Semantic Web Fred

SWF Framework

Net Dynamics & University of Innsbruck

Michael Stollberg
Reinhold Herzog
Peter Zugmann

March 2004

SWF-Deliverable:
D1

Version:
final

Date:
13-May-04

Net Dynamics

Executive Summary

This document defines the framework and architecture of the Semantic Web Fred. Herein, the system components and the functionality of the system are defined. This is the basis for the development to be performed in the Semantic Web Fred project.

This document is Deliverable No. 1 of the Semantic Web Fred project.

The project is funded by the "Wiener Wirtschaftsföderungsfonds" under the programme 'Call CoOperate 2003'. The project proposal has been awarded as the 2nd best project proposal in that call. 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 Tables

1 Introduction

The Semantic Web Fred project (short: SWF) aims at extending the functionality of the FRED system for goal-driven, automated resolution of tasks in a cooperative environment. This means that agents, called Freds in the system, perform tasks automatically on behalf of their owners as electronic representatives. According to the paradigms of agents as autonomously acting entities in a software environment, a Fred has to interact with other Freds in order to resolve the tasks it has been assigned. Therefore, a Fred has to find interaction partners that provide the capabilities needed to resolve the task and solve the task in a cooperative manner. The functional model of such collaborative relies on real-world cooperation models: for solving a complex task, several parties have to interact because the required facilities might be hold by different parties. Establishing cooperation between different parties can only be achieved when the cooperation is profitable to all partners because nobody offers a service without getting something in return. Following this, the general model of a cooperative agent system requires that each agent has at the same time goals and services: the former represents a desire that an agent wants to obtain as profit when joining a cooperation with other agents, and the latter represents the capability that the agent offers which is used by a cooperation partner for resolving a task it is assigned. This model is already realized in the FRED system and the SWF project aims at improving the functionality of the system and aligning it within emerging technologies around the Semantic Web and Semantic Web Services.

The FRED system, developed by Net Dynamics, is an environment for cooperative agent-based applications that allows import of ontologies and integration of external Web Services. Moreover, it integrates technologies for mediation of possibly heterogeneous resources with agent-based techniques for automated execution of tasks, thereby combining the core technologies identified for the Semantic Web. Thus, the FRED system can serve as an integrated platform for Semantic Web applications. As shown in [Stollberg et al., 2003], the FRED system in its current status of development provides an appropriate overall architecture for automated execution of tasks that are delegated to agents as electronic representatives (called Freds in the system) with a goal-driven approach for resolutions of tasks by implementations available in the system. This means that technological components are defined for dynamic and automated discovery and execution of suitable services available in the system for solving the tasks that are assigned to Freds. Although the overall architecture of the system seems to be appropriate – especially the architecture of the Goal-Service-Resolution technology, particular technological constructs are rudimentary in the

current status of development. Thus, the functional quality of the FRED system can be improved by enhancing particular components without need of changing the overall architecture.

Following this starting position, the approach for SWF is to drill up the existing FRED system and to incorporate more advanced goal resolution technologies. Therefore, technologies will be used that are emerging in the field of Semantic Web Services. The reason for this is that the problems arising within Semantic Web Services are very similar to those arising in SWF. Thus, usage of emerging standards is evident for SWF in order to ensure conformability with possible future standards. On the other hand, as these technologies are in their very early stage of development and the SWF project aims at providing a significant contribution to research and development of Semantic Web Services technologies and platforms. More precisely, the SWF will be based on the Web Service Modeling Ontology WSMO.[1]

Regarding the impact of SWF to research and development of platforms for agent-based Semantic Web applications, inspection of related work has shown that there does not exists any comparable system or development effort that integrates agent technology, ontology technology, and goal-driven cooperative Goal-Service-Resolution with support for Semantic Web Services so far. Thus, the contribution of the SWF project is an implementation of a novel type of runtime environment for agent-based applications that utilize Semantic Web resources. Its essential features are that SWF will provide a platform for automated cooperation that will be based on most recent technologies, it will provide means for integration of external Semantic Web resources, it will be comprised of functionalities for dynamic discovery, composition and execution of services, and, last but not least, it will include mediation facilities for establishing interoperability between possibly heterogeneous resources.

This document defines the SWF Framework which defines the components, the architecture, and the overall functionality of SWF. As the first Deliverable of the project it will serve as the basis for detailed specification and implementation of components and mechanisms of SWF.

The document is structured as follows: Section 2 defines the objectives of the SWF project and outlines the approach for development of SWF; Section 3 specifies the architecture of SWF by identifying the building blocks and defining their functionalities as well as the overall specification of the system. Besides, the arising requirements for

[1] WSMO Website: www.wsmo.org

the components and mechanisms to be developed in the course of the project are identified. Section 4 exemplifies the functionality of the system by a detailed example; Section 5 defines the scope of the SWF project, determining achievable goals with regard to the resources of the project; finally Section 6 concludes the SWF Framework. Additionally, Appendix A – Glossary lists the building blocks of SWF as well as other important technical terms and defines their meaning in the SWF Framework.

2 Objectives

The overall objective of the SWF project is to enhance the technologies for goal-driven cooperation of agents in the FRED system. Therefore, SWF will incorporate emerging technologies for the Semantic Web and Semantic Web Services in order to support Semantic Web applications. Thus, SWF will utilize the benefits of semantic technologies for advanced processing facilities on the one hand and conform to upcoming future standards. Besides, the SWF project will provide a significant contribution for the development of technologies for the Semantic Web and Semantic Web Services.

2.1 Mission

According to the general mission of the SWF project has been described before, the measurable objectives are:

- Specification of a Framework for the SWF which identifies the building blocks and the functionality, and which depicts to novelty of the system.
- Specification and implementation of enhanced technologies for automated cooperation and dynamic service resolution:
  - Description Language for Goals and Services; the latter can be Plans, Processes, or external Web Services.
  - Resolution Mechanisms for establishing and executing automated cooperation and dynamic discovery of appropriate services as well as basic mediation facilities needed
  - An architecture and execution environment for automated cooperation based on dynamic service discovery
- Showcasing SWF in Use Case Implementations
- Proclamation of SWF and the project results in scientific and industrial boards and events.

2.2 Starting Position

Although the objectives of the SWF are very high and may seem to be unattainable, we believe that we can achieve them because of the elaboration status of the starting points for the project – the FRED system on the one hand and the research expertise in Semantic Web Service technologies on the other. So, the implementation bases as well as the conceptual foundations for SWF are existent, and the duty of the SWF project is to further elaborate them, to combine them and to integrate them into a coherent system. In the following we outline the starting points of SWF and explain the approach followed in the project.

2.2.1 The Fred System

The FRED system is an integrated platform for agent-based applications that has been developed for 3 years before the start of the SWF project. The system is comprised of the following building blocks:

1. **FredBase:** the agent runtime environment. The interaction of Freds takes place in Meeting Rooms: a Fred is only called into a meeting when it has to interact with another agent in order to solve the task it is assigned. Partners called in a meeting are checked before for their complete interoperability. Besides, the FredBase provides interfaces for connection to external systems.
2. **Smart Objects:** ontology handling technology. Ontologies are applied as the grounding data model throughout the whole system. A Smart Object is an ontology object which is transformed into a Java object in order to allow usage of conventional, sophisticated technologies for data handling. Smart Objects are as expressive as OWL for describing knowledge structures. The management techniques for Smart Objects comprise all features needed to ensure correct, secure and scalable handling of Smart Objects.
3. **Goal-driven Task-Service-Resolution:** the resolving a task that is assigned to a Fred is done by specifying a Goal that represents the tasks from the user's perspective. To solve a Goal, an automated mechanism detects suitable problem solving services that can solve the Goal and/or other Freds with suitable cooperative Goals (e.g. buyer needs seller) and appropriate services. This is referred to as Service Discovery respectively Goal solving. Further mechanisms exist that determine appropriate Partners for a meeting, referred to as Meeting Creation mechanisms. A Goal can be solved either by a Plan, an implementation for functionality with dynamic control flow, or by a Process that contains a user

specified workflow procedure which can itself be comprised of several Plans, external services or sub-processes. Also external Web Services can be integrated as problem solving resources into the system via their WSDL descriptions.

While the FredBase and the Smart Objects technology is considered to be 'mature' with regard to the functional needs for the FRED system, the Task-Service-Resolution technology provides a suitable architecture for dynamic and automated goal resolution. But some of its underlying technological components are only rudimentary developed at this point in time, which hampers the quality of the resolution mechanisms. Especially, the description language – which is the essential basis for goal-driven systems – is very basic; this implies that the currently existing goal resolution mechanisms are also very basic because they work upon the description language. A detailed description of the FRED system in its current status of development is provided (Stollberg et al., 2003)

2.2.2 The Web Service Modeling Ontology WSMO

The Web Service Modeling Ontology WSMO is a Semantic Web Services research effort of the SDK-Cluster.[2] The aim of research efforts around WSMO is to define an overall framework that covers all aspects relevant for Semantic Web Services, and subsequently develop needed technologies for handling Semantic Web Services (Roman et al., 2004).

The conceptual foundation of WSMO is the Web Service Modeling Framework WSMF (Fensel and Bussler, 2002) which identifies 4 main components needed for Semantic Web Services: Ontologies provide the formal semantics of the information used by all other components, Goals specify objectives that a client may have when consulting a Web Service, Web Services represent the functional part as a computational resource accessible over the Internet, and Mediators which are used as connectors between components along with facilities to establish interoperability (see Figure 1).

[2] SDK-Cluster: joined research and development effort of the 6th framework IST projects SEKT, DIP, and Knowledge Web. SDK-Cluster Homepage: www.sdk-cluster.org

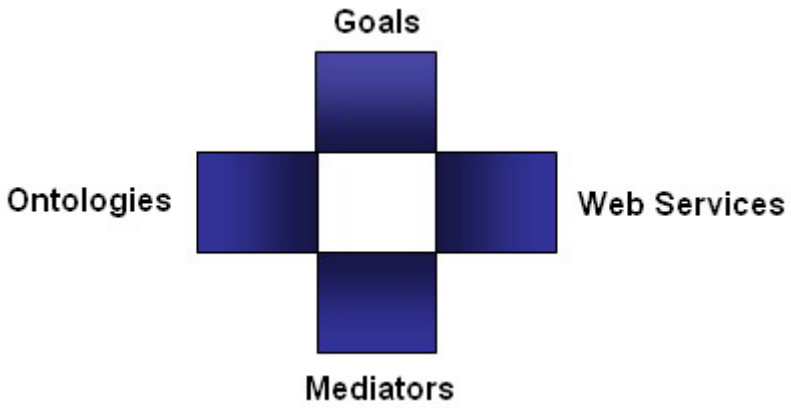

Figure 1: WSMF Components

These components have to be described semantically in order to support automated discovery, composition, and execution of Web Services – the aim of Semantic Web Service technologies. While WSMF defines the components, the requirements for a suitable description language and the general approach for Semantic Web Services technologies, WSMO is an ontology that explicitly defines the description elements for each component. We shortly summarize the most important description elements of WSMO for the particular components. The complete WSMO ontology design rationales and further explanation can be found at the WSMO project homepage (see footnote 1). [3]

The objective of WSMO is to define the description language needed for inference-based Semantic Web Services technologies. This means that logic-based reasoning mechanisms are envisioned for all technologies that handle Semantic Web Services. The formal language used in WSMO is F-Logic (Kifer and Lausen, 1995).

2.2.2.1 Ontologies

Ontologies provide the machine-readable semantics of the information used by all other components, thus ensuring semantically correct information interchange between the components. According to common definition of ontologies, ontologies in WSMO are described by:

- **Concepts:** the conceptual entities of a domain

[3] WSMO is currently under development. This document confirms to WSMO version 0.1 as specified at: http://www.wsmo.org/2004/d2/v01.

- **Relations:** properties of concepts and relationships between concepts. Relations, so also concepts properties, have range restrictions and a list of parameters
- **Axioms:** axiomatic constraints on concepts, relations, and instances
- **Instances:** concrete individual, type of a specific concept

2.2.2.2 Goals

A Goal specifies objectives that a client may have when he consults a web service, i.e. functionalities that a Web Service based system provides from the user perspective. Goals are described by:

- **Post Condition:** the state of the information space that is desired. This means that the status of the world after the goal will be solved is described with regard to the input information provided, so a mapping functions of the input data to its desired state.
- **Effects:** state of the world after the Goal is solved, without regard to the input information.

  Example: when I buy a ticket with a web service, the post-condition is that the connection conforms to my traveling desire and the ticket is valid for the journey. An effect is that I have the real ticket in my hands – but this is not important relevant of the technical functionality of the Web Service based application.

2.2.2.3 Web Services

Web Services are the computational resource that can be invoked over the Internet as a piece of functionality for a software application. In order to provide all information needed for automated Semantic Web Services technologies, a Web Services description in WSMO consist of 3 parts: the **Web Service Capability** which describes the functionality of a Web Service – needed to discover a suitable Web Services for a specific Goal, the **Web Service Interface** which describes the externally visible behavior of the Web Service – needed for dynamic composition of Web Services and compensation during execution. Besides this, information is needed on the input and output, the message patterns for communication, and the technical accessibility of Web Services. These are included in the Interface description of WSMO, but are commonly referred to as the **Web Service Grounding** that describes the information needed to actually access and use a Web Service. We shortly summarize the description elements for these 3 parts of a Web Service description:

Web Service Capability (counterpart to Goal Descriptions)

- **Pre-Conditions:** description of state of the information space (i.e. the input data) for enabling it to provide its service. Pre-conditions define conditions over the input.
- **Post-Conditions:** description of state of the information space after execution of the service. Post-conditions define the relation between the input and the output.
- **Assumptions:** describe the state of the world (without relation to the input) expected before execution of the service
- **Effects:** describe the state of the world (without relation to the input) after the execution of the service.

Web Service Interface (the "grey-box description" – not completely specified in the current version of WSMO)

- **Errors:** describe exceptions occurring during execution, according to an error framework.
- **Choreography:** describes the external business process of a Web Service, i.e. it decomposes the Capability of a Web Service into separate processing steps (sub-capabilities) and defines the information needed for cooperation. A choreography instantiates a generic Message Exchange Pattern as an act of cooperation, thereby defining the actual message formats for an MEP.
- **Message Exchange Pattern (MEP):** describes a sequence of conditional speech acts for communication, i.e. the pattern of messages interchanged needed for communication between two entities.
- **Orchestration:** describes the decomposition of the internal process of a Web Service with special regards to other Web Services invoked. Therefore, so-called Proxies are used that represent the invoked Web Service on the side of the invoking Web Service. An orchestration instantiates a Problem Solving Pattern (PSP).
- **Compensation:** describe the state of the world (without relation to the input) expected before execution of the service

Web Service Grounding (not explicitly specified in the current version of WSMO)

The Web Services Grounding holds information on how to access a Web Service, concerning protocol and message formats, serialization, transport and addressing.

Although the Web Service Grounding is incorporated in the Web Service Interface (see above) in the current version of WSMO, we list the related aspects here explicitly because these are very important aspects when utilizing Web Services:

- is a mapping from abstract descriptions in WS Capability and WS Interface into concrete descriptions of elements required for interacting with the service, i.e.:
    - inputs/outputs
    - error messages in accordance to an error handling concept
    - compensation information (alternative paths for same functionality)
- can link to various sorts of implementation, e.g. WSDL, CORBA, ebXML and others.
- to be relying on SOAP and WDSL as W3C recommendations with strong industrial backing.

2.2.2.4 Mediators

Mediators in WSMO fulfill 2 purposes: on the one hand, they define the connection between two components of WSMO and, on the other hand, they provide the possible needed mediation service between the connected components. This general architecture allows the linking of resources and, at the same time, includes description of the mediation facility and is employed in whenever heterogeneous WSMO components are linked. All interoperability and mediation problems are focused in Mediators which allows developers to concentrate on a specific aspect when defining a WSMO based application. WSMO differentiates the following types of Mediators, with regard to the type of connection realized and the type of mediation service that is needed to establish interoperability for the connection type (also see Figure 2):

- **OO Mediator:** linking 2 Ontologies, i.e. import of an ontology by another ontology or use of an ontology by a component. Mediation facility: ontology integration.
- **GG Mediator:** linking 2 goals, i.e. definition of Sub-Goal hierarchies. Mediation facility: reduction and extension of post conditions of Goals.
- **WW Mediator:** link 2 Web Services when they have to interact directly. Mediation facility: data, process, and protocol mediation.

- **WG Mediator:** link between a Web Service and a Goal, more precisely between the Capability of a Web Service and a Goal. Medication facility: reduction and extension of post conditions of Web Services.

In general, a Mediator is described by the following notions (we omit the non functional properties here - of course every Mediator carries these as well):

- **Source Component:** this is the one of the 2 WSMO elements to be combined that "initiates" the connection
- **Target Component:** the WSMO element that gets connected to the Source Component.
- **Mediation Service:** this describes the possibly needed mediation facility in order to make the Source and Target Component interoperable. Depending on the type of Mediator, a specific type to mediation service is needed (see above).

