FRED Whitepaper

ABSTRACT

This paper presents the FRED system, a development environment for agent-based applications that utilize Semantic Web resources. The FRED system consists of an agent runtime environment based on ontologies as the underlying data model and offers a tool suite for application development. It uses progressive technologies for task-service-resolution that detect suitable problem solving implementations to solve tasks assigned to agents. Furthermore, FRED allows integration of external Semantic Web Services into the system, thus admitting the assimilation of key technologies which is compulsory for creating Semantic Web applications. We give an overall description of the FRED system by describing its technological solutions, explain the system functionalities and relate this to contemporary approaches. The aim of this paper is to expose requirements on agent platforms for the Semantic Web and to show how these are attained in the FRED system.

TABLE OF CONTENTS

1. INTRODUCTION

In (Berners-Lee et al., 2001), an example is used to introduce the idea of the Semantic Web wherein Pete and Lucy's agents automatically perform assignments for every day's life problems. For such Semantic Web driven applications, different technologies have to be conjoint, as stated later in that article. In this sense, the key technologies identified for the Semantic Web are intended to fulfill distinctive purposes: ontologies provide highly expressive, machine-readable semantics and thus allow semantically correct information interchange between Semantic Web enabled applications. In the example, the parties involved can exchange information because they rely on a common ontology. Via Semantic Web Services, implemented functionality can be accessed and invoked remotely over the Internet, like the appointment arrangement carried out by Pete and Lucy's agents with the doctor's service in the refereed example. The agents perform tasks assigned by their owners automatically by invoking services and cooperating with other agents as illustrated copiously in the example.

This example indicates that agents constitute the interface between their owners, which can be humans as well as machines, and existing services offered for solving a problem. With respect to the quality of tasks that agents shall be able to solve, exceeding requirements arise for agent-based systems which operate on the Semantic Web. In general, an agent platform comprises a run time environment where agents can interact and techniques for resolving assigned tasks to agents with the services existing. In the context of the Semantic Web, an agent platform additionally has to support Semantic Web technologies in order to facilitate information exchange and interoperability with other Semantic Web applications. More precisely, an agent platform for the Semantic Web has to be able to process ontology data for semantically enhanced information interchange and it has to support discovery, invocation, and execution of web services as well as their integration into its task-service-resolution technology. In doing so, the ontologies and the web services required for an application can be assembled and then be utilized by the agents in a system.

The FRED system realizes this concept and thus is applicable as a mediation platform for agent-based Semantic Web applications. It provides a stable and scalable run time environment for agents and advanced techniques for task-service-resolution similar to approaches followed in Semantic Web Services research at present. Ontologies are employed as the underlying data model throughout the entire system, and the Smart Objects technology

has been developed to handle ontology instance data. Thereby, the expressiveness of ontologies is transmitted into the system on the one hand, and, on the other, existing ontologies required for a specific application can easily be imported into the system. Furthermore, the FRED system allows invocation and integration of external web services as problem solving services into the system. In order to expose how the FRED system can serve as a mediator platform for agent-based Semantic Web applications, we present the technical components of the FRED system in detail and explicate the goal-driven approach followed for resolving tasks. We also relate these solutions to the state of the art in Semantic Web driven applications and depict future developments planned for improvements and further extensions of the FRED technology.

The FRED system is developed by Net Dynamics[1], a SME based in Vienna and run by experienced Software Engineers. Net Dynamics was founded in 2000, immediately starting to develop the FRED system. Net Dynamics has received awards for their innovative work around the FRED platform, namely the Mercur Award [2] in 2002 and Constantinus Award [3] in 2003.

The paper is structured as follows. Section 2 introduces the FRED agent runtime environment and the tool suite for application development; section 3 presents Smart Object, the solution developed for managing and handling ontology data in the FRED system; section 4 describes the task-service resolution techniques of the FRED system and the underlying concepts of Goals, Plans, and Processes; section 5 discusses the usability of the FRED system as a mediator platform for Semantic Web applications; section 6 depicts related work in agent systems as well as in ontology and Semantic Web Service technologies; section 7 presents existing applications built upon the FRED system and points out directions for future developments; finally section 8 concludes the paper.

---

[1] Net Dynamics Internet Technologies GmbH & Co KG, see: www.netdynamics-tech.com.

[2] Every year the Chamber of Commerce awards companies in Austria that develop superior innovative products and methods. Net Dynamics has achieved this award for it's "Fred Platform" product.

[3] The Constantinus Award is an important and well known award for consulting- and IT projects in Austria. Every year, the most outstanding projects, performed by the consulting companies together with their customers, are awarded. Net Dynamics, together with Wiener Stadtwerke have achieved this Award for the eCoach project.

2. THE FRED PLATFORM

This section presents the general architecture of the FRED agent platform and denotes the main building blocks of the system. We first explain the objectives underlying the design of the system and expose related approaches. Then we depict the system architecture with special attention to the runtime environment as the heart of the FRED system and the interfaces provided for connections to external systems. Finally, we present the tool suite offered by the FRED system for application development. The terms and notions introduced in the following will be referred to for the more detailed investigations in the subsequent sections.

2.1 Objectives and Design Principles

The main objective persecuted with developing the FRED system is to create a generic platform as a foundation for agent-driven applications that enable automation for different use case scenarios. The elementary functionality of the system is to allow users, which can be either humans or machines, defining an agent as an electronic representative that performs assigned tasks automatically on behalf of its owner.

These agents, called **Freds** in the system, can be defined for any purpose, from simple arithmetic calculations up to complex procedures like booking flights. In order to resolve such a task, there has to be an implemented functionality available which a Fred can invoke. The FRED system realizes a goal-driven approach for this detection of suitable existing services for a given tasks. Therefore, **Goals** are specified which are general descriptions of problems that can be assigned to Freds for automated solving. The task-resolving services are called **Plans** which encompass several problem solving services, from single programs in so-called **Plan Modules** over combinations of them up to invocation of external Web Services. Further, more complex workflows can be specified in so-called **Processes** which can also be invoked to solve a Goal. The **FredBase** is the agent runtime environment of the system wherein Freds can interact to solve their Goals. When a Fred has been assigned with a Goal, a **Meeting** is called in the FredBase with another Fred that has a corresponding Goal. For example, a Fred A has been assigned with the Goal ´`add 3 and 7`´ and a Fred B carries a Goal ´`add(x,y)`´, A and B will have a meeting wherein A's Goal is solved.

Upon this basic functionality, the intended purpose of the FRED system is to serve as a development environment for agent-based applications that allow delegation of tasks to

electronic representatives. Such solutions can be beneficial in several fields wherein automated task delegation provides surplus value for users. Given the capabilities of the FRED system in managing a given task autonomously by using the capabilities of semantics for "Intelligence" and the management power of agents, the potential applications areas can be divided into the following:

Table 1: FRED Application Areas

	Collaborative Process Automation	Mediation-based Integration	Reuse of Process Knowledge
Companies	Business processes which demand high flexibility and supervision	Demand driven Enterprise Application Integration	Business Process standardization and reuse
Single Users	User centric active process support which lead to simplified usage	One Stop Service tasks which benefit from Semantic Web Services Integration	Coach based Services like Competency Knowledge Management

Thereby, the delegation paradigm inherent in the FRED system provides benefits in the simplification and therefore significant cost saving for a lot of business problems of today. The functionalities of the FRED system, in particular automation of task execution and integration of external resources, facilitate much more advanced support than recent technical solutions. Existing applications are presented in section 7.1.

The technical design objectives of the FRED system are to establish an agent platform for effective and scalable collaboration of Freds and a suitable tool suite for application developers. With respect to the usage objectives of the FRED system illustrated, the requirements listed in Table 2 have determined the technical design of the system.

