system, which also includes defining the security policies. Therefore, 4i must allow for that, and special MetaProviders have to be developed for creating interfaces to perform such administrative functions. As can be seen, the research on smart interfaces will affect and will be affected by research and results from other UBIWARE project workpackages, such as WP3: SURPAS "Smart Ubiquitous Resource Privacy and Security" and WP4:COIN "Self-Management, Configurability and Integration".

5.4 Application of 4i (FOR EYE) Technology: Semantically enhanced browsing across multimedia contents.

Let us choose the multimedia content management area for the case study as a one of the fastest growing area of the Web.

5.4.1 Resource semantic track

A number of vocabularies that deal at some level with multimedia content currently exist (Geurts et al., 2005): MPEG-7, Dublin Core Element Set, VRA, Media Streams, Art and Architecture Thesaurus (AAT), MIME, CSS, Composite Capabilities/Preference Profiles (CC/PP), PREMO, Modality Theory, Web Content Accessibility Guidelines. Of course, it is very important to develop appropriate format for semantic annotation of multimedia content. But, from the other hand, it is more natural to find the way to build-in full semantics to the digital formants of multimedia (image, video, audio). Nowadays, production houses shoot high-quality video in digital format; organizations that hold multimedia content (such as TV channels, film archives, museums, and libraries) digitize analog material and use digital formats. Maybe it is a time to reach all the digital media formats with a Semantic Track, which will contain not just content structure, but full semantic content annotation including: content structure, concepts, objects, actions and etc.

Considering the main aspect of the discussions around a multimedia, Human is a main customer of multimedia services and an end-user of a multimedia content. With a sustainable multimedia content growing, Human/User needs new intelligent techniques for multimedia content browsing, search/retrieving and adapted representation. At the same time, the stated goal of the Semantic Web initiative is to enable machine understanding of web resources. However, it is not at all evident that such machine-readable semantic information will be clear and effective for human interpretation. Hence, in order to effectively harness the powers of the semantic web, it needs a “conceptual interface” (Naeve, 2005), that is more comprehensible for humans. Such conceptual interface can improve multimedia content retrieving process and together with well elaborated Semantic Track of the multimedia resources, can provide a unique basement for semantically enhanced across multimedia contents browsing.

The sub-symbolic abstraction level covers the raw multimedia information represented in well-known formats for video, image, audio, text, metadata, and etc., which are typically binary formats, optimized for compression and streaming delivery. They aren’t well suited for further processing that uses, for example, the internal structure or other specific features of the media stream. A structural (symbolic) layer on top of the binary media stream provides this information. The standards that operate in this middle layer for the representation of multimedia document descriptions are: Dublin Core, MPEG-7, Visual Resource Association, and so on. The problem with this structural approach is that the semantics of the information encoded in the XML are only specified within each standard’s framework. MPEG-7 was not built specifically for web applications and thus does not facilitate embedding links to other resources and interoperability between them. A possible solution to resolve the

interoperability conflict is to add a third layer (the logical abstraction level) that provides the semantics for the middle one, actually defining mappings between the structured information sources and the domain's formal knowledge representation based on semantically enriched languages (RDF and OWL). RDF-based languages and technologies provided by the W3C community is well suited to the formal, semantic descriptions of the terms in a multimedia document's annotation. A combination of the existing standards seems to be the most promising path for multimedia document description in the near future. For these reasons, the W3C has started a Multimedia Annotation on the Semantic Web Task Force[20] as part of the Semantic Web Best Practices and Deployment Working Group. The new task force operates within the framework of the W3C Semantic Web Activity group[21]. One goal is to provide guidelines for using Semantic Web languages and technologies to create, store, manipulate, interchange, and process image metadata. Another is to study interoperability issues between multimedia annotation standardization and RDF- and OWL-based approaches. Hopefully, this effort will provide a unified framework of good practices for constructing interoperable multimedia annotations.

Research towards a multimedia content and content description bounding has been going during the last several years. Commonwealth Scientific and Industrial Research Organization have developed an open source family of technologies ANNODEX (Pfeiffer et al., 2003) for embedding annotations and hyperlinks directly within digital audio and video files. Such embedding allows the combined resource to become just like any web document which has content and content description bound into one. Also, the idea of a media semantic track utilizing has been elaborated in another research (Khriyenko, 2005), which concerns issues of multimedia smart messaging in an environment of limited devices. Semantic annotation of multimedia content is performed by using appropriate domain specific ontologies that model the multimedia content domain. Ontologies typically represent concepts by linguistic terms. However, also multimedia ontologies can be created, that assign multimedia objects to concepts. At the same time with semantic content metadata annotation, annotation of the concepts of: people (artist, owner, restorer, author, producer, etc.), art objects and representations (painting, sculptures, films, digital representations, etc.), events and activities, places, methods and techniques, and etc., we should provide a basis for multimedia content features to be presented in semantic annotation also. This gives a possibility for better automatic annotation of the multimedia content. Further we try to specify the features of the multimedia content that can be detected and presented in Semantic Track. In (Bertini et al., 2005) authors present a list of systems of automatic semantic annotation, most of them in the application domain of sports video. Among these, there is an approach, where MPEG motion vectors, playfield shape and players position have been used with Hidden Markov Models to detect soccer highlights.