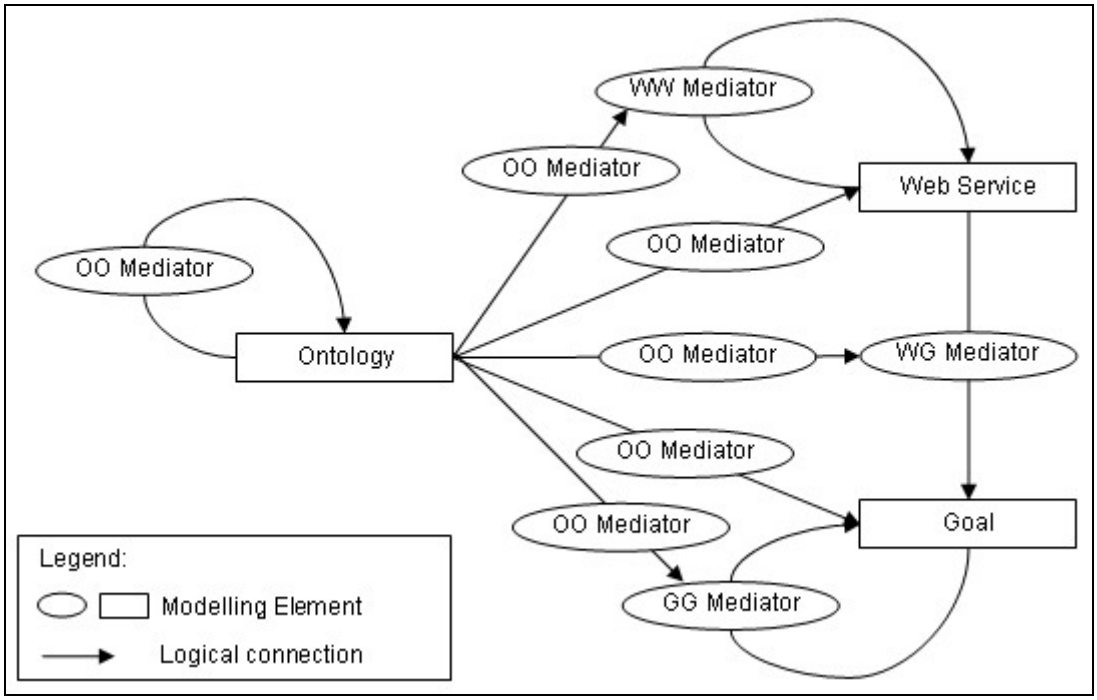

Figure 2: Concept of Mediators in WSMO

More detailed explanations of the Mediator concept in WSMO and the usage of the different types of Mediators as well as their specific modeling elements is provided in the WSMO Primer.[4]

[4] WSMO Primer: http://www.wsmo.org/2004/d3/d31-primer/

2.2.3 Emerging Semantic Web Service Technologies

Numerous research and development efforts are currently concerned with the development of technologies for the Semantic Web and Semantic Web Services apart from WSMO. We briefly summarize the most relevant ones and show how they relate to SWF.

At first, ontology specification languages are of interest that provide representation techniques for ontologies, the basic building block of semantically enhanced IT-systems. The most relevant languages are the recommendations of the W3C, regarding the 'language layer cake' for the Semantic Web (Koivunen and Miller, 2001). Apart from XML for information structuring in order to support automated processing of web content, RDF serves as the basic language for defining meta-data on for web resources. Along with RDFS, it provides the basic ontology language that allows definition of concept taxonomies (Manola and Miller, 2004). Extending the expressive power in order to provide a richer technology for specification of ontologies for the Semantic, the Web Ontology Language OWL was invented, which also is a recommendation of the W3C (McGuiness and van Harmelen, 2004). The Smart Objects as existent in the Fred system provides a similar expressive power as OWL (Stollberg et al., 2003).

The OWL-S is, apart from WSMO, the other major initiative in Semantic Web Services (OWL-S Services Coalition, 2004). It defines a model for describing Web Services by their Service Profile (intended purpose of the service), the Service Model (represents how the service works) and the Service Grounding (gives the details of how to access the service). In contrast to WSMO, OWL-S only defines a model for describing Web Services. A detailed comparion of WSMO and OWL-S in given in (Lara et al. 2004).

A couple of other technologies are concerned with specific aspects of Semantic Web Services. A sophisticated summary is provided in (Peltz, 2003):

- **WSDL:** The Web Service Description Language defines a language for describing the access to Web Services and message exchange using different Internet protocols (Chinnici et al., 2003). In the context of Semantic Web Services, WSDL is mostly applied for the Service Grounding.
- **BEPL4WS:** a technology for describing the behaviour of Web Services in a business interaction, specifying a XML-based grammar for control logic need to coordinate web services participating (Curbera et al., 2002). BEPL4WS is based on traditional business process and workflow technologies, commonly referred to as the most adequate process language for Semantic Web Services.

- **WSCI, WSCL:** The Web Service Choreography Interface along with the corresponsing Web Service Conversation Language is an effort of the W3C that defines a technology for specifying message flows between Web Services on top of WSDL (Arkin et al., 2002; Banerji et al., 2002). WSCI / WSCL will serve as the basis for specifying Choreography and Orchestration in WSMO and thus in SWF.

Besides, useful work is provided by different working groups around the W3C Web Service Activity[5]. All these technologies are analysed, used and incorporated into the WSMO project, thus they provide the basis for the SWF project.

2.3 SWF Approach

With respect to the starting point of the SWF project, the approach to be followed in the project is evident. Basically, it consists of 3 steps:

1. *Drill the current architecture of the FRED system and make it conform to WSMO, respectively emerging Semantic Web and Semantic Web Service technologies.*

   This is step is mainly performed in "SWF D1: Semantic Web Fred Framework" (this document).

2. *Develop a suitable Description Language and advanced resolution mechanisms for SWF, thus extent and improve the cooperation technology of the FRED system.*

   This step is mainly performed in the Deliverables "SWF D3: Goal and Service Description Language" and "SWF D4: Resolution Mechanisms".

3. *Test and showcase SWF in use case implementations.*

   This step is mainly performed in the SWF Use Case implementations.

The development of SWF will be a stepwise refinement, i.e. starting with the core technologies and enhance them in next development iterations.

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

3 Semantic Web Fred Architecture

This section identifies the building blocks of SWF, specifies their functionality as well as the overall workflow of the system. Before we commence with the specifications of components and mechanisms, we first illustrate the overall approach of SWF.

The "traditional" approach that dominated IT-system design over decades is the request-provider model. Therein, a request carries the information of the service, invokes the service and collects the result of the service. In a goal-driven approach, Goals are introduced to de-couple the request from the service logically, meaning that the requester does not have to know about the service. The request carries "Request information", while the service carries the "Service information".

The objective of SWF, and the reason for employing a goal-driven approach for automated cooperation, is to de-couple the requester and the service provider in order to overcome the shortcomings of traditional IT-systems. These are that request and service have to be defined for every single application scenario; this hampers the re-use of computational resources, thus requiring re-inventing the wheel for every car, resulting in traditional IT-systems not to be application independent. The goal-driven approach underlying the FRED system and the SWF project thus aims at creating a new type of IT-system.

This has of course very far-ranging implications on system design. First, the interacting parties are now equal in their status of control and initiation of system activity. First, there is **symmetry** between the entities of a system instead of a hierarchy, meaning that a requester and a provider interchange their roles continuously. Secondly, the connection between the functionality of a requester and the resolving services is **de-coupled** and derived **dynamically** during design or runtime, thereby enabling maximal **re-use** of existing computational resources. One the other hand, such systems require more complex control structures than traditional IT-systems – but they provide a general infrastructure for IT-supported applications. The combination of agent technologies with most recent techniques currently under development for goal-driven systems, the *SWF project will provide a context-independent, goal-driven system for automated execution of tasks that are delegated to electronic representatives along with dynamic Service usage*.

To illustrate the SWF Cooperation Model, we imagine the cooperative environment in a company. The CEO wants to maximize the company's success, and thus requests his salesman to sell more. It is a clear requester-provider scenario, but the reality is: not just the CEO has a goal and the salesman has the capability to do so, but both have a goal

and both must have some services to offer. The CEO might have the goal:"I want my salesman to sell more". To achieve this, he needs to offer some services to his salesman, for example "motivation, provision of money and compensation, support of the salesman, etc." On the other hand, also the salesman will have a goal, for instance just "sell to earn money", and he certainly also needs services to reach that goal, like "react on motivation and salary, use the provided support, etc.". Figure 3 shows this understanding of Cooperation.

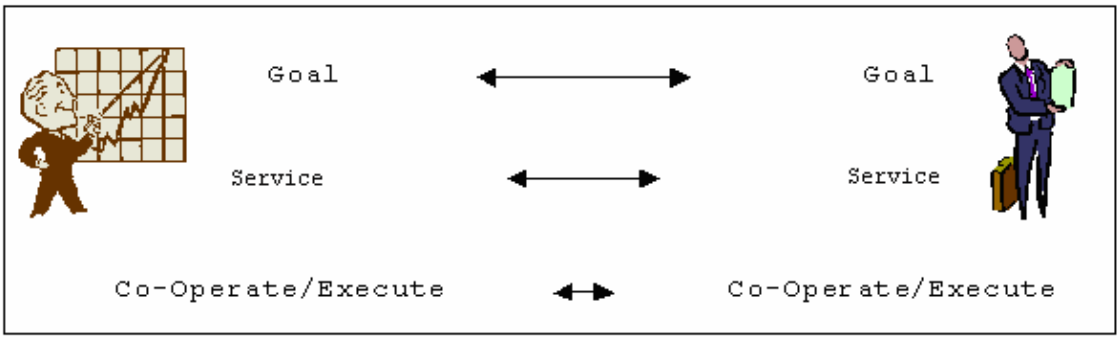

Figure 3: Basis of SWF Cooperation Model

Based on this real-world Cooperation Model, the technical architecture of SWF is defined by mapping this understanding of cooperation into a software system. We therefore define the necessary software components and resolution mechanisms and resolve the technical implications arising on basis of the starting position outlined in Section 2.

3.1 Cooperative Goal and Service Resolution

With regard to the Cooperation Model explained above, the general architecture and workflow of SWF works like this: in the Goal Repository the Fred System lists all available Goals. Over time, by users, i.e. user applications, or by Services Freds get Cooperative Goal Instances assigned which are derived from available Goals in the Repository. Each Fred may hold none, one or several Cooperative Goals. The Goals specify what the Fred is expected to do, typically in cooperation with one or more other Freds.

Given such a Goal Instance, a Goal Solving Process gets started. The first step is to identify compatible Cooperative Goals and the Freds who own those Goals (GG Discovery). Then, for each of the identified Goals, suitable Services are detected together with the Freds (Resources) who provides the Services (GS Discovery).

The discovered Services have to be compatible, meaning they have to have compatible external behaviors in order to allow automated interaction. This is checked and established via the WW Discovery mechanism. This all together is a dynamic iterative process partly executed at design time, partly at runtime. Most independence, but also risk of errors, is given if the resolution happens at runtime only. Part of the discovery mechanism is to include Mediators to ensure interoperability between the components.

Once all components are identified, i.e. the Cooperative Goals, the associated Services together with their external behavior to solve the Goals, and the Resources (i.e. Freds) to execute the Services, a "Cooperative Contract" gets established, which is referred to by all identified partners during the following Service Execution, and the Services get invoked in all partners.

Service Execution itself takes place in so called Meetings or within a Process Engine, the runtime environments of Freds. Services of different partners referring to the same contract are executed contemporarily (which is in a meeting) to ensure maximum success of cooperation. When errors occur during execution either rollback mechanisms are initiated (in a Meeting), or Error Handling and Compensation mechanisms are called (in a Process).

If everything works fine after execution of the Services the Goal enters a state of *solved,* controlled by the Goal Solving Process.

For all SWF components, distinctive repositories are created to hold the information or instances. Figure 4 shows this architecture in a comprehensive manner, introducing the components of the SWF. For graphical notation, we obey the graphical representation of UPML (Fensel et al., 2003) to the most possible extent.

In the following we introduce and define the components and mechanisms of the SWF architecture. We do not provide exhaustive rationales and scientific definitions in this section because it is intended for internal project specification. The scientific foundation will be elaborated in a later stage of the project.

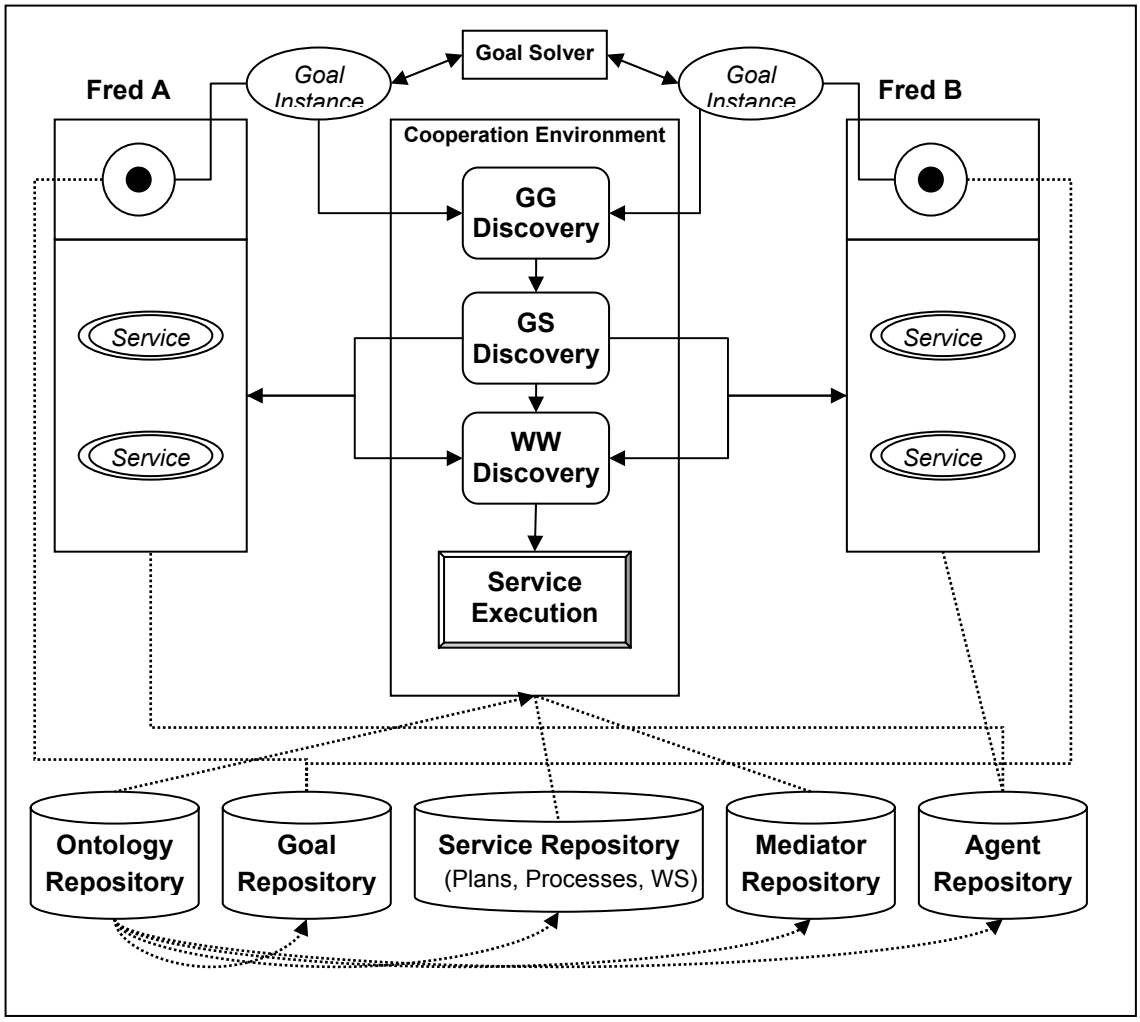

Figure 4: SWF - General Architecture

3.1.1 SWF Components Definition

At first we define the distinctive components of the Cooperative Goal and Service Resolution technology, while the methods and mechanisms that work with these components are specified later. We provide exhaustive explanations on the understanding of particular components and provide rationales why they are needed in SWF.

3.1.1.1 Ontologies

Ontologies define the terminology for an application. They provide the machine-processable semantics of all information needed.

We differentiate between different types of ontologies:

- **Domain Ontologies:** definition of concepts in the domain of discourse
- **System Ontologies:** dffinition of concepts of the technological components in order to allow semantic enhanced processing. The distinct system technologies needed are a Goal Ontology and a Service Ontology.

For handling ontologies and ontology instance data in SWF, WSMO technology and the existing Smart Objects technology of the FRED system is used.

3.1.1.2 Cooperative Goal

A Goal is understood as an objective that a user may have when he consults a service or a system. A Goal defines the WHAT of a problem to be solved, not the HOW or WHO or WHERE – the latter aspects are resolved dynamically by the Discovery Mechanisms. With introducing Goals, the de-coupling for a functionality requester and the provider is realized as the main objective of goal-driven systems.

In SWF, the concept of Goals serves for several purposes.

1. **Goal for Task Assignment:** A Goal specifies a desire that is to be solved by an agent. This means that a Fred owner assigns a Goal to a Fred as the task that the owner delegates to his electronic representative.
2. **Request for Cooperation:** Goals are used for establishing a Cooperation in order to solve Goals of Freds. This means that the Goal is used to find other Freds as possible collaboration partners. Therefore, the Goals that Freds carry have to be compatible, meaning that it has to be possible to determine that two Goals, and thus the Freds that carry them, are suitable cooperation partners. This usage of Goals is the heart of the SWF Cooperation technology and thus further explained in the following (see section 3.1.1.2.1).
3. **Goal-driven Service Resolution:** A Goal is the description of a generic desire for functionality which can be solved arbitrarily by a Service or by a manual activity. This usage of Goals in SWF is equivalent to goal-driven Web Service discovery as proposed in WSMO.

In fact, SWF uses Goals as central components which is an extension of the concept of Goals in WSMO – therein Goals are simply a request for functionality used to discover appropriate Web Services. As the concept of Goals in SWF is more extensive, additional conceptual aspects are required which we define below.

