Semantic Web Fred – Automated Goal Resolution on the Semantic Web

Michael Stollberg[1], Dumitru Roman[1], Ioan Toma[1], Uwe Keller[1], Reinhold Herzog[2], Peter Zugmann[2], Dieter Fensel[1]

[1]*DERI – Digital Enterprise Research Institute*
University of Innsbruck, Austria
{michael.stollberg, dumitru.roman, ioan.toma, uwe.keller}@deri.at

[2]*Net Dynamics Internet Technologies GmbH & Co KG*
Vienna, Austria
{reinhold.herzog, peter.zugmann}@netdynamics-tech.com

Abstract

Semantic Web Fred, SWF for short, is a context-independent, goal-driven system for automated execution of tasks that are delegated to electronic representatives along with dynamic service usage. A task is assigned to an agent for automated resolution, represented as a Goal. This is used to determine potential partners for collaborative task resolution, and for discovery of suitable goal-resolving services that can be internal implementations as well as external Semantic Web Services. The SWF technology integrates agent technology, ontologies, and Semantic Web Service technologies – the technological building blocks identified for the Semantic Web – into a coherent system. This paper describes the architecture of SWF, explains the mechanisms for establishing automated and cooperative goal resolution, and the alignment of SWF with the Web Service Modeling Ontology WSMO, a well-structured overall framework for Semantic Web Services. We also outline the contribution of SWF to the development of Semantic Web technologies.

1. Introduction

The objective of the Semantic Web is to develop enhanced information processing technologies for the Internet. Key technologies have been identified with different functional purposes: ontologies for semantically enhanced information exchange over the web between different agents (which can be individuals, organizations, or machines), Web Services for reuse and interoperability of computational functionality over the web, and agent technology for automated execution of tasks [2]. In order to exploit the full potential of Semantic Web enabled applications, these technologies have to be combined into coherent frameworks, utilizing the specific functional benefits of each technology. At this point in time, numerous research and development efforts are concerned with developing technologies for the Semantic Web, whereby only a few address the challenge of an integrated approach.

Semantic Web Fred combines agent technology, ontologies and Web Services into a goal-driven approach for resolution of tasks by services, applying emerging technologies for Semantic Web Services. Agents, called Freds in the system, interact in order to perform tasks automatically on behalf of their owners. Therefore, Freds have to find appropriate cooperation partners, as well as the computational resources for automated task resolution. With regard to service-oriented architectures as envisioned for Semantic Web Services [6], the main building blocks of SWF are Goals and Services: a Goal represents a task that a Fred has been assigned, and a Service is a computational resource that allows automated resolution of Goals. Services to be used in the system can be internal implementations, processes, and external, semantically described Web Services. Ontologies are used as the data model throughout the whole system, providing machine-processable terminology used in all other components. The objective of the SWF project is to develop advanced mechanisms for cooperative goal resolution, thereby providing an integrated system that combines the key technologies of the Semantic Web into a coherent framework.

The starting position of the SWF project is the FRED platform, developed by Net Dynamics. The FRED system is an environment for agent-based applications that support delegation of tasks to electronic representatives. Its main building blocks is at first the FredBase as the agent runtime environment, consisting of Meeting Rooms wherein interactions between agents (Freds) take place, management facilities for Freds, and interfaces for connection to other systems. The second one is the Smart Object ontology technology: ontologies are transformed into Java Objects in order to allow usage of conventional, sophisticated technologies for ontology data usage and management. The third building block is a goal-driven technology for service resolution, meaning that a Goal is specified as the request for functionality that is resolved by services available in the system. A complete description and analysis of the FRED system is provided in [24]. The aim of the SWF project is to extend the goal-driven technology for service resolution with advanced mechanisms and to align it with emerging technologies for Semantic Web Services – with regard to the obvious similarities in technological challenges, and in order to support Semantic Web technologies within SWF. As the foundation for alignment with Semantic Web Service technologies, SWF is based on the Web Service Modeling Ontology WSMO, an initiative that aims at developing an overall framework for Semantic Web Service description and technologies. [1]

The focus of this paper is to explain the SWF Framework for automated and cooperative task resolution, with special attention to how distinct technologies are arranged into a coherent system as well as the approaches for technological realization of the components. The aim is to point out how the key technologies for the Semantic Web are combined in SWF as a substantial contribution to the development of architectures and technologies for Semantic Web applications.

The paper is structured as follows: Section 2 introduces the design principles and the overall architecture of SWF; Section 3 describes the SWF components and their technological realization; on basis of this, Section 4 explains the mechanisms for automated and cooperative goal resolution; Section 5 discusses the contributions to Semantic Web development and positions SWF within related work; finally, Section 6 concludes the paper and points out directions for future development of SWF.

2. SWF Architecture

The following introduces the design principles underlying the SWF Framework, and explains the overall functionality for automated, cooperative goal resolution.

2.1. System Design Principles

The traditional approach that dominated IT-system design over decades is the request-provider model: a request carries the information of the service, invokes the service and collects the result of the service. In order to overcome the limitations in regard to support of the user perspective and reusability of services, the "rational agent approach" enforced in the AI-community introduces the notion of goals. The major motivation for this novel approach is that the user desire is **decoupled** from the resolving services, and the connection is **dynamically derived** during runtime by intelligent mechanisms [20]. This allows **context-independent** architectures with maximal **reuse** of existing computational resources. The technological challenge for such systems is an expressive, well-defined description language and, on this basis, a coherent framework of mechanisms for goal resolution. An additional challenge arising in open and de-centralized environments like the Internet is **heterogeneity**, leading towards mediator-oriented architectures for resolving mismatches on a syntactic, semantic, or behavioral level [27]. With respect to this, WSMO envisions a goal-driven, mediator-oriented approach for Semantic Web Services: inference mechanisms for discovery, composition, and execution of Web Services are based on a profound description framework for *Ontologies*, *Goals*, *Web Services*, and *Mediators* as the top-level components [8], [22].

