S e m a n t i c   W e b   F r e d

SWF Goal and Service Description Language (version 1)

Net Dynamics & University of Innsbruck

Michael Stollberg
Dumitru Roman
Reinhold Herzog
Peter Zugmann

June 2004

SWF-Deliverable:
D3

Version:
1.2

Date:
25-Jun-04

Executive Summary

This document defines the description language for Cooperative Goals and for SWF Services in the Semantic Web Fred project.

The general approach for a close and long-lasting alignment of SWF with WSMO is achieved by taken the overlapping technologies directly from WSMO, whereupon specific features needed for SWF are added. Thus, SWF will always stick to WSMO during the development of WSMO, without need to change the similar technologies twice. Aprt from re-use and concurrent development, this allows complete interchangeability with other WSMO-enabled technologies as well as appliance of more advanced technologies that might be developed within WSMO efforts.

The Semantic Web Fred project is funded by the "Wiener Wirtschaftsföderungsfonds", short: WWFF. The project partners are:

- Net Dynamics Internet Technologies GmbH, project leader
- Next Web Generation Group at Institute for Computer Science at the University of Innsbruck (DERI Austria), scientific cooperation partner.

Table of Contents

Index of Figures

Index of Listings

1 Introduction

This deliverable specifies the Goal and Service Description Language for Semantic Web Fred. The description language is the basis for the SWF Mechanisms for determining and executing Automated Cooperation, which will be specified in SWF Deliverable D4 "SWF Tools and Mechanisms".

In principle, SWF applies the WSMO Description elements for Goals and Web Services. This ensures a strong alignment with WSMO on the one hand, and, on the other it allows using WSMO technologies for Semantic Web Services inside the SWF framework. Besides, interchageability with other WSMO other WSMO-enabled technologies is ensured straight away. More precisely, Goal Schemas as defined in the SWF Framework (Stollberg et al., 2004a) are modeled as WSMO Goals. For using such a Goal as a task assigned to a Fred, it is instantiated into a Goal Instance. This carries further descriptive information needed for the SWF goal resolution process. SWF Services (the 3 types of Services defined in the SWF Framework) are described as WSMO Web Services, as specified in WSMO. This allows not only re-use of the WSMO description elements, but also usage of WSMO technologies as the core of the SWF Mechanisms.

The structure of the document is as follows: Section 2 recalls the architecture and the components of SWF, pointing out the need for semantic descriptions in the system; Section 3 explains the alignment strategy of SWF with WSMO; Section 4 specifies the description elements for Goals and Services in SWF, as well as the formal language to be used; Section 5 exemplifies the usage of the SWF Goal and Service Description Language within a use case. Section 6 concludes the paper. The Appendix shows the technical architecture of SWF.

This version of the deliverable specifies all description elements of SWF Goals and Services apart form Service Choreography description, as this is not specified at this point in time. This will be added in version 2 of the deliverable.

2 Need for Semantic Descriptions in SWF

In order to rationalize the SWF Goal and Service Description Language, we briefly recall the components and architecture of SWF in order to point out where and why semantic descriptions are needed. We refer to the SWF Framework (Stollberg et al, 2004a) for exhaustive discussion of the SWF Architecture and Components.

The aim of the SWF project is to support automated cooperation between Freds (that are agents that act as electronic representatives on behalf of there owners, i.e. a Fred automatically fulfills tasks assigned to him by his owner). According to the real-world cooperation model underlying the SWF architecture, each Fred needs to have a has a Goal (what he wants to achieve) and respective Services (what he has to provide / support for automated goal resolution in a cooperation). In a cooperation, the Goals are solved by executing the respective services. In order to make the SWF technology dynamic and generic, intelligent mechanisms determine possible candidates for a cooperation. These work on semantic descriptions, therefore a powerful Description Language for specific SWF components is needed.

Figure 1 shows the SWF conceptual model, as defined and exhaustively explained in the SWF Framework (see technical architecture in Appendix). Below, we shortly summarize the components and explain for which ones semantic descriptions are needed.

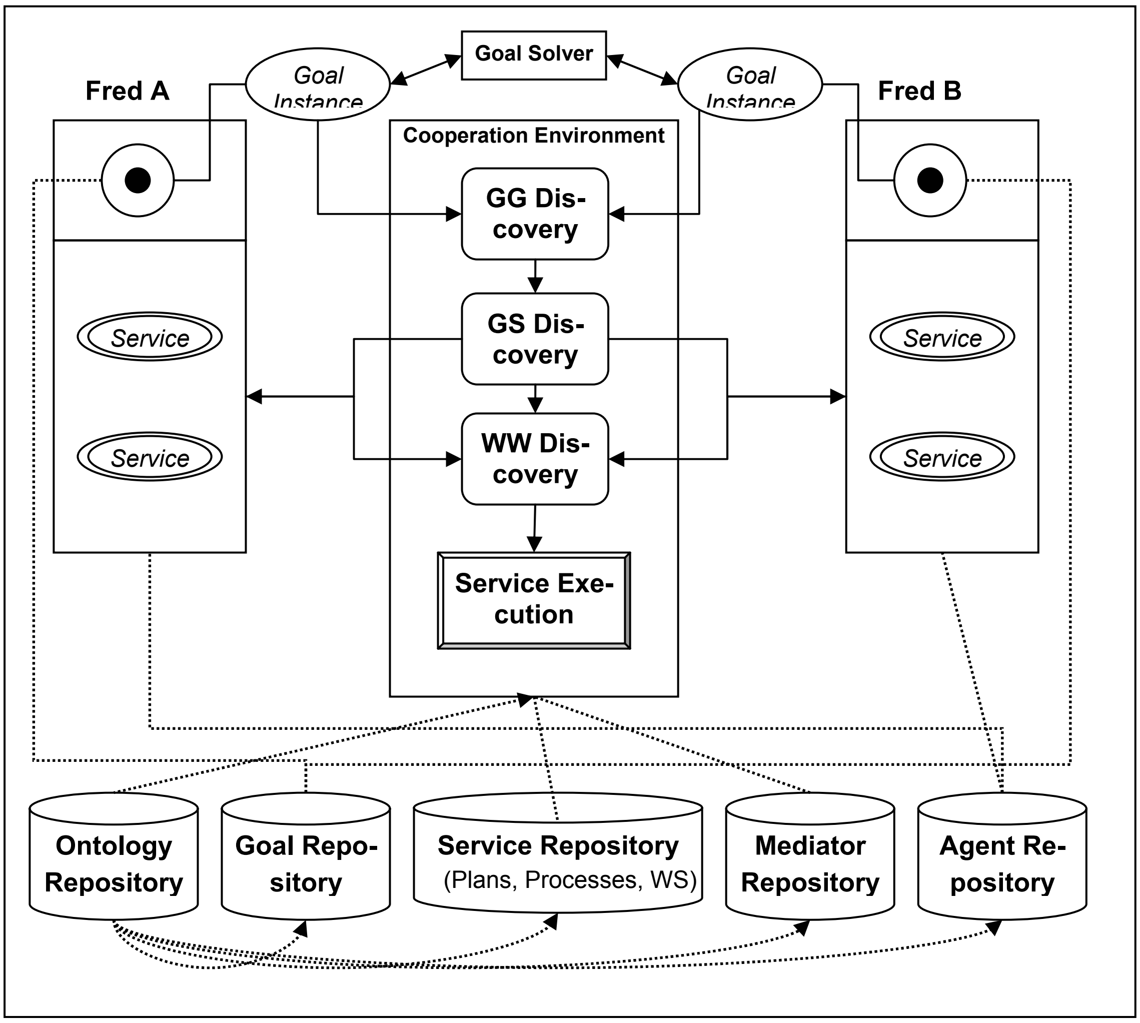


Figure 1: SWF Conceptual Model

The components of SWF are (for detailed description see section 3 of the SWF Framework):

- **Ontologies:** these represent the formally described knowledge used in all other components. In SWF, the FRED Smart Objects technology is used for ontology management (see Stollberg et al., 2004b). Also, external ontologies can be incorporated (e.g. WSMO ontologies), wherefore a transformation technology is provided in the FRED system. Section 3.1.3 explains the usage of WSMO ontologies and OO Mediators in more detail.
- **Goals:** a Goal describes a task that is assigned to a Fred for automated resolution. It describes the WHAT, while the HOW is to be dynamically detected via the SWF Discovery Mechanisms. SWF distinguishes between Goal Schemas and Goal Instances: the former is a generic description of a Goal, while the latter is a concrete Goal that is created by a party, indicating a concrete expression of a desire at a certain point in time that carries additional information. In contrast to WSMO Goals, a Goal in SWF carries more information (namely: the ontology object that will be submitted as input information to a service, the owner of a Goal Instance, i.e. a Fred, notion of the resolution status, and additional preferences on the resolution defined by the user).

  Changes to the SWF Framework:

  - Goals in SWF are called "Goals", not Cooperative Goals as specified in the SWF Framework
  - "Cooperative Goals" are Goals that only can be resolved in a cooperation. Therein, the compatible Goal Schemas are defined: Goal Instances of these can be created by potential cooperation partners.
- **SWF Services:** The overall notion of all problem solving implementations in SWF, i.e. the super-term for all kinds of Services in SWF (these can be Plans, Processes, or external Web Services – see the SWF Framework for further explanation). The type of a SWF Service is only of interest for the execution of the service (as therefore the accurate execution facilities have to be used), while during the discovery phase (= design time of a cooperation) only functional and behavioural information are of interest.
- **Mediators:** a Mediator is used to resolve mismatches by an appropriate mediation facility. It can be optionally included in the connection between connected

or interacting components. Mediators are supported by the SWF Architecture, but development of mediation facilities is out of the scope of the SWF project.

- **Agents / Freds:** a Fred is an agent, i.e. an electronic representative that acts on behalf of its owner. The agent technology for handling Freds is already existent in the FRED system (see Stollberg et al., 2004).
- **Repositories:** for all the components mentioned above, repositories are needed that hold the components or link to the actual locations. While agents are hold and managed by the facilities already existing in the FredBase, Ontologies, Goal Schemas, Service Descriptions, and Mediators are stored in the WSMO Registry, which is an extended UDDI Registry for WSMO components (Herzog et al. 2004). This is facilitated by the fact that these elements of the SWF Framework are identical to the respective WSMO components.