3.1.1.2.1 Cooperative Goals, Complete and Partial Description

A main merit of the Cooperation Model that has been introduced above as the basis of SWF is that cooperation will only work successfully when it is profitable for both partners. This means that cooperation partners both have a Goal that they want to achieve – see the introductory example of the CEO and a salesman.

In SWF, we want to detect suitable cooperation partners automatically during runtime of the system. Therefore, information on the compatibility for cooperation has to be carried by a Goal.

Goals in WSMO describe a desire for functionality based on (one or more) domain ontologies. Such a Goal specifies "I want X to be achieved", in terms of an ontology. In order to support detection of potential cooperation partners, there has to be more information. More precisely, a Cooperative Goal has to specify: "I want X to be achieved and my role in a cooperation can be Y". For detection of potential cooperation partners, Goals have to be checked in two regards: at first whether two Goals specify the same object of interest, i.e. they both want to deal with X, and secondly whether to owners of the Goals have compatible cooperation roles, i.e. the Y.

A small example for clarification: some Fred *A* has the Goal "I want to buy a chair and I am a Buyer", a Fred *B* has the Goal "I want to sell a chair or a bed and I am a Seller", a Fred *C* has the Goal "I want to sell a car and I am a Seller", a Fred *D* has the Goal "I want to sell a bed or a cupboard and I am a Seller". Regarding the cooperation roles, Freds *B, C,* and *D* would be suitable partners for Fred *A*, but with regard to the object of interest only Fred *B* is a potential cooperation partner.

According to WSMO, the core description element of Cooperative Goals is the postcondition that specifies the state of the information space that is desired. Therein, the object of interest (i.e. X) as well as the cooperation role (i.e. Y) are described in terms of ontologies. The information on the compatibility of cooperation roles can either be incorporated into the respective domain ontology or be defined in a separated "cooperation role compatibility ontology".

These extended Goals are called **Cooperative Goals** in SWF. They are used for detecting potential cooperation partners via compatible Goals (GG Discovery, see section 3.1.2.2.1) as well as for discovery of the required services for cooperation partners (GS Discovery, see section 3.1.2.2.2).

Regarding the structure of Cooperative Goals, a further aspect is important. When a Goal is specified by a Fred (or its owner), maybe not all information on the object of

interest are known a priori. Referring to the example above, this means that Fred *A* only specifies in it's the Goal that "I want to buy a chair and I am a Buyer", but does not give information about the payment method. Fred *B* defines a Goal "I want to sell a chair and I am a Seller that expects credit card payment only". For successful cooperation, Fred *A* has to be able to pay with credit card, therefore he needs to provide a proper Service in the cooperation.

Thus, we differentiate between Complete and Partial Goals (see also Figure 5):

- **Complete Description of Cooperative Goals:** the Goal defines the view on the Goal from the Goal Owner as well as from a possible partner's perspective.
- **Partial Description of Cooperative Goals:** a Goal only describes the view of the Goal from the owner's perspective.

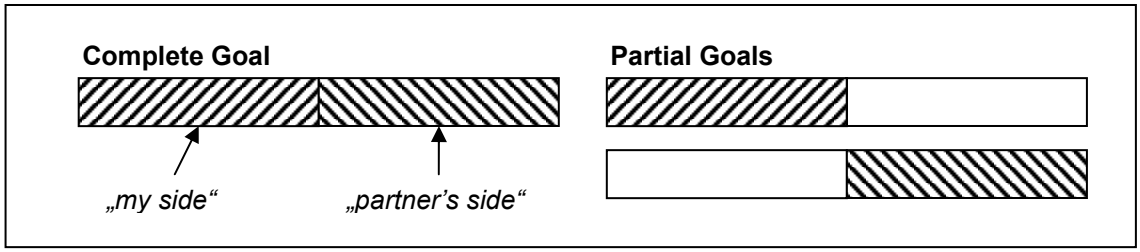

Figure 5: Complete and Partial Goals

After suitable cooperation partners have been detected, the Complete Goal is needed for discovering appropriate Services for the Cooperation at both partners. This means, if Fred *A* has defined his Goal as a Partial Goal as in the example above, this Goal has to be completed with the Goal information from a suitable partner to ensure that Fred *A* will provide a suitable service for payment (as the additional information from the partners Goal).

Actually, it is the normal case that a Goal is described from the point of view of its creator without regard to possible cooperation partners. The actual pre-requisites for cooperation are discovered during the negotiation phase of the cooperation contract. Thus, the approach of partially described Cooperative Goals provides a more realistic approach for cooperation than completely described Cooperative Goals.

3.1.1.2.2 Cooperative Goal Schema and Instances

In order to make Cooperative Goals usable as a central component of SWF, we have to distinguish Goal Schemas and Goal Instances. While a Goal Schema describes a generic Goal, a Goal Instance is "thrown" into the system as a specific request for functionality

of a SWF Fred; on this basis, Cooperation and automated Goal Solving are performed. The following gives the definition of the two components:

- A **Cooperative Goal Schema** is a generic specification that describes the schema of functionality / task that is solvable by the Services existing in the system. Goal Schemas are pre-defined in the system and kept in the Goal Repository (see section 3.1.1.4.2) wherefrom Goal Instances are created.
- A **Cooperative Goal Instance** is a particular instance of a Goal that specifies a concrete Goal, i.e. a request for cooperation with another Fred. A Goal Instance can only be created for a Goal Schema existent in the Goal Repository. Goal Instances have additional information, similar to the Process Ontology design existent in the FRED system: they have an owner, access restrictions, and further elements needed for computation.

3.1.1.2.3 Description Language Constructs

Cooperative Goals are described similar as WSMO Goals for supporting GG and GS Discovery. The definite description language constructs will be specified in SWF Deliverable D3 (Goal and Service Description Language) – the following is only a pre-selection:

- **Imported Ontologies / Used Mediators:** a Goal imports an ontology form the Ontology Repository as its data model. For including Mediators, this can be replaced by "used Mediators" that link the needed components together, thus following the Mediator concept of WSMO.
- **Post Condition:** the state of the information space that is desired. This means that the status of the world after the goal will be solved is described with regard to the input information provided, so a mapping function of the input data to its desired state.
- **Effects:** state of the world after the Goal is solved, without regard to the input information.
- **Input / Output:** Although captured by the definition of Goals post conditions, Goal In- and Output are possible needed for computational reasons (to be elaborated in SWF D3).

Additionally, the following descriptions elements are needed:

- Non-.functional Properties: Name / Title

- For Goal Instances, we have “Owner” and “Creation Date”, needed for Meeting Creation / Agent Cooperation.
- For resolution of a Goal Instance, we also need information on the state resolution of the Goal Instance. This would be, with respect to the resolution process: “created”, “pending”, “inProgress”, “resolved”.

3.1.1.3 Services

“Service” is the superordinate term for all types of problem solving resources that are available in the SWF system. All Services are described by a single description language for Services that is based the structure of WSMO for describing Semantic Web Services. This allows treating the different types of Services in SWF independently of their actual realization: moreover this approach allows applying the SWF technology for Semantic Web Services.

The understanding of Services in SWF is equivalent to the understanding of Web Services in WSMO – a service is a computational resource that can be invoked for solving a goal. In SWF, not only external Web Services can be used, but also problem solving resources that are inside the system (Plans and Processes). The idea of the SWF-Service Model is that every problem solving resource in the system is described equally in order to provide the descriptions needed for Service Discovery, while the actual execution of the Service that is actually used is independent of Service Discovery. The execution of Services is handled by the Execution Environment.

3.1.1.3.1 Service Types

The types of Services are shown in Figure 6 with further explanations below.

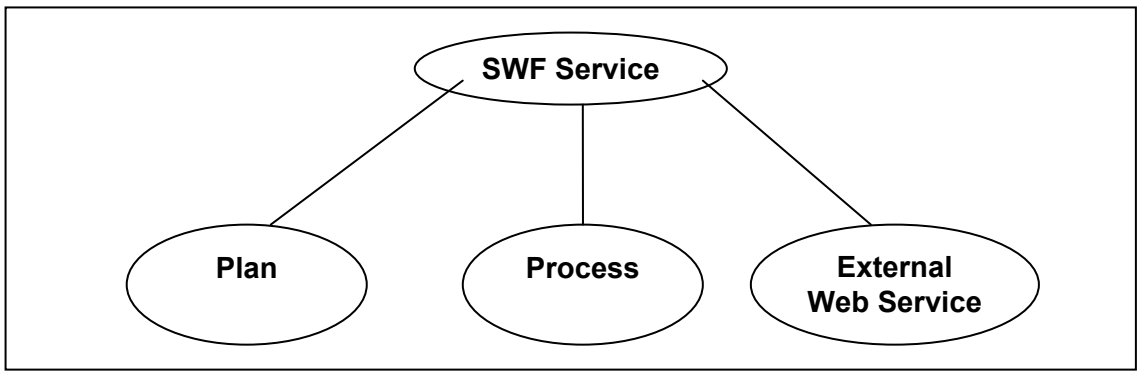

Figure 6: Types of Services in SWF

- **Plan:** A Plan is a problem solving service that is located inside the Fred system and is built on top of the Fred Plan Framework. Interaction is implemented using the behaviours and protocols of the Plan Framework API which is based on FIPA (see: http://www.fipa.org/specifications/index.html).
- **Process:** A process is a multi-step problem solving service, describing more complex functionality. The complexity alludes to the multiple steps of a process which requires more complex description and resolution mechanisms for Processes. Each step in a process can be solved arbitrarily, either by a Goal or by another Service. Processes can be invoked either during runtime by a Fred or another Process or they are triggered by an event or out of the application context.
- **External Web Service:** an external Web Services that can be invoked as a problem solving resource in SWF. Currently, there is the WSDLExecutorPlan which acts as a proxy and allows integrating external Web Services as Plans into the FRED system via their WSDL description.

3.1.1.3.2 Service Description Language Constructs

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 GS Discovery mechanism (see section 3.1.2.2.2) determines a suitable Service for a given Goal – that is a Goal Instance thrown by a Fred during runtime – and the Service Execution (see section 3.2) handles the execution of the particular Service according to its type, using the corresponding execution facilities. The general Service Description Language therefore has to be comprised of a Capability Description (1:1, i.e. each Service has one Capability) that provides the counterpart of a Goal Description in order to support GS Discovery. Furthermore, a Service Interface (1:n, i.e. each Service can have several Interfaces) describes the external visible behavior of Service, especially its messaging sequence for usage and interaction with a Service (called Choreography). Besides, the type of Service has to be declared, as well as the accessibility, and the error and compensation information that are needed for Service Execution.

With respect to WSMO, the basic building blocks of the SWF Service Description Language are the Capability, the Interfaces, and Grounding descriptions (note that in WSMO v0.2 the Web Service Grounding is hidden in the Web Service Interface!!). Thus, Services in SWF are described by the same description elements as Web Services in WSMO with some changes and extensions. We have the 6 aspects of a SWF Service Description:

3.1.1.3.2.1 Imported Ontologies / Used Mediators

A link to the domain ontologies needed to describe the data used in a Service. This is a connection to the ontology repository that allows import of Smart Objects. For including Mediators in order to resolve possible heterogeneities, this can be replaced by "used Mediators" that link the needed components together, thus following the Mediator concept of WSMO.

3.1.1.3.2.2 Capability

The Capability of a Service describes the functionality of a Service. A Service Capability is the counterpart of a Goal, needed for Service Discovery. According to WSMO, we define the following description elements for SWF Service Capabilities:

- **Pre-Conditions:** requirements on the information space (i.e. the input data) for enabling it to provide its service. Pre-conditions define conditions over the input.
- **Post-Conditions:** description of state of the information space after execution of the service. Post-conditions define the relation between the input and the output.
- **Assumptions:** describe the state of the world (without relation to the input) expected before execution of the service. [Unsure whether needed, to be worked out in SWF D3 "Goal and Service Description Language"].
- **Effects:** describe the state of the world (without relation to the input) after the execution of the service. [Unsure whether needed, to be worked out in SWF D3 "Goal and Service Description Language"].
- **Input / Output:** describes the input required for the Service as well as its output. [This might be caught be the description of pre- and postconditions - to be worked out in SWF D3 "Goal and Service Description Language"].

3.1.1.3.2.3 Interface

Describes the external visible behavior of Services as needed for WW Discovery (see section 3.1.2.2.3).

In general, the SWF Service Interfaces describe the Choreography of a Service, i.e. the externally visible activities of a Service along with their required messaging sequences. We differentiate single-state or stateless services and stateful services: A stateless service describes his messaging sequence only, while for stateful services also the externally visible activities have to be described. For Service Interface descriptions, we are only interested in the external behavior, not in the internal realization of a Service.

Besides, we assume that a Service provider offers a "correct" Interface description in the sense that the description is valid for real Service usage. In the future, this will be assured by automated generation the Service Interfaces.

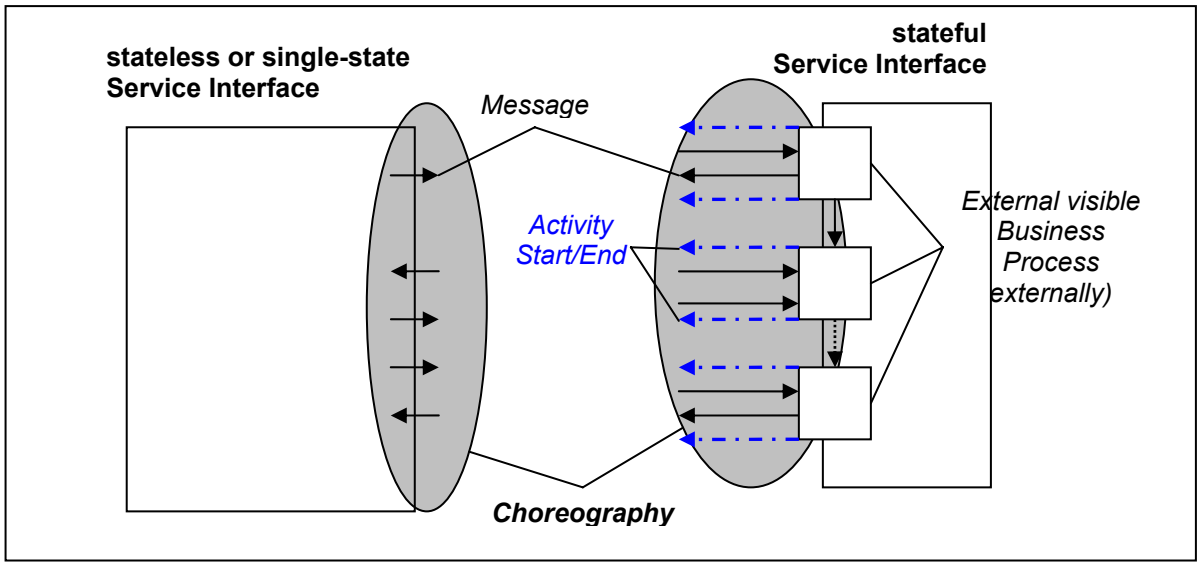

Figure 7: SWF Service Interface Structure

In SWF, we do not need the concept of Orchestration as part of a Service Interface that describes the request for functionality (or cooperation, respectively) by a Service. This is caught by the SWF Process Model, already existing in the FRED system as shown in Figure 8: a Process consists of several steps (called activities) which themselves can be solved by a Plan, a SubProcess, a manual activity, or by another Goal that can be solved arbitrarily.

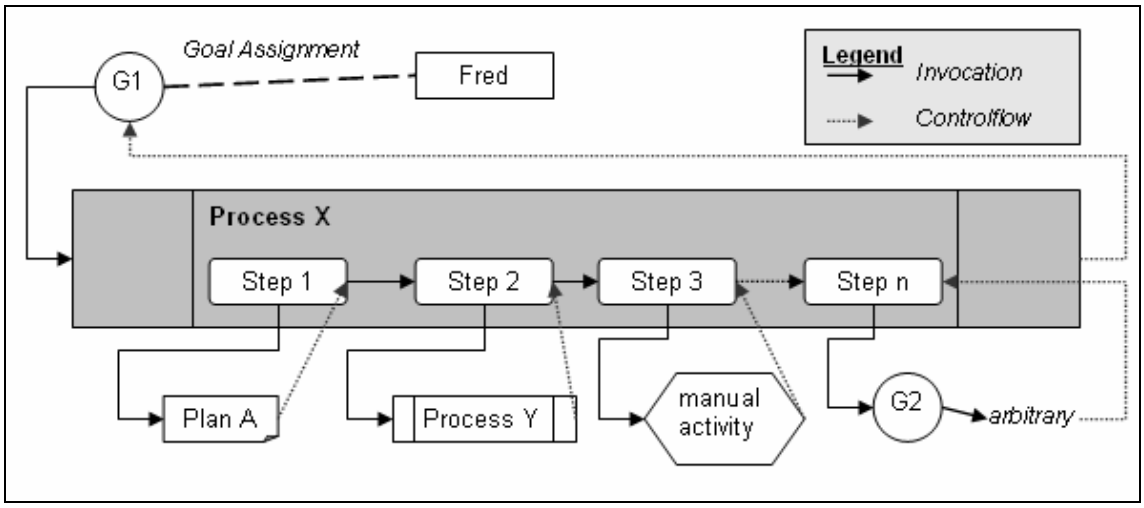

Figure 8: SWF Process Model

In the SWF Service Model, stateful services will be processes which request for functionality (or cooperation, respectively) independent of the SWF Service Model. For the other Service Types in SWF (Plans and external Web Services, executed via WSDLExecutorPlan) requests for are located inside the Service, but this is not of interest for the SWF Service Model.