Table 2: FRED System Design Principles

Requirement	Description	FRED Component
Stable & Scalable Agent Runtime Environment	A runtime environment is the core component of agent-based systems for task automation. The major requirements for this are support and utilization of standardized technologies for agent interaction, as well as a framework for stability and scalability for sizable application support.	**FredBase** *section 2.2*
Connectability to External Systems	A major requirement for modern applications, especially for Semantic Web driven applications, is information interchange between diverse systems. Therefore, a reasonable development environment	**Fred Location Interfaces** *see section 2.2.3*

	should provide interfaces for standard technologies to allow connection, interoperability, and integration of existing information and external systems.	
Tool Support for Application Development	In order to support application development on top of the platform in a sophisticated manner, tools should be provided that allow creation and management of the essential system technologies.	**Development Tool Suite** *section 2.3*
Ontologies as Data Model	Ontologies are the state-of-the-art knowledge representation technology and thus should be used as the underlying data model in the system to exploit their benefits in terms of expressiveness and information processing. A supplementary functional profit is that existing ontologies can be uploaded into the FRED system, thus allowing reuse of ontologies.	**Smart Objects** *section 3*
Goal-Driven Task-Service-Resolution	The most important and most challenging feature of agent platforms is the resolution of tasks and services, i.e. how a task assigned to an agent can be solved by services available in the system. The FRED system therefore realizes a goal-driven approach wherein information on solvable tasks and the corresponding solving services are described in Goal, Plans, and Processes. These can be combined to solve a concrete task assigned to an agent, thus enabling reuse to a very high extent. Á similar approach is applied for discovery and composition of Semantic Web Services.	**Goals, Plans, Processes** *section 4.2, 4.3* **Goal Resolution** *section 4.4*
Support and mediation facilities for Semantic Web applications	To support Semantic Web applications, techniques for integrating ontologies and Web Services have to be provided. Furthermore, the platform needs to ensure that invoked resources are interoperable in order to allow automated task execution.	**Smart Objects** *section 3* **WSDLExecutorPlan** *section 4.2.2* **Mediation Facilities** *section 5.1*

2.2 Architecture

In the following we describe the agent runtime environment of the FRED system. A complete FRED system consists of a FredBase which holds the components of the agent platform and several interfaces for accessing the system as well as for connection to external systems. Here, we only introduce the most important building blocks which are referred to in the subsequent sections.

2.2.1 FredBase

The FredBase is the place where Freds, i.e. the agents in the system, interact to solve the tasks they have been assigned. A FRED-application may consist of several different Freds, and each one may fulfill a different purpose. So, a *Fred A* may hold the functionality that a *Fred B* needs to solve his task. For this interaction, Freds act together in so-called Meeting

Rooms (see below). Figure 1 shows a physical overview of the FredBase architecture with further explanations below.

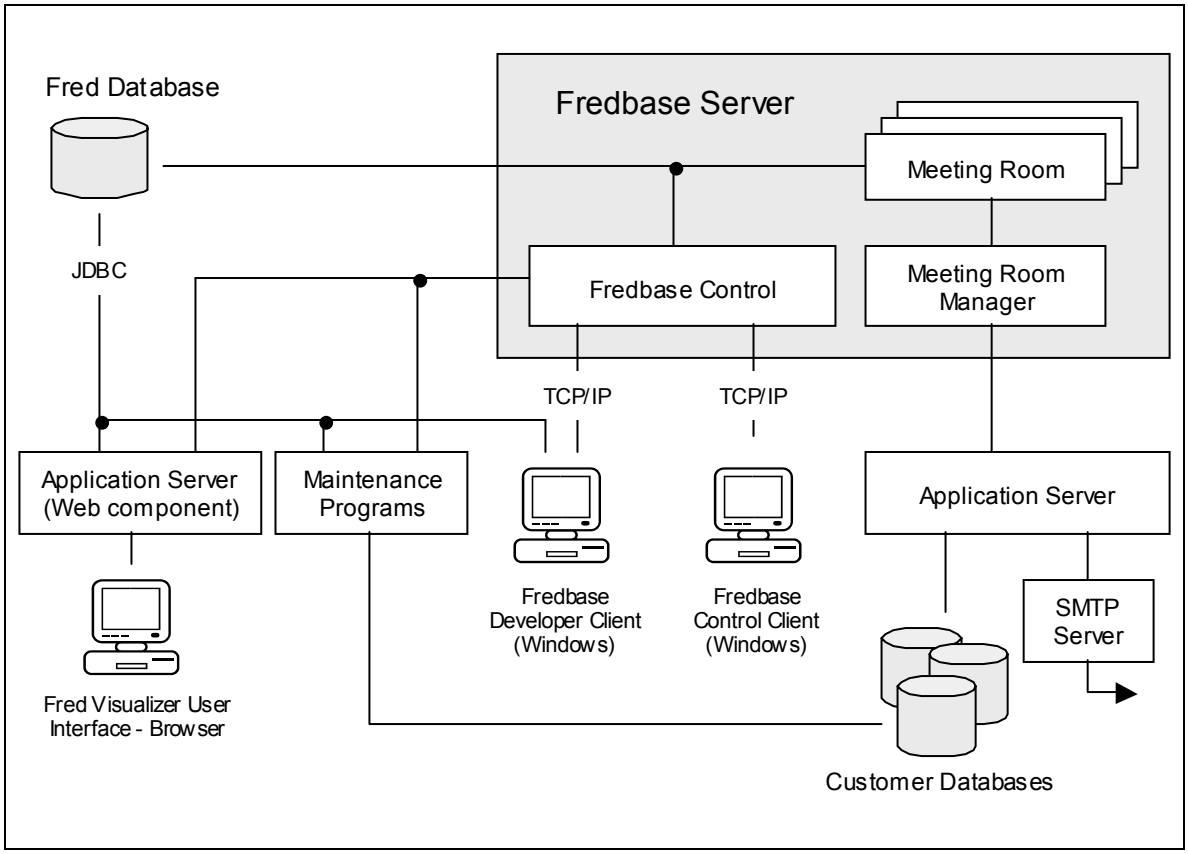

Figure 1: FredBase Architecture

The **FredBase Server** is the place where the **FredBase Control**, which manages interaction between Freds, is installed physically. The FredBase Server also comprises one or more **Meeting Rooms Manager** that maintains meetings between Freds (i.e. task execution, collaborations, etc. between the existing agents). The **Fred Database**, an conventional RDBS accessible via JDBC, stores Freds and all information related to them, i.e. the Smart Objects as the application information and customer related data. The **Application Server** implements the ´Fred Tools Connect´- interface and optionally connects to the Customer Databases and to an SMTP-Server for user interaction. The **Application Server (Web Component)** implements the user interface connection via the 'FredInform' interface (see section 2.2.3).

The FRED system is implemented in JAVA, making use of its functional benefits one the one hand and, on the other, to achieve technological consistence throughout the system. C++ is used in the FredBase Control and Visual Basic is used for User Interface implementations for

reasons of its particular technical benefits. In the following we explain the technical buildings blocks for execution and control of the FredBase in more detail.

2.2.2 FredBase Control

While transactions or dialogue oriented applications are usually started by user intervention, Freds are active all the time and can be invoked automatically. The management of agent interactions is performed by the FredBase Control. It provides the runtime environment which calls Freds into a meeting room whenever they need to interact top solve their assigned tasks and capabilities, ensures that Freds are withdrawn for the FredBase when they are not needed, and it provides a selection mechanism to determine those Freds out of potentially millions which are ready for meaningful processing. So the FredBase Control is the component which ensures stableness and scalability of the agent platform.

2.2.2.1 Meeting Rooms

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.

Goals are solved by executing the corresponding Plan or Process. The Meeting Room therefore provides the computational environment, consisting of two major elements. First it holds all services required for execution, for example a JVM for JAVA implementations and connections to the data repository to run Plans, or access to the Process Engine for execution of Processes. Second it provides the basic communication infrastructure through which agents exchange messages and information, called a Message Channel. The communication behavior of Freds is predefined in the Plan or Process that are to be executed, and the Message Channel is the environment for agent communication. While these components provide the technical environment needed to run a meeting, the Meeting Room Manager controls the execution procedure. It initiates a meeting by invoking the Plan or Process to be executed, returns the computational results to the meeting participants and releases the Freds after successful completion of the meeting. Furthermore, the Meeting Room Manager provides a moderation framework for meetings that handles exceptions in Plan or Process execution and solves contentions on interactions among the participating agents. It also ensures that all meetings in a FredBase are running logically completely independent of each other.