Semantic Web Fred realizes the WSMO approach, adding the notion of cooperative goal resolution. According to the paradigm of agents as autonomously acting entities in a software environment and with regard to task solving in real-world settings, more complex goals can only be achieved in cooperation as needed facilities might be held by different entities. Further, a cooperative goal resolution will only take place if it is profitable for all participants. In consequence, **symmetry** is a supplementary design paradigm of SWF: every acting entity in the system has Goals to be resolved as well as Services that provide the functionality the entity brings into a cooperative goal resolution. The SWF architecture shown in Figure 1 reflects this, with further explanations below.

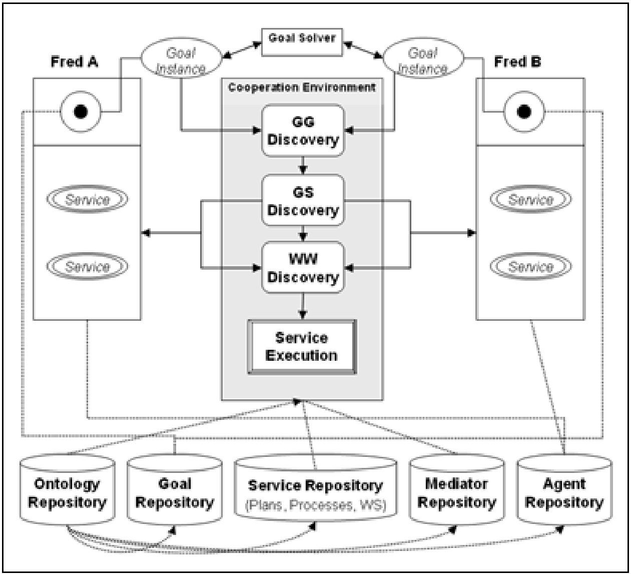

Figure 1: SWF Architecture

[1] WSMO is the Semantic Web Services research effort of the SDK-Cluster, a joined initiative of European research around the Semantic Web (see www.sdk-cluster.org). The aim of WSMO is to define an overall framework that covers relevant aspects for Semantic Web Services, and subsequently develop technologies Semantic Web Services. The conceptual foundation of WSMO is the Web Service Modeling Framework WSMF [8]. See WSMO homepage at: www.wsmo.org.

2.1. SWF Overall Workflow

Figure 1illustrates the structure of Freds, the component repositories, and the architecture of the cooperative goal resolution environment. For explaining the overall functionality, we refer to the common example of purchasing: a buyer wants to buy a chair, and a seller wants to sell furniture (including chairs). These two goals can only be achieved in a cooperation between the buyer and seller, which is realized in SWF as follows.

Image that both parties are represented by Freds in the system (Fred *A* as the buyer, Fred *B* as the seller), and services for purchasing are available. At a certain point in time, both create Goal Instances (the ball on top of a Fred-box), indicating their desire to solve the Goal. Out of possibly several Freds in the system, so-called Goal-to-Goal Discovery (short: *GG Discovery*) detects Fred *A* and Fred *B* for cooperation by determining the compatibility of the Goal Instances. Then, *A* and *B* have to find suitable services that provide the specific functionalities needed for cooperation. For instance, Fred *B* as the seller needs a browsing facility for his product catalogue, ability to place a contract of purchase, and a facility for receiving payment; Fred *A* as the buyer needs the opposite facilities. This is performed by Goal-to-Service Discovery (short: *GS Discovery*) which detects appropriate services for each Fred in the service repository, similar to Web Service Discovery. In order to allow interaction of the services for automated goal resolution, the services used by the cooperation partners have to be reconcilable with regard to their external behavior (the external visible business processes) and the expected messaging sequence. Thus, the third step in establishing cooperative goal resolution is Service-to-Service Discovery (short: *WW Discovery*, for societal reasons), establishing the behavioral compatibility of the services detected in GS Discovery. The result of the three discovery mechanisms is a *Cooperation Contract* that contains all information relevant for execution of the cooperation (the Freds, the Goals, and the Services to be used). Then, the cooperation partners are called into a Meeting Room (see above), wherein this contract is processed by the *Service Execution* component that handles all execution-related aspects (errors, compensation, etc.).

The goal resolution process is managed by the *Goal Solver*: it controls the discovery mechanisms, including iterations (e.g. re-call GG Discovery when GS Discovery fails) and rollbacks when the execution of services fails; it also monitors the resolution process of the Goal Instances of the cooperation partners. Although the goal resolution technology in SWF is designed for automated resolution of cooperative goals in particular, Goals that do not need cooperative resolution can be handled as well. Therefore, a Fred is called in a *Singleton Meeting* where the required services are executed without agent interaction.

This general overview reveals the design principles of SWF – a goal-driven approach for de-coupling of request and functionality, and the symmetry of Freds as the acting entities in the system. Moreover, it emphasizes the similarity of the SWF components and mechanisms with approaches currently considered for Semantic Web Services. In order to illustrate SWF as an integrated agent platform for the Semantic Web, the following sections explain the technological realization of the distinct SWF components and their interplay in more detail.