The Service Interface part of the SWF Service Description Language will be worked out in SWF D3. We will align and adapt to the specification of Choreography in WSMO, see WSMO Deliverable D14 at http://www.wsmo.org/2004/d14/v01/.

3.1.1.3.2.4 Grounding

The Service Grounding describes the input and output and the technical accessibility of Services – needed for actual access to an existing Service. In SWF, the Service Grounding refers to a specific type of SWF Service as defined above.

The Service Grounding has to provide the following information:

- **Service Access & Binding**, i.e. a link to the physical location and the protocol for accessing a particular Service. This is highly related to the Type of Service (see below).
- **Errors Messages:** messages returned when execution of a Service fails. This has to be conformant to a Error Handling Framework.
- **Compensation Information**: information for finding an alternative execution path that supplies the same functionality. Design Guidelins for a Compensation Franework are already existent in the FRED system.

In the end, the Service Grounding information might be incorporated in the Serivce Interface, conforming to WSMO. But conceptually, we have to separate this.

3.1.1.3.2.5 Non-functional properties

The Non-functional Properties of Services are used for selection of service out of result set from Service Discovery. Although these properties are not important for computational execution of a Service, they are the basis for advanced Service Discovery in order to determine the usability of the set of Services returned by the discovery mechanism. Also, the non-functional Properties of Services determine the quality of the Goal resolution, for instance that a request must be satisfied within a certain time.

Important Non-functional Properties of Service Descriptions are (not complete, refers to additional Non-functional Properties of Web Services as defined in WSMO):

- **Performance:** represents how fast a service request can be completed. Performance can be measured in terms of throughput, latency, execution time, and transaction time. The response time of a service can also be a measure of the performance. High quality web services should provide higher throughput, lower latency, lower execution time, faster transaction time and faster response time.
- **Scalability:** represents the ability of a service to process more requests in a certain time interval. It can be measured by the number of solved requests in a certain time interval.
- **Robustness:** represents the ability of a service to function correctly in the presence of incomplete or invalid inputs. It can be measured by the number of incomplete or invalid inputs for which the service still function correctly.
- **Accuracy:** represents the error rate generated by the web service. It can be measured by the numbers of errors generated in a certain time interval.

The measurement techniques for these properties will be adopted from WSMO. This aspect is not a core issue of the SWF project, and we will adopt emerging technologies for this from WSMO or other initiatives.

3.1.1.3.2.6 Type of Service

Defines the type of Service which realizes the Service. In order to execute them, further descriptive information for the different types of Services is needed (might be included in the Serivce Grounding, see above):

- **Plan:** a link to the computational resource needed (e.g. a JVM for Java programs).
- **Process:** invocation of the Process Instance.
- **Web Service:** link to the WSDLExecutorPlan.

3.1.1.4 Repositories

We need several repositories for storing, retrieving, and managing the components of the SWF technology. As shown in Figure 4, these are:

3.1.1.4.1 Ontology Repository

This holds the Smart Objects of the ontologies – both the domain ontologies and the system ontologies. The existent SMO technology is sufficient for this.

3.1.1.4.2 Goal & Goal Instance Repository

The Goal Repository holds Goal Schemas as well as the Goal Instances. The structure can be derived from the representation of ontologies in common ontology tools: the Goal Schemas define the grounding structure and the Goal Instances are assigned to their corresponding Goal Schema. The Goal Repository should be structured on basis of a domain ontology in order to allow users to select an appropriate Goal Schema for creating a Goal Instance. Here, users can be: system developers, users that want to assign a task to a Fred, or Freds or Processes that want to create a Goal Instance during runtime.

A possible future extension is to structure Goal Schemas hierarchically, meaning that a complex Goal is decomposed into Subgoals. These Subgoals are autonomous Goals themselves which are assigned as a Subgoal to the Supergoal. There is no sequence defined, but the Subgoals define that certain Goals have to be achieved in order to solve the Supergoal, without any prescription on the sequence. This would be a solution for semi-automated goal decomposition in order to support a broader variety of Goals solvable by the Service existent in the system, following the approach specified in the field of Problem Solving Methods (Fensel et al., 2003).

3.1.1.4.3 Service Repository

The Service Repository is a Registry that holds pointes to the descriptions of all Services available in the System. Services in SWF are only used by machines, so the Service Repository does not have to be human-friendly.

The SWF Service Repository will be a registry, i.e. only holding links to the actual Services with their descriptions. The general approach is to take existing registry technologies, namely UDDI (Bellwood et al., 2003) and the Web Service Inspection Language WSIL (Ballinger et al., 2001), and enhance these with semantic processing facilities. The Service Registry will consist of the following components:

- A data model, in coherence with the SWF Service Model
- A store/publish API
- A find/query API
- A subscription and notification API
- A maintenance API (delete, modify, etc.)

The Service Repository is used by the following SWF components:

- GS Discovery Mechanism: poses a query on the Service Capabilities, and retrieves a set of matching Services. The selection of the Service to be actually used for resolving a Goal is determined either by advanced a GG Discovery (using Non-functional Properties of the Services) or by the Fred that has thrown the respective Goal Instance. See section 3.1.2.2.1.
- WW Discovery: takes the Service Interfaces of the Service of potential coolaboration partners and determines compatibility of the Serivces for automated interaction, ending up in a Contract for Cooperation, see section 3.1.2.2.3.
- Execution Environment: uses the Service Grounding information to invoke and execute the Service selected for Goal Resolution, with regard to the Type of the particular Service. Also, the error and compensation information of the Service Groundings are used to handle errors that occur during runtime and to determine possible different ways for Goal Resolution if a Service is not available.

The Service Repository also holds Mediators as a Mediator is a special type of Service.

3.1.1.4.4 Agent Repository

In addition to the repositories needed of the SWF components, we also need a repository for Freds. Therefore, we apply the existing FredBase technology as existing in the FRED system.

3.1.2 SWF Methods and Mechanisms

In this section we identify the mechanisms needed for SWF Cooperation technology and specify the methods needed on a conceptual level. We first outline the architecture and overall workflow and then specify the particular Discovery Mechanisms.

3.1.2.1 Cooperative Goal Solving

As outlined introductory, 3 different Discovery Mechanisms are needed for the SWF Cooperation technology. Figure 9 shows the overall workflow of these mechanisms with further explanations below.

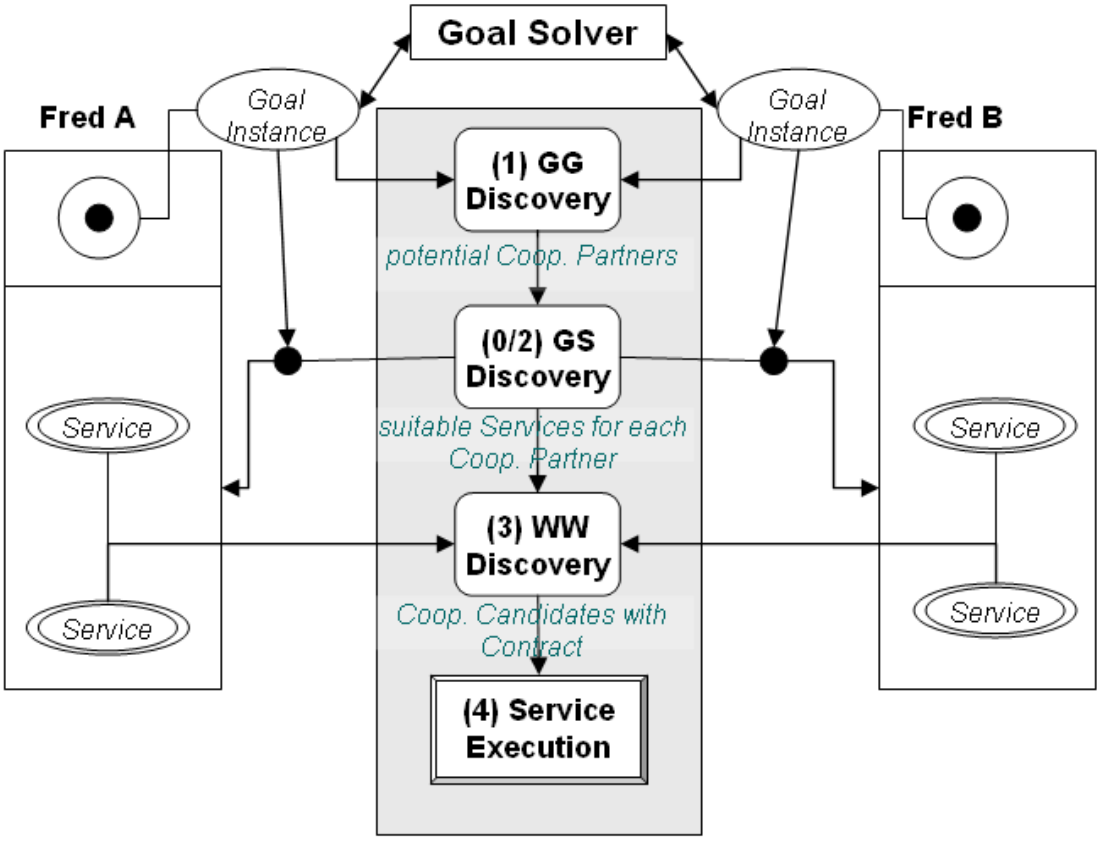

Figure 9: SWF Cooperation Workflow

For establishing a Cooperation, the first step is to detect potential Collaboration Partners. This is done by matching Cooperative Goal Instance that is thrown by several Freds during system runtime, called GG Discovery (Goal Instance to Goal Instance matching). A Goal Instance can be created any time by a Fred, and the GG Discovery Mechanism implements different strategies to determine potential Collaboration Partners. The Cooperation workflow depicted in Figure 9 starts when the GG Discovery has determined potential partners. After this, the second step is to determine the Services needed by each Partner to carry out the Cooperation automatically. This is done by the GS Discovery (Goal Instance to Service) that matches the Cooperative Goals (which are the same as for GG Discovery) with Service Descriptions, returning a set of suitable Services for each potential Cooperation Partner. GS Discovery can be performed at two different points in time: when a Fred creates a Goal Instance, i.e. independently of a concrete Cooperation, and / or when the GG Discovery has determined a potential Cooperation Partner. In conjunction with GS Discovery, the so-called Resource Resolution mechanism determines additional partners for a Meeting: if

a partner needs a certain Service for a Cooperation that he cannot provide by himself, another Fred that can provide the requested Service is called for the Meeting (further explanation on this in section 3.1.2.2.4). When each partner has detected suitable Services, the third step is to determine the compatibility of the Services. This is done by the WW Discovery (Service to Service matching), which investigates the external visible behaviours of the Services of each partner (i.e. a Choreography match) and determines which of the suitable Services detected in GS Discovery are compatible and thus can be used for automated Cooperation. Out of this, a Contract for Cooperation is created which specifies the Services to be used by each partner and the messages exchange sequence, as well as further aspects that need to be clarified.

When these 3 Discovery Mechanisms are completed, the Cooperation Partners are candidates for a Meeting. This means that a Meeting, wherein the Cooperation is actually performed, is instantiated and put on the Meeting List. The Meeting List is managed and controlled by the Cooperation Environment, which ensures that meetings have a minimal duration and that the execution resources for a meeting are available. The Cooperation is actually carried out in a Meeting, according to the Contract defined before. Thereby, the Execution Environment handles Invocation of the Service, checks possibly violated conditions (e.g. pre-conditions of Services), invokes Mediation Services when needed, and handles errors that might occur as well as compensation when needed and specified.

When the Meeting has been completed successfully, the Cooperation is finished and the Goal Instances of the participating Freds are satisfied – at least parts of the Goals since it might take several meetings to solve a Goal. This is controlled by the Goal Solver (see section 3.2.4). After a successful meeting, the Freds are withdrawn from the runtime environment (if they are not participating in aother cooperation), and the Freds will proceed with their activity list, acting autonomously on behalf of his owner.

3.1.2.1.1 The iterative discovery mechanism

The workflow described above is for a single Cooperation wherein everything “works”, i.e. the Discovery Mechanisms always detects suitable partners / services / contracts. In a real-world setting, the distinct steps work iteratively and concurrently. A specific Fred might be engaged in several potential Cooperation proposals, and GS-Discovery, Resource Resolution and WW Discovery work iteratively until a “fitting cooperation pair” is found.

The control of this is handled by both the Meeting List Manager (managing meetings, see section 3.2.4), and the Selection Engine (permanent search for possible

collaboration partners, see section 3.1.2.2.1). These functionalities already exist in the FRED system and will be adopted for SWF.

3.1.2.1.2 Mediation

In general, the resources applied by Services and Goals can be heterogeneous – especially when we use external Web Services and when we want to expand the SWF technology to a Semantic Web Environment. Thus, we need Mediation to resolve possibly occurring heterogeneities.

The idea of Mediation – at least to our understanding – is that there is a higher-leveled description technique for describing the structure of resources. Upon this works an mechanism that checks the resources to be integrated (i.e. to be made interoperable) according to their structure, then it provides mapping functionalities in order to make the resources interoperable. The basis of such mechanism first an exhaustive declarative description technique that allows describing all features of the resources (i.e. a suitable ontology language for ontologies, and a suitable process description language for business processes) and, on top of this, an algebra which allows to describe computable relations between the modelling primitives (Papakonstantinou et al, 1996).

Mediation is required on different levels, according to (Fensel and Bussler, 2002):

- **Data Level:** establishing interoperability between heterogeneous data sources. In SWF as an ontology-based system, the mediation facility is Ontology Integration.
- **Process Level:** establishing interoperability between heterogeneous processes. This means that in the external visible behavior of Web Services that are ought to interact, there are some mismatches in the sequence of activities. These have to be resolved in order to make them interoperable. In SWF, process mediation is not needed directly.
- **Protocol Level:** establishing interoperability between resources that request / use heterogeneous messaging patterns. Currently, the compatibility of Freds regarding protocol level mediation is ensured by the FIPA-agent protocols

A Mediator in SWF is understood as a special type of Service that establishes interoperability between potentially heterogeneous resources. The general approach for Mediators in SWF is that all resources are described by explicitly specified Description Languages. For establishing interoperability between 2 components, the types of mismatches are analyzed and mappings are created between the components that resolve the mismatches – if the mediation facility is able to resolve them. SWF follows

the WSMO concept of Mediators explained above (see section 2.2.2.4), applying the different types of Mediators in the different types of Cooperation Mechanisms.

In order to allow automated execution of Services, they have to be **"perfectly mediated"**. This means that resources applied for automated execution of Services in meeting between Freds in SWF have to be interoperable regarding all levels of mediation / possible heterogeneity. The concept of the Meeting Room as the point for establishing meeting only between "perfect mediated Freds" is the basis of the SWF Mediation Framework.

The idea of "Mediators" and a "Mediation Architecture" has first been announced in the 12-year-old article (Wiederhold, 1992). The idea is that a mediator resolves mismatches during runtime of the system and not wraps resources before they are used in a system. This is somewhere in literature named as the "next generation integration systems" because they provide mediation facilities based on higher-levelled techniques and thus allow integration (i.e. making things interoperable) without regard to applications contexts, thus they can be used universally. Thus, SWF implements a Mediation Architecture by the concept of Meeting Room concept that ensures that Freds that are called for a Meeting have perfectly mediated resources.

Elaboration of Mediation Mechanisms is not aim of the SWF project. But as Mediation is an important aspect of systems like SWF that deal with possibly heterogeneous resources, the integration of Mediation Mechanisms has to be considered in the framework and the overall architecture of SWF. There are already some Mediation facilities in the FRED system (e.g. Mapping and Morphing) – further Mediation Facilities will be adopted from external development efforts.

3.1.2.2 Discovery

In the following we describe the different Discovery Mechanisms needed in SWF. In general, the Discovery Mechanisms are implemented as algorithms that work on the component descriptions and determine a set of suitable partners, Services or a contract. After the Discovery Mechanisms are completed, the Freds are called as candidates for a Meeting. The execution of the Services and all runtime resolutions are located in the Execution Environment described in section 3.2.

The most reasonable technology for the SWF Resolution Mechanisms is logic-based inference machines because of their suitability for resolution tasks like this. At least for the core resolution functionalities, i.e. matching of declarative descriptions for Meeting Creation and Service Discovery, logic-based inference mechanisms seem to be the

appropriate choice – the surrounding control should be realized by "conventional" technologies.

The reasoning technologies to be applied in SWF is planned to be the WSMO Reasoner. The primary aim of the WSMO Reasoner, at least at this stage of WSMO development, is to provide a tool that allows determining whether a WSMO Goal Description and a WSMO Web Service Capability match. The input is a Goal and the link to a Service Repository to be searched, and the output is the set of Service Descriptions that match the Goal. Therefore, Goal and Service Description have to be defined in F-Logic or at least in a subset of F-Logic in order to guarantee computability.[6]

It is not clear yet if the WSMO Reasoner will meet the industrial strength requirements of SWF, thus it is necessary to evaluate alternative technologies for Goal – Capability matching. This will be an ongoing task during the next month, and we might also have some BA-students working in this area as well as the support of Uwe Keller who is developing the WSMO Reasoner. The choice of the Reasoning technology and the concrete specification of the Discovery Mechanisms will be provided in SWF Deliverable D4 "SWF Mechanism and Tools".

For specifying the functionality of the particular mechamisms, we apply to the following specification template:

1) **Matching.** What the Mechanism does, i.e. what parts of SWF components description it works on
2) **Input and Output.** What the input and the output of the Mechanism is
3) **Invocation.** When and how it is invoked

   Additionally, we discuss specific aspects for each Discovery Mechanism, specify the type of WSMO Mediators required, and possible extensions for future development.