The aim of SWF is to develop a framework and implementation for automated cooperation with dynamic resolution of Cooperative Goals and SWF Services. Therefore, the following mechanisms are needed, as depicted in Figure 1, for detailed description see section 3 of the SWF Framework (Stollberg et al, 2004):

- **GG Discovery:** determines whether Goal Instances created by different Freds are compatible. If yes, the owners of the Cooperative Goals Instances start the Cooperation process.
- **GS Discovery:** detects the Services needed for the automated cooperation for each partner. This is similar to what commonly is referred to as Web Service Discovery. The result is that each partner has discovered all the services that he needs for the automated cooperation, possible a couple of usable services.
- **WW Discovery:** for the Services discovered for each partner by GS Discovery, we now have to check whether they are compatible with respect to their external visible behaviour and message exchange patterns (i.e. Choreography Matching). If there are such compatible services for each partner, a "Cooperation Contract" is defined that holds all information detected within the Discovery Mechanisms and information needed for execution.
- **Service Execution:** after the Discovery Process, the Cooperation Contract is executed, meaning the services of the partners are executed.
- **Goal Solver:** if the execution of the Services is completed successfully, then the Cooperative Goals of the respective cooperation partners are solved, so the Fred has accomplished his mission.

As the 3 Discovery Mechanisms are intended to be generic, following the approach of declarative descriptions and intelligent mechanisms working on this. Thus, there is need for semantic descriptions. As the Discovery Mechanisms only work on Goals and on SWF Services, we only need a Description Language for these two SWF components.

For the other SWF components, the following techniques are used (mainly already existent):

- **Ontologies:** for execution of services inside a FRED Meeting Room, the ontologies are transformed into Smart Objects. Everywhere else, ontologies are represented in a WSMO representation format (WSML, F-Logic, or any other compatible format). For using ontologies as the terminology definitions in other SWF components, they are used / imported via WSMO OO Mediators using the "usedMediators"-tag. This is specified in the modeling only, while for execution this information are transformed into the proper FRED Smart Objects technology. Thus, no additional description language is needed for ontologies in SWF.
- **Mediators:** in WSMO, these are only conceptual entities defined in models. For applying mediation facilities, these specifications have to be transformed into the Smart Objects technology as well.
- **Agents:** the FredBase technology for handling Freds is already existing, and is not further extended in the SWF project.

3 Strategy for SWF - WSMO Alignment

SWF is supposed to be a "WSMO implementation". More precisely, SWF applies the description elements and technologies developed within WSMO for the goal resolution technologies. Therein, the 'WSMO-parts' of SWF are isomorphic with the corresponding WSMO components and technologies. Consequently, SWF is divided into 2 parts: those corresponding to WSMO (called the 'WSMO-part'), and those which are specific for SWF.

This strategy ensures that SWF is completely WSMO-compliant. Besides, WSMO is not completed yet and under ongoing development – by this strategy we do not have to change all the SWF architecture and technologies when WSMO is updated (only some specific aspects might have to be aligned). Furthermore, SWF can apply the WSMO technologies to the most possible extent, and we ensure compatibility and interchangeablity with other WSMO-compliant systems straight away.

3.1 'WSMO-Part' of SWF - Usage of WSMO elements in SWF

Drawing the distinction between the 'WSMO-part' and the 'FRED-part' in SWF is very important and not a trivial task. In essence, the WSMO-Part of SWF is comprised of the following elements:

- SWF Goal Schemas are WSMO Goals (NOT SWF Goal Instances)
- SWF Services are described as WSMO Web Services
- WSMO Ontologies are used / supported
- WSMO Mediators are used (at least as conceptual specifications)
- A common repository for WSMO components and the WSMO-part of SWF

The following explains this in detail.

3.1.1 SWF Goal Schemas as WSMO Goals

A Goal in SWF is a task delegated to a Fred for automated resolution. As explained above, SWF distinguishes between 3 notions of a Goal:

1. **<u>Goal Schema:</u>** this is the generic description of a Goal, which only specifies the desire to be achieved. These are pre-defined in the system, and users can choose a Goal Schema of which they want to assign a task to a Fred.
2. **<u>Goal Instance:</u>** of a Goal Schema, Goal Instances are created as concrete expression of a desire to be solved. Therein, the desire specified in the corresponding Goal Schema can optionally be refined. A Goal Instance carries additional information needed for management and the goal resolution in SWF.
3. **<u>Cooperative Goal:</u>** are Goals that only can be resolved in a cooperation. Therein, the compatible Goal Schemas are defined: Goal Instances of these can be created by potential cooperation partners.

As SWF Goal Schemas are exactly the same as WSMO Goals, they are modeled as WSMO Goals (meaning they have exactly the same structure and meaning of distinct elements).

3.1.2 SWF Services as WSMO Web Services

The general idea of the SWF Service Model is that there is a general Description Language for Services, independent of the Type of Service (the 3 types of SWF Services are: Plans, Processes, external Web Services). In fact, the type of Service in SWF is only of interest for the actual execution of a Service, while for the discovery phase in the SWF goal resolution process (= design time of cooperative automated goal resolution) only descriptive information on the functionality of the service and its behavior with regard to service usage are needed.

As this is exactly the same intention as the one of the Web Service description language in WSMO, a SWF service description is a WSMO Web Service description. The resolution of the service type, needed for invoking the appropriate service execution facility is provided in the grounding information of the service description (most likely in the WSDL-part).

This allows usage of different services: Plans and Processes as FRED-specific service types, and WSMO-enabled as well as WSDL-described Web Services. WSMO-enabled Web Services are executed remotely at the service provider, while for all other service types the execution facilities are located inside the FRED system (JVMs for Plans, the Process Engine for Processes, and the WSDLExecutorPlan for WSDL-described Web Services).

3.1.3 Support for WSMO Ontologies in SWF

In WSMO, ontologies are represented in WSML, as defined in [Oren, 2004]. This contains management information (basically the non-functional properties), and the ontology specifications wherefore the representation format is based on F-Logic [Kifer et al., 1995] (in fact, the representation of all logical expressions in WSML has a isomorphic mapping into F-Logic). In the FRED system, ontologies are represented by Smart Objects. A transformation from Smart Objects to WSML / F-Logic, and vice versa, allows the interchange of WSMO ontologies with SWF. The expressiveness of Smart Objects in relation of WSMO ontologies, as well as the quality of the transformation technology will be described in a technical report of the SWF project (to appear).[1]

Apart from the representation quality, the distinction between Smart Objects and WSMO ontologies in terms of usage is important. Here, the WSMO representation format is used in modeling as well as within the Discovery Mechanisms. The Smart Objects representation is used for service execution inside the FRED-system, i.e. for Plans, Processes, and WSDL-described Web Services. The reason is that the execution facilities for these service types, as existing in the FRED system, rely on Smart Objects. As

[1] The Smart Objects technology relies on OXML, the internal ontology representation of OntoEdit [Erdmann et al., 2004]. OXML is based on F-Logic – this facilitates the transformation without loss of information.

WSMO-enabled Web Services are executed outside the system, the WSMO ontology representation is sufficient here.

Besides, as the Smart Object ontology provides stable and performant means for ontology management, the Smart Object representation and technology is used to store and handle ontology instance data within SWF.

3.1.4 Usage of WSMO Mediators in SWF

The concept of Mediators is regarded as the most important feature of WSMO, in comparison to other frameworks for Semantic Web Services. The aim is to catch the problem of heterogeneity as arising in open and distributed environments like the Internet by a clear and coherent concept. Of course, heterogeneity plays a major role within SWF, as the targeted application field is the (Semantic) Web. Thus, SWF applies the concept of Mediators in WSMO completely.

The conceptual model as well as the technologies for WSMO mediators are still under development at the time of writing. Basically, a WSMO Mediator connects one or more source components with a target component, and it contains the mediation facility needed to resolve mismatches between the source and target component, or between the source components.[2] In general, mediators are only defined explicitly if some mismatches occur – mediators without a mediation facility are omitted (the source component is used directly in the target component).

The general strategy for usage of WSMO Mediators within SWF is that existing Mediators are applied, if they are beneficial within an SWF application (meaning: development of mediation facilities is not targeted within SWF). As WSMO Mediators only describe a conceptual entity, they are transformed into the SWF technology (in fact, this has to be done for all systems that want to apply WSMO Mediators). Here, the source and target components are connected with their registry entries, and the mediation facilities are invoked as additional resources in meetings, and applied in service execution.

The basic structure of WSMO Mediators is defined, and thus it is possible to define their usage with SWF. The major aspect of WSMO Mediator usage in SWF is within which component / mechanism specific WSMO Mediators are applied, and the timing

[2] note: this is not yet specific within WSMO standard, but a proposal concerning this matter is currently under discussion in the WSMO working group.

aspects of mediation facility definition (design time) and mediation facility execution (runtime). The following explains this for all 4 types of WSMO Mediators.

3.1.4.1 OO Mediators

OO Mediators have 2 functional purposes: (1) re-use / modularization and integration of ontologies, and (2) providing the "information space" as the terminology definitions for all other components. Several OO Mediators can be "concatenated" in order to receive the final information space. The general idea is that the OO Mediator takes care of all ontology management related issues (access & integration & evolution & … of the source components), and presents the information space as "virtual, integrated ontology" to the OO Mediator user. So the user does not have to care about namespaces, etc. What the OO Mediator user sees is only the virtual ontology / information space that he uses as the terminological basis for specifying his descriptions – without having to care about any ontology management issues.

WSMO OO Mediators define:

- 1 or more ontologies or other OO Mediators as source component
- 1 target component (= OO Mediator user) which receives the desired information space
- mappings that integrate the source components (i.e. resolve the mismatches).

OO Mediators are applied within all SWF components that use ontologies as their terminological basis (these are: Goal Schemas and SWF Service Descriptions). For execution of the mappings specified in a OO Mediator, the dedicated WSMO ontology mapping execution facility will be applied (which is not defined yet). This will be realized by invoking this execution facility as an additional service into a meeting.

3.1.4.2 GG Mediators

A GG Mediator is used to define Goals by refining existing Goals, thus allowing specification of Goal structures on an ontological level. Therefore, new Goals can be created out of existing Goals, as a reuse. For refinement of the information spaces of the source-Goals, a Reduction is specified. In general, there can be one source-Goal (refinement via simple restriction) or several (refinement is a restriction along with a con- or disjunction of the information spaces of the source-Goals).

WSMO GG Mediators define:

- 1 or more source Goals
- 1 target Goal (the newly created Goal)
- ontology mappings that integrate the possibly heterogenic information spaces of separate source Goals. This can also be a OO Mediator.
- a reduction that restricts the information space of the target Goal, defined as an axiom.