With regard to performance and scalability mandatory for real world applications, design principles for Meeting Rooms and affectionate elements are defined. In general, Meeting Rooms are designed for meetings with many Freds, so-called multiple-chair meetings. As most meetings are only with one or two participants and the rest of the chairs would be wasted, a Meeting Room can be spilt up and used contemporarily for as many meetings as chairs are available at a time. If a Fred does not hold a meeting at a time, it is withdrawn from the FredBase and serialized in the Fred Database. So only those Freds who actually interact at certain a time are hold in the FredBase which improves the scalability of the Fred system. A further convention is that a Plan should be written in a way that its execution in a meeting only occupies the Meeting Room for a short time, i.e. in the range of milliseconds. If a meeting is interrupted, for instance because user input required, it should be ended and re-scheduled for later.

2.2.2.2 Selection Engines

The decision which Freds are sent into a meeting is realized by several mechanisms, so-called Selection Engines. Common to all the mechanisms is that they select one or more Freds from the database, create a meeting suggestion, and then hand over this meeting suggestion to the scheduler mechanism for the Meeting Rooms. The technical operations mode of these engines relies on rules and heuristic patterns for detection of potential meeting partners. In order to allow developers to customize the selection functionality, the design of the FredBase allows adding and removing mechanisms for meeting selection dynamically. Table 3 shows the Selection Engines currently available in the FRED system.

Table 3: FRED Selection Engines

Selection Engine	Description
Seleng	This is the primary Selection Engine which selects Freds for a meeting. From a Fred's point of view, the meeting takes place by accident.
Event Manager	Event-driven selection of meetings. Triggering events can be timer events or external events.
Meeting List Manager	Creation of meetings as requested by invitations of Freds by a Plan or Process. That is, when a Fred holds a certain capability he might be called into a meeting to solve a Goal that requires this capability.

The employment and the operation mode of the Selection Engines will be investigated in detail in section 4.3 after the underlying technologies will have been introduced.

2.2.3 Interfaces

The FRED system provides two components to support the connectability to external systems and information sources requested for modern development platforms. These interfaces are not simple connection points for external systems but are designed as integrated units for managing data interchange between internal and external components.

The **Fred Tools Connect** enables access to external resources. It consists of JAVA-APIs and standard technology interface for access to Fred application databases, external customer databases, and other external systems like internet resources, eMail and SMS servers. These resources can be directly used in Meeting Rooms or the Process Engine (see section 4.3.2) via the Fred Tools Connect API. The **Fred Visualizer** is the central component for user interface management, comprised of components for driving user interfaces either server based (web application or browser) or client based (server functions for fat clients). The Visualizer allows concurrent support for several, possibly remote user interfaces to a FRED application. The application state is controlled centrally and then sent via SOAP (Mitra, 2003) to particular user interfaces, thus employing web services technology to ensure information consistency. Furthermore, a Validation Framework is attached to the Visualizer which is generated from ontology definition and constraint definitions (see section 3.2) to check a user's input against the specified constraints.

The Fred Visualizer components use Fred Tools Connect components as the Model for receiving the application state according to the Model–View-Control paradigm. Figure 2 shows the architectural configuration of the Fred Tools Connect and the Fred Visualizer.

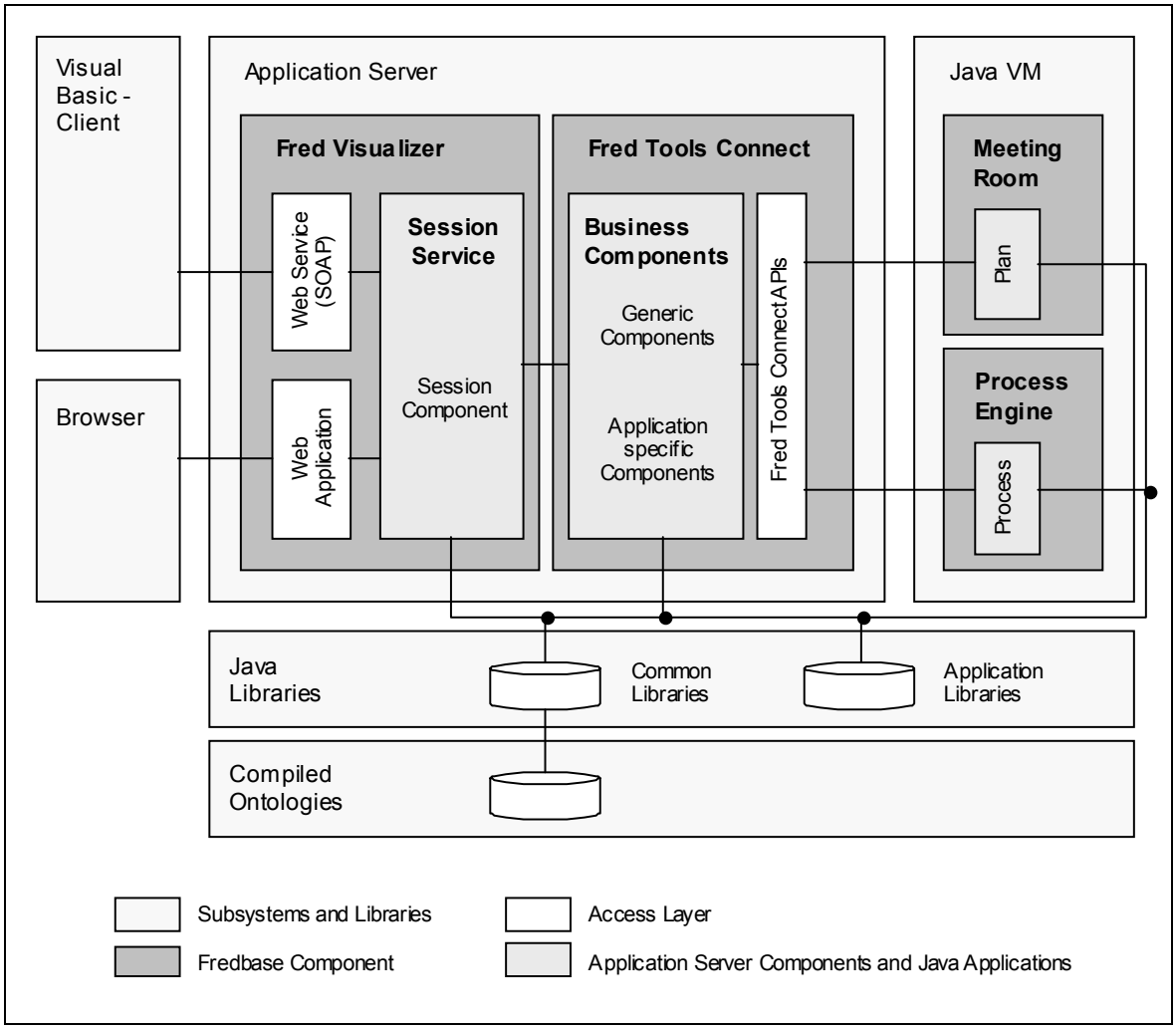

Figure 2: FRED Interfaces Architecture

Another interface provided by the FRED system is the so-called **Fred Link Interface** for interconnecting several FRED systems. This allows exchanging all kinds of FRED-specific information like Freds, Smart Objects, Goals or Plans in between various FRED applications without sumptuous transformation or loss of information. Thus, a 'virtual Fred World' can be created, following the concept of nested and interoperating system which is the general idea of Semantic Web driven applications.

2.3 FRED Development Kit

For application development, a Fred Application Programming Interface (FAPI) is available which wraps the Agent API (for manipulation of Freds), the Ontology API (for accessing and managing Fred's data), the Goal API (manipulation of goals) and transaction services for an interface programmer. Moreover, graphical development tools are provided to support application developers in building application upon the FRED system. They support control

of the FredBase, edition of Freds, Smart Objects (the knowledge basis), Goals and Plans as well as assignment of goals and plans for testing purpose. Here, these tools are listed and explained briefly while the underlying technologies will be investigated in more detail in the following sections.

Fred Control Client

This is the control client for the FredBase of a Fred Location. It allows starting and stopping a FredBase and provides information of the current state of a FredBase, i.e. how many meeting rooms are running, how many Freds they contain and the state of each Fred in the system.