3.1.2.2.1 Cooperative Goal to Goal Discovery (GG Discovery)

GG Discovery is the detection of potential cooperation partners by matching Cooperative Goals, thus determining Compatibility of Cooperative Goals.

According to the description template defined above, the GG Discovery works like this.

[6] The WSMO Reasoner is still under developed, the progress can be found in WSMO Deliverable D5.x at www.wsmo.org/2004/. A first version of Deliverable D5.1 about the theoretical requirements on proof obligations can be found in (Keller, WSMO D5.1).

1) **Matching.** matches the postconditions and effects of Goals, determining whether the object of interest is the same (i.e., if the desired states of the information space of Cooperative Goals are compatible), and if the cooperation roles of the Goal owners are compatible.

2) **Input and Output.** Gets 1 Goal Instance as input and returns a set of potential cooperation partners.With each potential partner that is deteceted, the Goal Instance Owner starts a negotiation for Cooperation, following the steps outlined above.

3) **Invocation.** GG Discovery is initiated when a Goal Instance is created by any Fred in the system. Thereby 3 types of engines are used for GG Discovery (according to the already existing Meeting Creation Mechanisms in the FRED system):

   - permanent search: an engine that permanently seraches the system / the Goal Repository for potential partners
   - event driven: an engine that starts GG Discovery when a specific event (internal or external) occurs
   - application context driven: starts GG Discovery when anadditional partner is needed for a Cooperation, i.e. when a Service defines a Goal during Serive Execution (mostly within Processes, s.a.)

Figure 10 shows the GG Discovery Mechanism comprehensively.

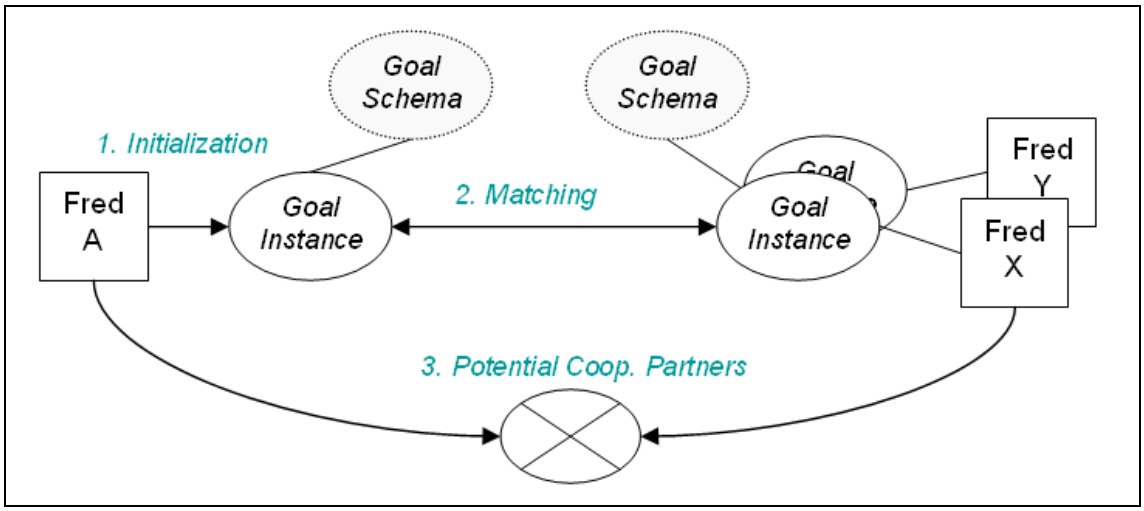

Figure 10: GG Discovery Mechanism

3.1.2.2.1.1 GG-Discovery with Complete and Partial Cooperative Goals

For matching Cooperative Goals, we need to have knowledge about the cooperation roles of the Goal Instance Owner. As discussed above, we distinguish completely described Cooperative Goals (the Goal is specified with regard to the perspectives of both sides for cooperation), and partially described Goals wherein only the perspective of the Goal Instance owner is defined (see Figure 5). The 2 types of Cooperative Goals require slightly different GG Discovery:

- **GG Discovery for Complete Goals:** This means that both parties describe their Goal completely. The GG Discovery Mechanism is executed "normally".
- **GG Discovery for Partial Goals**: This means that at least one party describes its Goal only from his perspective. GG Discovery can be performed on partial Goals as well (i.e. determining compatibility wrt object of interest and cooperation role). After detection of potential partners, their Partial Goals have to be composed into a Complete Goal in order to support the subsequent Cooperation Mechanisms.

3.1.2.2.1.2 Usage of WSMO GG Mediator

For resolving heterogeneity, we have to use a Mediator equivalent to a WSMO GG Mediator. A GG Mediator resolves conceptual mismatches by reduction, i.e. weakening of the postconditions / effects or the cooperation role of Goals in order to make them compatible. The GG-Mediator in SWF reduces the Goal Descriptions wrt to the object of interest and the cooperation role, if necessary. Figure 11 shows the GG Discovery mechanism with usage of a GG Mediator.

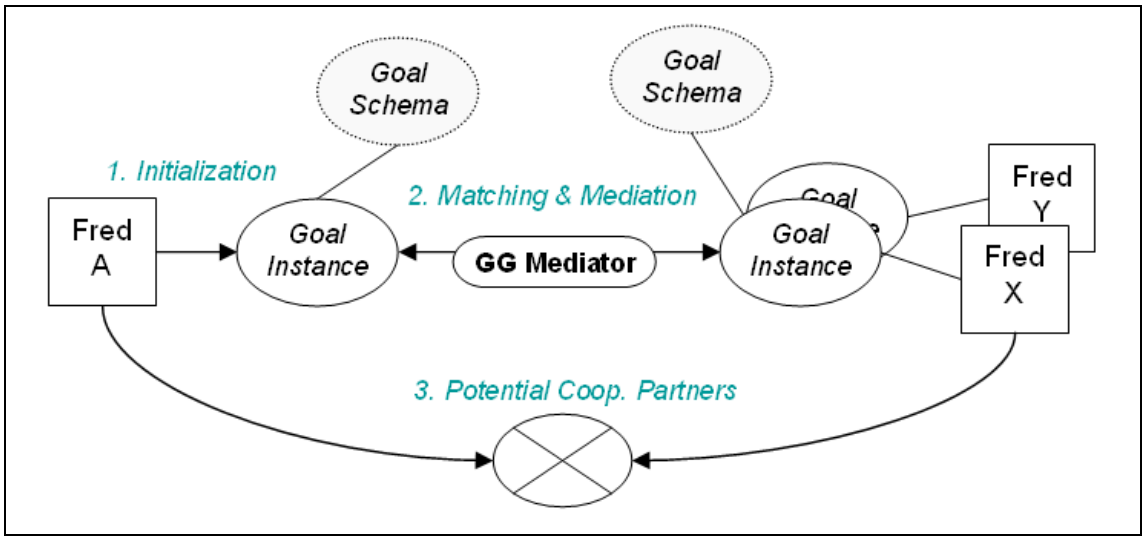

Figure 11: GG Discovery Mechanism with Mediation

3.1.2.2.1.3 Possible Extensions

- perform a pre-GG Discovery on Goal Schemas without regard to Goal Instances, i.e. independent of concrete requests for cooperation.
- provide additional information in Goal Schema / Instance descriptions to increase the quality of GG Discovery (e.g. non-functional Properties for Cooperation Requests, etc.)
- increase matching rate by surrounding constructs (the conditions for matching success should be very weak in order to facilitate Cooperation)

3.1.2.2.2 Cooperative Goal to Service discovery (GS Discovery)

GS Discovery is the detection of suitable Services that a party has to provide in a Cooperation in order to resolve his Goal. GS Discovery is the second step after GG Discovery and takes place at each potential Cooperation Partner. The core of Service Discovery is matching of Goals and Service Capabilities, more precisely querying the Service Repository for Service Capability descriptions which match the postconditions and effects of the given Cooperative Goal Instances of a Partner. The Services detect have to be executable by the Owner of the Goal Instance. In fact, GS Discovery is equivalent to what is called "Discovery" within Semantic Web Services.

According to the description template defined above, the GS Discovery works like this.

1) **Matching.** matches the Goals and Service Capabilities. A suitable Service has to be able so solve the Goal according to the object of interest (i.e. its Capability must be able to satisfy the Goal), and the Service has to support the cooperation role of respective partner (i.e., for a cooperation role "Buyer", a payment Service has to be able to "pay with a payment method", and for a "Seller" the payment Service has to be able to "perform payment for a product").
2) **Input and Output.** GS Discovery is performed for each partner of a potential Cooperation. It takes the Goal Instance that has been the starting point for GG Discovery as input and returns a set of suiatble Services for each partner, possible several usable Services for one Capability (the actually used Service is then detected by WW Discovery, see below). The Goal Instances have to be Complete Coopertative Goals, so that each partner will have all Services that he needs for Cooperation. These Services have to be executable by the respective partner, meaning that they either have to be Services by the Partner himself or the partner has to have the rights and ability to use that Service.

3) **Invocation.** GS Discovery is invoked by the Cooperation Environment as the second step after GG Discovery. It can also be invoked iteratively in correspondence to Resource Resolution or WW Discovery when the detected Services do not have compatible Service Interfaces.

Figure 12 shows the GG Discovery Mechanism comprehensively.

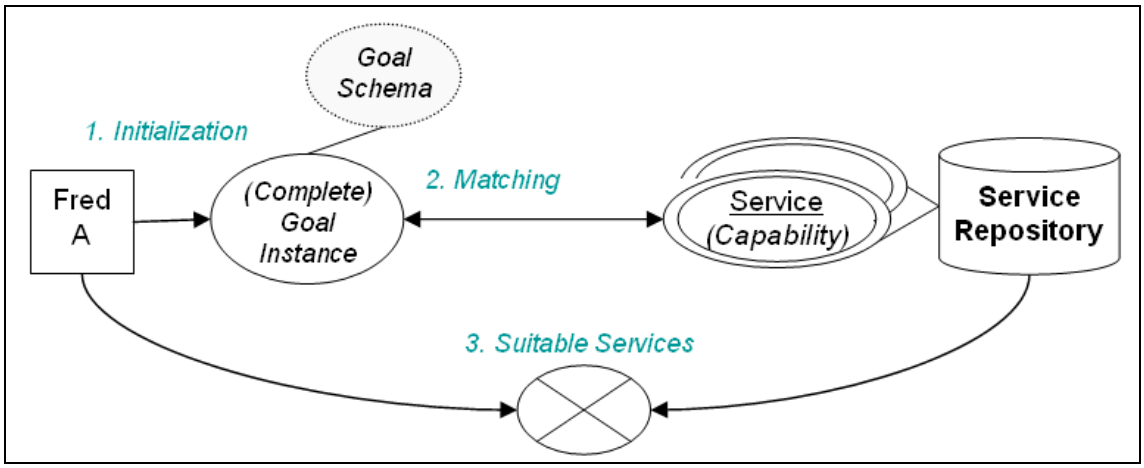

Figure 12: GS Discovery Mechanism

3.1.2.2.2.1 Levels of GS Discovery

We differentiate the following levels for Goal-Capability matching.

1. Simple static linking: the Resolution is specified, i.e. a hard-wired link defines the suitable Services for a Goal Resolution.
2. Simple linking: dynamic detection of suitable Services that have a simple behaviour, i.e. Plans or Web Services (if used via the WSDLExecutorPlan).
3. Complex static linking: same as 2., but Services have complex behaviour, i.e. Processes or Web Services (if used with more complex techniques than the WSDLExecutorPlan).
4. Complex dynamic linking: dynamic detection of suitable Services, that have a complex behaviour and compose them into the Service needed.

The algorithms for the different GS-Discovery Mechanisms will be specified in SWF Deliverable D 4 “SWF Mechanism and Tools”. Additional, the SWF Service Discovery Mechanism has to provide the GS Discovery Results in adequate manner for further processing by the subsequent mechanisms.

3.1.2.2.2.2 Usage of WSMO WG Mediator

Similar to GG Discovery, compatibility of Goals and Service Capabilities might be established by weakening specific conditions. For this, we apply a WSMO WG mediator that provides the needed reduction. Figure 13 shows the GS Discovery with an WG Mediator.

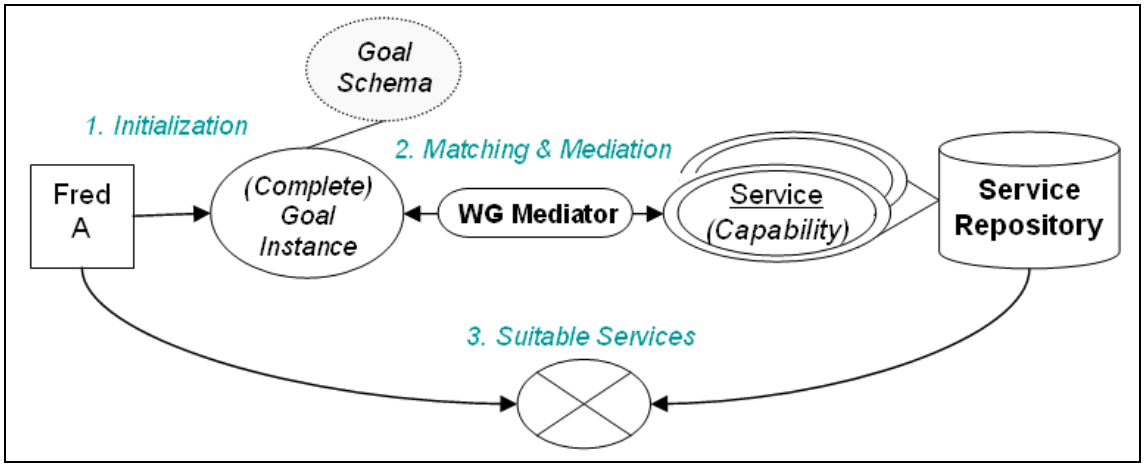

Figure 13: GS Discovery Mechanism with Mediation

3.1.2.2.2.3 Possible Extensions

- apply non-functional Properties for advanced quality GS Discovery
- pre-GS Discovery: when a Fred gets created (i.e. pre-select usable Services) or when a Goal Instance is created, but before Cooperation Workflow starts

3.1.2.2.3 Cooperative Service to Service discovery (WW Discovery)

WW Discovery is the detection of the Services that will actually be used for automated Collaboration. Therefore, the Services detected for each Partner by the GS Discovery are checked according to their Interfaces, determining pairs of Services that have compatible Service Interfaces. If WW Discovery is successful, the determined Services, their execution sequence, and the messaging sequence are delineated in the Contract for Cooperation (see section 3.2.2).

According to the description template defined above, the GS Discovery works like this.

1) **Matching.** matches Services Interfaces of the suitable Services detected by GS Discovery. This includes determination of compatible Message Sequences as well as the compatibility of the external visible Business Processes of Services.

2) **Input and Output.** takes the Services of each Partner as detected by GS Discovery and returns a set of compatible Services. The GS Discovery possibly can return several Services for one specific activity in the external visible business process of the overall Service of a Fred. WW Discovery selects one of the set of suitable Services that will be used for Cooperation.

3) **Invocation.** WW Discovery is invoked by the Cooperation Environment as the third step after GS Discovery. It can also be invoked iteratively in correspondence to Resource Resolution or GS Discovery when none of the Services do have compatible Service Interfaces.

Figure 14 shows the WW Discovery Mechanism comprehensively. This figure assumes that both Cooperation Partners use Services of the type "Process" which has predefined activities within data and control flow. The blue, dotted lines denote activity start / end messages, and the black arrows denote the message flow. A further assumption is that each Cooperation Partner always has 1 Service that is comprised of all the functionality that a Partner needs for a Cooperation. This can be a concrete Service (a Plan, Process, or external Web Service), or a composition of different Services existent in the system.

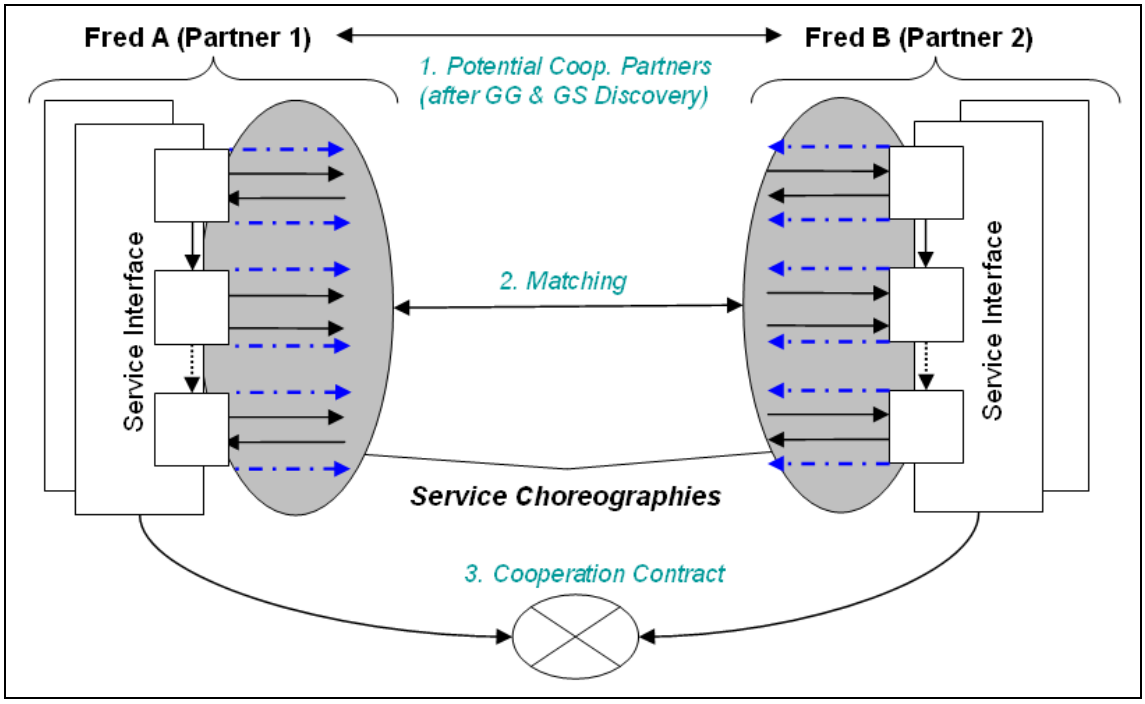

Figure 14: WW Discovery Mechanism

3.1.2.2.3.1 Choreography Matching: Messaging, Behavior and Dynamic Creation

The techniques needed for WW Discovery seem to be quite unexplored at this point in time. Here, we list what we need and the initial ideas how it could work.