GG Mediators are applied in SWF for within Cooperative Goals, and in GG Discovery. In the former case, a GG Mediator is used to restrict the information spaces of the compatible Goal Schemas defined in a Cooperative Goal to an exact match. Within GG Discovery at runtime (which checks the matching of Goal Instances of the Goal Schemas defined in a Cooperative Goal), a GG Mediator is used to restrict the concrete objects of interest of the Goal Schemas to an exact match. For instance, if the Goal Schemas define "furniture" as the exactly matching object of interest, and the Goal Instance instantiated this to "chair" as a subclass of "furniture", than the GG Mediator between the Goal Instance restricts the information space to "chair". This latter aspect can be determined automatically during runtime, while the GG Mediator in a Cooperative Goal is specified at design time.

3.1.4.3 WG Mediators

WSMO WG Mediators are used to handle partial matches in Goal-Capability Matching, the heart of Web Service Discovery. Therein, a Reduction restricts the information space of objects to be passed between a Goal and a Service to an exact match – based on the same approach as within GG Mediators.

Also, the structure of a WG Mediator is similar to GG Mediators, only there are different source & target components:

- 1 source component and 1 target component, whereof one is a Goal and the other one is Web Service. Here, the source component is always the component with the narrower information space, so that the reduction can be determined easily. For only partial overlaps (meaning that only parts of the Goal information space overlap with only parts of the Web Service information space, the Web Service is the source component by default).

- ontology mappings that integrate the possibly heterogenic information spaces of separate source and target component. This can also be a OO Mediator.
- a reduction that restricts the information space of the Goal and the Web Service to an exact match, defined as an axiom.

In SWF, WG Mediators are applied for handling partial matches in GS Discovery – similar to Web Service Discovery in WSMO. GS Discovery is intended to be separated between design time and runtime: When a new Goal Schema is created, possibly usable Service are determined (design time); during runtime, only this pre-defined set of possibly usable Services is searched to really usable services. Here, the mappings and reductions required for design time GS Discovery (Goal Schema to SWF Service) are defined at design time, maybe requiring human intervention. The WG Mediators between Goal Instances only carry and additional reduction, as all the terminology issues as specified in the Goal Schemas. This can be determined automatically during GS Discovery.

3.1.4.4 WW Mediators

WSMO WW Mediators are indented to resolve mismatches between Web Service in order to establish interoperability of Web Service which are not interoperable a priori. This is concerned with the WSMO Web Service Interface description, which distinguishes the notions of Choreography and Orchestration. Although these aspects are not fully specified in WSMO at the time of writing, it is obvious that they have different functional purposes, and thus different needs for mediation arise. In general, *Choreography* defines how to communicate with a Web Service in order to consume its functionality, and *Orchestration* defines how the overall functionality of a Web Service is achieved by the composition of other Web Services. The respective needs for mediation are to resolve the possible mismatches therein.

As the conceptual model is not completely defined yet in WSMO, neither the description elements, we skip the specification of WW Mediator usage with SWF at this moment.

3.2 Common Repository for WSMO and SWF

The ‘WSMO-part’ of SWF allows to use a common repository for WSMO and SWF components, as these are identical with regard to their structure and content.

Therefore, the WSMO Registry is used as defined in (Herzog et al., 2004). This is an enhanced UDDI-Registry that implements a hybrid architecture in order to combine the benefits of central and decentralized approaches. More precisely, the non-functional properties of each WSMO component are mapped into the corresponding UDDI information types, while the functional description elements

4 SWF Goal and Service Description Language

This section specifies the description elements for Cooperative Goals and for SWF Services in SWF. In order to provide a complete overview of the SWF Description Language, this comprises all description elements; those ones of the 'WSMO-part' are repeated from WSMO Standard (Roman et al., 2004). For definition of the description elements, we apply the same syntax used in WSMO Standard: "=>>" denotes a set valued attribute, "=>" a single valued one.

We show the usage of the distinct description elements by a complete use case modeling in section 5 of this document.

4.1 Goal Description Elements

SWF distinguishes between Goal Schemas and Goal Instances: the former is the generic description of a Goal, which only specifies the desire to be achieved. These are predefined in the system, and users can choose a Goal Schema of which they want to assign a task to a Fred. A Goal Instance is created from a Goal Schema as concrete expression of a desire to be solved. Therein, the desire specified in the corresponding Goal Schema can optionally be refined. A Goal Instance carries additional information needed for management and the goal resolution in SWF. In addition, the notion of Cooperative Goals is introduced. These are Goals that only can be resolved in a cooperation. Therein, the compatible Goal Schemas are defined: Goal Instances of these can be created by potential cooperation partners.

4.1.1 Goal Schema Description Elements

A SWF Goal Schema is modeled as a WSMO Goal. Thus, it has exactly the same description elements as WSMO Goals defined in WSMO Standard [Roman et al., 2004].

```
swfGoalSchema:goal[
 nonFunctionalProperties => nonFunctionalProperties
 usedMediators =>> {ooMediator or ggMediator}
 postCondition => axiomDefintion
 effects =>> axiomDefinition
]
```

Listing 1: Description Elements SWF Cooperative Goal Schema

- **Non-Functional Properties:** WSMO Core Properties defined in WSMO Standard, section 2.1 in [Roman et al., 2004]. For SWF Goal Schemas, all WSMO Core Properties elements can be defined, whereby the following are mandatory (to be displayed in user-interface for assigning a Goal to a Fred, or for information management): *title, creator, description, subject, date, time, identifier, source, rights, version*.
- **Used Mediators:** in WSMO Goals 2 types of WSMO Mediators can be used. OO Mediators for importing ontologies as the terminology definition for the Goal description, and GG Mediators can be used when the Goal is specified by reuse of existing Goals. See also section 3.1.4 for further investigation on the usage of WSMO Mediators in SWF.
- **Post Condition:** the object of interest as the desire expressed in the Goal i.e. what the user wants to achieve. This is modeled as an axiom that describes an unambiguous ontology object with regard to specified domain ontologies that restricts the set of instances which can resolve the Goal. These restrictions might reflect only a subset of the attributes of a chair in the referred ontology, but they completely described the desire from the user's perspective.
- **Effects:** state of the world desired after the Goal is solved, i.e. side-effects that restrict possibly resolutions for the Goal. The modeling is equivalent to the modeling of Goal postconditions.

4.1.2 Goal Instance Description Elements

A Goal Instance is created out of a Goal Schema (i.e. a WSMO Goal), and is "thrown" into the system by a party at a certain point in time *t0*. Thereby, the owner of the Goal Instance (which is a Fred) indicates that this specific Goal shall be resolved now. When a new Goal Instance is created, the engines that invoke GG Discovery start (i.e. they can find this Goal Instance and initiate GG Discovery – see SWF Framework).

A SWF Goal Instance takes a Goal Schema (with optional refinements), and carries additional information needed for the SWF goal resolution process.

```
swfGoalInstance:[
 nonFunctionalProperties => nonFunctionalProperties
 instanceOf => swfGoalSchema
 owner => Fred
 submission => instanceDefinition
 postcondition => axiomDefinition
 effects =>> axiomDefinition
 status => {open OR solved OR not solved OR processing OR can-
celled}
].
```

Listing 2: Description Elements SWF Cooperative Goal Instance

- **Non-functional Properties.** in addition to those inherited from Goal Schema (here, *creation* specifies the point of creation), the following non-functional properties can be defined as user requirements on the SWF goal resolution process:
  - *timeConstraints*: restricts the time frame for resolution of the Goal Instance
  - *resourceConstraints*: defines preferred cooperation partners
  - *goalResolutionConstraints*: other user preferences for goal resolution
- **Instance Of:** link to the Goal Schema (WSMO Goal) that the Goal Instance instantiated. Thereby, all descriptions of the corresponding Goal Schema are inherited into the Cooperative Goal Instance.
- **Owner:** defines the party that creates the Goal Instance. This a Fred or any other party that is known as an actor in the system (this might move to the creator-element of non-functional properties).
- **Submission:** this contains the information to be submitted to service(s) during service usage / execution. The submission is hold in a temporary *Information Container*, wherefrom the needed information are taken during service execution as required by the service's choreography. The service output returned after service execution is also hold in this Container, as well as additional input information requested by the service that are not existing in the Submission a priori.

  An entity for interacting with a service specified in WSMO Standard at this point of time. Nevertheless, there needs to be such an active counterpart for service usage, that submits the input information to the service, receives the service result, and support the interaction with the service. The design decision of placing this facility into the Goal Instance in SWF allows a better support for task delegation to Freds as electronic representatives (otherwise, the Fred as the Goal Instance owner would have to hold this facilities). The realization as an temporarily Information Container has several benefits: at first, the user specifies the submission data once at design time (if there is nothing missing, then the service usage can be performed without any further need to user intervention); secondly, he data flow information required by the service choreography are designed independently from specific services; thirdly, the Container centrally handles all data flow between a service and its user, so that interaction is restricted to control flow issues only (e.g. selections, acknowledgements).

- **Postcondition:** the concrete desire that the Goal Instance owner wants to get resolved. The postcondition of a Goal Instance is the same as defined in the corresponding Goal Schema, and it can optionally be refined. The meaning and the modeling of Goal Instance postconditions are the same as for Goal Schemas (i.e. WSMO Goals).
- **Effects:** similar to the postcondition, the effects are inherited from the Goal Schema. These can be either refined or be omitted. The meaning and the modeling of effects is the same as for Goal Schemas (i.e. WSMO Goals).
- **Status:** information on the "resolution status" of the Goal Instance. Values can be:
    - *open*: the Goal Instance has been created, but the resolution process has not started yet (i.e. search engines search for compatible Goal Instances)
    - *processing*: the goal resolution process is going on
    - *cancelled*: the goal resolution process has been cancelled because of an error during services execution. The meeting for execution is rescheduled.
    - *solved*: Goal Instance Resolution has been successfully (i.e. the Goal is solved)
    - *not solved*: Goal Instance Resolution has not been successfully, because of an unexpected / not handled error during service execution.