3. SWF Components

This section presents the different SWF components as basis for the discussion of the SWF mechanisms for automated goal resolution. Apart from the technological realization, we emphasize the demarcation of technologies for different functional purposes.

As the central components, we first discuss the description model and usage of Goals and Services. In order to ensure conformity and interchangeability with other Web Service-based applications, the SWF Goal and Service Description Language is a specialization of the WSMO description elements for Goals and Web Services [22]. SWF uses WSML for modeling, a concise, platform-neutral representation format for WSMO components based on F-Logic [13] that can be transformed into different backend technologies [16].

3.1. Goals

A Goal expresses a desire that a user delegates to a Fred for automated resolution. When assigned to a Fred, the information kept in the Goal is used to determine potential cooperation partners and the SWF Services needed to perform the resolution automatically.

The most important information of Goals is the specification of the *desire*, i.e. what the user wants to achieve. This is modeled as an unambiguous ontology object with regard to specified domain ontologies that restricts the set of instances which can resolve the Goal. For instance, the desire of Fred *A* in the example above would specify a brown-colored, wooden chair with measures 50 x 100 x 60 cm. These restrictions might reflect only a subset of the attributes of a chair in the referred ontology, but they completely described the desire from the user's perspective. If a service can return instances that are valid for desire of the Goal, then this service can be used for executing the goal resolution automatically (see section 4.2). Goals are kept in the SWF Component Repository (see section 3.3.4).

For managing and usage of Goals in SWF, additional constructs are used that extend the notion of Goals in WSMO (wherein a Goal models the desire only). The following gives an overview of these constructs, explaining their architectural denotation as well as their semantic description elements.

3.1.1. Goal Schema. A Goal Schema defines the ontological structure wherefrom Goal Instances are derived as concrete expressions of a desire. Goal Schemas are predefined in the system, and a task assigned to a Fred is a Goal Instances created out of an existing Goal Schema.

A Goal Schema is equivalent to WSMO Goals, described by the following elements:

non-functional properties. WSMO Core Properties, based on the Dublin Core Metadata set [26]. This is information for managing items in the system, here: *title, creator, description, date, time, identifier, rights, version*.

usedMediators. import of the ontologies used as terminology definitions, and resolution of possible mismatches.

postcondition. ontological structure of the desire to be achieved with conditions over this, as explained above.

effects. side-effects on the world that are expected to hold after the Goal is solved.

3.1.2. Goal Instance. A Goal Instance is the concrete expression of a desire, created by a Fred during runtime by instantiating a Goal Schema (i.e. definition of specific values for certain attributes of the ontology objects and refinement of conditions defined in the Goal Schema). A Goal Instance represents a task assigned to a Fred, and it gets resolved in the goal resolution process. It might take several cooperation meetings to resolve a Goal Instance.

A Goal Instance inherits the descriptive information of the corresponding Goal Schema, instantiates the ontological structures and refines the respective conditions, and carries additional information needed for managing the goal resolution process:

instanceOf. link to corresponding Goal Schema.

non-functional properties. in addition to those inherited from Goal Schema: *timeConstraints* (time frame for resolution), *resourceConstraints* (preferred cooperation partners), *goalResolutionConstraints* (other user preferences).

owner. link to the Fred that created the Goal Instance.

submission. the information that will be handed over as input to a service.

postcondition. instantiation of the Goal Schema postcondition, with refinements.

effects. refined and extended from Goal Schema.

status. resolution process information, possible states: *open, processing, solved, cancelled, not solved*.

3.1.3. Cooperative Goal. A Cooperative Goal describes a Goal that has to be solved in cooperation of multiple partners. It defines compatible Goals, i.e. the Goals that individual entities carry as partners in cooperative goal resolution. Referring to the example above, a Cooperative Goal 'purchase' has the compatible Goals 'buy chair' held by Fred *A* and 'sell furniture' held by Fred *B*. Thus, a Cooperative Goal defines compatible cooperation roles on an ontological level (see section 4.1).

Cooperative Goals are described by the following elements:

non-functional properties. same as for Goal Schemas.

compatibleGoals. links to the compatible Goal Schemas, along with constraints on specific Goal Schemas (e.g. cardinality constraints for the object of interest).

cooperativeGoalConstraints. conditions that resolve structural mismatches between compatible Goal Schemas (e.g. restricting the object of interests in compatible Goal Schemas to an exact match).

3.2. SWF Services

The SWF Service Model is comprised of three different service types that can be used for automated goal resolution: (1) *Plans* are internal services implemented as Java programs, normally used for simple functionalities; (2) *Processes* are multi-step services wherein each activity can be resolved arbitrarily by a Goal or another Service, thus allow definition of complex, nested services; also (3) *external Web Services* can be used, invoked via their WSDL description. Each service type requires a different execution technology: a JVM for Plans, a Process Engine for Processes, and an execution facility for WSDL-descriptions of external Web Services. These technologies already exist in the FRED system, see [24].

From a user's perspective, the service type and its technological realization are not of interest; what is needed is unambiguous information on what the service provides and how it can be used. Thus, SWF Services are described by a common description scheme of all types of services, and the resolution of the service type is left for service execution. SWF Services are described as WSMO Services by the following notions: *non-functional Properties* (management information), a *Capability* as a functional description, and an *Interface* that specifies the behavior interfaces of a service needed for service usage [22]. The following explains the conceptual model and the description elements of these notions.