- **Messaging Compatibility Check.** This is concerned with determining whether 2 Services can cooperate according to their message sequences. Therefore, the following aspects have to be checked:
  - Compatibility according to the Message Exchange Pattern (MEP): a MEP is a application context independent definition of the semantics of Message Types, i.e. differentiation of a REQUEST, CONFIRMATION, etc. For example, the FIPA-ACL primitives are an MEP, see: http://www.fipa.org/specifications/index.html. For automated cooperation, the types of the messages have to be compatible according to the MEP of the Service Interfaces.
  - Compatibility of the Message Sequences: this is concerned with the actual message exchange between the Services. The sequences have to compatible, meaning that the messages send out by Service *A* have to be the messages that the partner Service *B* expects as incoming messages in the right sequence.
- **Behavior Compatibility Check.** A Choreography does not only specify the Messaging, but also the external visible business processes of Services. These can to compatible as well for automated Cooperation. Depending the Service Interface specification, the technique for Process / Activity Compatibility Check has to determine whether the activities of the Services of the cooperation Partners are compatible, resulting in a cooperation business process specification that will be part of the Contract for Cooperation. In fact, Process Mediation comes into play here.
- **Dynamic Choreography Creation.** In order to facilitate Cooperation, it should be possible that one of the Partners adopts his Choreography to the partner's Choreography. This means that a Choreography is created on-the-fly during runtime, ensuring Compatibility by an opposite or collateral means than Mediation.

3.1.2.2.3.2 Usage of WSMO WW Mediator

As within the other Discovery Mechanisms, the behaviors of Services or the resources they use can be heterogenous. For resolution, we apply a WSMO WW Mediator which resolves possibly occuring mismatches.

The general structure of WW Discovery with a WW Mediator is shown in Figure 15.

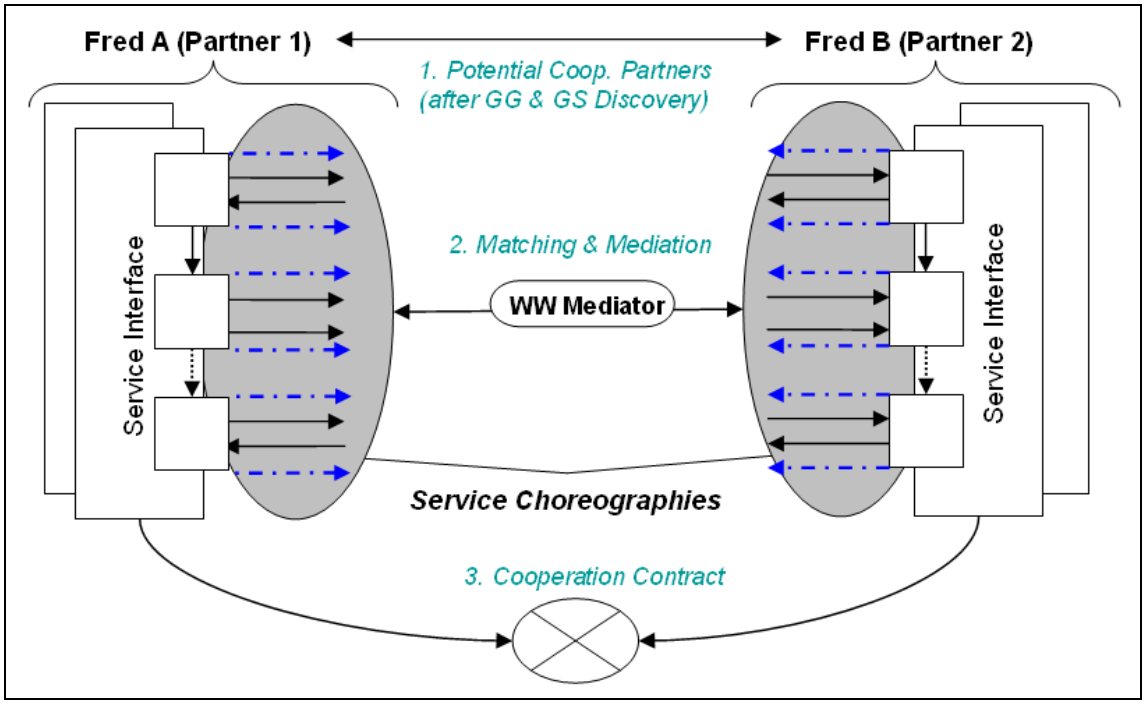

Figure 15: WW Discovery with Mediation

The different types of mismatches can concern any of the mediation levels mentioned above (see section 3.1.2.1.2):

- **Data Level:** i.e. different ontologies, resolved by Ontology Integration techniques / OO Mediators
- **Process Level:** process mediation results in compatibility of the Services accoring to their external visible behaviors, see above
- **Protocol Level:** this can be either on the MEP – therefore a mapping between different MEPs is needed – or on the Message Sequence, resolvable by message sequence adaptation.

3.1.2.2.3.3 Possible Extensions

The WW Discovery is very complex and not too much related work exists here. Therefore we follow a stepwise development of this mechanism in SWF:

1) Specification of Service Interface Description (see section 3.1.1.3.2.3)
2) Specification & Development of Core Features (Messaging and Behavior Compatibility Check)
3) Enhancements:

- Core Functionalities
- On-the-fly Choreography Creation
- Mediation Facilities

The further specification of WW Discovery is highly related to the specification of the Service Interface Descriptions in SWF, and thus strongly related to the Web Service Interface description specification in WSMO, see (Roman et al., 2004b)

3.1.2.2.4 Resource Resolution

To resolve a Goal, not only the service itself has to be discovered, but also the Resource that will provide or perform the Service has to be resolved. Similar to Service resolution, the Resource may already be specified in advance (it is part of the Goal that a Service has to be provided by a certain Resource) or it may be resolved during Discovery.

In Fred, Resource resolution is always about resolving a Fred. A Fred may represent himself, a human resource or even a group of resources like an organizational unit. In the later two cases Fred acts as a proxy.

In principal there are two ways to get to the Resource:

3.1.2.2.4.1 Goal -> Service -> Resource

In this case, first the Services for a Goal are resolved and then the Resource resolution is performed on the result. This approach is preferred if the Resource is not specified, so all potential providers of a matching Service are retrieved and the Discovery can select the best matching Service independent of the Resource. When there is a list of Services that equally match the Goals' conditions, one Services can be issued after the other until the Goal is satisfied (e.g. buy 100 chairs, first Service can only deliver 20, next 50 and so on).

3.1.2.2.4.2 Goal->Resource->Service

In this approach, first all Services for a given Resource are resolved. This is a better approach when the Resource is already specified and different Services have to be resolved for the given Resource.

In many cases the Resource will not be specified in a Goal, at least not in the first step, but subsequent steps may need to involve the same Resource as the first step did.

3.1.2.3 Web Service Delegation

When a Service has been discovered, it implicitly has a Resource that is responsible for Service Execution (e.g the Provider).There are cases that the Provider may want to delegate the Service to another party (e.g. delivery of a goods outsourced to another company) by invoking other Services for his own functionality. If a Service has been delegated is especially important to know when the next step in a Process must be executed together with the Resource the Service was delegated to.

The usage of other Services might be hidden to users or it might be explicitly stated as a special feature of the Service (e.g. "I use Service X which has the trust-certificate YZ"). Such explicitly stated Service Delegation can also be used in Goals to explicitly allow or deny the delegation of a Service. This is a step towards trust management in Service Oriented Architectures (Erl, 2004).

3.2 Execution Environment

For execution of the Services determined for an Automated Cooperation by the discovery mechanisms, an execution environment is needed. This has to hold the following functions:

- Goal resolution mechanism
- Service Discovery Mechanism
- Repository for Freds, Plans, Processes and Resources
- Service invocation
- Execution environment to execute Plans, Processes and a proxy for external Web Services that
  - manages and controls cooperative execution of services (Meetings)
  - watches and handles protocol violations during cooperation
  - provides transaction security
  - handles Errors and some Compensation during runtime
  - provides and invokes necessary Mediators if required

In general, the Execution Environment executes the Services as defined in the Contract., the result of the WW Discovery, and it manages Meetings as well as the Freds in the system. All runtime-related issues of Service Execution and Goal Resolution are handled here, except execution of external Web Services (they are only invoked here). If Service Execution fails for whatever reason, the Meeting between the Freds is postponed and the respective Cooperation Mechanisms are called again.

The SWF Service Execution Environment holds the computational resources for execution of the different types of Services. Most of the technologies needed for the Cooperation Execution Environment are already existent in the FRED system Thus, it is not a completely new developed environment, but it is a reorganization of the Service processing facilities.

3.2.1 Invocation

Invocation in SWF covers 3 different aspects:

1. **Request for Cooperation, i.e. Creation of a Cooperative Goal Instance.** For a given Goal, matching cooperative Goals and suitable services for the Goals are discovered and a Meeting (specified by the goals and services) is scheduled.
2. **Invocation of the Services for Execution.** The created Meeting will be executed by invoking all involved services. Execution performs according to the underlying Contract of the services.
3. **Cooperarion of Service Invocation during runtime.** When, during execution of a Meeting, a Cooperation is requested (i.e. a Cooperative Goal Instance is thrown by a Service) or another Service is invoked (e.g. a Process Activity invokes a Plan).

3.2.2 Contract

The Contract for Cooperation holds all information that Cooperation Candidates have agreed on, which basically is the result of the Discovery Mechanisms. The content of the Contract are:

- the Cooperation Partners
- the Cooperative Goal Instances
- the Services to be used (i.e. the result of WW Discovery, incl. the MEP, Messaging Sequence, Business Process for Cooperation)
- Mediators if needed
- Further information id needed

3.2.3 Error Handling

This is concerned with errors and exceptions that might occur during the execution of Services, i.e. with error handling and with compensation when a service is not available. The Error Handling Mechanism monitors the execution of Services and handles all

kinds of errors that occur during runtime, i.e. during execution of Services. Service Execution Mechanisms deal with Error Handling, Compensation, and Transaction Control for multi-step or composed services. This is especially important in Service Orientated Architectures with autonomous services that interact.

In SWF, the Service Grounding Description holds information of error messages and possible compensation activities. The Service Execution Mechanism monitors the execution of Services during runtime and delegates occurring errors, etc. to the respective handling techniques. There are already techniques exiting for this in the FRED system, and the development of further Error Handling technologies is not part of the SWF Framework.

3.2.3.1 Error Framework

A framework that specifies types of error that can occur and general ways for handling them. In general, there are 2 types of errors that might occur:

1. **Missing Information.** some input data required by the Service is not availlable. Either the additional information is provided by the user / a cooperation partner during runtime, or the execution fails
2. **Service Failure.** The Service can not be executed because of any kind of runtime exception (not accessible, internal failure in Service, ect.). Either an alternative path is provided as compensation information or the execution fails.

Errors of the first type can be resolved by interaction with the user / Fred owner during execution, and errors of the second type are handled by compensation information (see below).

3.2.3.2 Compensation

Techniques for determining alternative execution paths that return the sane functionality when a Service is not executable. This works on the Compensation Information specified in the Grounding Description of a Service: the compensation information specify a Service that can be executed instead of the failed Service and that will return the same result; this alternative path is executed as a normal Service. The definition of the compensation information is provided by the Service provider.

3.2.4 Cooperation Control Unit

This is concerned with management of Meetings and the Goal Resolution Process, as well as the management of Freds in the system. In general, the technologies existing in

the FRED system already provide this functionality and will be slightly adopted for SWF.

3.2.4.1 Meeting Room

A Meeting Room is the place where agents interact, more precisely the environment where Freds called for a meeting exchange information in order to solve the Goals they have been assigned.

A Meeting Room holds the following facilities:

- **Execution Facilities** for the different types of Services (JVM for Plans that are Java Programs, the Process Engine for executing Processes, and the WSDLExecutorPlan for invoking external Services)
- **Meeting List Manager**: manages the meeting list, i.e. scheduling of meetings with regard to priorities and disposability of the cooperation participants.
- **Meeting Room Manager**: manages and controls the execution of meetings, i.e. the execution of the Cooperation as defined in the Contract.

The Meeting Room technologies are already exiting in the FRED system.

3.2.4.2 Goal Solver

The Goal Solver controls the resolution process of Goal Instances and manages the Freds existing in a system.

The control of the Goal Resolution process relies on the processing states of a Cooperative Goal Instance (see section 3.1.1.2.3). For each Goal Instance of each Fred, resolution is controlled separately. The management of Freds is performed by the FredBase Control. It ensures that Freds are withdrawn for the FredBase when they are not needed (i.e. when they are not involved in any ongoing Cooperation), and it withdraws Freds currently not needed for the execution environment has keeps them in a sleeping mode in the Agent Repository. This ensures stableness and scalability of the system.

3.2.5 Mediation Execution

The mediation facilities required for a cooperation are determined within the discovery mechanisms (GG Mediators in GG Discovery, WG Mediators in GS Discovery, and WW Mediators in WW Discovery). Each Mediator calls a third Service into the meeting which provideds the needed mediation facility (defined in the Mediation Service), and

these Services are executed during Service Execution in the same way as other SWF Services.

3.2.6 SWF on different Platforms

In aim of the SWF project is first to enhance the Cooperation technology of the FRED system by more dynamics, and secondly to expand this technology into an Semantic Web Service Environment. The overall functionality in both settings is the same, while only specific parts need to be adjusted for the different environments.

3.2.6.1 FredBase Environment

SWF in the FredBase Environment means that the Execution Environment is located in a Fred system. This means that the "binding" of the SWF technologies is the FRED system, including the following aspects:

- **Cooperation in Fred Meeting Room:** The existing Meeting Room technologies are used as for Service Execution.
- **FIPA Message Exchange:** the Freds use the FIPA-MEP for communication
- **Shared Contract:** the Contract for Cooperation is specified for the FredBase

3.2.6.2 SWF in Semantic Web Environment

When exporting the SWF technology to a Semantic Web Environment (i.e. in an open environment in the Internet), only specific aspects of SWF need to be adopted while the rest of the technology and mechanisms remain the same:

- **Cooperation in Virtual Meeting Room:** for Service Execution, the SWF Meeting Room technologies have to be adopted for Web Service Execution. This is basically requires additional Service Execution facilities for Web Services.
- **Semantic Web Service Message Exchange:** cooperation partners use MEP that are (will be) designed for communication between (Semantic) Web Services.
- **Virtual Shared Contract:** the Contract for Cooperation is specified according to the Service Description and Execution and for (Semantic) Web Services.

This shows that the SWF technology can very easily be adopted as an architecture for Semantic Web Services.

3.3 Alignment of Fred with Semantic Web Service Technologies

The final section of the SWF Architecture specification outlines how it obeys to the design principles and state-of-the-art technologies for Semantic Web Services. We briefly recall the aims of development around Semantic Web Services and explain

The aim of research and development effort around Semantic Web Services is to provide advanced technologies for usage of Web Services. Current development considers especially the following technologies:

- **Automatic Web Service Discovery**: Location of services that abides to the service requester specification for a concrete task
- **Automatic Web Service composition**: Assembly of services based on its functional specifications in order to achieve a given task and provide a higher order of functionality
- **Automatic Web Service execution:** Invocation of a concrete set of ser-vices, arranged in a particular way following programmatic conventions that realizes a given task

Therefore, suiatble and expressive enough Description Techniques for Web Services are required as well as intellingent mechanisms that work on this discriptions. We discuss how these issues are addressed in SWF and point out the major contributions that the SWF project will provide for research and development of Semantic Web Service technologies.

3.3.1 SWS-based Common Service Description Language

The SWF Service Model defines different types of Services: Internal Services (Plans and Processes) and External Web Services. Therefore, a common description language is defined that supports the discovery mechanisms for automated Cooperation and Service Execution (The SWF Description Language for Goals and Services will be elaborated in SWF D3, see This deliverable specifies the overall approach and architecture of SWF and points out the development issues of the project. In the subsequent deliverables, specific technological building blocks will be specified. Those deliverables which introduce WSMO technology to Fred system are listed below.

Table 2). A suitable Description Language is the pre-requiste for all intelligent techniques that are used in SWF, and as well within Semantic Web Services.

The SWF Description Language relies on WSMO with some modifications and extensions. This ensures the suitability of the SWF Description technique. An additional

outcome of the SWF project will feedback and further insights for development of description techniques for Semantic Web Services. Also, the SWF technology and / or parts of it will be provided as Semantic Web Services.