4.1.3 Cooperative Goals

A Cooperative Goal describes a Goal that has to be solved in cooperation of multiple partners. It defines compatible Goal Schemas, whereof individual entities (Freds) can carry Goal Instances of as partners in cooperative goal resolution. For instance, a Cooperative Goal 'purchase' has the compatible Goals 'buy chair' held by Fred *A* and 'sell furniture' held by another Fred *B*. A Cooperative Goal defines compatible cooperation roles on an ontological level, and it is used as a accommodating construct to determine potential cooperation partners within GG Discovery. Listing 3 show the structure of a Cooperative Goal in SWF:

```
swfCoopGoal:[
 nonFunctionalProperties => nonFunctionalProperties
 compatibleGoals =>> swfGoalSchema
 constraints => ggMediator
].
```

Listing 3: Description Elements Cooperative Goal

- **Non-Functional Properties:** WSMO Core Properties defined in WSMO Standard, section 2.1 in [Roman et al., 2004]. For SWF Cooperative Goals, all WSMO Core Properties elements can be defined, whereby the following are mandatory: *title, description, subject, date, time, identifier, source, rights, version*.
- **Compatible Goals:** links to SWF Goal Schemas (i.e. WSMO Goals) whereof individual entities (Freds) can carry Goal Instances of as partners in cooperative goal resolution.
- **Constraints:** a GG Mediator can optionally be applied to resolve mismatches between the Goal Schemas declared to be compatible. There can be terminological mismatches (resolved via ontology mappings / a OO Mediator), and the objects of interest of compatible Goal Schemas have to exactly match.

4.2 SWF Services Description Elements

As explained above, the general idea of the SWF Service Model is that there is a general Description Language for Services, independent of the Type of Service, as this is only of interest for the actual execution of a Service).

According to the WSMO-alignment strategy, SWF Services are described as WSMO Web Services, thus having the following description elements as defined in WSMO Standard, section 6 [Roman et al., 2004]:

```
swfservice:webservice[
 nonFunctionalProperties => nonFunctionalProperties
 usedMediators =>> ooMediator
 capability => capability
 interface =>> interface
].
```

Listing 4: Description Elements SWF Service

4.2.1 Non-functional Properties

Non-Functional Properties for SWF Services consist of the same properties as WSMO Web Service Descriptions, with the same meaning (see WSMO Standard).

4.2.1.1 Core Properties

For SWF Services, all WSMO Core Properties elements can be defined, whereby the following are mandatory for management: *title*, *creator*, *subject*, *description*, *publisher* (= Service Owner in SWF), *date*, *time*, *format*, *identifier*, *rights* (here: usage permission), *version*.

4.2.1.2 Service Specific Properties

The specific Non-functional Properties of Services are intended to be used for filtering / selection on the Service Discovery Result set of. Although these properties are not important for computational execution of a Service, they are the basis for advanced Service Discovery, improving the quality of the discover result. These properties will be taken over from WSMO, when there is a usable definition for measurement and description of them. These properties are not further considered in this version of the SWF Service Description Language.

4.2.2 Used Mediators

Importing / usage of domain ontologies as the terminology definition to be used in a service description. Therefore, a WSMO OO Mediator is used. NOTE: here, all terminologies needed for the service description are defined, so that no additional terminology import is needed in the sub-elements.

4.2.3 Capability

The Capability of a Service describes the functionality of a Service (1:1, i.e. each Service has one Capability). It describes what the service does, for service advertisement. WSMO defines following description elements as functional aspects of a Service (NOTE: non-functional properties and used mediators are omitted, as these are already defined in the overall service description).

```
capability:[
 precondition => axiomDefinition
 assumptions =>> axiomDefinition
 postcondition => axiomDefinition
 effects =>> axiomDefinition
].
```

Listing 5: Description Elements SWF Service Capability

- **Pre-Condition:** specifies the input requested by the service for enabling it to provide its service, and conditions on this. Similar to the notions of Goals, the precondition is defined as an axiom that defines an unambiguous ontology object in accordance to the information space defined by the OO Mediator of the overall service description. The conditions are specified by specific values for some attributes of some input. E.g., the input of a furniture-selling service is "furniture" and "customer" (both ontology concepts), and the condition is that the type of furniture is either "chair" or "bed" (as subclasses of furniture), i.e. the types of furniture sold by the service owner. Note that there is exactly one precondition of a web service.
- **Assumptions:** describe arbitrary condition on the state of the world that have to hold before execution of the service. Assumptions are not related to the input information needed for computation. However, they are related to the functionality of a service. Assumptions are modeled the same way as preconditions (and as all other notions of the Capability description).
- **Post-Conditions:** describes the result of service execution, i.e. the output of the service along with conditions on this. The postcondition is modeled as a relation to the input, i.e. the ontology objects defined in the precondition. For example, the result of a service is a contract of purchase for a piece of furniture, then the postcondition states that there will be a contract of purchase for the furniture-item that is provided as input (i.e. that fulfills the precondition and assumptions).

  Effects: describe changes in the state of the world to result for the execution of a service. Effects are not related to the output of a service; however, they are related to the functionality of a service in the same way as assumptions are.

4.2.4 Interface

The Web Service Interface in WSMO is the behavioral description of a Service. As exhaustively explained in the SWF Framework [Stollberg et al., 2004], the SWF Service

Interface description does only consist of the Choreography description; WSMO Orchestration, i.e. usage of other Services by an Service, is covered by FRED Processes (at least for the moment – it is planned to develop specific Service composition technologies for SWF / FRED in a follow-up project of Semantic Web Fred).

The Service Choreography describes the external visible behavior of Service in terms of a process representation, as well as the Message Exchange Patterns used for data and control flow within this process (see Choreography in WSMO, [Roman et al., 2004a], under construction at the time of writing). This shall support establishing global interaction models for collaboration of several Web Services. In SWF, this is the aim of WW Discovery (see SWF Framework).

The underlying model of a SWF Service Choreography shown in Figure 2 is that there is an invocation message at the beginning of service-user interaction (the user is the Goal Instance Owner, i.e. a Fred, in SWF), and a result message at the end. In between, there is an arbitrary exchange of messages according to the external visible behaviour of the Service. The information exchange between the Goal Instance Owner and the Service is realized by the Information Container as described above.

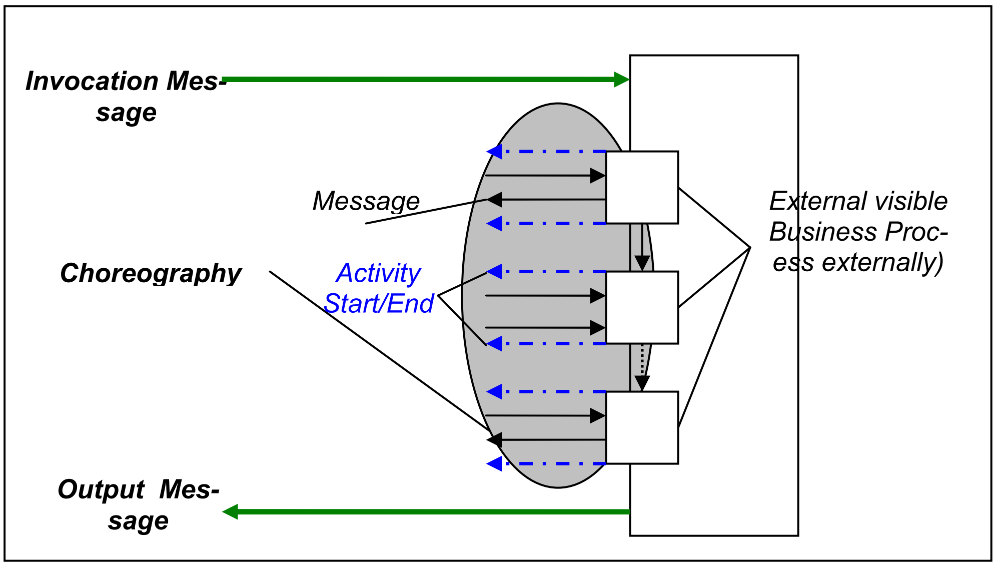


Figure 2: SWF Service Interface Structure

As the conceptual model and modeling of Service Choreographies is not specified at the time of writing, this is postponed to the follow-up version of this Deliverable. The general requirements for Service Choreography description in SWF are the following:

- a concrete technology must be supported, most likely WSDL

- a Choreography has to encapsulate error handling and compensation description
- a Choreography description must contain the physical access information of the service and its binding (communication protocol supported)
- the Choreography description has to support resolution of the service type, in order to invoke the specific execution facility for the different service types supported in the SWF Service Model.

4.3 Formal Language

As part of the 'WSMO-part' of SWF, the formal language used for modeling SWF components is WSML, including all features of this language developed in with the WSML working group.[3]

Apart form providing appropriate expressiveness and a sound logical foundation, this facilitates the usage of WSMO-specific technologies for the 'WSMO-part' of SWF on the one hand. On the other, the interchangeability support provided within WSML for other Semantic Web technologies and standards is by default inherited to SWF.

5 Use Case

We refer to the buyer-seller use case specified in the SWF Framework, section 4, for showcasing the SWF Goal and Service Description Language.

As specified above, WSML is used for the models of the SWF components below. All models listed here are also available in the CVS for the SWF project, accessible under "Links" on the project website at http://deri.at/research/projects/swf/.

Please note that the use case specification below is only an initial version which will be enhanced, corrected, updated, and completed in future versions.

5.1 Ontologies

Ontologies are modeled as WSMO Ontologies. W e need 4 domain ontologies:

(1) furniture - modeled below

(2) purchase - taken from WSMO D3.2

(3) person / address - taken from WSMO D3.2

3 The WSML working group is part of the WSMO working group, aiming at development of a language called Web Service Modeling Language (WSML) that formalizes the Web Service Modeling Ontology, see WSML homepage at: http://www.wsmo.org/wsml/