Ontologies are used as terminology definitions in service description via the *usedMediators*-construct as in Goal Schemas. Services are kept in the SWF Component Repository (see section 3.3.4).

3.2.1. SWF Service Capability. As a functional description, the Capability describes what the service does. It defines the input required for the service, the result the service returns when executed successfully, and different conditions. The Capability of a Service serves as the information for detecting suitable services of a given Goal in GS Discovery (see section 4.2).

The description elements for SWF Service Capabilities coincide with those for WSMO Web Service Capabilities:

precondition. input required by the service along with conditions.

assumptions. arbitrary constraints on state of the world that has to hold before the service can be executed.

postcondition. result of the service in relation to the input, and conditions over this.

effects. arbitrary constraints of the state of the world that result as a change after execution of the service.

3.2.2. SWF Service Interface. This describes the behavior of a Service, needed for service usage. WSMO distinguishes Choreography and Orchestration in a Web Service Interface: the former describes the external visible behavior as needed for using the Service; the latter describes the composition of several Web Services into the functionality of the described service [22]. As Orchestration is realized by the processes in SWF and this information is not relevant for service usage, only the Choreography of a service is described in the SWF Interface.

The Choreography Interface of a single service describes its external visible behavior to allow interaction with the service user. While a simple service behavior consist of invocation and result communication only, the Choreography of a service with a more complex behavior describes those steps of the service's business process wherein interaction with the service user is required. For instance, the selling service for Fred *B* in the example above needs an acceptance notification form the user for the contract of purchase before it can proceed with the payment activity. Thus, the Choreography description of a service is comprised of the externally visible activities and transitions of its business process along with the messaging sequence expected for each activity.

The underlying model of a SWF Service Choreography Interfaces is that there is an invocation message at the beginning of service-user interaction, and a result message at the end. In between, there is an arbitrary exchange of messages according to the external visible behavior of the service. This individual Choreography Interface description is the basis for WW Discovery, wherein a global interaction model for the services of cooperation partners is determined.

Besides the behavior interface description, a Choreography Interface comprises information on the physical access to the Service, compensation information for alternative execution paths, the binding to the used communication protocol, and the resolution of the SWF service type.

In accordance to Choreography Interface description in WSMO, an SWF Service Interface is described by the following elements:

invocationMessage. proposition of usage by service user, including the input information required by the service.

resultMessage. notification of service finalization, handing over the result to the service user (output or error).

choreography. external visible behavior along with the messaging sequence expected. According to Choreography description in WSMO, activities and transitions are described by common workflow patterns [28] in conjunction with communication patterns for describing the messages expected to be interchanged [20].

access. link to physical location of service.

binding. communication protocol of the service (supported protocols: *FIPA, HTTP, SOAP*).

compensation. alternative execution paths (other services that return the same output).

serviceType. type of SWF Service, as explained above: *plan, process, webservice*.

3.3. Other Components

The other SWF components are Freds, Ontologies, Mediators, and the SWF Component Repository. We briefly outline their technological realization in order to complete the technical overview of the system.

3.3.1. Freds. A Fred is a software agent that acts as an electronic representative on behalf of its owner. Goals can be assigned a Fred for automated resolution, and a Fred has usage permissions for Services existing in the system.

As stated above, the FredBase is the agent runtime environment of the FRED system with efficient management facilities for Freds. Interactions between Freds take place in Meeting Rooms, wherein all computational resources are available for Service Execution. We refer to [24] for further information on the FRED agent technology.

3.3.2. Ontologies and Smart Objects. Ontologies are transformed into Java Objects in the FRED system, so-called Smart Objects. As exhaustively discussed in [24], the expressiveness of Smart Objects for ontology representation is comparable to OWL-DL. Smart Objects are used as the data model throughout the whole SWF system. The Smart Object technology also comprises facilities for management, evolution, mismatch-handling, and persistent storage for ontology data.

Import and export of ontology schema and instance data to standard ontology languages (RDF and OWL) is performed in the *Ontology Tower*, the central ontology management unit. Besides, conventional web content can be imported into Smart Objects by semantic annotation on basis of an existing ontology.

3.3.3. Mediators. Mediators are the central component in WSMO for handling heterogeneity. In general, a WSMO Mediator connects one or more *sources components* with a *target component*, whereby optionally a *mediation services* can be included for resolving possibly occurring heterogeneities between the components. WSMO defines four types of Mediators: *OO Mediators* connect ontologies and import them as terminology definitions into other components, *GG Mediators* for connecting Goals, *WG Mediators* connect Goals and Web Services, and *WW Mediators* connect Web Services [22].

SWF applies the concept of WSMO Mediators, using OO Mediators to import domain ontologies into component descriptions, and the other mediator types within specific discovery mechanisms. As the development of mediation facilities is out of the scope of the SWF project, existing Mediators are applied.

3.3.4. SWF Component Repository. This is an UDDI registry which holds all SWF components, apart form Freds. It realizes a hybrid architecture in order to combines the benefits of central and peer-to-peer repositories. Therein, the non-functional properties of each component are mapped to equivalent UDDI information types and are stored centrally in the registry, while the detailed semantic descriptions are kept locally in a persistent repository at the respective owner. Publishing and retrieval is supported by the regular UDDI functionalities [11].