Smart Objects Browser

The Smart Object Browser allows browsing and editing the Smart Objects of a Fred. A Smart Object is generated from an ontology as the knowledge structure definition on the concept level and holds ontology data at the instance level. All knowledge of Freds is based on Smart Objects. Smart Objects will be further investigated in section 3.

Goal Editor

In the FRED system, a 'Goal' is a task that a Fred shall fulfill. Goals are also based on Smart Objects. The Goal Editor assigns existing Goal Descriptions to a Fred for testing and problem verification during development. Before a Goal can be assigned to a Fred with the Goal Editor, the Goal Description as well as a corresponding Plan Description need to be created using the Description Editor. The goal-driven approach for task-service resolution will be further investigated in section 4.1.

Plan Editor

In the task-service-resolution technique of the FRED system, an implementation for problem solving can be a Plan. A Plan consists of one or more Plan Modules which are the elementary functional building blocks. A Plan Module consists of the problem solving implementation (which can be an internal implementation as well as an external service) and specifies the communication behavior necessary for the Freds to access and execute the Plan Module. Plans and Plan Modules will be examined in detail in section 4.2. The Plan Editor provides pre-defined JAVA code elements and templates for supporting Plan Module development.

Description Editor

This development tool is a graphical editing environment for Goals- and Plan Descriptions. These descriptions contain all necessary information (naming, versioning, functional and non-functional information) that are required to make Plans and Goals usable for Freds. These descriptions are specified in XML. A detailed explanation of the description language for Goals and Plans is provided in section 4.2.1.

Policy Browser

The Policy Browser allows defining security and privacy rules for Smart Objects. The idea behind this is that not everybody, i.e. not every Fred, should be allowed to read, write or change the Smart Objects of another Fred in terms of privacy and data security. The policy rules are directly defined on the properties of the Smart Objects in order to avoid unintended access to secure information. Policies can be defined on five different levels, starting from policies for the whole FredBase via policies for particular Goals and Applications up to restrictions for special meetings.

Process Composer

This tool supports workflow specification, called 'Processes' in Fred terminology. Processes are procedural definitions for the execution order of tasks (called 'Activities' in Fred). An Activity is a superordinate term which covers plans, external services, and non-automated actions that have to be performed by the user. Each Activity is described by input and output parameters which are based on Smart Objects and which determine possible activity compositions. The Process concept in Fred will be further explained in section 4.3.

3. SMART OBJECTS

In this section we introduce the **Smart Objects** technology which has been developed for managing and handling ontology data in the FRED system. Ontologies have been chosen as the underlying data model throughout the whole FRED system. The reason for this is to make use of the benefits of ontologies as a highly expressive, up-to-date knowledge representation technique in order to enhance the informational significance of knowledge carried by Freds. Furthermore, this approach allows integration of existing ontologies into a FRED-based application. Thereby support for Semantic Web application development is assured without disrupting the knowledge representation format from external ontologies into the internal system representation, thus ontological notions can be preserved. A Smart Object is an ontology instance compiled into a JAVA object that carries the ontological notions defined

for the corresponding concept. This allows storing and handling ontologies via conventional JAVA technologies which are more mature compared to currently existing ontology management technologies; besides, technology consistency of the FRED system as a JAVA-based platform is assured. Techniques for handling and management of Smart Objects have been developed which represent the all aspects required for ontology management in ontology-based applications. In the following, we explain the notions of Smart Objects and the transformation method for ontologies into JAVA objects. We further present the technologies for Smart Objects management and relate them to the state of the art in ontology-based applications.

3.1 The concept of Smart Objects

In order to avail the competence of ontologies as a highly expressive knowledge modeling and representation technique on the one hand and to make use of the potential for enhanced information processing, the complete data model of the FRED system relies on ontologies. Thereby, ontologies are developed outside the system and then compiled into JAVA-Classes, called **Smart Objects (short: SMO)**. During runtime of a FRED-application, Smart Objects as the knowledge base of the system are modified on the ontology instance level. Additionally, further technical constructs of the FRED system are derived form ontology definitions, demonstrating an exhaustive usage of ontologies with respect to the envisioned employment of ontologies in Semantic Web applications.

The mostly accepted understanding of ontology is “explicit specification of a shared conceptualization” (Fensel, 2003). This means an ontology is on the one hand a commonly accepted conceptual model of a domain which provides a terminology and an understanding of a domain for humans and on the other hand it can be specified in a machine-readable format and thus can be used as a data model for an application. Because of this, ontologies are considered to be the backbone technology for the Semantic Web for the reason that they allow highly expressive modeling of knowledge domains and present machine-processable semantics for information.

3.1.1 Types of Smart Objects

An ontology consists of the following notions (Chaudhri et al., 1998): *Concepts* describe the entities of a domain by their *Properties*, i.e. attributes that characterize an entity. *Relations* describe the associations between concepts, whereby subsumption and membership

relationships define the taxonomic structure of an ontology and other kinds of relations can be defined to identify further knowledge structures. An *Instance* is a concrete individual of a concept. To allow more sophisticated handling of instances, *Axioms* can be defined for logical constraints and coherences between concepts, properties and relations. With respect to this, the FRED system distinguishes the following three types of Smart Objects.

OOI – Ontology Object Instance

An OOI is an instance of an ontology concept. It contains all attributes defined for that concept in the corresponding ontology, so instantiating an OOI always gives a complete and exact representation of that instance. Further, an OOI carries the information about usage, ownership and modifications which appear during runtime.

Attributes (corresponding to ´Properties´ in ontology theory) are the place where specific values of a concept are stored which determine the current state of an OOI. Fred differentiates two kinds of OOI-Attributes: first the *simple-OOI-Attribute* which contains simple data type and second the *OOI-Relationship* which describe relationships between OOIs, hence the values of the OOI-Relationship are references to OOIs. These OOIs are either instances of the concept of the type of the relationship, or instances of any sub-concept of the type of the relationship. The expressiveness of OOIs will be described in section 3.3.

OOC – Ontology Object Constraint

An Ontology Object Constraint (OOC) is a logical expression defined for a certain ontology concept. This expression can be used as generic selection mechanisms in different scenarios. OOCs are a powerful concept since they allow defining logical expressions on OOIs which are reusable for different purposes. OOCs will be explained in more detail in section 3.3.2.

OOG – Ontology Object Group

OOIs and OOCs can be grouped together into an OOG for reasons of managing and organizing SMOs in a specific application context. OOGs only contain pointers to SMOs, thus an OOI may be contained in several OOGs, and an OOG can be contained in another OOG. OOGs do not fulfill any special technical functionality, but they support SMO management in applications with possibly numerous SMOs.

3.1.2 Fred Universe

Smart Objects are used as for representing all knowledge in the Fred system. That means that all features and components that utilize or process data use SMOs as their data format. Thus,

the knowledge described by the employed ontologies is referred to as the **Fred Universe** because the ontologies determine all information available in the system. More precisely, each Fred carries its knowledge as SMOs, the information in Goals, Plans and Processes – the task-service-resolution components in the FRED system – are described by SMOs, as well as every other construct in the system that uses or processes data. Thus, the Fred Universe ensures that the semantic affiliation of information and the consistency of data are guaranteed in a FRED application.

SMOs are generated from ontologies that are being built outside the Fred system. In the Fred terminology, the ontology developed outside the system is referred to as the "static aspect of ontology" and the compiled JAVA-SMOs, which represent the ontology instance data, are called the "dynamic aspect" (this indicates that only data at the instance level are used and modified during runtime). The FRED system offers an **Ontology API** for accessing and processing of SMOs, containing features for handling of OOIs, OOCs, and OOGs as well as exception handling and storing SMOs. The techniques for storage, maintenance for usage of SMOs will be described in detail in section 3.4.

In order to support sharing and exchange of Business Objects between different applications, so-called **Shared Smart Objects** are provided. Shared Smart Objects are ordinary SMOs which are not assigned by singles Fred only, but are hold in a Shared Container in Fred Tools Connect (see section 2.2.3). For each application one Shared Container is can be allocated which allows all Freds in that application to use the Shared Smart Objects. Distinct from Smart Objects, which are exclusively accessed through the Ontology API, Shared Smart Objects are stored as Enterprise Java Beans, thus accessible by every Fred via EJB methods.