(4) date and time - taken from WSMO D3.2

5.1.1 Furniture Ontology ´

This ontology defines the domain of furniture. This version is an initial version, and it will be improved and aligned with existing domain ontologies.

```
comment: Furniture Ontology (simplified, to be enhanced)
ontology http://www.deri.at/research/projects/swf/ontologies/furniture.wsml

namespaces
  default = http://www.deri.at/research/projects/swf/ontologies/furniture.wsml#
  dc = http://purl.org/dc/elements/1.1#
  wsml = http://www.wsmo.org/2004/d16/d16.1/v0.2/20040418#
  loc = http://www.wsmo.org/2004/d3/d3.2/v0.1/20040607/resources/loc.wsml#

non-functional-properties
  dc:title "Furniture Ontology (simplified)"
  dc:creator "SWF Project"
  dc:subject "Furninture", "Chair", "Table", "Bed"
  dc:description "describes different types of furniture"
  dc:publisher "SWF Project"
  dc:contributor "Michael Stollberg"
  dc:date "2004-06-07"
  dc:type http://www.wsmo.org/2004/d2/v0.3/20040329/#ontos
               comment: ontologies are modeled as WSMO Ontologies
  dc:format "text/plain"
  dc:language "en-US"
  dc:relation
               comment: add some existing ontology for furniture / furnishing (dtl. = Woh-
nungseinrichtung)
  dc:coverage "project specific / general"
  dc:rights http://www.deri.org/privacy.html
  dc:version "0.1"


usedMediators
  ontology http://www.wsmo.org/2004/d3/d3.2/v0.1/20040607/resources/loc.wsml


conceptDefinitions

concept furniture
  non-functional-properties
    dc:description "general class of furniture"
  width oftype integer
  height oftype integer
  depth oftype integer
  manufacturer oftype manufacturer
  material oftype set {material}

concept manufacturer
```

```
  non-functional-properties
    dc:description "furniture manufacturer"
  name oftype string
  address oftype loc#address

concept material
  non-functional-properties
    dc:description "material a piece of furniture is made of, includes material-type and quality"
  type oftype string
  quality oftype integer

concept chair memberof furniture
  non-functional-properties
    dc:description "chair as a subclass of furniture"
  color oftype string

concept armchair memberof chair
  non-functional-properties
    dc:description "armchair as a subclass of chair"


concept table memberof furniture
  non-functional-properties
    dc:description "table as a subclass of furniture"


concept bed memberof furniture
  non-functional-properties
    dc:description "table as a subclass of furniture"
  color oftype string


instanceDefinitions

comment: these are specific type of tables
instance kitchenTable subconceptof tableType
instance diningTable subconceptof tableType
instance bedstand subconceptof tableType

comment: these are specific types of beds
instance single subconceptof bed
instance double subconceptof bed
instance canopyBed subconceptof double
```

Listing 6: Use Case - Furniture Ontology

5.1.2 Other Domain Ontologies

The other three domain ontologies needed are taken from the WSMO Deliverable D3.2 – WSMO Use Case and Testing (Stollberg et al., 2004b). We specify the links here:

```
comment: Purchase Ontology(taken from WSMO D3.2, not copied here)
ontology http://www.wsmo.org/2004/d3/d3.2/v0.1/20040531/resources/po.wsml

comment: Location, Person and address Ontology (taken from WSMO D3.2, not copied here)
ontology http://www.wsmo.org/2004/d3/d3.2/v0.1/20040531/resources/loc.wsml

comment: Date and Time Ontology (taken from WSMO D3.2, not copied here)
ontology http://www.wsmo.org/2004/d3/d3.2/v0.1/20040531/resources/dt.wsml
```

5.2 Goal Descriptions

As in the use case description, there shall be a Buyer who wants to buy a chair, and a seller (which actually is a shop) that wants to sell specific pieces of furniture.

In accordance to the WSMO-SWF Alignment Strategy described above, SWF Goal Schemas are modeled as WSMO Goals, while SWF Goal Instances carry the additional descriptions specified above.

5.2.1 Buyer

The Buyer Goal Schema is a general goal for buying one piece of furniture. This is refined in the Buyer Goal Instances to a specific type of armchair.

5.2.1.1 Buyer Goal Schema (WSMO Goal)

```
comment: Goal Schema for buying furniture
swfGoalSchema memberof goal http://www.deri.at/research/projects/swf/ontologies/buyer-
goalSchema.wsml

namespaces
  default = http://www.deri.at/research/projects/swf/ontologies/buyer-goalSchema.wsml#
  dc = http://purl.org/dc/elements/1.1#
  wsml = http://www.wsmo.org/2004/d16/d16.1/v0.2/20040418#
  comment: using all 4 domain ontologies
  furn = http://www.deri.at/research/projects/swf/ontologies/furniture.wsml#
  per = http://www.wsmo.org/2004/d3/d3.2/v0.1/20040531/resources/loc.wsml#
  po = http://www.wsmo.org/2004/d3/d3.2/v0.1/20040531/resources/po.wsml#
  dt = http://www.wsmo.org/2004/d3/d3.2/v0.1/20040531/resources/dt.wsml#

non-functional-properties
  dc:title "buying one piece of furniture"
  dc:creator "SWF Project"
  dc:subject furn#furniture, po#buyer
  dc:description "Goal Schema for buying one piece of furniture"
  dc:publisher "SWF Project"
  dc:contributor "Michael Stollberg"
  dc:date "2004-06-21"
  dc:type http://www.wsmo.org/2004/d2/v0.3/20040329/#L3958
```

```
              comment: definition of Goals in WSMO Standard
  dc:format "text/plain"
  dc:language "en-US"
  dc:relation http://www.deri.at/research/projects/swf/ontologies/furniture.wsml,
    http://www.wsmo.org/2004/d3/d3.2/v0.1/20040531/resources/loc.wsml,
    http://www.wsmo.org/2004/d3/d3.2/v0.1/20040531/resources/po.wsml,
    http://www.wsmo.org/2004/d3/d3.2/v0.1/20040531/resources/dt.wsml
  dc:coverage
  dc:rights http://www.deri.org/privacy.html
  dc:version "0.1"


usedMediators
  comment: all 4 domain ontologies are used
                              as no mismatches have to be resolved, the ontologies are used directly
  ontology http://www.deri.at/research/projects/swf/ontologies/furniture.wsml
  ontology http://www.wsmo.org/2004/d3/d3.2/v0.1/20040531/resources/loc.wsml
  ontology http://www.wsmo.org/2004/d3/d3.2/v0.1/20040531/resources/po.wsml
  ontology http://www.wsmo.org/2004/d3/d3.2/v0.1/20040531/resources/dt.wsml


postcondition
 axiom buy-one-furniture-postcondition
  non-functional-properties
    dc:description "the desire is to buy one piece of furniture, information:
                              item (the desired piece of furniture)
                              buyer
                              delivery of the product"
  logical-expression
  "item memberof furn#furniture,
   buyer memberof po#buyer[
    purchaseIntention ofvalues {item}
    billTo ofvalue BillAddress
    shipTo ofvalue BillAddress
    hasPayment ofvalue Payment
  ]
  and
  BillAddress memberof per#address[
              country ofvalue austria] and
  Payment memberof po#paymentMethod
  and
  item-delivered memberof po#delivery[
    products ofvalues {item}
    receiver ofvalue buyer
  ]."

effect
 axiom buy-one-furniture-effect
   non-functional-properties
     dc:description "there shall be a trade for the bought furniture
                              (i.e. a contract for the purchase)"
   logical-expression
   "aTrade memberof po#trade[
            items ofvalues {item},
        buyer ofvalue buyer,
        payment ofvalue Payment
    ]."
```

Listing 7: Use Case - Buyer Goal Schema

5.2.1.2 Buyer Goal Instance

```
comment: Goal Instance for buying a chair as an instantiation of the Goal Schema 'buying one
piece of furniture'
goalinstance http://www.deri.at/research/projects/swf/ontologies/buyer-goalInstance.swf

namespaces
  default = http://www.deri.at/research/projects/swf/ontologies/buyer-goalInstance.swf#
  dc=http://purl.org/dc/elements/1.1#
  comment: this should hold the SWF Goal and Service Description language
  swf = http://www.deri.at/research/projects/swf/ontologies/swf.wsml#
  comment: namespace of all ontologies inherited from the Goal Schema

non-functional-properties
  dc:title "buying one red leather armchair"
  dc:creator "SWF Project" comment: could also be the owner (Fred-ID)
  dc:subject furn#chair, po#buyer
  dc:description "Goal Instance for buying one red leather armchair"
  dc:publisher "SWF Project"
  dc:contributor "Michael Stollberg"
  dc:date "2004-06-21"
  dc:time "09:55"
  dc:source comment: here must be UDDI-address
  dc:type http://www.deri.at/research/projects/swf/ontologies/swf.wsml#goalInstance
              comment: Goal Schema modeled as defined in Goal Instance Description
  dc:format "text/plain"
  dc:language "en-US"
  dc:relation
  dc:coverage
  dc:rights http://www.deri.org/privacy.html
  dc:version "0.1"
  timeConstraints currentDate:dt#dateandtime + 5:dt#dayOfMonth


instanceOf http://www.deri.at/research/projects/swf/ontologies/buyer-goalSchema.wsml
  comment: this is the identifier of the related Goal Schema

owner
comment: must be id of the Fred, if not moved to non-functional properties

submission buy-one-furniture-submission
  non-functional-properties
    dc:description "the data to be handed over as input to a service, this is:
                item (piece of furniture), and the of the buyer"
  instanceDefinition
  "submissionItem memberof furn#armchair[
     width ofvalue 60
     heigth ofvalue 110
     depth ofvalue 100
     material ofvalues Material[
       type ofvalue "leather"
       quality ofvalue "high"
     ]
     color ofvalue "red"
   ]
  and
  buyer memberof po#buyer[
    shipTo ofvalue myAddress,
    billTo ofvalue myAddress
    payment ofvalue Payment
  ]
and
myAddress memberof address[
   name ofvalue "Michael Stollberg"
   street ofvalue "Technikerstrasse"
   number ofvalue 13
   zipcode ofvalue 6020
   city ofvalue innsbruck
   state ofvalue tirol
   country ofvalue austria
]
and
Payment ofvalue myCreditCard memberof po#creditCard[
   name ofvalue "Dieter Fensel"
   number ofvalue 1234567890
   expMonth ofvalue 9
   expYear ofvalue 2006
   type ofvalue "MasterCard"
]."


postcondition
  axiom buy-a-chair-postcondition
  non-functional-properties
    dc:description "the desire is to buy a red armchair made of leather, information:
          - item (red armchair)
          - buyer
          - delivery of the product"
  logical-expression
  "wantedItem memberof furn#chair[
    width ofvalue 60
    heigth ofvalue 110
    depth ofvalue 100
    material ofvalues Material[
       type ofvalue "leather"
       quality ofvalue "high"
    ]
  ]
  and
 buyer memberof po#buyer[
   purchaseIntention ofvalues {item}
   billTo ofvalue myAddress
   shipTo ofvalue myAddress
   hasPayment ofvalue Payment
  ]
 and
  myAddress memberof address[
   zipcode ofvalue 6020
   city ofvalue innsbruck
   state ofvalue tirol
   country ofvalue austria
   ]
 and
  Payment ofvalue myCreditCard memberof po#creditCard[
   name ofvalue "Dieter Fensel"
   expMonth ofvalue 9
   expYear ofvalue 2006
   type ofvalue "MasterCard"
  ]
 and
  item-delivered memberof po#delivery[
    products ofvalues {item}
    receiver ofvalue buyer
  ]".


effect
 axiom buy-a-chair-effect
   non-functional-properties
     dc:description "there shall be a trade for the chair"
   logical-expression
   "aTrade memberof po#trade[
       items ofvalues {wantedItem},
       buyer ofvalue buyer,
       payment ofvalue Payment
    ]."

status  open memberof swf#status
```

Listing 8: Use Case - Buyer Goal Instance

5.2.2 Seller

The Goal Schema for the Seller is a general Goal for selling furniture with delivery restricted to Austria. This is refined in the Goal Instance to the product catalogue of the Austrian furnishing shop “Leiner”, who is the seller in this example.