In order to support interchangeability, the SWF component repository is aligned with the WSMO Registry (which has the same architecture). For concurrent publishing, SWF component descriptions are transformed into the corresponding WSMO component description, facilitated by the fact that the SWF description language is a specialization of WSMO.

4. SWF Mechanisms

After presenting the building blocks of SWF, this section explains the mechanisms developed to establish automated and collaborative goal resolution. We introduce the steps successively, explaining the functionality of each mechanism in the resolution process as well as the approaches for matchmaking in the distinct discoverers.

4.1. GG Discovery

As the first step in the cooperative goal resolution, GG Discovery detects potential cooperation partners by determining the compatibility of their respective Goal Instances. The compatibility is given when the objects of interest (i.e. the Goal postconditions) match, and when the cooperation roles of the Goal Instance owners are compatible.

Figure 2 shows the structure of GG Discovery. A *Cooperative Goal* defines compatible Goal Schemas, thus specifying compatible cooperation roles and constituting a pre-selection of possible cooperation partners at design time. Optionally, a WSMO GG Mediator can be defined that resolves mismatches between Goal Schemas that are not compatible a priori. For checking the compatibility of the objects of interest of Goal Instances created from compatible Goal Schemas, the ontology objects defined in the Goal Instances have to be either the same or be a superset or subset of each other. The matchmaking between Goal Instances is realized in the FOL-theorem-prover VAMPIRE [19] by checking whether the Goal Instance of the initiating Fred (Fred *A* is the figure) logically entails the Goal Instances of Goal Schemas that are compatible to the one of Fred *A*. The result of this mechanism is a set of Freds (the Goal Instance owners) as potential partners for cooperative goal resolution. For non-cooperative goals, the GG Discovery step is skipped.

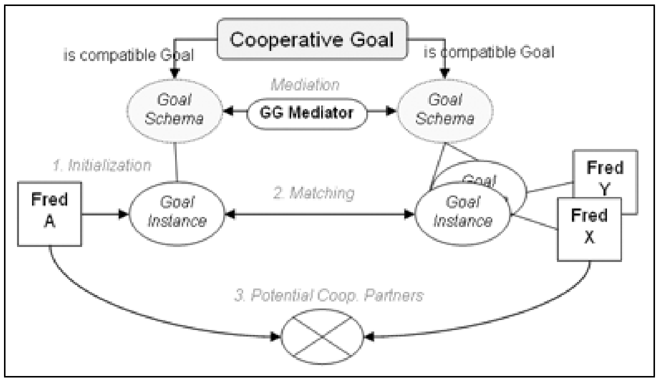

Figure 2: GG Discovery Overview

Three separate search engines initiate GG Discovery: one explores the system for new Goal Instances created by Freds permanently, the second one triggers GG Discovery on basis of explicit events, and the third one initiates GG Discovery dynamically during runtime, e.g. when a new Goal Instance is created by a process. The separation of GG Discovery via Cooperative Goals at design time and the compatibility check of Goal Instances at runtime improves the performance this mechanism.

4.2. GS Discovery

The second step in the resolution process is GS Discovery which detects suitable services for automated goal resolution separately for each cooperation partner identified in GG Discovery. The result of GS Discovery is a set of services that a partner can use for automated cooperation. To use a service, a Fred needs to be either the owner of the service or have appropriate usage permissions. GS Discovery is analogous to Web Service Discovery, determining whether a service can fulfill the desire specified in the Goal Instance. Figure 3 shows the structure of GS Discovery with further explanations below.

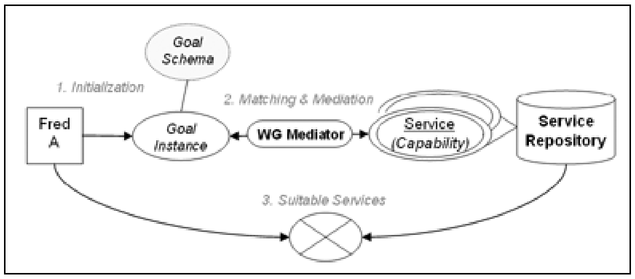

Figure 3: GS Discovery Overview

For Fred *A* as one cooperation partner, the SWF Component Repository is searched for suitable SWF Services. The proof obligation for matchmaking between the Goal Instance of Fred *A* and Service Capabilities as functional descriptions of available Services is that under consideration of all Ontologies and Mediators used in the Goal and the Capability description, if the submission of the Goal Instance satisfies the precondition and assumption of the Capability, and if the Goal Instance postcondition implies the Capability postcondition as well as if the Goal effects imply the Capability effects, then that the Service matches the Goal. For resolving partial Goal-Capability matches into exact matches, a WSMO WG Mediator defines the required reduction similar to the usage of GG Mediators in GG Discovery. This approach is a specialization of the Web Service Discovery defined in WSMO [12].

GS Discovery is realized by determining the logical entailment of the Capability by the Goal, meaning that a match is accomplished if the description elements of the Capability are logical consequences of the corresponding elements of the Goal. Similar to GG Discovery, the matchmaking is realized in VAMPIRE. GS Discovery is as well performed subsequently at design time and runtime: when a new Goal Schema is created (design time), a set of suitable services it detected. Out of this, a subset is determined for specific Goal Instances during runtime.

4.3. WW Discovery

As the last step for establishing an automatically executable cooperation, WW Discovery determines the behavioral compatibility of the services detected separately for each partner in GS Discovery. In order to allow automated interaction of services as the realization of cooperative goal resolution, the Choreography Interfaces of cooperation partners have to be compatible with regard to behavioral models and messaging sequences. The result of WW Discovery is a global interaction model of the partners' services that allows automated execution of the cooperation.