This exhaustive usage of SMOs, respectively ontologies as the grounding data model throughout the whole system exemplifies the potential of ontologies as a modern knowledge representation technique. First, ontologies are a highly expressive knowledge modeling technique, secondly they provide machine-processable semantics for information, and thirdly they can be applied as the underlying data model for enhanced information processing and data consistency checking in an application via constructs like SMOs in the Fred system. These potential benefits are the reason why ontologies have been claimed as the basic building block for the Semantic Web. The major challenge for ontology-based system is to develop techniques which allow managing of ontologies one the one hand and making use of the complexity of ontologies in an application (Fensel et al., 2002b). We will present the respective solutions of the FRED system in the subsequent sections.

3.2 Smart Objects Compilation

The use of ontologies in Fred Base has two flavors. The conventional view of utilizing ontologies as the underlying data model for enhanced information processing on the one hand, and on the other to make use of ontologies as the common basis for the whole Fred system. This means that as many system components as possible as well as components needed for application development (APIs, SOAP message formats, plans, processes) are derived automatically from ontology definitions. This section describes the deployment process of Smart Objects and the generation of other components in the FRED system. First we explain the procedure and technologies for compiling ontologies into SMOs and then we specify the other technical components generated out of ontology definitions.

3.2.1 Smart Objects Generation

As stated before, the ontology schema data (called the ´static part of the ontology´ in Fred terminology) are created outside the system and then complied into JAVA-objects, the so-called ´dynamic part´ of SMOs. For editing the external ontologies, OntoEdit is used.[4] The ontologies are kept in the OXML-format, the in-house XML-representation for ontologies used by OntoEdit.[5] Then the OXMLCompiler, a part of the Ontology API, transforms the OXML-file of an ontology into JAVA objects which carry the ontological notions for each concept. These JAVA objects are the OOIs which are subsequently available as basis of the Fred Universe in an application. Figure 3 illustrates this process while we give further insights below.

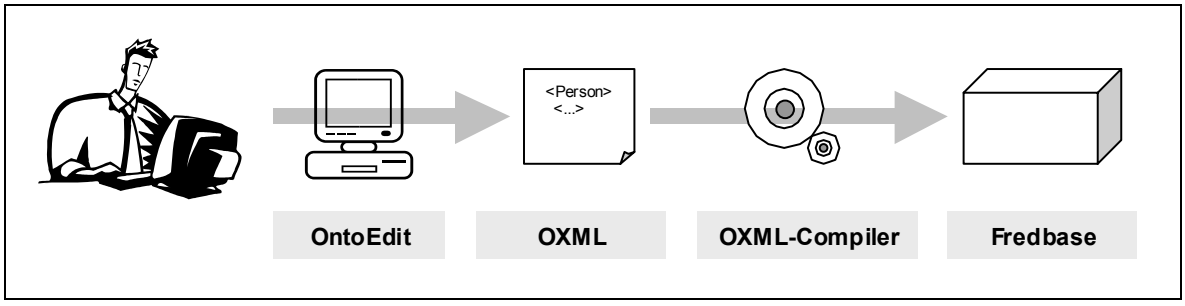

Figure 3: Ontology Deployment Process in the FRED system

[4] OntoEdit: Ontology Editor developed by Ontoprise GmbH, Karlsruhe (Germany) , see: www.ontoprise.de. In the FRED system, the professional version of OntoEdit is used, release no. 2.6.

[5] OXML has been designed as a proprietary XML-format for ontology representation in the OntoEdit because no standard representation format was available at design time of OntoEdit (Erdmann, 2003).

More precisely, the generation of SMOs out of the ontology specification in an OXML-file works like follows. First, the OXML-file is transformed into an XML Schema, short XSD-file. Into this XSD-file, application specific constraints not supported in OXML as well as further XML-datatype restrictions are added to the ontology specification. Then this extended ontology definition is used to compile the OOIs, the EJBs for Shared Smart Objects as well as the generation of other technical constructs (see below). The reason for this intermediate step via XSD in Smart Object compilation is that OXML is a proprietary, not easy extendable format while XSD allows application specific refinement of ontology definitions. Furthermore, existing XML-technologies can be used for further processing of the extended ontologies.

The Ontology Deployment process in its current implementation is obviously a proprietary solution for transforming ontology data into JAVA objects as OXML and XML Schema is used instead of standard formats like RDF or OWL. But apart from the fact that this can be easily changed by a generator for SMOs out of other XML-ontology-representations, the approach shows that compiling ontologies into JAVA-objects is a reasonable method to make ontology data handle by conventional and settled technologies.

3.2.2 Generation of Other Components

The result of the preparation process for compiling SMOs are numerous automatically generated technical building blocks and description files all serving as a base for using Smart Objects and Shared Smart Objects in Fred applications.

The generation of the technical building blocks is done in the Ontology Tower, the central ontology management unit (see section 3.4.1). The code packages generated out of the XSD-file apart from the OOI-Java classes are: SQL-statements for creating RDBs including object-relational mappings, complied EJBs representing Business Objects in Shared Smart Objects and providing access to them, compiled Visual Basic classes representing Smart Objects to be used in GUI-implementation, description files with rules for constraint processing in the Validation Framework (see section 2.2.3), WSDL descriptions for Smart Objects, and HTML Java documentation.

This demonstrates that the FRED system does not only make use of ontologies for knowledge representation and processing, but also utilizes them as the structuring principle whenever information management is concerned (in the storage, the business logic and the GUI tier).

Figure 4 summaries the utilization of ontologies as the grounding data model of the FRED system.

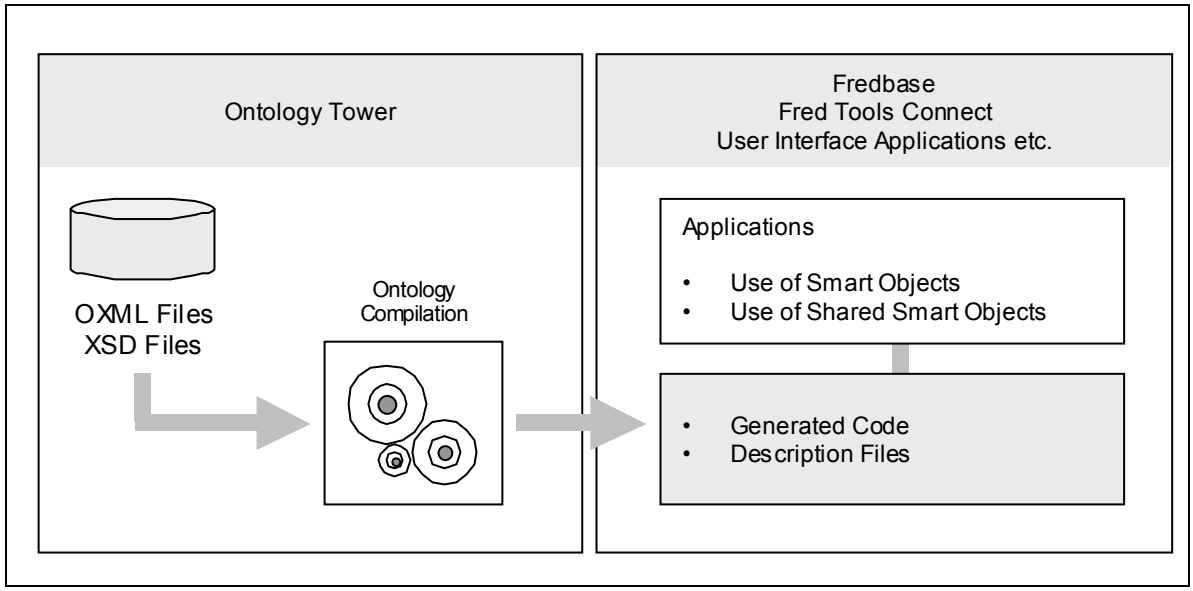

Figure 4: Ontology Compilation in FRED

The extensive usage of ontologies exemplifies the essentiality of ontologies in progressive systems because they do used to provide the underlying data model as the Fred Universe but also for structuring and generating other components of the system. An essential prerequisite for this technique is to preserve the expressive power of the ontologies which we will analyze in the following.