5.2.2.1 Seller Goal Schema (WSMO Goal)

```
comment: Goal Schema for selling furniture (a WSMO Goal)
```

```
swfGoalSchema memberof goal http://www.deri.at/research/projects/swf/ontologies/seller-
goalSchema.wsml

namespaces
  default = http://www.deri.at/research/projects/swf/ontologies/seller-goalSchema.wsml#
  dc = http://purl.org/dc/elements/1.1#
  wsml = http://www.wsmo.org/2004/d16/d16.1/v0.2/20040418#
  comment: using all 4 domain ontologies
  furn = http://www.deri.at/research/projects/swf/ontologies/furniture.wsml#
  per = http://www.wsmo.org/2004/d3/d3.2/v0.1/20040531/resources/loc.wsml#
  po = http://www.wsmo.org/2004/d3/d3.2/v0.1/20040531/resources/po.wsml#
  dt = http://www.wsmo.org/2004/d3/d3.2/v0.1/20040531/resources/dt.wsml#

non-functional-properties
  dc:title "selling one piece of furniture"
  dc:creator "SWF Project"
  dc:subject furn#furniture, po#buyer
  dc:description "Goal Schema for selling one piece of furniture"
  dc:publisher "SWF Project"
  dc:contributor "Michael Stollberg"
  dc:date "2004-06-21"
  dc:type http://www.wsmo.org/2004/d2/v0.3/20040329/#L3958
              comment: definition of Goals in WSMO Standard
  dc:format "text/plain"
  dc:language "en-US"
  dc:relation http://www.deri.at/research/projects/swf/ontologies/furniture.wsml,
    http://www.wsmo.org/2004/d3/d3.2/v0.1/20040531/resources/loc.wsml,
    http://www.wsmo.org/2004/d3/d3.2/v0.1/20040531/resources/po.wsml,
    http://www.wsmo.org/2004/d3/d3.2/v0.1/20040531/resources/dt.wsml
  dc:coverage
  dc:rights http://www.deri.org/privacy.html
  dc:version "0.1"


usedMediators
  comment: all 4 domain ontologies are used
         as no mismatches have to be resolved, the ontologies are used directly
  ontology http://www.deri.at/research/projects/swf/ontologies/furniture.wsml
  ontology http://www.wsmo.org/2004/d3/d3.2/v0.1/20040531/resources/loc.wsml
  ontology http://www.wsmo.org/2004/d3/d3.2/v0.1/20040531/resources/po.wsml
  ontology http://www.wsmo.org/2004/d3/d3.2/v0.1/20040531/resources/dt.wsml


postcondition
 axiom sell-one-furniture-postcondition
  non-functional-properties
    dc:description "the desire is to sell one piece of furniture, information:
                          - item (the desired piece of furniture)
                          - seller
                          - delivery of the product"
  logical-expression
  "salesItem memberof furn#furniture,
   seller memberof po#seller[
    address ofvalue SellerAddress
    salesIntention ofvalues {salesItem}
    acceptsPayment ofvalue Payment
  ] and
  SellerAddress memberof per#address[
    country ofvalue austria] and
  Payment memberof po#creditcard
 and
 item-delivered memberof po#delivery[
   products ofvalues {salesItem}
   sender ofvalue seller
  ]."

effect
 axiom buy-one-furniture-effect
   non-functional-properties
    dc:description "there shall be a trade for the sold furniture
                          (i.e. a contract for the purchase)"
   logical-expression
   "aTrade memberof po#trade[
             items ofvalues {salesItem},
       seller ofvalue seller,
       payment ofvalue Payment
    ]."
```

Listing 9: Use Case - Seller Goal Schema

5.2.2.2 Seller Goal Instance

```
comment: Goal Instance for selling a piece of furniture (different types of furniture)
                        as an instantiation of the Goal Schema 'selling one piece of furniture'
goalinstance http://www.deri.at/research/projects/swf/ontologies/seller-goalInstance.swf

namespaces
  default = http://www.deri.at/research/projects/swf/ontologies/seller-goalInstance.swf#
  dc=http://purl.org/dc/elements/1.1#
  comment: this should hold the SWF Goal and Service Description language
  swf = http://www.deri.at/research/projects/swf/ontologies/swf.wsml#
  comment: namespace of all ontologies inherited from the Goal Schema

non-functional-properties
  dc:title "selling one piece of specific furniture: a chair, a table, or a bed"
  dc:creator "SWF Project" comment: could also be the owner (Fred-ID)
  dc:subject furn#chair, po#buyer
  dc:description "Goal Instance for selling specific pieces of furniture "
  dc:publisher "SWF Project"
  dc:contributor "Michael Stollberg"
  dc:date "2004-06-21"
  >dc:time "09:55"
  dc:source comment: here must be UDDI-address
  dc:type http://www.deri.at/research/projects/swf/ontologies/swf.wsml#goalInstance
          comment: Goal Schema modeled as defined in Goal Instance Description
  dc:format "text/plain"
  dc:language "en-US"
  dc:relation
  dc:coverage
  dc:rights http://www.deri.org/privacy.html
  dc:version "0.1"


instanceOf http://www.deri.at/research/projects/swf/ontologies/buyer-goalSchema.wsml
  comment: this is the identifier of the related Goal Schema

owner
comment: must be id of the Fred, if not moved to non-functional properties

submission sell-one-furniture-submission
  non-functional-properties
    dc:description "the data to be handed over as input to a service, this is:
                    - specific piece of furniture (this would be the product catalogue of the
seller
                              - information on the seller
                              - accepted payment method"
  instanceDefinition
  "submissionItem ofvalue aFurniture and
   ((aFurniture memberof furn#chair[
       manufacturer ofvalue Manufacturer
    ],
    (Manufacturer memberof furn#manufacturer[name ofvalue "Gucci", address ofvalue Guc-
ciAddress] or
     Manufacturer memberof furn#manufacturer[name ofvalue "Kika", address ofvalue KikaAd-
dress] or
      Manufacturer memberof furn#manufacturer[name ofvalue "Leiner", address ofvalue Lein-
erAddress]
    )
    comment: this means that only request for chairs produced by certain manufacturers are
sold,
          addresses not further modelled here

   or
   (aFurniture memberof furn#kitchentable[
       material ofvalue Material] and
    Material memberof furn#material[
      name ofvalue "wood",
      quality ofvalue "high"
    ]
   )
            comment: means that only request for wooden kitchen tables of high quality are sold
           or
   (aFurniture memberof furn#double)
   ),
   seller memberof po#seller[
    address ofvalue SellerAddress
    salesIntention ofvalues {salesItem}
    acceptsPayment ofvalues {AcceptedPayment}
  ]
  and
  SellerAddress memberof per#address[
    name ofvalue "Leiner"
    street ofvalue "Mariahilferstrasse"
    number ofvalue 18
    zipcode ofvalue 1070
    city ofvalue vienna
    state ofvalue vienna
   country ofvalue austria
  ]
  and
  (AcceptedPayment memberof po#creditcard[
     type ofvalue CCType] and
   (CCType ofvalue "MaterCard" or CCType ofvalue "Visa"))."


postcondition
 axiom sell-one-furniture-postcondition
  non-functional-properties
    dc:description "the desire is to sell one piece of furniture, information:
                          - item (the desired piece of furniture)
                          - seller
                          - delivery of the product"
  logical-expression
  "salesItem memberof furn#afurniture
   and
   ((aFurniture memberof furn#chair[
      manufacturer ofvalue Manufacturer
    ],
    (Manufacturer memberof furn#manufacturer[name ofvalue "Gucci", address ofvalue Guc-
ciAddress] or
     Manufacturer memberof furn#manufacturer[name ofvalue "Kika", address ofvalue KikaAd-
dress] or
     Manufacturer memberof furn#manufacturer[name ofvalue "Leiner", address ofvalue Lein-
erAddress]
    )
     comment: this means that only request for chairs produced by certain manufacturers are
sold,
                                 addresses not further modelled here

   or
   (aFurniture memberof furn#kitchentable[
               material ofvalue Material] and
    Material memberof furn#material[name ofvalue "wood", quality ofvalue "high"]
    )
     comment: means that only request for wooden kitchen tables of high quality are sold
   or
   (aFurniture memberof furn#double)
   )
   and
   leiner memberof po#seller[
    address ofvalue SellerAddress
    salesIntention ofvalues {salesItem}
    acceptsPayment ofvalues {AcceptedPayment},
  ] and
  SellerAddress memberof per#address[
    name ofvalue "Leiner"
    city ofvalue vienna
    country ofvalue austria
  ]
  and
  (AcceptedPayment memberof po#creditcard[
    type ofvalue CCType] and
   (CCType ofvalue "MaterCard" or CCType ofvalue "Visa")),
 and
  item-delivered memberof po#delivery[
    products ofvalues {salesItem}
    receiver.shiptoAddress.country = austria
    sender ofvalue leiner
   comment: the seller will be leiner, and delivery is only supported within Austria
  ]."

effect
 axiom sell-one-furniture-effect
   non-functional-properties
    dc:description "there shall be a trade for the sold piece of furniture
                (i.e. a contract for the purchase)"
   logical-expression
    "aTrade memberof po#trade[
    items ofvalues {salesItem}
      seller ofvalue leiner
      buyer ofvalue receiver
      payment ofvalue AcceptsPayment
    ]."

status  open memberof swf#status
```

Listing 10: Use Case - Seller Goal Instance

5.2.3 Cooperative Goal "Purchase"

In order to showcase cooperative goal resolution, we specify the Goal "Purchase" as a Cooperative Goal in SWF. The compatible Goal Schemas identified here are the Goal Schema of the Buyer (see Listing 7) and the one of the Seller (see Listing 9).

```
comment: Cooperative Goal 'Purchase'
comment: identifies Goal Schemas 'buying one piece of furniture' and 'selling one piece of fur-
niture'
                              as compatible Goals
swfCoopGoal http://www.deri.at/research/projects/swf/ontologies/coopGoal-purchase.swf

 nonFunctionalProperties
  dc:title "a purchase for a piece of furniture"
  dc:creator "SWF Project"
  dc:subject furn#furniture, po#buyer, po#seller
  dc:description "the Goals of buying and selling a piece of furniture establish a purchase
              which is to be resolved in a cooperation"
  dc:publisher "SWF Project"
  dc:contributor "Michael Stollberg"
  dc:date "2004-06-07"
  dc:type http://www.deri.at/research/projects/swf/ontologies/swf.wsml#cooperativeGoal
              comment: Goal Schema modeled as defined in Cooperative Goal Description
  dc:format "text/plain"
  dc:language "en-US"
  dc:relation http://www.deri.at/research/projects/swf/ontologies/buyer-goalSchema.wsml,
              http://www.deri.at/research/projects/swf/ontologies/seller-goalSchema.wsml
  dc:coverage "general"
  dc:rights http://www.deri.org/privacy.html
  dc:version "0.1"

compatibleGoals ofvalue {
              http://www.deri.at/research/projects/swf/ontologies/buyer-goalSchema.swf
              http://www.deri.at/research/projects/swf/ontologies/seller-goalSchema.swf
  }
compatibleGoalConstraints ofvalues {}
   comment: not needed her, as the compatible Goal Schemas are heterogenic
```

Listing 11: Use Case - Cooperative Goal 'Purchase'

5.3 SWF Service Descriptions

For the automated execution, we need at least one service of each partner. According to the WSMO-SWF alignment strategy, SWF Services are modeled as WSMO Web Services.