Another approach has been aimed to detect the principal soccer highlights, such as shot on goal, placed kick, forward launch and turnover, from a few visual cues. Additionally, the ball trajectory also has been used in order to detect the main actions like touching and passing and compute ball possession by each team; a Kalman filter is used to check whether a detected trajectory can be recognized as a ball trajectory. But, in all these approaches a model based event classification is not associated with any ontology-based representation of

[20] http://www.w3.org/2001/sw/BestPractices/MM/
[21] http://www.w3.org/2001/sw/

the domain. However, although linguistic terms are appropriate to distinguish event and object categories, they are inadequate when they must describe specific patterns of events or video entities. In this case, high level concepts, expressed through linguistic terms, and pattern specifications represented instead through visual concepts, can be both organized into new extended ontologies, that will be referred to as pictorially enriched ontologies. Ontologies can be extended to multimedia enriched ontologies where concepts that cannot be expressed in linguistic terms are represented by prototypes/patterns of different media like video, audio, etc. The audio features used to characterize the sound signal and classify the sample by instrument. The CUICADO project (Peeters, 2003), provided a set of 72 audio features, and research has shown that some of the features are more important in capturing the signal characteristics: temporal shape, temporal feature, energy features, special shape features, harmonic features, perceptual features and MPEG-7 Low Level Audio Descriptors (spectral flatness and crest factors). Now we can see how many multimedia-specific features and properties can enrich a Semantic Track of multimedia resources.

5.4.2 Across multimedia content semantic browsing in a sense of concept based semantic search

We have to consider another developing trend on the Web – a growth in multimedia content. Technological progress has meant that we have never had access to so much media content as now. Future challenges for the Web will be the meaningful organization of this huge amount of online media content as well as its meaningful delivery to the user. However, the present state of the art of media and Web technologies prevents richer integration. A multimedia semantic browsing, as a sub-class of general resources browsing is a complex process that combines a set of sub-processes. This process can be performed based on presented 4i (FOR EYE) technology. Figure (FIGURE 5.6) shows us an example of an across multimedia contents semantic browsing architecture. In the left center of the figure, a GUI-Shell is presented as a combination of the tools that take parts in the process: multimedia content player, Semantic Track visualization component, concept browser and semantic search query builder/creator.

FIGURE 5.6 Multimedia semantic browsing

Let us consider an example, where user is watching an episode of a movie with some song (soundtrack) at the background. User likes this song/melody and would like to find more songs of this author (or even more complex goal – find similar songs to the initial one). To achieve the goal, user should browse Semantic Track of this video instance, which contains a structure of a video file, objects, actions, soundtracks, etc.; and find a reference to the searched song. Then, utilizing a concept browsing tool, which is connected to remote ontology, user can specify a semantic query for a needed multimedia resource (in our case - a song). Such query specification can be considered as a creation/construction of a resource semantic pattern (virtual nested resource with specified properties). As a result of the search process, appropriate audio resource will be returned and even lyrics of the song can be displayed based on its' Semantic Track.

But it was just a simple case of semantic search/browsing process. Multimedia Resource Semantic Track usually contains just a structure of content and descriptions of multimedia content specific features. And because of this, very often we can not specify direct linking between the contents of two Semantic Tracks of the different resources. The "glue" for these two semantic annotations is situated in Semantic Knowledge Bases (for example semantically-enhanced Wikipedia or different ontologies). It can be useful in the next example. Now we are looking for an image of the house of the first wife of some actor from a movie that we are watching. Firstly, we stop the movie on a scene where this actor is presented and, based on Semantic Track, find a link to this person. Then we browse a semantic knowledge base via the concept browser and find a link to his first wife and her house. After semantic search query generation we get the searched image on the browser.

At the same time, approach of instance based search via MetaProviders can be beneficially utilized in multimedia content searching/browsing. Let us consider a case, when we would like to see other houses, which are located nearby the house of the mentioned wife. We can use some MetaProvider – Wikimapia kind of service, which provide an access to the registered resources via showing them on a map. If the image is registered on this service/platform, then we easily can find other registered images in the same area (location), especially if final visualization will be filtered in a context that searched resource is an image of a house.

Accordingly to the GUN approach, all the parts of searching/browsing process presented in GUI-Shell can be developed as separate functional modules (resource) and can be chosen by user to allow personalization of a browsing interface. In this particular use case of the OntoEnvironment, with resources of the real world (people, objects and etc.) we face new semantically-enhanced media-file resources. As was mentioned, these resources contain not just internal structure in their Semantic Tracks, but also links to other resources. Thus, with a purpose to be competitive in the open market of the media resources and have big rank of use, resources should be self-maintained and all the time should have up-to-date links in Semantic Track. Here we see the necessity of resource proactive behaviour. Supplied with an agent-based GUN Platform, behaviour of the resource can be configured in a way that gives resource a possibility to communicate with other resources and change/update own Semantic Track in real time (see FIGURE 5.7).