Figure 4 illustrates the structure of WW Discovery for two Freds (*A* and *B*) that have been determined as potential cooperation partners, and each of them has discovered a set of possibly usable Services. The oval boxes represent the Choreography Interfaces with the external visible behavior and the related messages.

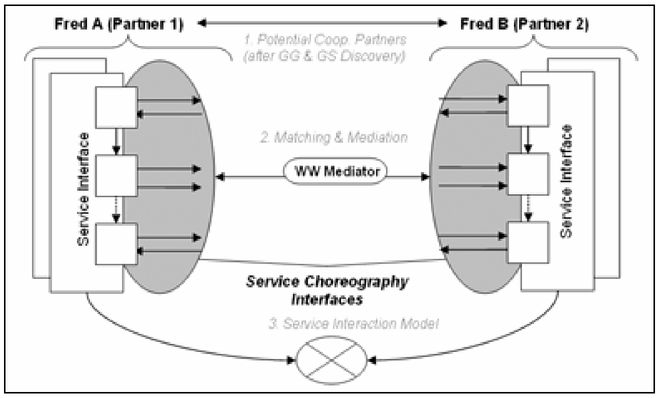
Fred A (Partner 1)
Fred B (Partner 2)
1. Potential Coop. Partners (after GG & GS Discovery)
Service Interface
2. Matching & Mediation
WW Mediator
Service Interface
Service Choreography Interfaces
3. Service Interaction Model

Figure 4: WW Discovery Overview

Determining the compatibility of Choreography Interfaces consists of two aspects: first, the workflow of the Service Choreographies have to compatible (process level compatibility), and second the expected messaging sequence has to be compatible (protocol level compatibility). The former is considered to hold if either the process models are the same (sequence of activities and transitions), or if one of them subsumes the other. For the latter, the Service Choreography Interfaces have to be symmetric. This compatibility check works on the formal model of Choreography Interface descriptions defined in [23]: each Choreography Interface is represented in an Abstract State Machine (ASM) [3], and their compatibility is determined by proving that the ASM representation of a service validates the ASM representation of a partner's service. For establishing a valid and deadlock-free interaction protocol for the services of cooperation partners, a WSMO WW Mediator can be included that resolves mismatches on the process and protocol level.

4.4. Service Execution

When the three discovery mechanisms are completed successfully, a *Cooperation Contract* is created. This contains all information needed for automated execution of the cooperative goal resolution: the Freds as the interacting cooperation partners, their particular Goal Instances that are to be resolved in the cooperation, and the Services to be used along with the interaction model derived in WW Discovery. Then, the Freds are called into *Meeting* wherein the Cooperation Contract is executed.

The SWF Service Execution module resolves the type of Service specified in the Grounding description, and calls the respective execution environment (see above). It monitors the execution process, including an error handling framework. This identifies specific types of errors (service not available, wrong or missing input, accessibility errors). If such an error occurs, the meeting for service execution is re-scheduled and the Goal Instance status is set to 'cancelled'; for all other errors, the meeting is aborted, and the Goal Instance status is set to 'not solved'. Also, alternative execution paths are conducted when a service fails, if suitable compensation information is specified in the service Grounding. When the service execution is completed successfully, the goal resolution is finished. The processing status of the respective Goal Instances is set to 'solved', and the Freds are withdrawn from the meeting, taking over their next tasks.

In a FRED Meeting Room, the services communicate via the FIPA ACL [7], while web protocols are used for communicating with external resources (HTTP or SOAP). The Meeting Room technology provides means for scheduling and managing meetings, described in [24].

5. Discussion and Related Work

The delineation of the SWF technology in the preceding sections exposes several similarities and connections to Semantic Web and Semantic Web Service technologies. This section reveals these, points out the contribution of SWF to the development of the Semantic Web, and positions it within related work.

5.1. SWF as a Semantic Web Environment

To make the vision of the Semantic Web become reality, the identified key technologies have to be integrated into a coherent system in accordance to their functional purposes and benefits. Apart from applying sophisticated technologies for realization, the success criteria for Semantic Web applications are support for reuse of and interchange with external resources via standardized formats, and employment of capable inference techniques for advanced processing of semantically annotated information [25].

The SWF technology presents such an integrated system. Agents perform tasks automatically as electronic representatives on behalf of their owner, interacting in a stable and scalable runtime environment. The Smart Objects technology allows using ontologies as machine-readable, semantically described terminology throughout the whole system. It provides secure and performant management facilities as well as import and export of ontologies via standard ontology representation formats for interchange with other Semantic Web enabled applications. The goal resolution technology realizes a mediator-orientated approach as envisioned for Semantic Web Services, amending the notion of cooperative goal resolution for more complex tasks. Reuse and interchangeability of computational resources such as service descriptions, goal specifications, and mediation facilities is supported by the alignment of the SWF Component Repository with the WSMO Repository, a general registry for Semantic Web resources.

A major merit of WSMO is the definition of an unambiguous and well-structured framework for Semantic Web Services, which is a prerequisite for effective inference mechanisms like those for establishing cooperative goal resolution in SWF. Although the matchmaking algorithms are improvable with regard to their technical realization, the SWF discoverers are concise inference mechanisms based on a well-defined description language for Goals and Services. The matchmaking facilities rely on approaches developed within WSMO, but their architectural assimilation into the SWF goal resolution technology presents the realization of a goal-driven approach within dynamic service usage. Thus, the main contribution of SWF to the development of Semantic Web Services technologies is the combination of emerging technologies into a coherent, working system for automated goal resolution.