The models below only model the overall service description as well as the Service Capability, as the modeling specification for Interfaces / Choreography is not fully specified in WSMO at the time of writing.

NOTE: In contrast to the proportion of Services and Goals in SWF is opposite as in WSMO. In SWF, the Buyer Service has to satisfy the Buyer Goal Instance (similar for the Seller Services and the Seller Goal Instance). In WSMO, the Web Service Discovery looks for a Seller Services as the resolution service for a Buyer Goal.

5.3.1 Buyer Service

The Buyer Service is a general Service that allows purchasing of pieces of furniture automatically. It is to be implemented as a FRED Plan (could, of course, be any other type of service type supported in SWF).

comment: Service for Buyer

```
swfservice memberof webservice http://www.deri.at/research/projects/swf/ontologies/buyer-
service.wsml

comment: this service is to be implemented as a FRED Plan

  namespaces
    default= http://www.deri.at/research/projects/swf/ontologies/buyer-service.wsml#
    dc=http://purl.org/dc/elements/1.1#
    comment: this should hold the SWF Goal and Service Descriptionlanguage
    wsml = http://www.wsmo.org/2004/d16/d16.1/v0.2/20040418#
    furn = http://www.deri.at/research/projects/swf/ontologies/furniture.wsml#
    per = http://www.wsmo.org/2004/d3/d3.2/v0.1/20040531/resources/loc.wsml#
    po = http://www.wsmo.org/2004/d3/d3.2/v0.1/20040531/resources/po.wsml#
    dt = http://www.wsmo.org/2004/d3/d3.2/v0.1/20040531/resources/dt.wsml#

  non-functional-properties
    dc:title "SWF Service for buying a piece of furniture"
    dc:creator "SWF project"
    dc:subject furn#furniture, po#buyer, po#paymentMethod, po#delivery
    dc:description "SWF service all facilities a buyer needs for buying a piece of furniture"
    dc:publisher "SWF project"
    dc:contributor "Michael Stollberg"
    dc:date "2004-06-21"
    dc:type  http://www.wsmo.org/2004/d2/v0.3/20040329/#L3958
               comment: this is the link to web service specification in WSMO Standard
               comment: actually, we could put the 'SWF Service Type' infomation here
    dc:format "text/plain"
    dc:language "en-us"
    dc:relation http://www.deri.at/research/projects/swf/ontologies/furniture.wsml,
      http://www.wsmo.org/2004/d3/d3.2/v0.1/20040531/resources/loc.wsml,
      http://www.wsmo.org/2004/d3/d3.2/v0.1/20040531/resources/po.wsml,
      http://www.wsmo.org/2004/d3/d3.2/v0.1/20040531/resources/dt.wsml
    dc:coverage "general"
    dc:rights http://deri.at/privacy.html
    dc:version "1.0"


  usedMediators
    comment: all 4 domain ontologies are used
                            as no mismatches have to be resolved, the ontologies are used directly
    ontology http://www.deri.at/research/projects/swf/ontologies/furniture.wsml
    ontology http://www.wsmo.org/2004/d3/d3.2/v0.1/20040531/resources/loc.wsml
    ontology http://www.wsmo.org/2004/d3/d3.2/v0.1/20040531/resources/po.wsml
    ontology http://www.wsmo.org/2004/d3/d3.2/v0.1/20040531/resources/dt.wsml


  capability
    non-functional-properties
      dc:description "defines the capability of the Buyer Service"


   precondition
    axiom buyerServicePrecondition
```

```
      non-functional-properties
        dc:description "the input to the Service has to be:
                                                  - an object of furn#furniture
                                                  - a buyer with a shipping address in Austria
                                                  - a valid (not expired) credit card"
      logical-expression
       "inputItem memberof furn#furniture
        and
        inputBuyer memberof po#buyer[
          purchaseIntention ofvalues {inputItem}
          payment ofvalue BuyerCreditCard memberof po#creditCard
          shipTo ofvalue aAddress memberof pers#address[
          country ofvalue austria
               ]
                           and
        ((currentDate.date.year < BuyerCreditCard.expYear) or
           ((currentDate.date.year = BuyerCreditCard.expYear) and
              ((currentDate.date.monthOfyear < BuyerCreditCard.expMonth) or
               (currentDate.date.monthOfyear = BuyerCreditCard.expMonth))
           )
         )."

  comment: the current date is an instance re-defined in the Date and Time Ontology
           because we do not have a built-in funtion yet

   assumption
    axiom buyerServiceAssumption
      non-functional-properties
        dc:description "the bank account of the buyer's credit card has to hold enough informa-
tion"
      logical-expression
            comment: not modeled here, as the ontological definitions are missing


   postcondition
    axiom buyerServicePostcondition
      non-functional-properties
        dc:description "the output of the service is:
                                    - the piece of furnitute specified in the input, is a piece of furni-
ture
                                    - the buyer will be the buyer provided as input
                                    - the payment method will be the one provided as input
                                    - the purchased item will be delivered to the shipping address of
the buyer provided as input"
      logical-expression
       "bougthItem ofvalue inputItem
        and
        purchaser ofvalue inputBuyer[
          shipTo ofvalue aAddress memberof loc#address[
          country ofvalue austria
          ]
          hasPayment ofvalue BuyerCreditCard
                      ]
```

```
                              and
                              itemDelivery memberof po#delivery[
                                products ofvalues {boughtItem},
                                receiver ofvalue purchaser
                              ]."

   effect
    axiom buyerServiceEffect
      non-functional-properties
        dc:description "there is a trade for the piece of furniture bought,
                              i.e. a contract of purchase"
      logical-expression
                    "aTrade memberof po#trade[
                              items ofvalues {boughtItem}
                              buyer ofvalue purchaser
                              payment ofvalue BuyerCreditCard
                     ]."


 interface
    non-functional-properties
      dc:description "defines the Choreography of the Buyer Service "
    comment: not specified yet
```

Listing 12: Use Case - Buyer Service

5.3.2 Seller Service

The Seller Service described here is a service of automatically selling pieces of furniture by a specific Austrian-based furnishing shop, called ‘Leiner’. It allows to sell all items available in the shop’s product catalogue (imaginary at the moment).