3.3 Expressiveness

A very important quality criterion for ontology-based systems is to what extent they make use of the expressive power ontologies provide for knowledge representation. In order to determine the ontological potency of SMOs we position them within conventional ontology representation languages. The expressive power of an ontology representation language is determined by the modeling primitives - Concepts, Properties, Relations, and Axioms – supported. In order to rank the expressiveness of Smart Objects as an ontology representation technique, we describe the ontological notions covered in the OXML-format, those which are transferred into Smart Objects by the OXMLCompiler and position this within ontology representation standards.

3.3.1 OOIs – Ontology Object Instances

OXML as the internal ontology representation format of OntoEdit comprises representation facilities for all ontological notions supported by OntoEdit (Sure et al., 2002). Apart from facilities for defining the taxonomic structure of ontologies, OntoEdit allows to define highly expressive axioms as F-Logic expressions.[6] Table 4 summarizes the ontology representation primitives complied into OOIs out of the OXML ontology definition.

Table 4: Ontology Modelling Primitives compiled into Smart Objects

Feature Group	Modelling Primitives
Concept-modelling (all of OXML)	- concept names (as unique identifier in the Fred system) - concept-hierarchy definitions (`subConceptOf`) - instance consistency information (`disjointWith`)
Properties & Relations (all of OXML)	- characteristics (name, range, domain, min. / max. cardinality) - logical Characteristics (transitivity, symmetry, inverse) - property / Relation hierarchy definitions (`subRelationOf`)
Instance Information (parts of OXML)	- concept membership - pre-defined attribute values and datatype restrictions - relation declarations
Additional information, manually added into XSD-file	- versioning information: current ontology version number - language Support: for English and German, descriptions - `<securityInformationGroup>`: data security information on accessibility - `<priority>`: specifies application importance of a concept / attribute (i.e. the importance of an attribute in terms of characterizing a concept or an instance) - `<required>`: specifies if a property or relation is mandatory - `<deprecated>`: define compatibility between instances of different ontology versions

Information not compiled is proprietary aspects that are used in OntoEdit as well as axiom definitions in the OXML-file. The former does not affect the expressiveness, while the latter causes the loss of axiom definitions. Integrity constraints and other logical expressions can be

[6] F-logic (Frame-logic): logic that combines the object-orientated paradigm and frame-based approaches for ontology specification and reasoning (Kifer et al, 1995).

added to SMOs by the concept of OOCs. As these are different from axiomatic expressions in ontology specifications, we discuss them separately in the following section.

In comparison to the standards for ontology languages recommended by the World Wide Web Consortium W3C, OOIs can be ranked to be equivalent to the one of OWL Lite in terms of expressiveness. OWL Lite (McGuiness et al., 2003) is a subset of the full OWL language constructors that allows defining taxonomic structures along with simple cardinality constraints (just 0 or 1). It provides the ontology modeling primitives shown in Table 5:

Table 5: OWL Lite Modeling Primitives

Feature Group	Primitives
RDFS-features	`class, property, individual, subClassOf, subPropertyOf, domain, range`
(In)-Equality	`equivalentClass, equivalentProperty, sameAs, differentFrom, allDifferent`
Property Characteristics	`inverseOf, TransitiveProperty, SymmetricProperty, FunctionalProperty, InverseFunctionalProperty`
Property Type Restriction	`allValuesFrom, someValuesFrom`
Cardinality	`minCardinality, maxCardinality, cardinality` *(each only 0 or 1)*
Class Intersection	`intersectionOf` *(for specifying composite concepts)*
Datatypes	`DataTypeProperty` *(uses RDF Datatype mechanism)*

It is important to note that these ontological notions are only those which are compiled into the OOIs. This means that the constituting basis of the Fred Universe contains facilities for expressing taxonomic structures, property typologies as well as additional, non-functional attributes. As mentioned above, the concept of OOCs allows to define further constraints and other logical notions upon OOIs, thus the expressive power of Smart Objects is to be rated as the one of OWL Lite plus axiomatic expressions.

3.3.2 OOCs – Ontology Object Constraints

OOCs are based on an ontology concept and their purpose is to filter and validate instance according to a certain criteria. We will give an example for illustration, based on the concept ´Car´. After applying the OOC, only cars with more then 100 PS will be valid. A simple path expression is used because the amount-attribute is not directly accessible in the concept Car, but via the concept ´Power´.

```
(hasPower.amount (ALL) > 100)
AND
(hasPower.hasMeasure (ALL) LIKE PS)
```

Listing 1: OOC Example

Out of this we can obtain the most relevant features of OOC, namely:

(1) Subsumption definitions according to the meta-data model of RDFS, as the OOC is defined using the Car concept but can be applied to all subsequent instances

(2) Path expression, as the condition expressed in an OOC is not limited to its own concept. Starting at the car concept it uses the relation `hasPower` and accesses its properties amount and `hasMeasure`

(3) Basic logical operators are provided, namely: `FULLFILS, VIOLATES, ASSIGNABLE, NOT_ASSIGNABLE, <, <=, >, >=, ==, !=`

(4) Operators for processing String-Values: `LIKE, NEAR`

(5) OOCs consider the cardinality of a particular relation (respectively property). The quantifiers ANY and ALL allow explicit specification how to handle the condition in case of a cardinality bigger then 1.

Further, the example demonstrates the usage of OOCs in the FRED-system. An OOC is defined for one ontology concept in the Smart Object Browser (see section 2.2.3). After creation, it is compiled into a Java Object Instance and handled the same way as OOIs. For processing OOCs, so called Predefined Constraint Processors (PCP) are defined which perform certain selection operations on the set of OOIs of the ontology concept the OOC is defined on. Currently, two PCPs are provided by the FRED system: the PCP 1 returns the set of OOIs of a specific OOG which confirm to the OOC (i.e. all ontology instances in an application context), while the PCP 2 returns all OOGs which wherein at least one OOI holds for the OOC. Referring to the example, the OOC can be used to pose a selection query on OOIs of the concept Car in terms of a query to the knowledge base by applying PCP 1. In another context, the PCP 2 can be employed for using the OOC as an integrity constraint stating that only OOGs are of interest wherein at least one OOI of the concept Car exists that has more than 100 PS. Further PCPs can be defined for supplementary selection mechanisms if requested.

The idea behind this approach is that an OOC is a logical snippet defined upon an ontology concept processable by PCPs which can be invoked in an arbitrary application context as

generic selection mechanisms to ease data handling. The treatment of OOCs as Java Objects and the execution of PCPs as conventional Java programs ensures performance of OOCs-processing since no intermediate machine for inferencing on OOC is needed which has been proven by testing OntoBroker (Fensel et al., 1998) as an inference machine for OOC processing. In terms of Deductive Databases, OOIs can be seen as the extensional knowledge base and OOCs represent the intensional knowledge, i.e. the knowledge that is deducible from OOIs. Regarding the expressive power, the OOC-concept is an implementation of propositional logical (logical expressions within predicates, functions and quantifiers), enriched with path expression facilities and means for processing string data (Ullman, 1998). An OOC is defined for one single concept, and a PCP only processes one OOC at a time while the combination of several OOCs is handled in the application logic. As propositional logic is decidable and the concept of OOCs and PCPs does not allow definition of nested expressions between independent classes, the OOC-processing technique is decidable and thus cannot interrupt the system by infinite loops.

With the concept of OOCs and PCPs, the FRED system provides a technique for enhanced ontology data processing on basis of logical expressions. The strength of this approach is the universal applicability and its ease of use since no complicated reasoning environment has to be considered in OOCs design. Although, due to the different technological realization, the approach is hard to compare with the model-theoretic techniques considered of inference-based ontology reasoning (Horrocks and Patel-Schneider, 2003) it enables logic-based support for enhanced ontology data processing, thus extending the expressiveness of Smart Objects with axiomatic capabilities.