In conclusion, SWF presents a prototypical environment for the Semantic Web that integrates key technologies into a coherent framework, as it does not exist at this point in time. The strong alignment with WSMO ensures association with WSMO-based techniques as well as compatibility with other WSMO-compliant applications. For substantiation, the following compares our work with related efforts.

5.2. Related Work

Working fields related to SWF are agent technologies with support for the Semantic Web, overall frameworks for Semantic Web Services, and approaches for inference mechanisms to handle Semantic Web Services. We briefly summarize promising efforts for each aspect, and discuss benefits and shortcomings in relation to SWF.

'TAGA – Travel Agent Game in Agentcities' (http://taga.umbc.edu/) is research activity aiming at development of an agent environment with support for the Semantic Web. Based on a FIPA-compliant agent framework, TAGA simulates the global travel market on the Web [29] within Agentcities (a worldwide network of agent platforms for intelligent service composition and execution over the Internet, http://www.agentcities.org/). In contrast to SWF, the TAGA system is an open environment, but it does not provide support for ontology usage. Web Services usage is restricted to WSDL, and the system does not provide means for dynamic service handling. Other established FIPA-based agent platforms are JADE [1] which applies ontologies as terminology definitions, and ZEUS [5] that implements a goal-driven approach for task resolution. Similar to TAGA, the support for Semantic Web resources in these systems is very limited.

Concerning overall frameworks for Semantic Web Services, the most relevant ones existing at this point in time are OWL-S [17], successor of DAML-S Semantic Web Services effort (http://www.daml.org/services/owl-s/), and WSMO [22]. OWL-S defines an ontology system for describing Web Services, comprised of three top-level notions: the *Service Profile* includes information for 'service advertisement' which is used for Web Service Discovery; the *Service Model* contains descriptive information on the functionality of a service and its composition out of other services, whereby the service functionality is conceived as a process; the *Service Grounding* gives details of how to access the service, mapping from an abstract to a concrete specification for service usage. Although OWL-S serves as a basis for various research and development activities, it is heavily criticized for conceptual weaknesses and incompleteness. Especially, the meaning of the OWL-S description elements is not clearly defined, leading to misinterpretations and incompatible models [14]. WSMO aims at overcoming these deficiencies by providing an unambiguous description framework for Semantic Web Services. Furthermore, WSMO is not restricted to Web Service description only, but includes additional top-level notions: Goals for representing user desires in a service-oriented architecture, and the concept of Mediators for handling heterogeneity. Thus, WSMO provides a more complete framework for aspects and challenges arising within Semantic Web Services [9], which rationalizes the choice for WSMO as the foundation of SWF.

Several approaches have been developed for Web Services Discovery, applying reasoning technologies. For example, [18] presents an approach based on subsumption reasoning of requests and OWL-S Service Profiles. A profile matches a request if all inputs and output of the profile are equal to or subsume those of the request, extended with notions for handling partial matches. An analogous approach applies DL subsumption reasoning based on $\mathcal{SHIQ}(\mathcal{D})$, a Description Logic with a semantics equivalent to OWL for ontology representation [15]. Although these approaches are very similar to the one used in GS Discovery (subsumption reasoning, handling of partial matches), they naturally inherit the insufficiencies of OWL-S for service description, as well as the absence of a clear concept for handling partial matches. Besides, both approaches only use T-Box reasoning (i.e. on the ontology schema level), while the WSMO-based technique used in SWF also considers instance level information.

With respect to WW Discovery, there are some approaches for determining behavioral compatibility. [4] introduces a formalization of WSCI, the W3C effort for describing Web Services Choreographies (not continued anymore). Single behavioral service descriptions are represented in the process algebra CCS, whereupon the compatibility is determined with regard to the order and the content of messages defined in the Choreography. Also, heuristic mediation facilities for establishing compatibility of a-priori not compatible Choreographies are outlined. Another approach for determining global interaction models for Web Services develops heuristics for a "Synchronizability Analysis" [10], similar to the compatibility analysis applied within WW Discovery. Apart from these efforts being of academic nature only, the formal processes representation used are questionable whether being appropriate for Choreography description [23].

6. Conclusion and Future Work

This paper has presented the Semantic Web Fred framework and technology for automated, cooperative goal resolution on the Semantic Web. We have outlined the motivation and system design principles, the technological realization of system components, and mechanisms developed for the goal resolution process.

The main contribution of the SWF technology is a concise framework that combines the key technologies of the Semantic Web into a system for automated and cooperative goal resolution with support for Semantic Web resources. An agent runtime environment allows delegation of tasks to electronic representatives, and emerging technologies based on the Web Service Modeling Ontology WSMO are applied in a coherent architecture for establishing automated goal resolution with dynamic service usage. Ontologies are used as the semantic data model throughout the whole system. Interchangeability with other Semantic Web resources is supported by import and export facilities as well as alignment with the WSMO repository.

The SWF technology is currently designed for a FRED environment, meaning that the interaction of agents and service execution takes place in Meeting Rooms of the FRED system. The aim for future development is to expand the SWF technology to a web environment, meaning that agent interaction and service execution is performed in virtual meeting rooms on the Internet. Therefore, only specific building blocks of the SWF technology have to be modified (namely the Meeting Room technology), while the overall architecture of the system can remain. This extendibility illustrates the quality of SWF in a nutshell, emphasizing it to be a prototypical environment for the Semantic Web.