This service is to be implemented as a FRED Process.

```
comment: Service for Seller
swfservice memberof webservice http://www.deri.at/research/projects/swf/ontologies/seller-
service.wsml

comment: this service is to be implemented as a FRED Process

  namespaces
    default= http://www.deri.at/research/projects/swf/ontologies/seller-service.wsml#
    dc=http://purl.org/dc/elements/1.1#
    comment: this should hold the SWF Goal and Service Descriptionlanguage
    wsml = http://www.wsmo.org/2004/d16/d16.1/v0.2/20040418#
    furn = http://www.deri.at/research/projects/swf/ontologies/furniture.wsml#
    per = http://www.wsmo.org/2004/d3/d3.2/v0.1/20040531/resources/loc.wsml#
    po = http://www.wsmo.org/2004/d3/d3.2/v0.1/20040531/resources/po.wsml#
    dt = http://www.wsmo.org/2004/d3/d3.2/v0.1/20040531/resources/dt.wsml#

  non-functional-properties
    dc:title "SWF Service for selling one piece of furniture"
    dc:creator "SWF project"
    dc:subject furn#furniture, po#buyer, po#paymentMethod, po#delivery
    dc:description "SWF service all facilities a seller needs for selling one piece of furniture"
    dc:publisher "SWF project"
    dc:contributor "Michael Stollberg"
    dc:date "2004-06-21"
    dc:type  http://www.wsmo.org/2004/d2/v0.3/20040329/#L3958
             comment: this is the link to web service specification in WSMO Standard
             comment: actually, we could put the 'SWF Service Type' infomation here
    dc:format "text/plain"
    dc:language "en-us"
    dc:relation http://www.deri.at/research/projects/swf/ontologies/furniture.wsml,
      http://www.wsmo.org/2004/d3/d3.2/v0.1/20040531/resources/loc.wsml,
      http://www.wsmo.org/2004/d3/d3.2/v0.1/20040531/resources/po.wsml,
      http://www.wsmo.org/2004/d3/d3.2/v0.1/20040531/resources/dt.wsml
    dc:coverage "general"
    dc:rights http://deri.at/privacy.html
    dc:version "1.0"


  usedMediators
    comment: all 4 domain ontologies are used
                       as no mismatches have to be resolved, the ontologies are used directly
    ontology http://www.deri.at/research/projects/swf/ontologies/furniture.wsml
    ontology http://www.wsmo.org/2004/d3/d3.2/v0.1/20040531/resources/loc.wsml
    ontology http://www.wsmo.org/2004/d3/d3.2/v0.1/20040531/resources/po.wsml
    ontology http://www.wsmo.org/2004/d3/d3.2/v0.1/20040531/resources/dt.wsml


  capability
    non-functional-properties
      dc:description "defines the capability of the Seller Service"


  precondition
    axiom sellerServicePrecondition
      non-functional-properties
        dc:description "the input to the Service has to be:
                                      - the type of furniture sold (i.e. the product catalogue
of a furnishing house)
                                      - information on the seller (located in Austria)
                                      - accepted payment methods"
      logical-expression
       "inputItem ofvalue aFurniture
        and
        ((aFurniture memberof furn#chair[
            manufacturer ofvalue Manufacturer
               ],
         (Manufacturer memberof furn#manufacturer[name ofvalue "Gucci", address ofvalue
GucciAddress] or
        Manufacturer memberof furn#manufacturer[name ofvalue "Kika", address ofvalue Ki-
kaAddress] or
```

```
        Manufacturer memberof furn#manufacturer[name ofvalue "Leiner", address ofvalue
LeinerAddress]
        )
        comment:  this means that only request for chairs produced by certain manufacturers are
sold,
                                                addresses not further modelled here
      or
      (aFurniture memberof furn#kitchentable[
         material ofvalue Material] and
       Material memberof furn#material[name ofvalue "wood", quality ofvalue "high"]
      )
       comment: means that only request for wooden kitchen tables of high quality are sold
      or
      (aFurniture memberof furn#double)
     )
     and
     vendor memberof po#seller[
        address ofvalue SellerAddress
        salesIntention ofvalues {inputItem}
                        acceptsPayment ofvalues {AcceptedPayment}
     ]
     and
     SellerAddress memberof per#address[
                        name ofvalue "Leiner"
                        country ofvalue austria
     ]
     and
     (AcceptedPayment memberof po#creditcard[
          type ofvalue CCType] and
       (CCType ofvalue "MaterCard" or CCType ofvalue "Visa"))."

  assumption
   axiom sellerServiceAssumption
    non-functional-properties
      dc:description "there must be a buyer who is a registered customer,
                 and who has accepted payment methods"
    logical-expression
            "customer memberof po#buyer[
        shipTo ofvalue CustomerAddress memberof loc#address[country ofvalue austria]
        hasPayment ofvalues AcceptedPayment
     ],
     customerNumber oftype string."


  postcondition
   axiom sellerServicePostcondition
    non-functional-properties
      dc:description "the output of the service is:
                              - a piece of furniture, which cane be any type of furniture existing
in the product catalogue.
                              - the seller (which is Leiner here)
                              - the payment method accepted by the seller
                              - a delivery of the product, only in Austria"
    logical-expression
    "soldItem ofvalue inputItem
            and
            seller ofvalue vendor
            and
            itemDelivery memberof po#delivery[
              products ofvalues {soldItem}
                    sender ofvalue vendor
                    receiver ofvalue customer
            ]."


  effect
   axiom sellerServiceEffect
    non-functional-properties
     dc:description "there is a trade for the piece of furniture sold,
                    i.e. a contract of purchase"
    logical-expression
            "aTrade memberof po#trade[
                    items ofvalues {soldItem}
                    buyer ofvalue customer
                    payment ofvalue AcceptedPayment
             ]."


 interface
  non-functional-properties
   dc:description "defines the Choreography of the Seller Service "
  comment: not specified yet
```

Listing 13: Use Case - Seller Service

6 Conclusions

In this deliverable we have defined the description elements for SWF Goals and Service along with the formal languages to be used for modeling and the technological and we have showcased the concrete modeling of SWF Goals and Service within the use case scenario used throughout the SWF Deliverables. Besides, and we have defined the strategy for long-lasting and efficient alignment of SWF with WSMO: therefore, the 'WSMO-parts' of SWF are modeled and handled as WSMO components.

A second, updated version of this Deliverable will add the description elements for Service Choreographies, and possibly refine the specification of existing description elements.

References

Erdmann, M.: OXML 2.0. *Reference manual for users and developers of OXML - the XML-based Ontology Representation language for OntoEdit.* Karlsruhe: Ontoprise GmbH, 2003. available at: http://www.ontoprise.de/documents/tutorial_oxml2.0.pdf

Herzog, R.; Zugmann, P.; Stollberg, M.; Roman, D.: WSMO Registry. WSMO Deliverable D10, WSMO Working Draft 26 April 2004.
available at: http://www.wsmo.org/2004/d10/v0.1/

Kifer, M.; Lausen, G.; Wu, J.: *Logical foundations of object-oriented and frame-based languages.* Journal of the ACM, 42(4):741-843, 1995.

Oren, E.: *BNF grammar for WSML user language*. WSMO Working Draft D16.1, 05 April 2004 (under construction at point of writing).

Roman, D.; Vasiliu, L.; Bussler, C., Stollberg, M. (Ed.): *Choreography in WSMO.* WSMO Deliverable D14. available at: http://www.wsmo.org/2004/

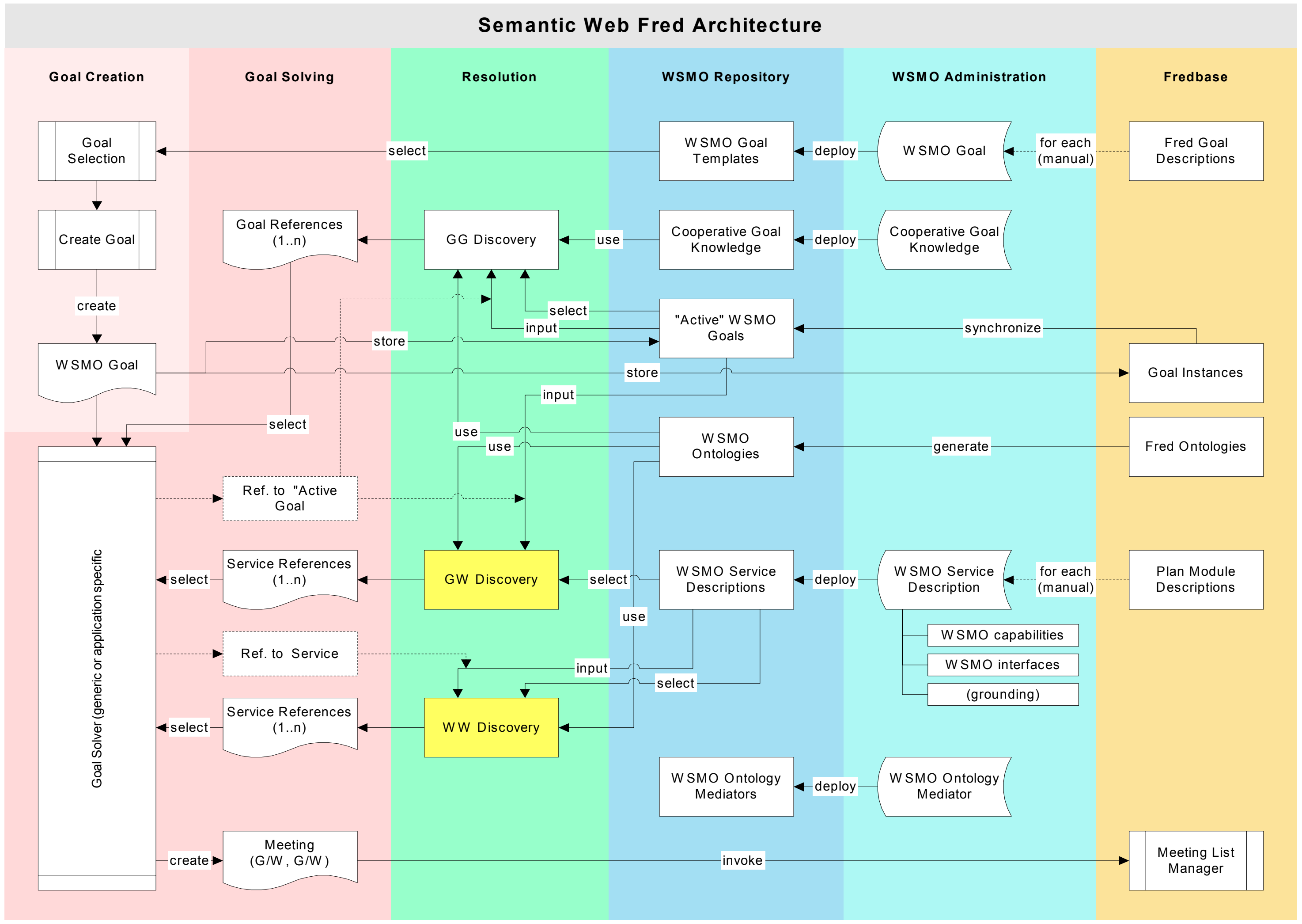
Roman. D. et al.: *Web Service Modeling Ontology WSMO, WSMO Standard, Version 0.3*. WSMO Deliverable D2, WSMO Working Draft 29 March 2004. available at http://www.wsmo.org/2004/d2/v0.3/20040329/

Stollberg, M.; Herzog, R., Zugmann, P.: SWF Framework. Semantic Web Fred Deliverable D1. available at: http://nextwebgeneration.com/projects/swf/papers/SWF-D1-SWFFramework-final.pdf

Stollberg, M.; Lausen, H.; Arroyo, S.; Herzog, R.; Smolle, P.; Fensel, D.: *Fred Whitepaper*, 2004.
available at: http://www.netdynamics-tech.com/media/downloads/FRED-WhitePaper.pdf

Stollberg, M.; Lausen, H.; Lara, R.; Polleres, A.: WSMO Use Case and Testing. WSMO Deliverable D3.2, WSMO Final Working Draft 28 June 2004.
available at: http://www.wsmo.org/2004/d3/d3.2/v0.1/

Appendix – SWF Technical Architecture


Semantic Web Fred Architecture
Goal Creation
Goal Solving
Resolution
WSMO Repository
WSMO Administration
Fredbase
Goal Selection
select
WSMO Goal Templates
deploy
WSMO Goal
for each (manual)
Fred Goal Descriptions
Create Goal
Goal References (1..n)
GG Discovery
use
Cooperative Goal Knowledge
deploy
Cooperative Goal Knowledge
create
select
input
"Active" WSMO Goals
synchronize
store
WSMO Goal
store
Goal Instances
input
select
use
use
WSMO Ontologies
generate
Fred Ontologies
Ref. to "Active Goal
Goal Solver (generic or application specific
select
Service References (1..n)
GW Discovery
select
WSMO Service Descriptions
deploy
WSMO Service Description
for each (manual)
Plan Module Descriptions
use
WSMO capabilities
Ref. to Service
input
WSMO interfaces
select
(grounding)
select
Service References (1..n)
WW Discovery
WSMO Ontology Mediators
deploy
WSMO Ontology Mediator
create
Meeting (G/W , G/W)
invoke
Meeting List Manager