FIGURE 5.7 Semantically enhanced multimedia resource infrastructure

5.5 Conclusions

Now, when human becomes very dynamic and proactive resource of a large integration environment with a huge amount of different heterogeneous data, it is quite necessary to provide a technology and tools for easy and handy human information access and manipulation. Semantically enhanced context-dependent multidimensional resource visualization provides an opportunity to create intelligent visual interface that presents relevant information in more suitable and personalized for user form. Context-awareness and intelligence of such interface brings a new feature that gives a possibility for user to get not just raw data, but required information based on a specified context. Now, when unlimited interoperability and collaboration demand data and information sharing, we need more open semantic-based applications that are able to interoperate and collaborate with each other. Ability of the system to perform semantically enhanced resource search/browsing based on resource semantic description brings a valuable benefit for today Web and for the Web of the future with unlimited amount of resources. Proposed resource visualization approach can find a place and can be utilized in various visual systems and especially in next-generation human-centric open environments for resource collaboration with enhanced semantic and context-based visual resource browsing. Presented 4i (FOR EYE) technology quite fits the demands of a new generation of integration systems. It is an ensemble of Platform Intelligent GUI Shell and visualization modules – MetaProviders that provide context-dependent representation view of resource data and integration on two levels. These are: information

(data) integration of the resources to be visualized; and integration of resource representation views with a handy resource browsing in different dimensions. It can be considered as a new valuable extension of text-based Semantic MediaWiki to Context-based Visual Semantic MediaWiki. With the idea of the GUN we come to the environment where all the resources are semantically interoperable and have own semantic description – Resource Semantic Track. With the growing ubiquity of digital media content, ability to combine continuous media data with its own multimedia specific content description into the one source brings the idea of a true multimedia semantic web one step closer. 4i is a good basis for the different business, production, maintenance, healthcare, social process models creation and multimedia content management.

In the context of UBIWARE, 4i (FOR EYE) technology is a part of it. From one side, the technology is a base for Human-Resource adaptation and will be elaborated accordingly to the principles and vision of UBIWARE. The intelligence of this smart interface is a result of collaboration of multiple agents: the human's agent, the agents representing resources of interest (those to be monitored or/and controlled or requesting a human), and the agents of various visualization services – MetaProviders via an agent of Human-Resource Adapter. From the other side, accordingly to the 4i, specific interfaces (MetaProviders) will be developed to provide functionality for a human (Platform administrator) to configure functionality of a UBIWARE-based system.

5.6 Dissemination of the results

The idea and materials that were included in this technical report were presented in several international conferences and published as scientific papers:

- ***ICEIS-2007*:** 9th International Conference on Enterprise Information Systems.
  Khriyenko O., "4I (FOR EYE) Technology: Intelligent Interface for Integrated Information", In: Proceedings of the 9th International Conference on Enterprise Information Systems (ICEIS-2007), Funchal, Madeira – Portugal, 12-16 June 2007.

- ***SIGMAP-2007*:** International Conference on Signal Processing and Multimedia Applications.
  Khriyenko O., "4I (FOR EYE) Multimedia: Intelligent semantically enhanced and context-aware multimedia browsing", In: Proceedings of the International Conference on Signal Processing and Multimedia Applications (SIGMAP-2007), Barcelona, Spain, 28-31 July 2007.

- ***VIIP- 2007***: 7th IASTED International Conference on Visualization, Imaging, and Image Processing.
  Khriyenko O., "Context-sensitive Multidimensional Resource Visualization", In: Proceedings of the 7th IASTED International Conference on Visualization, Imaging, and Image Processing (VIIP 2007), Palma de Mallorca, Spain, 29-31 August 2007.

Bibliography

Bertini, M., Cucchiara, R., Bimbo, A. and Torniai, C., (2005). Ontologies Enriched with Visual Information for Video Annotation, In: *Multimedia and the Semantic Web workshop as part of the 2nd European Semantic Web Conference (ESWC2005-MSW)*, Heraklion, Crete, May-June, 2005.

Blakley, B., Heath, C., and members of The Open Group Security Forum. (2004). Security design patterns. Technical Guide No. G031. The Open Group.

Borselius, N., (2002). Mobile agent security, Electronics & Communication Engineering Journal, October 2002, Volume 14, no 5, IEE, London, UK, pp 211-218

Bosse, T., Treur, J. (2000). Formal interpretation of a multi-agent society as a single agent. Journal of Artificial Societies and Social Simulation 9(2)

CASCADAS. CASCADAS - Component-ware for Autonomic Situation-aware Communications, and Dynamically Adaptable Services, FP6 Integrated Project http://www.cascadas-project.org/

Collier, R., Ross, R., O'Hare, G. (2005). Realising reusable agent behaviours with ALPHA. In: Eymann, T., Klugl, F., Lamersdorf,W., Klusch, M., Huhns,M.N. (eds.) MATES 2005. LNCS (LNAI), vol. 3550, pp. 210-215. Springer, Heidelberg

Dastani, M., van Riemsdijk, B., Dignum, F., Meyer, J.J. (2004). A programming language for cognitive agents: Goal directed 3APL. In: Dastani, M., Dix, J., El Fallah-Seghrouchni, A. (eds.) PROMAS 2003. LNCS (LNAI), vol. 3067, pp. 111-130. Springer, Heidelberg

Deliver. Deliver: Intelligent Software Knowledge Management and Delivery, http://www.cwi.nl/htbin/sen1/twiki/bin/view/Deliver

Freitas, C. M. D. S., Luzzardi, P. R. G., Cava, R. A., Winckler, M. A., Pimenta, M. and Nedel, L. P., (2002). Evaluating Usability of Information Visualization Techniques. In: 5TH SYMPOSIUM ON HUMAN FACTORS IN COMPUTER SYSTEMS (IHC), 2002, Fortaleza - CE Proceedings. Fortaleza:SBC, 2002. p. 40-51.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J., (1995). Design patterns: elements of reusable object-oriented software. Addison-Wesley professional computing series. Addison-Wesley. Boston, Mass.