3.4 Smart Objects Management

Management of Smart Objects comprises techniques for storage and maintenance of ontologies and SMOs as well as techniques for mediating SMOs, i.e. to allow interoperability of different SMOs. These technologies are very important for ontology-based applications in order to make ontologies usable at all on the one hand, and, on the other, to enable long-term usage of ontology data in the application as the corresponding knowledge requirements might change or evolve over time (Gómez-Pérez et al., 2003). In the following we explain the solutions for this provided in the FRED-system and relate them to the techniques used in current ontology-based applications. It is to note that the technical solution, although they

might not confirm to the latest trends and developments in this area, present an operating and integrated solution for ontology data handling as it rarely exists at this point of time.

First we investigate the storage mechanisms for ontologies and SMOs in the Fred system. A sophisticated storage technology should provide secure and scalable storage of data, and a proper query and retrieval mechanism. Further, maintenance techniques have to be supplied to allow updating of the knowledge base of the system. In the Fred system, separate techniques are used since ontology data are separated into schema definition, the ´static part of the ontology´, and the SMOs as the instance data, called the ´dynamic part´.

3.4.1 Ontology Tower

As stated before, ontology schemas (called “ontologies” or “static part of the ontology” in Fred terminology) are kept as OXML-files in the Fred system. The ontologies are maintained in the so called “Ontology Tower”. This is a central location where all ontologies are edited, stored, and the compilation into SMOs is performed.

Furthermore, a versioning mechanism is applied to allow ontology updating (Klein and Fensel, 2001). The versioning information is manually added to the OXML file. Confirming to current approaches in research, two kinds of changes are distinguished (Kiryakov et al., 2002). A minor change adds a concept to the ontology which does not affect the interoperability between different versions of the ontology. Thus a minor change is stored by simply overriding the old version of the OXML file. The second kind of change is a major change which is a deletion or modification of an ontology concept which might affect the interoperability between the new and the old version. Therefore a major changed ontology is stored in a new OXML file. The system checks the validity of manual versioning so that a major change can not be stored as a minor change.

The overall aim of ontology storage and maintenance systems is to support distributed, multi-user maintenance of ontologies with the purpose of enabling collective evolvement of an ontology as a shared conceptual model of a domain. The major outcome of an exhaustive survey of existing tools for ontology storage and querying in (Magkanaraki et al., 2002) is that currently available tools for ontology management are not mature enough to support this in a secure and scalable manner, thus none of them is appropriate to be employed in a commercial system. However, as ontology management solutions are needed in ontology-based system one way or the other, most existing applications apply a central component for ontology management similar to the Ontology Tower in the Fred system. Therein the

necessary ontology maintenance tasks are performed – i.e. persistent storage, a retrieval mechanism and methods for ontology updating. As long as there is no sufficient tool support, the correctness of all maintenance activities relies on human beings. This obviously is very hazardous as the correctness of ontology management activities determines the functional quality of the whole system – especially if ontologies are used as extensively as in the Fred system. But regarding the fact that the external ontologies are only used to define the schema for SMOs and are kept outside the FRED system, this dependence on human power of judgment can not hamper the system functionality directly.

3.4.2 Persistent Storage of Smart Objects

One of the major advantages of the Smart Objects concept is that SMOs can be stored as Java-Objects, thus conventional storage technologies can be applied. In the FRED system, all SMOs data are stored in the Fred Database (see Figure 1).

An essential requirement for sizable applications is to provide a scalable storage technology. The FRED system therefore implements a storage approach based on the "persistence by reachability" – paradigm which means that only those SMOs (that is OOIs, OOGs and OOCs) are stored permanently which are presently used. With respect to the ´Fred Universe´-principle, the Freds, i.e. the agents, are the components that hold knowledge of the system as SMOs. All knowledge of a Fred is enclosed in an **OO-Container**. An OOContainer has exactly one **Root OO Group**. The Root Group is stored in the Fred Database as the starting point a Fred's knowledge. The knowledge of a Fred is extended by adding new SMOs to the Root OO Group. As all SMOs belonging to the Root OO Group are stored permanently, only the actual knowledge of each Fred is stored permanently.

By this construct, only the currently needed SMOs data are stored one the one hand and, on the other, SMOs which need to be stored persistently are determined automatically during the system runtime. If, for example, a SMO ´A´ is referenced by another SMO ´B´ (through a relationship in SMO ´B´ with a value of OOI ´A´) and SMO ´B´ is persistent (because it is a member of the Root OO Group or of another OO-Group belonging to an OO-Container), then SMO A is automatically made persistent as well. In other words: SMOs are only made persistent if it is actually used by a Fred. Thus there are three memory states of SMOs (see: Figure 5)

(1) **Persistent Objects:** Persistent Objects are SMOs currently assigned to the OO-Container, thus persistently stored in the database.

(2) **Transient Objects:** Transient objects are SMOs which are needed during a meeting. After the meeting, they either become Persistent or Orphaned Objects.

(3) **Orphaned Objects:** Orphaned objects are transient objects no longer in use, and therefore not accessible.

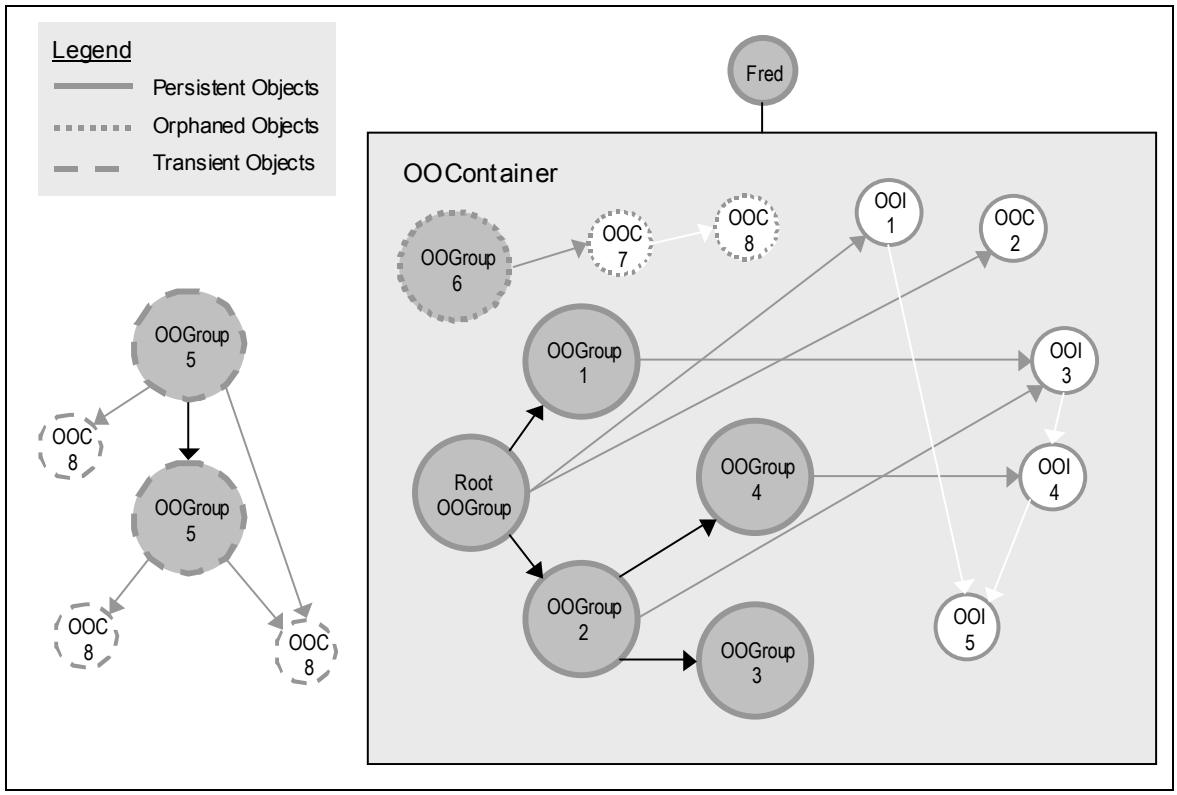

Figure 5: Persistency Concept in Fred

In this aspect, the Fred system differs from most of the other approaches in currently existing ontology-based systems by instantiating all ontology data into Smart Objects, the FRED system bypasses the need for scalable and reliable ontology instance storage and querying devices because settled techniques for storing Java-Objects can be employed. On the other hand, changes in the ontology (on the schema level / static part) require a re-compilation of the Smart Objects, eventually also realigning existing SMO-data to new ontology concepts. Techniques for enabling interoperability of different versions of ontologies and SMOs with regard to long-term usability of FRED-applications in evolving environments are discussed in the following section.