Acknowledgement.
The SWF project is funded by Austrian government under the CoOperate programme 2003, and was awarded as the 2nd best proposal in the call. We like to thank the members of the WSMO working group for support and fruitful input to the work presented here.

7. References

[1] Bellifemine, F.; Rimassa, G.; Poggi, A.: *JADE – a FIPA-Compliant Agent Framework.* In: Proceedings of the 4th International Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agents, London, 1999.

[2] Berners-Lee, T.; Hendler, J.; Lassila, O.: *The Semantic Web. A new form of Web Content that is meaningful to computers will unleash a revolution of new possibilities.* In: Scientific American May 2001.

[3] Borger, E.: *High level system design and analysis using abstract state machines.* Springer Lecture Notes in Computer Science 1641, 1{43), 1999.

[4] Brogi, A.; Canal, C.; Pimentel, E.; Vallecillo, A.: *Formalizing Web Services Choreographies.* First International Workshop on Web Services and Formal Methods (WS-FM 2004), Pisa February 23-24, 2004.

[5] Collis, J.; Ndumu, D.; van Buskrik, C.: *The ZEUS Technical Manual, Release 1.04*, 2001.

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

[7] FIPA: *Agent Communication Language Specification.* Foundation for Intelligent Physical Agents, Geneva, 1997.

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

[9] Flett, A.: *A Comparison of DAML-S and WSMF.* Internal Report, Vrije Universiteit Amsterdam, 2002.

[10] Fu, X.; Bultan, T.; Su, J.: *Analysis of Interacting BPEL Web Services.* In: Proceedings of the 13th World Wide Web Conference 2004, New York, pp. 621-630.

[11] Herzog, R.; Zugmann, P.; Stollberg, M.; Roman, D.: *WSMO Registry.*, WSMO Working Draft D10, 26 April 2004.

[12] Keller, U. (ed.): *Inferencing support for Semantic Web Services. Proof Obligations.* WSML Working draft D5.1, April 15 2004.

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

[14] Lara, R.; Lausen, H.; Arroyo, S.; de Bruijn, J.; Fensel, D.: *Semantic Web Services. description requirements and current technologies*, International Workshop on Electronic Commerce, Agents, and Semantic Web Services, In conjunction with the Fifth International Conference on Electronic Commerce (ICEC 2003), Pittsburgh, USA, 2003.

[15] Li, L.; Horrocks, I.: *A Software Framework for Matchmaking based on Semantic Web Technology.* In Proc. of the 12th World Wide Web Conference, May 20-24, Budapest, Hungary, 2003.

[16] Oren, E.: *BNF grammar for WSML user language*. WSMO Working Draft D16.1, 05 April 2004.

[17] OWL Services Coalition: *OWL-S: Semantic Markup for Web Services*, version 1.0, 2004; available at: http://www.daml.org/services/owl-s/1.0/owl-s.pdf .

[18] Paolucci M., Kawamura T., Payne T.R.; Sycara K.: *Semantic Matching of Web Services Capabilities.* In Proc. of the First International Semantic Web Conference. Sardinia, Italia, 2002.

[19] Riazanov, A.; Voronkov, A.: *The design and implementation of VAMPIRE.* AI Communications 15(2), Special issue on CASC, pp. 91 -110, 2002.

[20] Ruh, W.A.; Maginnis, F.X.; Brown, W.J.: *Enterprise Application Integration: A Wiley Tech Brief.* Boston: John Wiley and Sons, Inc, 2001.

[21] Russel S.J., Norvig P., *Artificial Intelligence*, a Modern Approach, Prentice Hall. 1995.

[22] Roman, D.; Lausen, H.; Keller, U. (eds.): *Web Service Modeling Ontology - Standard (WSMO - Standard)*, WSMO Working Draft D2, 16 August 2004.

[23] Roman, D., Vasiliu, L.; Stollberg, M.: *Choreography in WSMO*, WSMO Working Draft D14, 17 July 2004.

[24] Stollberg, M.; Lausen, H.; Arroyo, S.; Herzog, R.; Smolle, P.; Fensel, D.: *Fred Whitepaper*. DERI Technical Report DERI-TR-2004-01-09, 2004; available at: http://www.deri.at/publications/techpapers/documents/DERI-TR-2004-01-09.pdf.

[25] Stollberg, M.; Lausen, H.; Lara, R.; Ding, Y.; Sung-Kook, H.; Fensel., D: *Towards Semantic Web Portals*. WWW 2004 Workshop on Application Design, Development and Implementation Issues in the Semantic Web, May 2004.

[26] Weibel, S.; Kunze, J.; Lagoze, C.; Wolf, M.: *RFC 2413 - Dublin Core Metadata for Resource Discovery*, 1998; available at: http://www.faqs.org/rfcs/rfc2413.html.

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

[28] van der Aalst, W. M. P.; Ter Hofstede, A. H. M.; Kiepuszewski, B.; Barros, A. P.: *Workflow Patterns.* In: Distributed and Parallel Systems 14(1), 2003, p 5 – 51.

[29] Zou, Y.; Finin, T.; Ding, L.; Chen, H.; Pan, R.: *Using Semantic Web technology in Multi-Agent Systems: a case study in the TAGA trading agent environment*. 5th International Conference on Electronic Commerce: Technologies, Pittsburg, 1-3 October 2003.