Geurts, J., Ossenbruggen, J., and Hardman, L., (2005). Requirements for practical multimedia annotation. In: *Multimedia and the Semantic Web workshop as part of the 2nd European Semantic Web Conference (ESWC2005-MSW)*, Heraklion, Crete, May-June, 2005.

Harrison, C. G., Chess, D. M. and Kershenbaum, A., (1995). Mobile Agents: Are they a good idea, technical report, 1995, IBM Research Division.

Horn, P., (2001). Autonomic computing: IBM's perspective on the state of information technology. IBM Corporation, Tech. Rep., 15 Oct. 2001. Available: http://www.research.ibm.com/autonomic/manifesto/autonomic_computing.pdf

Jansen, W., (2000). Countermeasures for Mobile Agent Security, Computer Communications, Special Issue on Advanced Security Techniques for Network Protection, Elsevier Science BV, November 2000.

Jansen, W. and Karygiannis, T., (1999). Mobile Agent Security, National Institute of Standards and Technology, Special Publication 800-19, August 1999.

Kaykova, O., Khriyenko, O., Klochko, O., Kononenko, O., Taranov, A., Terziyan, V. and Zharko, A., (2004). Semantic Search Facilitator: Concept and Current State of Development, In: InBCT Tekes Project Report, Chapter 3.1.3 : "Industrial Ontologies and Semantic Web", Agora Center, University of Jyväskylä, January – May 2004. URL: http://www.cs.jyu.fi/ai/OntoGroup/InBCT_May_2004.html

Kaykova, O., Khriyenko, O., Kovtun, D., Naumenko, A., Terziyan, V. and Zharko, A., (2005). 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.

Kaykova O., Khriyenko O., Terziyan V., General Networking Framework, Technical Report (Deliverable D 3.1), SmartResource Tekes Project, Agora Center, University of Jyvaskyla, January-May, 2006.

Kephart and Chess, D. M., (2003). The vision of autonomic computing. Computer, vol. 36, no. 1, pp. 41--50, Jan. 2003. Available: http://portal.acm.org/citation.cfm?id=642200

Khriyenko, O., (2005). SemaSM: Semantically enhanced Smart Message, In: *Eastern-European Journal of Enterprise Technologies*, Vol. 1, No. 13, 2005, ISSN: 1729-3774.

Larson, J. (2006). Intel (editor). W3C Working Group Note: "Common Sense Suggestions for Developing Multimodal User Interfaces", 11 September 2006, URL: http://www.w3.org/TR/2006/NOTE-mmi-suggestions-20060911/

Luostarinen, K., Naumenko, A., and Pulkkinen, M., (2006). Identity and Access Management for Remote Maintenance Services in Business Networks, in IFIP International Federation for Information Processing, Volume 226, Project E-Society: Building Bricks, eds. R. Suomi, Cabral, R., Hampe, J. Felix, Heikkilä, A., Järveläinen, J., Koskivaara, E., (Boston: Springer), pp. 1-12.

Marcos, G., Smithers, T., Jiménez, I., Posada, J., Stork, A., Pianciamore, M., Castro, R., Marca, S., Mauri, M., Selvini, P., Sevilmis, N., Thelen, B. and Zechino, V., (2005). A Semantic Web based approach to multimedia retrieval. In: Fourth International Workshop on Content-Based Multimedia Indexing (CBMI 2005), Riga, Latvia, 21-23 June, 2005.

Mazhelis, O., and Naumenko, A., (2006). The Place and Role of Security Patterns in Software Development Process, In Medina, E.F. and Yagüe, M.I. (Eds.), Security in Information Systems Proceedings of the 4th International Workshop on Security in Information Systems, WOSIS 2006 In conjunction with ICEIS 2006 Paphos, Cyprus, May 2006. INSTICC Press, pp. 91-100.

Naeve, A., (2005). The Human Semantic Web: shifting from knowledge push to knowledge pull, In: *International Conference on Computational Science (ICCS 2005)*, Atlanta, USA, 22-25 May, 2005.

Naumenko, A., (2006). Contextual rules-based access control model with trust. In Shoniregan C. A. and Logvynovskiy A. (Eds.), Proceedings of the International Conference for Internet Technology and Secured Transactions, ICITST 2006, 11-13 September, London, UK, ISBN 0-9546628-2-2, e-Centre for Infonomics, pages 68-75.

Naumenko, A., (2007), Semantics-Based Access Control – Ontologies and Feasibility Study of Policy Enforcement Function, In: J., Filipe, J., Cordeiro, B., Encarnacao, and V., Pedrosa (Eds.), In Proceedings of the 3rd International Conference on Web Information Systems and Technologies (WEBIST-07), Barcelona, Spain - March 3-6, 2007, Volume Internet Technologies, INSTICC Press, pp. 150-155.

Naumenko, A., Katasonov, A., Terziyan V., (2007), A Security Framework for Smart Ubiquitous Industrial Resources, In: Goncalves, R., Müller, J., Mertins K., and Zelm, M., (Eds.), In Enterprise Interoperability II: New challenges and Approaches, Proceedings of the 3rd International Conference on Interoperability for Enterprise Software and Applications (IESA-07), March 28-30, 2007, Madeira Island, Portugal, Springer, pp. 183-194.

Naumenko, A., and Luostarinen, K., (2006). Access Control Policies in (Semantic) Service-Oriented Architecture, In Schaffert S. and Sure Y. (Eds.), Semantic Systems From Visions to Applications, Proceedings of the SEMANTICS 2006, Austrian Computer Society, Vienna, Austria, pp. 49-62.