3.3.2 Automated Cooperation Discovery

The SWF Discovery Mechanisms work on different parts of the Descriptions Language for Cooperative Goals and Services as described in section 3.1.2.2. These mechanisms in conjunction with the overall workflow for Automated Cooperation will indeed show utilization of the SWF Description Language as a coherent solution for intelligent mechanisms that realize a goal-driven approach for dynamic and automated service usage along with automated cooperation of agents.

Apart from the fact that there is no other system known that combines agent technology, ontology technology, and goal-driven service usage at this point of time, the SWF project will showcase a running implementation of different discovery mechanisms. At this point in time, there is no widely accepted approach for such mechanisms.

3.3.3 Service Usage

While the GG and GS Discovery mechanisms are concerned with the WHAT (i.e. the functionality of possibly usable Services) the WW Discovery is concerned with the HOW, i.e. the Service Usage. Currently, only SOAP and WSDL are (more or less) widely used for Service Usage, lacking of semantics and thus support for "usage-negotiation" (i.e. can a Service be used by a requester according to interaction behavior).

WSMO works on technologies for enhanced Service Usage, which is covered in the Web Service Interface description (see section 2.2.2.3). We recall the understanding of the most relevant aspects and point out how SWF treats this, showing the contribution to further development of WSMO.

3.3.3.1 Choreography

Choreography in WSMO describes the Interface of a Service with regard to how a user can communicate with a WS. More precisely, this is concerned with the exchange of information (i.e. semantically described data) between 2 parties by messages. A Message is a Communicative Act wherein certain information are passed from a sender to a receiver as part of a Conversation. A Conversation defines a sequence of messages that are needed to fulfill a Cooperation. A Cooperation is an interaction of parties that together fulfill some task or goal.

This means a WS describes his provider-interface as a Choreography. A Choreography is concerned with the messaging sequence (i.e. (1) receive request, (2) acknowledge, (3) send reply, ...), and about states / activities of the external visible behavior of a WS (i.e. a WS that sells a product provides his external visible behavior openly to the world, like: (1) you browse by product catalogue, (2) you select a product for purchase and (3) confirm your purchase intention, (4) you pay, (5) the good is delivered to you). Technologies needed for Choreography are:

1. A semantic description of the messaging sequence (NOTE: this is not an MEP - an MEP is a set of messaging types independent of an application context. For example, FIPA is a MEP. A MEP can be used to determine compatible messages - i.e. a REQ and a ACK, without notion on the actual sequence; a messaging sequence defines which message is when send by whom, thus very dependent of an application context).
2. A semantic description of the states / activities of the external visible behavior of a WS in order to align the external visible business processes of interacting WS. This is concerned with Process technologies and also with Process Mediation.

For WSMO, Choreography is elaborated in WSMO D14.[7] As Choreography is a major issue in SWF (especially in WW Discovery), the SWF project will certainly provide additional insight in the needs and technologies for WS Choreography.

3.3.3.2 Orchestration

In WSMO, Orchestration is concerned with the composition of Web Services that are used / invoked by another Web Service. For the invoked Web Service, the invoker has to determine a proper execution sequence according to the Capabilities and the Interfaces of the invoked Web Services. In order to allow dynamic determination of suitable Web Service invocations by another WS, composition techniques are required. In WSMO, Orchestration is defined in D15 (not available at this point in time).

In SWF, Orchestration is incorporated within the Process technology; see the discussion in section 3.1.1.3.2.3. This technology might be a starting point / contribution for development of more advanced, dynamic technologies for Orchestration.

[7] Choreography in WSMO, current version at: http://www.wsmo.org/2004/d14/v0.1/

3.3.4 Plan Framework version 2.0

The new version of the FredBase Plan Framework will reflect the requirements of combining existing Fred technology with SWF technology. Among the framework extensions are:

- (Semi-)Automatic creation of choreography: a Plan consists of behaviors representing control flow or interaction protocols. Both contribute to the external behavior of the Plan, the Choreography. Tools will support to (semi-)automatically create the choreography as part of the interface description of the Plan.
- Binding with FIPA message channel: a Plan's interaction behavior can be bound to SOAP or other Internet based transport mechanisms, or it can be bound to the internal transport mechanisms used so far for agent communication: the FIPA message channel.
- Management of choreography: for given choreographies the Plan Framework must include mechanisms which make sure that the protocols are obeyed. This is similar to what the Meeting Room Moderator does today, which will be extended as necessary.
- Meeting Room Moderator: today's implementation of the Meeting Room Moderator as some kind of Mediator will be clarified and adjusted to WSMO concepts.

3.3.5 RDF and Smart Objects

With Semantic Web Fred, the FredBase will have to deal with two "styles" of ontological data models and languages: WSMO based and Smart Object based. There are several reasons to stay with both models. The main reason is that Smart Objects got developed for ease of use by application developers, which is a major advantage writing customer applications based on FredBase technology. On the other side discovery services require more powerful representation provided by WSMO technology. Both "styles" can be easily integrated by Ontology mapping.

As a first step a RDF/S to SMO Converter is introduced. In theory this is a typical ontology mapping mechanism; but internally it does not just cover different ontologies, but different ontology languages. The focus of the converter is converting instances according to mapping rules.

Such mechanisms are required for mappings between descriptions of services and goals in new SWF technology and application specific Plans written in Plan Framework using the Ontology API.

4 Use Case Scenario: Buyer – Seller

In the following we outline a use case scenario for SWF that will be used throughout the project for specification and refinement of the SWF components and mechanisms. The use case describes a general Buyer and Seller scenario.

We imagine a Buyer that wants to buy a chair and a Furniture Selling Store that sells different kinds of furniture, including chairs. Both, the Buyer and the Seller, have Freds as electronic representatives that shall fulfill their specific Goals – the Buyer. For describing how the Cooperation takes place within SWF, we first describe the real-world perspective of the use case and then explain the workflow of automated cooperation in SWF.

4.1 Use Case Description

At first, we describe how this cooperation scenario works in the real world. We describe the participants, their Goals and Roles, the Services they have to provide and the how workflow for the Cooperation.

4.1.1 Participants

We identify 2 participants (or general types of parties) for cooperative purchasing:

1) A Buyer: a "normal" customer
2) A Seller: a retail trader that runs a shop

4.1.2 Goals and Roles

For cooperation, each party has a goal and a cooperation role:

1) Buyer:
   - **Goal:** wants to buy a chair with the following properties: wooden, Measures (W x H x D): 50 x 100 x 60, color: light brown, max. price: 150 €
   - **Cooperation Role:** will be a Buyer, i.e. the one who purchases the chair, pays for it, and owns it in the end
2) Seller:
   - **Goal:** wants to sell furniture, offering different kinds of furniture (beds, chairs, cupboards, etc.). Especially, he wants to sell the highest possible amount of his product that is possible.

- **Cooperation Role:** will be a Seller, i.e. the one who disposes the chair, and receives the payment for it

4.1.3 Services

As outlined in the Cooperation Model underlying the design of SWF, each cooperative party has to provide some capabilities (called Services) in order to interact with another party as a partner in a Cooperation. The 2 types of parties in the use case need to have the following Services:

1) Buyer:
   - Ability to express Search Desire
   - Payment Service for paying
   - Ability to receive Good Delivery
2) Seller:
   - A Search Service for available furniture
   - Payment Service for excepting payment by different payment methods
   - Good Delivery Service

4.1.4 Cooperation

The cooperation between the Buyer and the Seller for purchase would abide the following sequence:

1. The Cooperation starts by expressing the Purchase Intention, i.e. is initiated by the Buyer. For the Cooperation to start, the Seller's shop ha to be open (whatever kind of customer service is used – a "real" shop where the customer walks in, a telephone center for taking purchase orders, or an Internet Online Shopping Services, etc.)
2. The Buyer expresses his purchase desire, and the Seller offers his product list. Then, the product list is checked on whether the desired chair is available
3. Assuming that the chair is available, the Buyer decides if he wants to buy the chair. If yes, then the two partners sign a purchase contract on the chair, incl. payment and delivery method. In our use case, the Buyer says that he wants to pay after delivery, and the Seller accepts several business models for this, incl. payment after delivery.
4. Next, the chair is delivered to the Buyer.

5. Finally, the Buyer performs the payment process, which is by payment by credit card.
6. Then, the Cooperation is completed successfully, having solved the Goals of the participants.

4.2 Use Case as Automated Cooperation in SWF

According to the SWF components, mechanisms and methods specified in this document, we now describe how the use case cooperation scenario would be solved automatically within SWF. We therefore assume that the cooperation is resolvable and that the resources applied are homogeneous, i.e. we do not need Mediators. The numbered actions below specify the logical workflow of SWF.

4.2.1 Pre-Requisites

We assume that there is a SWF-application for automated shopping; wherein Buyers and Sellers interact and perform purchases automatically (at least those parts of purchasing that can be performed automatically). We assume that there are several Freds in the system that can be potential Buyers or Sellers.

We also assume that there are suitable SWF-Services (i.e. implementations of functionality) existing in the system, and there are respective ontologies and general Cooperative Goals Schemes for purchasing existing:

- Ontologies for products and for the purchasing process that are applied as the domain terminology definitions by all other SWF components involved.
- Goal Schemas that provide the general structure of completely described Cooperative Goals for the desires of users / the functionalities provided by the available SWF-Services. There are 2 Cooperative Goal Schemas in the use case:
    - **"Buying Product":** defines a complete Cooperative Goal, specifying the object of interest (a product), and the Cooperation Role "Buyer" (saying that the Goal Instance owner will purchase the product, pay for it, and receive the good by delivery).
    - **"Selling Product":** specifies a complete Cooperative Goal, specifying the object of interest (a product), and the Cooperation Role "Seller" (saying that the Goal Instance owner will sell the product, receive the payment for it, and deliver the good to the buyer by a delivery service).

Furthermore, we assume that the participants defined above have defined Freds as electronic representatives that behave of their behalves. These Freds are able to execute the needed Services for automated purchasing, i.e. the Freds have usage permissions for the respective Services or the Services are provided by the participants (the Fred owners).

4.2.2 Cooperation Initialization

Automated Cooperation is SWF is initialized by creating a Cooperative Goal Instance, which then is taken by the GG Discovery in order to detect potential Cooperation Partners.

For the use case, we assume the following situation of initializing a Cooperation:

- The Buyer creates Cooperative Goal Instance from the given Cooperative Goal Schema "Buying Product" at a certain point in time (t0). The Goal Instance defines specific values for the ontological notions specified in the Goal Schema, e.g. defining "Chair" as a type of product with certain properties (wooden, Measures (W x H x D): 50 x 100 x 60, color: light brown), and a maximal price of 150 € - see Use Case description.
- The Seller has a Cooperative Goal Instance of "Selling Product" that specifies the different types of furniture, the different modes for payment and delivery supported, and different ways of the purchasing process supported. This Goal Instance is created and remains ´online´ all the time when the ´shop is open´, as the Seller intents to have several purchases by several Buyers.

4.2.3 Determine Potential Cooperation Partners (GG Discovery)

A Cooperation in SWF is invoked by the GG Discovery that determines potential Cooperation Partners by checking the compatibility of Cooperative Goals (see section 3.1.2.2.1). The Cooperation initialization situation described above allows different scenarios for starting a Cooperation: a Cooperation can be invoked either by the Seller or by the Buyer. Therefore, the different types of GG Discovery invocation engines are used; we describe the different possible scenarios below.

4.2.3.1 GG Discovery invoked by Buyer

Cooperation Invocation by the Buyer is performed by taken the Goal Instance of "Buying Chair" owned by the Buyer and to search for compatible Goal Instances of "Selling Chairs" (or for a superconcept of Chair, e.g. furniture). This is done at time *t0*, i.e. when the Buyer creates a Goal Instance. Concerning the type of search mechanism

of GG Discovery, this is an event-based Cooperation Invocation (Event: creation of "Buying Chair" Goal Instance). In a general case, this can also be invoked out of an application-context, when a Goal Instance is thrown during Service Execution of another Cooperation.

A pre-requisite for Cooperation Invocation by the Buyer is that the Seller Goal Instance is ´online´.

4.2.3.2 GG Discovery invoked by Seller

Also, the Seller can initiate Cooperation, i.e. searching for possible Customers. This would be a permanent search for potential Cooperation Partners as long as the Seller Goal Instance is ´online´. Therein, Buyer Goal Instances are checked for compatibility on the object of interest (i.e. if the Seller can satisfy the purchase intention), and several Cooperations are initiated for satisfying the Goal of the Seller to sell as much furniture as possible.

4.2.4 Detect Suitable Services (GS Discovery)

According to the Automated Cooperation workflow of SWF (see section 3.1.2.1), the second step after GG Discovery is to detect the Services that each Cooperation Partner needs for the automated interaction, called GS Discovery (see section 3.1.2.2.2).

GS Discovery has to be done separately for every Cooperation Partner. Therefore, the Goal Instances of a Cooperation Partner is taken as the input and a set of Services is returned for each functionality that has to be provided by the specific partner for a Cooperation.

4.2.4.1 GS Discovery for each potential Cooperation Partner

The core of detecting suitable services in GS Discovery is Goal-Capability match, i.e. checking whether a Service can satisfy the desire expressed in the given Goal. In general, GS Discovery can work in different flavors: either it returns a set of Services that match the whole Goal (i.e. the support all aspects specified in the Goal) or it returns sets of Services for different sub-aspects of the Goal (see further explanation below). Note that GS Discovery works on the SWF Service Description Language only without regard to the type of Services.

Besides the functional aspect determined by Goal-Capability match, the Services returned by GS Discovery must be usable by the distinct Fred, meaning that the Freds have usage permissions for the respective Services or the Services are provided by the participants (the Fred owners).

4.2.4.2 Detected SWF Services (imaginary for use case)

In the following we describe possible Services that could be detected by GS Discovery in order to illustrate how SWF Services could look like and work. These are only imaginary SWF Services for the use case, but they should show how different types of real-world Services might be reflected as SWF Services.

4.2.4.2.1 Buyer Services

As stated in the use case description above, the Buyer has to provide 3 Services:

1. *Ability to express Search Desire*

   This is a Service wherein the Buyer can specify his object of interest as a query of search, including all property values desired. Most likely, this can be a SWF Service provided by the SWF application for automated purchasing. The Buyer only needs the usage permission for this (which he gets when he registers for that application), and does not have to care about the type of Service and its internal functionality.

2. *Payment Service for paying*

   For the different payment methods that the Buyer may have, there are different types of Services, for example: an external Web Service of some credit card institute for credit card payment, and Processes for Cash and Check payment including the manual activities (filling out check, etc.). All these Services are returned by GS Discovery if the Buyer as usage permissions.

3. *Ability to receive Good Delivery*

   This Service can most likely be a Plan because it is only concerned with the Buyer side. It gets invoked when the Buyer receives the good and notifies the Seller about this.

4.2.4.2.2 Seller Services

We assume that the Seller owns a SWF Service in the system that holds all his functionalities and his external visible business behavior. This would be a Process that specifies the business process supported for selling furniture, and this Process would have a lot of conditional possibilities for the different steps in order to support various types of Services that potential Customers (i.e. Buyers) have.

For the different Services identified in the use case description above, the Seller Process might include the following:

1. *A Search Service for available furniture*

   Here, the same Service provided by the SWF purchase application can be used, as the Buyer does. This ensures compatibility of the Services right away.

2. *Payment Service for excepting payment by different payment methods*

   The Seller should be able to support as many payment methods as possible in order to broaden his support for potential Buyers. Therefore he should have disjunctive support for different payment methods, with similar Services types as identified for the payment Service at the Buyer side (see above).

3. *Good Delivery Service*

   For Delivery Service, either an external Web Service of a contracted Supplier can be used or another Fred is called that provides the delivery facility by specifying another Cooperative Goal in this Process activity.

Figure 16shows this Process within the FRED Process Composer.

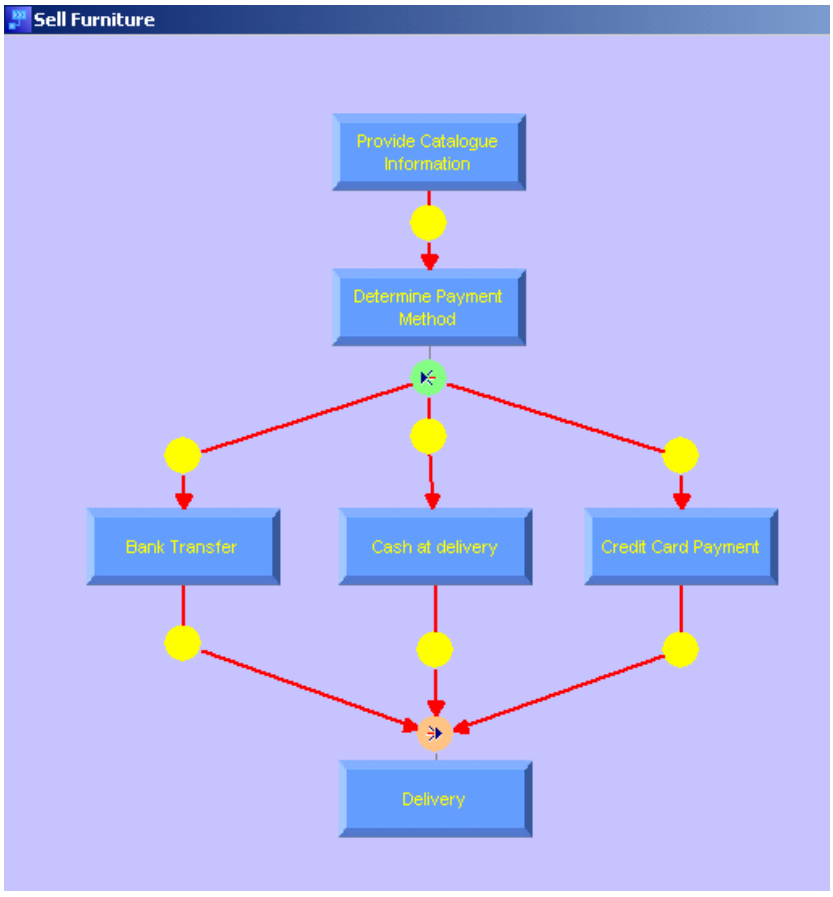

Figure 16: Process "Sell Furniture"

4.2.5 Determine Cooperation Contract (WW Discovery)

After the usable Services for each Cooperation Partner have been determined in GS discovery, they have to be checked for compatibility with regard to their interaction behavior that is specified in the Service Choreographies. This is done in WW Discovery (see section 3.1.2.2.3).