Naumenko, A., Nikitin, S., Terziyan, V., and Zharko, A., (2005). Strategic Industrial Alliances in Paper Industry: XML- vs. Ontology-Based Integration Platforms, In: The Learning Organization, Special Issue on: Semantic and Social Aspects of Learning in Organizations, Emerald Publishers, ISSN: 0969-6474, Vol. 12, No. 5, 2005, pp. 492-514.

Nixon, L., (2006). Multimedia, Web 2.0 and the Semantic Web: a strategy for synergy. In: First International Workshop on Semantic Web Annotations for Multimedia (SWAMM), as a part of the 15th World Wide Web Conference, Edinburgh, Scotland, 22-26 May, 2006.

OBIX. Obix framework – an open source Java library for software configuration http://obix-framework.sourceforge.net/

O'Reilly, T., (2005). What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software. September 2005. Online http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html

Peeters, G., (2003). A large set of audio features for sound description (similarity and classification) in the CUIDADO project, In: *CUIDADO project report*, 2003. URL: http://www.ircam.fr/anasyn/peeters/ARTICLES/Peeters_2003_cuidadoaudiofeatures.pdf

Patel-Schneider, P., Hayes, P. and Horrocks, I., (eds.), (2004). OWL Web Ontology Language Semantics and Abstract Syntax, W3C Recommendation, W3C; www.w3.org/TR/owl-absyn/ (26.02.2007).

Pfeiffer, S., Parker, C. and Pang, A., (2003). The Annodex annotation and indexing format for timecontinuous bitsreams, *Version 2.0 (work in progress).* http://www.ietf.org/internet-drafts/draft-pfeiffer-annodex-01.txt, December 2003.

Rao, A., Georgeff, M. (1991). Modeling rational agents within a BDI architecture. In: KR'91. Proc. 2nd International Conference on Principles of Knowledge Representation and Reasoning, pp. 473-484

Rao, A. (1996). AgentSpeak(L): BDI agents speak out in a logical computable language. In: Perram, J., Van de Velde, W. (eds.) MAAMAW 1996. LNCS, vol. 1038, pp. 42–55. Springer, Heidelberg

SELFMAN. SELFMAN – Self Management for Large-Scale Distributed Systems based on Structured Overlay Networks and Components, FP6 Project, 2006 – 2009. http://www.ist-selfman.org/wiki/index.php/SELFMAN_Project

Shoham, Y. (1993). Agent-oriented programming. Artificial Intelligence 60(1), 51–92

Srirama, S., and Naumenko, A., (2007). Secure Communication and Access Control for Mobile Web Service Provisioning, In CD-ROM Preprints of Proceedings of

International Conference on Security of Information and Networks (SIN2007), 8-10th May, 2007.

Terziyan, V., and Katasonov, A., (2007). Global Understanding Environment: Applying Semantic and Agent Technologies to Industrial Automation, In: M. Lytras et al (eds.), Emerging Topics and Technologies for in Information Systems, IGI Global, 2007, 35 pp. (Book chapter, submitted 14 May 2007).

Tetlow, P., Pan, J., Oberle, D., Wallace, E., Uschold, M., and Kendall, E., (eds). (2006). Ontology Driven Architectures and Potential Uses of the Semantic Web in Systems and Software Engineering. W3C editors draft.

Thevenin, D. and Coutaz, J., (1999). Plasticity of User Interfaces: Framework and Research Agenda. In Proceedings of Interact'99, vol. 1, Edinburgh: IFIP, IOS Press, 1999, pp. 110-117.

Vazquez-Salceda, J., Dignum, V., Dignum, F. (2005). Organizing multiagent systems. Autonomous Agents and Multi-Agent Systems 11(3), 307-360

Web 3.0. In: Wikipedia - a multilingual, web-based, free content encyclopedia. URL: http://en.wikipedia.org/wiki/Web_3.0

Yuxin, M., Zhaohui, W., Zhao, X., Huajun, C., Yumeng, Y., (2005). Interactive Semantic-Based Visualization Environment for Traditional Chinese Medicine Information. In: Web Technologies Research and Development - APWeb 2005, 7th Asia-Pacific Web Conference, Shanghai, China, March 29 - April 1, 2005, Springer, volume 3399/2005, ISBN 978-3-540-25207-8, pp. 950-959.

Appendix A: UBIWARE WP7 Status

Introduction

The objective of the 7-th workpackage is to trial UBIWARE on real industrial cases. This has two major goals for such case studies. The first goal is to evaluate the scientific concepts behind UBIWARE and to find problems and issues in UBIWARE that would otherwise be overlooked. The second goal is to facilitate the further utilization of UBIWARE in the industry. Several specific cases, proposed by the industrial partners, will be analyzed, designed and prototyped based on the UBIWARE platform. The reasons for prototyping are the same: to identify issues in UBIWARE that would get overlooked if the work was only theoretical and thus abstract, and to demonstrate the benefits of UBIWARE in a tangible way so to facilitate future industrial adoption.

There are three industrial cases, those of ABB, Fingrid and Metso Automation.

During the Year 1, with respect to all three cases the following tasks are to be performed:

Task T1.1_w7: Case analysis: identification of relevant industrial resources, their dependencies and interactions

Task T1.2_w7: Connecting to relevant industrial resources: Development of appropriate resource adapters

More detailed tasks for each industrial case have been defined during communication with Metso Automation and Fingrid (see below).

A.1 Metso Automation case

A.1.1 Background

The following systems are considered:

- **IP21** – Process history database – contains all process events.
- **ALP** – Alarms only database. Contains about 1% of IP21 messages.
- **Engineering DB –** object database that represents resources' structure and links between them. Database content is quite static, updated rarely.
- **MUST -** Multi-State Monitoring system.
- **DNA diary –** electronic diary with human comments related to events.

A.1.2 Opportunities

- **Timeline-based visualization of the alarms**
  Recognition and showing if similar alarm has already taken place. This will help expert in problem solving and decision making.

- **Clustering of the alarms in visualization**
  Time-based, place-based, action-based, etc. The alarms are shown in their own clusters, out of which it is easier for the user to select the important ones.

- **Automatic regulation (adjustment) of the alarm limits**
  A lot of time is spent on the alarm limits definition during the installation of the factory. The simplification of the procedure using semantic web would be a great benefit.

A.1.3 Working Plan

The working plan is defined as an initial set of features and steps needed to implement those features. Connection between Semantic Web storage and IP21 was implemented during previous project SmartResource 2004-2007. The goal of the first year of UBIWARE project is to connect two more resources and provide very simple analysis of alarms for demo purposes.

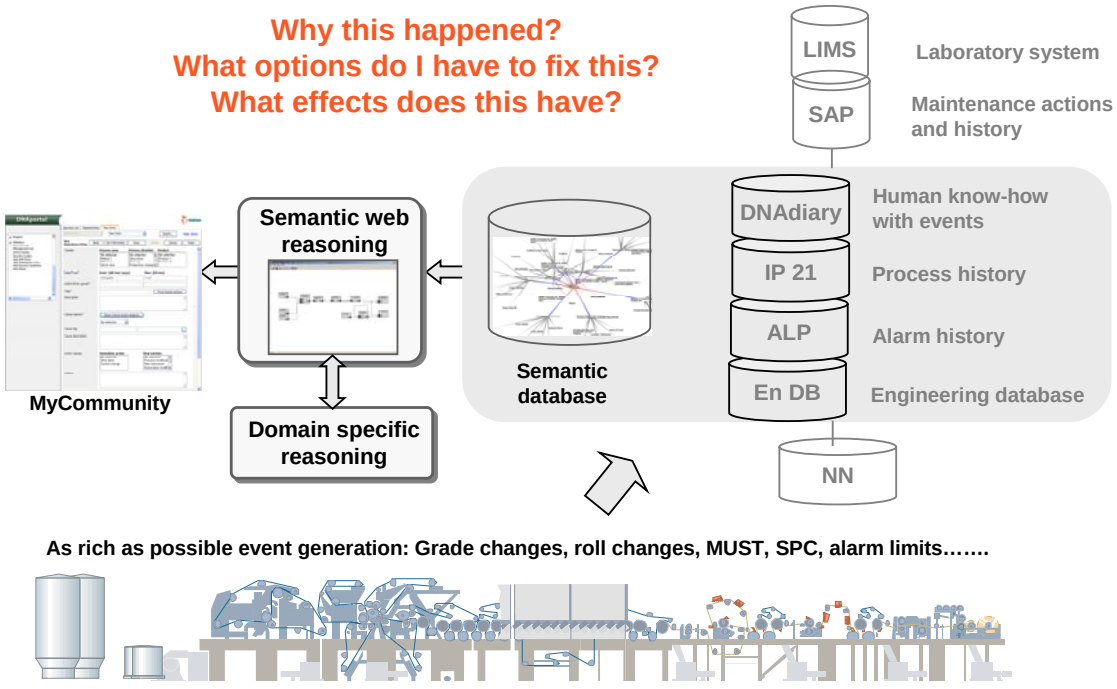

FIGURE A.17 Accumulation of cause-effect data

1. Connect Engineering database.
   - Get XML dump of DB, analyze design and extend existing ontology accordingly.
   - Implement a semantic adapter to database. Karaila promised to establish an interface to the database during the week 8-12.10.07. The interface will provide needed data in an XML format. We will extract the following information: Basic knowledge about factory, connections within the factory and alarm limits. The connection should be available in November.
   - Still to decide which factory floor will be a tested.

2. Connect Diary database.
   - Get data scheme from Metso and extend ontology accordingly.
   - Metso will ask permission from Heinola if their diary information can be used.
   - Metso already has a dump of the Heinola Fluting from year 2006.
   - The limitation is that Heinola has an old version of the diary meaning that the data is mainly textual descriptions, not categorized, so, reuse of the information is much lower.
   - Implement a semantic adapter to database.

3. Implementing needed Reusable Atomic Behaviors and Behavior Models needed for creating an agent able to:
   - Query the databases
   - Perform simple analysis of the alarms (for demo purpose).
4. Implementing another agent linked to a Web-Service so the analysis results could be accessed through the web (e.g. linked to the intranet).