Apart from the compatibility of the messaging sequences and the MEP, a suitable execution order for the Services has to be fixed for automated Cooperation. In the Services for the partners described above, the Buyer only has sets of Services for the distinct aspects of his Cooperative Goal. But we have a fixed sequence (although with options) at Seller side, therefore we have to determine the execution sequence at Buyer side, taken the given Services of the Buyer and check / determine their compatibility to the Seller Process. If this works out, the definite service execution sequence along with all other information that is needed are specified in the Contract for Cooperation; if not, the SWF Cooperation Workflow stops here and re-invokes either GS-Discovery for detecting other SWF Services or the complete Cooperation proposal is finalized without success.

4.2.6 Execute Cooperation (Service Execution)

If WW Discovery has been successful and the Contract for Cooperation is fix, than a Meeting is scheduled between the Freds of the distinct Buyer and the Seller. The management of scheduled meetings is handled by the Meeting Room and its preliminary technologies.

In the Meeting, the Services are executed as specified in the Cooperation Contract. All runtime related issues are handled here: the Invocation of the different types of Services, their Execution with the distinct execution technologies, requests for additional user input if required, as well as Error Handling (see section 3.2). When irresolvable errors occur (this can be any execution exceptions, not resolvable compensations, as well as irresolvable mismatches), the Cooperation is stopped here as a failure.

4.2.7 Finalize Cooperation (Goal Solver)

If the Cooperation Execution has been completed successfully, the so-called post-cooperation processing starts. This includes the management of Goal Resolution by the Goal Solver (i.e. setting the Cooperative Goal Instances of the participants to be “completed”), and the management of Freds in the system. All Freds that are currently not involved in any Cooperation are withdrawn from the runtime environment and holds

in the Fred repository in a sleeping mode, and called back for the next Cooperation. This improves scalability of the system (see Stollberg et al, 2004).

5 Scope of SWF Project

The area of research and development touched by the SWF project is very large, multi-faceted, and of infinite complexity. The aim of the SWF project is to tackle the related problem fields and create a suitable architecture for goal-driven cooperation between agents based on a Service-Resolution technology that allows dynamic discovery, composition and execution of problem solving services.

Due to limited resources, we have to restrict the scope of the SWF project to feasible objectives.

5.1 Scope Restrictions

In order to restrict the research and development efforts to the most important aspects at this point in time and to define a feasible scope for the project, we record the problems fields related to SWF and specify the main fields of activity in the project in Table 1.

Table 1: Scope Specification of SWF Project

Aspect / Problem field	SWF-Scope
Description Language for Goals and Services	central aspect, stepwise development
Discovery Mechanisms: Specification, Algorithms, and Execution Technology	central aspect, stepwise development **GG Discovery:** core mechanism with Complete Goal Description only **GS Discovery:** core mechanism with minimal extensions **WW Discovery:** core mechanism only
Mediation Facilities	not a central aspect
Mediation Environment	not a central aspect
Repositories	**Ontology Repository:** existent SMO

	technology **Goal Repository:** important, stepwise development **Service Repository:** important, stepwise development **Mediator Repository:** not essential
Service Execution	**Invocation:** central aspect **Contract**: not a central aspect **Error Handling:** basic features
SWF in FRED environment and architecture	central aspect
SWF in Web environment	not a central aspect

5.2 Project Plan / SWF Deliverables Overview

This deliverable specifies the overall approach and architecture of SWF and points out the development issues of the project. In the subsequent deliverables, specific technological building blocks will be specified. Those deliverables which introduce WSMO technology to Fred system are listed below.

Table 2: SWF Project Plan 2004

SWF Deliverable No. & Title	Contents	Planned Due Date
D1 "SWF Framework"	- architecture, components, workflow definition - requirements analysis - scope of project	**M3** (03/2004)
D2 "SWF Language Evaluation and Comparison"	background and rationale of SWF Language choice	**M4** (04/2004)
D3 "SWF Goal and Service Description Language"	- specification of description elements for Goals and Services - definition of system ontologies for Goals and	V.1: **M5** (05/2004) V.2: **M7**

	Services - specification of representation language	(07/2004)
D4 "SWF Mechanism and Tools"	- specification of SWF mechanisms and algorithms - requirements / specification of SWF tools	**M8** (08/2004)
D5 "SWF Semantic Data Processing"	requirements & specification of SWF Semantic Data Processing unit (mechanisms and tools)	

6 Conclusions

In this document we have worked out the Semantic Web Fred Framework which defines the building blocks of the SWF Architecture as well as the workflow of SWF. We also have outlined the requirements on the technologies to be developed in the course of the project and defined the scope of the Project as well as the work plan for the first year of the SWF project.

On basis of this exhaustive framework, the subsequent deliverables will specify the SWF technology in more detail along with their respective implementation.

References

Arkin, A. et al: Web Service Choreography Interface (WSCI) 1.0. W3C Note 8 August 2002. available at: http://www.w3.org/TR/wsci/

Ballinger, K. et al: Web Services Inspection Language (WS-Inspection) 1.0. IBM Specification 2001. available at: http://www-106.ibm.com/developerworks/webservices/library/ws-wsilspec.html

Banerji, A. et al: Web Service Conversation Language (WSCL) 1.0 W3C Note 14 March 2002. available at: http://www.w3.org/TR/wscl10/

Bellwood, T.; Clément, L.; von Riegen, C. (Ed.): UDDI Version 3.0.1. UDDI Spec Technical Committee Specification, Dated 20031014, available at: http://uddi.org/pubs/uddi_v3.htm

Chinnici, R.; Gudgin, M.; Moreaum, J.-J.; Weerawarana, S. (2003): *Web Services Description Language (WSDL) Version 1.2*, W3C Working Draft 3 March 2003.

Curbera, F.; Goland, Y.; Klein, J.; Leymann, F.; Roller, D.; Thatte, S.; Weerawarana, S. (2002): *Business Process Execution Language For Web Services*, BEA Systems &

IBM Coporation & Microsoft Corporation.

Erl. T.: *Service Oriented Architecture. A Field Guide to Integrating XML and Web Services*. London, Prentice Hall PTR 2004.

Fensel, D., Angele, J.; Decker, S.; Erdmann, M.; Schnurr, H.P.; Studer R., Witt, A.: *On2broker: Lessons Learned from Applying AI to the Web.* Technical Report Institute AIFB Research report no. 383, 1998.

Fensel, D.; Bussler, C.: *The Web Service Modeling Framework WSMF*, Electronic Commerce Research and Applications, 1(2), 2002.

Fensel, D.; Motta, E. ; van Harmelen, F.; Benjamins, V. R.; Crubezy, M.; Decker, S.; Gaspari, M.; Groenboom, R.; Grosso, W.; Musen, M. A.; Plaza, E.; Schreiber, G.; Studer, R.; Wielinga, R.: *The Unified Problem-solving Method Development Language UPML.* Knowledge and Information Systems Journal (KAIS), 5(1), 2003.

Keller, U.: Inferencing Support for the Semantic Web. Reasoning about Semantic Web Service Descriptions in WSMO and WSML. WSMO Deliverable D5.1 v0.1, 1st March 2004. available at: http://www.wsmo.org/2004/d5/d51/v01/20040301/

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

Koivunen, M.-R.; Miller, E.: W3C Semantic Web Activity. In: Proceedings of the Semantic Web Kick-off Seminar in Finland. Helsinki: 2001.

Lara, R.; Roman, D.; Polleres, A.: Conceptual Comparison WSMO / OWL-S. WSMO Deliverable D4.1, v01. available at: http://www.wsmo.org/2004/d4/d4.1/v01/

Manola, F.; Miller, E.: RDF Primer. W3C Recommendation 10 February 2004. available at: http://www.w3.org/TR/rdf-primer/

McGuinness, D.; v. Harmelen, F. (2004): *OWL Web Ontology Language Overview.* W3C Recommendation, 10. Feb. 2004. available at: http://www.w3.org/TR/owl-features/

Papakonstantinou, Y.; Garcia-Molina, H.; Ullman, J. (1996). *MedMaker: A Mediation System Based on Declarative Specifications.* In Proceedings of the International Conference on Data Engineering (ICDE 96), pp. 132-141.

Peltz, C. (2003): Web Service Orchestration. A review of emerging technologies, tools, and standards. Hewlett Packard Company.

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

Roman. D. et al.: *Web Service Modeling Ontology WSMO, Version 0.1*. WSMO Deliverable D2, DERI Working Draft 14 feb. 2004.

available at http://nextwebgeneration.com/projects/wsmo/2004/d2/v01

Stollberg, M.; Lausen, H.; Arroyo, S.; Herzog, R.; Smolle, P.; Fensel, D.: *Fred Whitepaper*, 2003.
available at: http://www.netdynamics-tech.com/media/downloads/FRED-WhitePaper.pdf

The OWL Services Coalition: OWL-S: Semantic Markup for Web Services, version 1.0 available at http://www.daml.org/services/owl-s/1.0/owl-s.pdf

Wiederhold, G. (1992). *Mediators in the Architecture of Future Information Systems*, IEEE Computer, 25(3): pp. 38- 49.

Appendix A – Glossary

This section defines the terminology used in the SWF project along with definitions. Italic words refer to other terms explicitly defined in the Glossary.

Table 3: SWF Terminology Glossary

Activity	A step in a *Process.* In the SWF Process Model, an Activity can be a Goal with our without an associated *Plan*, a Sub*Process, or a Goal with out without an associated Sub-Process.* We also call an Activity what is handed over to a human for manual processing, administrated by an Activity Manager.
Agent	An electronic representative that solves tasks on behalf of its owner. It acts autonomously and solves the tasks it is assigned in *Cooperation* with other agents.
Choreography	The visible *External Behavior* of a *Service*, describing the *Message Sequence* and the externally visible *Activities* of a stateful Service. The *WW Discovery* deals with the Choreography of Services for establishing the *Contract* for *Cooperation*.
Compensation	Undo effects created by a set of Services if one of the *Service Execution* fails.
Contract	Holds the agreement on *Services* used for a *Cooperation*, established after the *Discovery Mechanims* worked successfully and service invokation got called.
Cooperation	Interaction between *Agents* for executing tasks that have been

	assigned to *Freds* when trying to solve cooperative (compatible) Goals. Automated Cooperation is the central functionality of SWF. The SWF Cooperation Model relies on real-world cooperation models.
Description Language	A higher-leveled language that describes the structure of SWF components. The *Discovery Mechanims* as well as *Mediators* work on the Description Language contructs of SWF compontents. In SWF, a Description Language is defined for *Goals* and for *Services*.
Discovery Mechanism	Superordinate term for *GG Discovery, GS Discovery, and WW Discovery*. Mechanisms that discover suitable partners and services by matching specific constructs of the Description Language of SWF components, determining compatibility.
Error Handling	Handling of errors and exceptions that occur during *Service Execution*. The methods for Error Handling are defined in a Error Framework.
External Behavior	Describes the externally visible behavior of a *Service*, located in the *Service Interface Description*. The External Behaviors of Services are matched in *WW Discovery* in order to obtain the *Contract* of a *Cooperation*.
Fred	An *agent* in the FRED system
GG Discovery	*Discovery Mechanism* that determines potential cooperation partners by matching *Cooperative Goals*.
Goal Description (Cooperative Goal)	Specification of the desire for cooperation, describing the state of the information space and effects to be achieved by cooperation of Freds in the Fred system. A Cooperative Goal Description holds information about the objects of discourse as well as the owner (creator) of the Goal.
Goal Description, Complete	A Complete Goal Description specifies the desired information space and effects from the Goal owner's view as well as from the potential partners' view.
Goal	A Partial Goal Description specifies the desired information space

Description, Partial	and effects from the Goal owner's view.
Goal Instance	A particular instance derived from an existent *Goal Schema* specifying a concrete desire, i.e. a specific request for a solution. A Goal Instance can only be created from an existent *Goal Schema*.
Goal Resolution	General term for solving a Goal arbitrarily by *Cooperation* and dynamic *Service Resolution*.
Goal Schema	A generic Goal Description. Goal Schemas are pre-defined in the system and *Goal Instances* are created out of Goal Schemas.
Goal Solving Process	The overall process which gets executed, after a Goal Instance is created, to achieve the desired state of that Goal Instance. Once the desired state is entered the Goal is considered to be *solved.*
Goal Solver	Mechanism that executes and controls the resolution of *Goals.*
Goal-driven approach	Scientific designation of technologies that separate the requester and provider by using the concept of *Goals* and *Services*.
GS Discovery	*Discovery Mechanism* that detects all suitable services a Fred can use in a *Cooperation* by matching *Cooperative Goals* and *Services*.
Invocation	Invoking a particular *Service* for *Service Execution.*
Manual Activity	An Activity handed over to a human for manual execution.
Mediation	Techniques that make heterogeneous resources interoperable. Therefore, the mismatches between resources are identified and resolved by specific mechanisms that work on their descriptions (*Description Language*).
Mediator	A *Service* that has the *Capability* to resolve heterogeneties between resources of the same type in order to make them interoperable. A Mediator can be called into a *Meeting* when the resources used by the Freds are heterogeneous.
Meeting	A Meeting is part of the Service Execution and the place where *Agents* execute and interact with each other in an automated

	Cooperation in order to solve their respective *Goals*.
Meeting Room	The place where Meetings take place, i.e. the runtime environment of *Agents*. Apart from the computational environment for *Service Execution*, the Meeting Room holds facilities for establishing *Perfect Mediation* between *Freds*.
Message	A single communicative act that sends information from one to another interacting partner. A Message has a Sender and Receiver, thus is uni-directional.
Message Exchange Pattern - MEP	A set of message types that allows defining semantics for *Messages*. An MEP is independent of an application context. In the FRED system, the FIPA MEP is used for agent interaction.
Message Sequence	Sequence of *Messages* sent and received by a *Service*.
Ontology	An ontology defines the terminology of a domain of discourse in a machine-processable format. All SWF components use ontologies as the underlying data model. In SWF, the WSMO based technology and *Smart Objects* technology is used for handling ontologies.
Perfect Mediation	In order to allow automated *Cooperation*, the resources used by the *Freds* have to be completely interoperable. This means that there can not be mismatches of any kind, or the mismatches have to be resolvable by *Mediators*. Freds whose resources are completely interoperable are called “perfectly mediated”.
Plan	Type of a *Service.* A Plan is based on the Plan Framework and implements a problem solver.
Process	Type of a *Service.* It defines a multi-step problem solving implementation. The particular steps are called *Activities* which can be solved by an arbitrary *Service*, thus allowing interdependent specifications of Services.
Repository	Superordinate term for the place and device that holds SWF components. In SWF, there are Repositories for *Goals*, *Services and Service Descriptions*, *Ontologies*, and *Agents*.

Resolution Mechanisms	General term for the *Discovery Mechanisms*.
Service	A problem solving implementation available in the system. The specific Types of Services are defined in the *SWF Service Model*. A Service is discovered by the *GS Discovery* to be suitable to solve a *Goal*. Services are described semantically by a Service Description.
Service Capability	The functional description part of the *Description Language* for *Services*. It describes WHAT a Service does, while the HOW is described by the *Service Interface.*
Service Description	The description of a Service, consisting of Service Capability, Service Interface and Service Grounding.
Service Execution	Execution of a particular Service. This is the last step after the cooperation partners and *Services* have been determined and invoked by the *Discovery Mechanisms*. The Service Execution Environment, i.e. the Meeting Room or Process Engine, holds the computation resources to execute all types of *Services* and invokes possibly needed *Mediators* needed for execution. Moreover, the Service Execution Environment handles Errors and Compensation during runtime.
Service Grounding	The part of the *Description Language* for *Services* that describes the physical implementation of a Service. It describes the Service Usage and holds information about Error Handling.
Service Interface	The part of the *Description Language* for *Services* that describes how to interact with a Service, i.e. the *Choreography*.
Service Resolution	The process of identifying a Service through discovery and associating it with a given Goal Instance.
Smart Object	A Smart Object (SMO for short) is a Ontology Object that is transformed into a Java Object in order to make use of mature technologies for data management. There are three types of SMOs: OOIs (an ontology instance), OOCs (an axiomatic expression), and OOGs (SMOs grouped together within. an application context). SMOs are as expressive as OWL for ontology representation. The

	SMO technology is currently available within the FRED system.
WW Discovery	*Discovery Mechanism* that determines compatible Services of potential cooperation partners out of the Services that have been discovered by *GS Discovery* by comparing and matching *Service Interfaces*.
SWF Service Model	The SWF Service Model defines the *Type of Services*, which can either be a *Plan*, a *Process or* an *external Web Service.*
Web Service (external)	Type of *Service* that calls an external Web Service via its WSDL-description, executed by the WSDLExecutorPlan.
WSMO	Web Service Modeling Ontology. DERI technology for Semantic Web Services. Homepage: www.wsmo.org