A.2 Fingrid case

A.2.1 Background

The following systems are considered:

- **Event History Database** (Oracle) in the office environment, to which data is automatically replicated from SCADA's event history database (the replication will be implemented in December).
  - The focus is on **R1**-alarms, i.e. equipment alarms that require some maintenance actions to be performed.
  - The database also keeps events on when a maintenance worker entered or left a substation (billing is based on working time).
  - R3, R4, and R5 alarms, i.e. disturbances in the network, can also be considered.
  - Major R3, R4, and R5 alarms are also manually fed into Fingrid extranet (sähkömarkinnat => käyttöhäiriöt page)
- **Elnet system** (Oracle) that stores information about assets: towers, feeders, substations (i.e. the whole power network) + which maintenance service provider serves which working area.
  - **MapInfo** GIS is used for geographic representation of the power network. Data for MapInfo *is somehow synchronized*[22] with data in Elnet.
- **Tosu system** (MS Access) that is used by the maintenance service providers to report to Fingrid the costs for the work performed.
- **The lightning info-service** – FMI provides for Fingrid data on all the lightning events in Finland. The data *is probably stored* in a DB. Additionally, FMI provides a Google Maps –based application, which geographically shows lightnings data combined with the data on the locations of Fingrid power lines.
- **Microsoft SPS** is used for creating web-based intranet.
- **MS BizTalk** is used as the integration platform, e.g. lightning data comes through BizTalk.

A.2.2 Opportunities

In the present state of Fingrid practice, at least the following functions are performed manually (or not performed at all) while could potentially be automated:

- **Statistic analysis of the Event History data**.
  - Report on how many R1 alarms were happening per month /year per working area.

[22] Throughout the text, in italic are things that could be clarified

  - Analysis of efficiency of maintenance service providers. In case of an R1 alarm in their working area, the provider is notified automatically. One question is how much time it takes the provider to reach the substation to perform maintenance.
- **Integrating data from Event History, Elnet and Tosu.**
  - Analysis of the relationship between alarms (event history) and the types of equipment (Elnet).
  - Understanding what alarm has led to what maintenance actions at what cost. *Such matching is not possible for some reason (this is left unclear) at present.*
- **Integrating data from Event History and Lightening data**
  - Matching the locations and times of R3,4,5 alarms with the locations and times of lightning strikes to automatically filter out lightning-caused disturbances (normally require no action to be performed).

A.2.3 Special requirements

Data security is a central concern. It is major reason, along with **safety**, for putting the focus on historic analysis of events rather on the real-time operation. It is also the reason for that no real-time systems will be accessible for UBIWARE development for a while.

Any new functionality that UBIWARE can create should be linked to Fingrid web-based intranet through **SPS**.

A.2.4 Working Plan

The working plan is defined as an initial set of features (with priorities assigned) and steps needed to implement those features. At this stage, it is difficult to say if all features could be implemented in the UBIWARE timeframe. This also obviously depends on how deep we will go into the features.

The goal of the first year is to implement FE-1 (limited to only simple analysis in step 4).

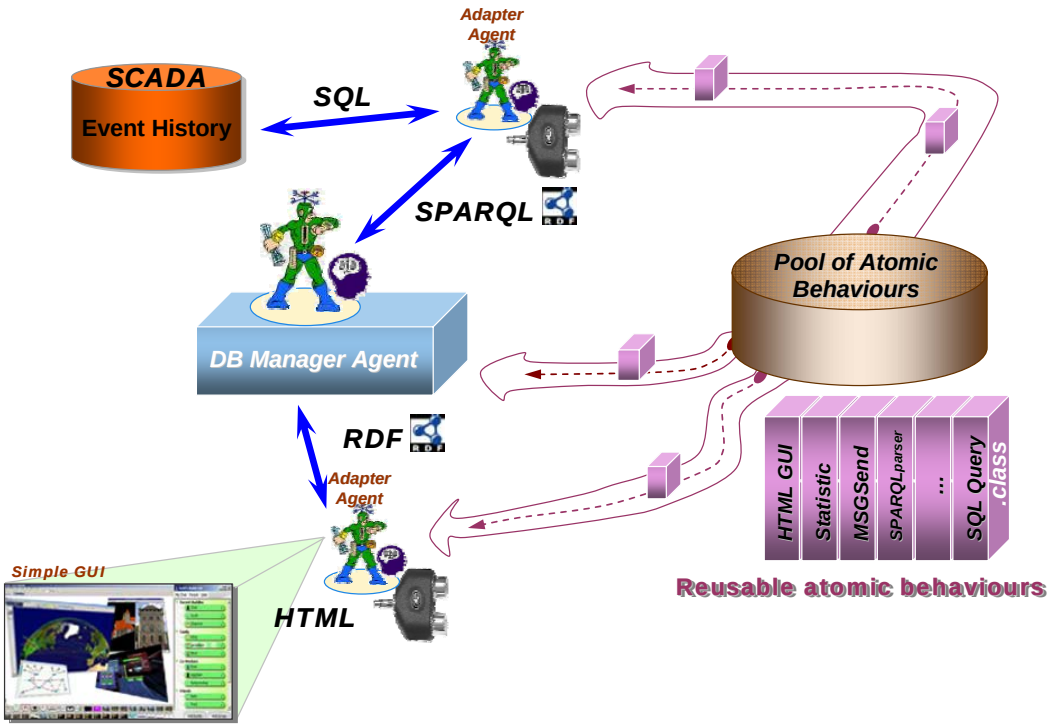

FIGURE A.2 Analysis of the Even Data

FE1 (priority high):

1. Get the dump of event data and load it into a test (Oracle) database.
2. Based on the data scheme design an ontological model to be used for classifying the events and their properties.
3. Implement a semantic adapter to the database.
4. Implementing needed Reusable Atomic Behaviors and Behavior Models needed for creating an agent able to:
   - Query the database
   - Perform statistical analysis of the data (see above)
5. Implementing another agent linked to a Web-Service so the analysis results could be accessed through the web (e.g. linked to the intranet).

FE-2: Integration with Elnet (priority: medium)

1. Extending the ontology from FE-1
2. Implementing an adapter to Elnet (*real, replica?*)
3. Implementing RABs and behavior models for the Elnet agent
4. Developing the integration scenarios (involving the Event DB agent and the Elnet agent and maybe others)

FE-3: Integration with Tosu (priority: medium)

Steps are similar to FE-2

FE-4: Integration with Lightnings data (priority: low)

Steps are similar to FE-2

A.3 ABB case

ABB has not yet confirmed its participation in UBIWARE and so far refused to have meeting to specify the case (situation as for 6 November 2007). We hope to have the confirmation from ABB and to schedule a specification meeting in near future.

Appendix B: UBIWARE Publications List

Terziyan V., Katasonov A., Global Understanding Environment: Applying Semantic and Agent Technologies to Industrial Automation, In: M. Lytras and P.O. Pablos (eds.), *Emerging Topics and Technologies in Information Systems,* IGI Global, 36 pp. (submitted 14 May 2007).

Terziyan V., SmartResource – Proactive Self-Maintained Resources in Semantic Web: Lessons learned, In: *International Journal of Smart Home*, Special Issue on Future Generation Smart Space, 2008, SERSC publisher, ISSN: 1975-4094, 18 pp. (submitted 5 September 2007).

Katasonov A., Kaykova O., Khriyenko O., Nikitin S., Terziyan, V., Smart Semantic Middleware for the Internet of Things, In: *Proceedings of the Internet of Things 2008 (IOT-2008), International Conference for Industry and Academia***,** 26-28 March, 2008, Zurich, Switzerland, Springer, LNCS, 17 pp. (submitted 20 September 2007).

Katasonov, A., Terziyan, V., SmartResource Platform and Semantic Agent Programming Language (S-APL), In: P. Petta et al. (Eds.), *Proceedings of the 5-th German Conference on Multi-Agent System Technologies (MATES'07)***,** 24-26 September, 2007, Leipzig, Germany, Springer, LNAI 4687 pp. 25-36.

Terziyan V., Predictive and Contextual Feature Separation for Bayesian Metanetworks, In: B. Apolloni et al. (Eds.), *Proceedings of KES-2007 / WIRN-2007*, Vietri sul Mare, Italy, September 12-14, Vol. III, Springer, LNAI 4694, 2007, pp. 634–644.

Nikitin S., Terziyan V., Pyotsia J., Data Integration Solution for Paper Industry - A Semantic Storing, Browsing and Annotation Mechanism for Online Fault Data, In: *Proceedings of the 4th International Conference on Informatics in Control, Automation and Robotics (ICINCO),* May 9-12, 2007, Angers, France, INSTICC Press, ISBN: 978-972-8865-87-0, pp. 191-194.

Salmenjoki K., Tsaruk Y., Terziyan V., Viitala M., Agent-Based Approach for Electricity Distribution Systems, In: *Proceedings of the 9-th International Conference on Enterprise Information Systems*, 12-16, June 2007, Funchal, Madeira, Portugal, ISBN: 978-972-8865-89-4, pp. 382-389.

Khriyenko O., 4I (FOR EYE) Technology: Intelligent Interface for Integrated Information, In: *Proceedings of the 9th International Conference on Enterprise Information Systems (ICEIS-2007)*, Funchal, Madeira - Portugal, 12-16 June 2007.

Khriyenko O., 4I (FOR EYE) Multimedia: Intelligent semantically enhanced and context-aware multimedia browsing, In: *Proceedings of the International Conference on Signal Processing and Multimedia Applications (SIGMAP-2007)*, Barcelona, Spain, 28-31 July 2007.

Khriyenko O., Context-sensitive Multidimensional Resource Visualization, In: *Proceedings of the 7th IASTED International Conference on Visualization, Imaging, and Image Processing (VIIP 2007)*, Palma de Mallorca, Spain, 29-31 August 2007.

Naumenko A., Semantics-Based Access Control in Business Networks, Jyvaskyla Studies in Computing, *PhD Thesis*, Volume 78, Jyvaskyla University Printing House, 215 pages, 2007.

Srirama, S., and Naumenko, A., (2007). Secure Communication and Access Control for Mobile Web Service Provisioning, In: *Proceedings of International Conference on Security of Information and Networks (SIN2007)*, 8-10th May, 2007.

Naumenko, A., SEMANTICS-BASED ACCESS CONTROL - Ontologies and Feasibility Study of Policy Enforcement Function , In: *Proceedings of the 3rd International Conference on Web Information Systems and Technologies (WEBIST-07)*, Barcelona, Spain - March 3-6, 2007, Volume Internet Technologies, INSTICC Press, pp. 150-155.

Naumenko A., Katasonov A., Terziyan V., A Security Framework for Smart Ubiquitous Industrial Resources, In: R. Gonzalves, J.P. Müller, K. Mertins and M. Zelm (Eds.), In: *Enterprise Interoperability II: New challenges and Approaches, Proceedings of the 3rd International Conference on Interoperability for Enterprise Software and Applications (IESA-07)*, March 28-30, 2007, Madeira Island, Portugal, Springer, pp. 183-194.

Katasonov A., Kaykova O., Khriyenko O., Loboda O., Naumenko A., Nikitin S., Terziyan V., The Central Principles and Tools of UBIWARE, *Technical Report (Deliverable D 1.1)*, UBIWARE Tekes Project, Agora Center, University of Jyvaskyla, May-October 2007, 118 pp.