3.4.3 Smart Objects Mediation

The final aspect of the Smart Objects concept to be investigated is mediation of ontologies and SMOs, i.e. techniques to ensure the interoperability between different versions of ontologies and SMOs in the Fred system (Wache et al., 2001). A prerequisite for this kind of interoperability is that there are no irresolvable ontological mismatches, and the system must allow processing of data that belong to different versions of an ontology.

In ontology research, this field of study is referred to as “ontology integration” (Wache et al. 2001). The general approach relies on a so called “ontology algebra” (Wiederhold, 1994) which defines operations like intersection, union and difference for ontologies as known from mathematical algebra. Upon this, rules are defined to resolve the ontological mismatches and thus enable interoperability of ontology instance data that belong to different versions or completely independent, but interoperable ontology schemas. Generally, three strategies for ontology integration are distinguished. In so called “ontology mapping”, rules are defines to enable interoperability between two or more ontologies. The rules and the source ontologies are kept separated after integrating. In “ontology alignment” one of the source ontologies is transformed into the other, while “ontology merging” unites the source ontologies into one ontology which comprises all information of the source ontologies. Figure 6 shows the difference between these approaches. The choice for one of these techniques is determined by the application field. Remark that every strategy is defined on the ontology schema level and instances are transformed according to the rules defined (Noy and Musen, 2003).

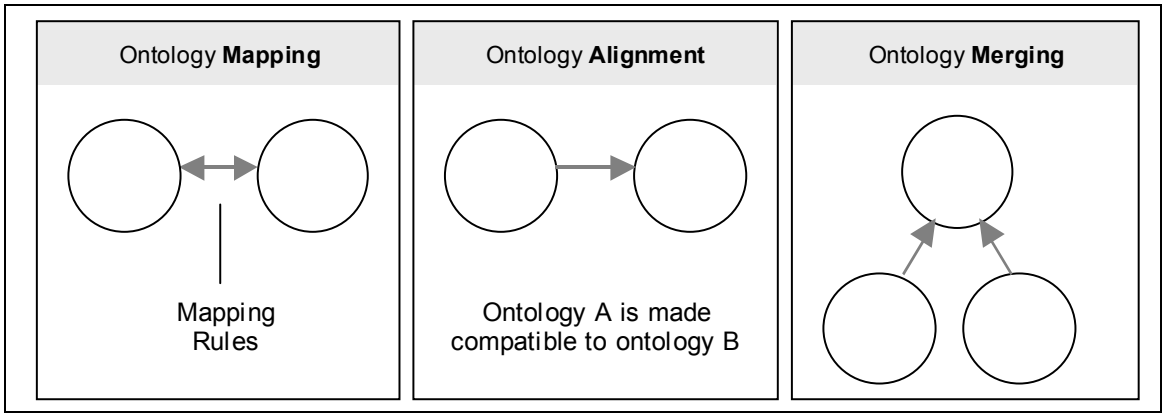

Figure 6: Ontology Integration Strategies

The Fred system provides a solution for solving this kind of interoperability problems whereby two strategies are implemented, called ´Mapping´ and ´Morphing´. The functional differences can be explained at best within small examples, starting with Mapping: Fred A

and Fred B want to communicate. If Fred B needs an OOI-instance of concept Person as input, the output of Fred A is an OOI-instance of Employee (sub-concept of Person). Then the missing information for the input of B (Employee) is added to the output of A (Person) via mapping rules. In Morphing, the complete OOI describing a Person is transformed into an OOI of Employee, which might be needed when as a consequence of a major version change. Thereby all properties including their values have to be transformed. Morphing is also realized by mapping rules. It is important to note that these strategies implemented in the Fred system do not perform integration of ontologies on the schema level, but transformations of SMOs, i.e. on the instance level. With regard to the three ontology integration strategies, mapping and morphing are both ontology mapping strategies since they are realized via mapping rules and they do not change the schema definitions of the source ontologies.

Mapping and morphing are realized by the same technologies. The integration rules are specified in XSLT[7], which then are either generated into JAVA-classes that are invoked during the runtime of the system (usually used for mapping) or they are used for a so called a ´batch´ transformation wherein all SMOs are of the specified concept are transformed. Therefore, the SMOs are serialized into an XML-document (using a specific XML-Schema for that defines the data structure necessary to represent all aspects of SMOs, called ´OO-XML´), then the XSLT-rules are applied to the XML-file, and then the SMOs are recompiled into the FredBase. This technique is usually used for morphing when a huge number of SMOs needs to be transformed. As for the other ontology management aspects presented, there are no sophisticated tools for ontology integration at this point of time. Thus, the FRED system – as well as other ontology-based systems – has been forced to create a proprietary solution to achieve interoperability of SMOs.

4. TASK-SERVICE-RESOLUTION

A major challenge for agent-based systems is to discover an appropriate problem solving service for a task that an agent is assigned to solve. For this resolution of tasks and services, the Fred system realizes a goal-driven approach which is a contemporary methodology also applied in technologies for discovery of Semantic Web Services.

---

[7] XSLT (Extensible Stylesheet Language Transformations): Language for transforming XML-Data Structures. W3C Recommendation, see: http://www.w3.org/TR/xslt

This section illustrates the concept of goal-driven resolution, introduces the underlying technical constructs and describes the techniques for resolving task-service-resolution supplied in the FRED system. We first illustrate the goal-driven approach, and then explain the concept of Goals and Plans as well as the initial solution for simple services and the Process-based approach which allows definition of more complex problem solving services. Finally, we present the technologies for Goal Resolution in section 4.4.

4.1 The concept of Goal-Driven Resolution

The idea of goal-driven system is to specify solvable user needs in a declarative way as pre-defined goals and similarly describe the problem solving capabilities available in the system with an operational specification. A selection mechanism is then mapping between both descriptions and initiates the execution of appropriate program code for a specific request. So goal-driven systems contain a mostly hierarchical ordered set of predefined goals along with operators which describe the relations between the goals, implemented services for solving problems, and selection mechanisms for dynamic determination of suitable implementations for a goal (John and Kieras, 1994).

The advantage of goal-driven systems in comparison to transaction-driven or user-dialogue-driven approaches is first that goals and services can be removed, added, or modified dynamically without changing the functional architecture, and second that particular goals and services can be reused for different application scenarios. Goal-driven approaches have been applied in several application fields. For the resolution of tasks and service implementations, Problem-Solving Methods (short: PSM) have been developed which rely on a goal-driven approach (Fensel, 2000). A PSM is a reusable reasoning pattern which decomposes a complex goal into subtasks that are solvable with the existing problem solving services, thus allowing automatic detection of suitable services for a given goal to a high extent. The idea of PSM is also applied in techniques for discovery and composition of Semantic Web Services, wherein so-called capabilities describe solvable goals and appropriate Web Services are determined by reasoning on the declarative descriptions. These descriptions are based on a web service description ontology to ensure semantic consistency, thus bringing up the concept of Semantic Web Services (Fensel et al, 2002a).

In the FRED system, three technical constructs underlying the goal-driven resolution of tasks and services are defined. According to the common arrangement of goal-driven systems, a **Goal** represents generic user goal which can be assigned as a task to a Fred. The service

implementations existing in a FRED application can either be **Plans** or **Processes**. Plans are implementations for some simple functionality, while a Process contains a more complex workflow procedure which can itself be comprised of several plans, external services or sub-processes, thus allowing recursive reuse of existing services to solve different Goals. Figure 7 illustrates the interrelations of Goals, Plans and Processes: a Goal can be solved by a Process, wherein the Process consists of several steps which themselves can be solved by a Plan, a SubProcess, a manual activity, or by another Goal that can be solved arbitrarily.

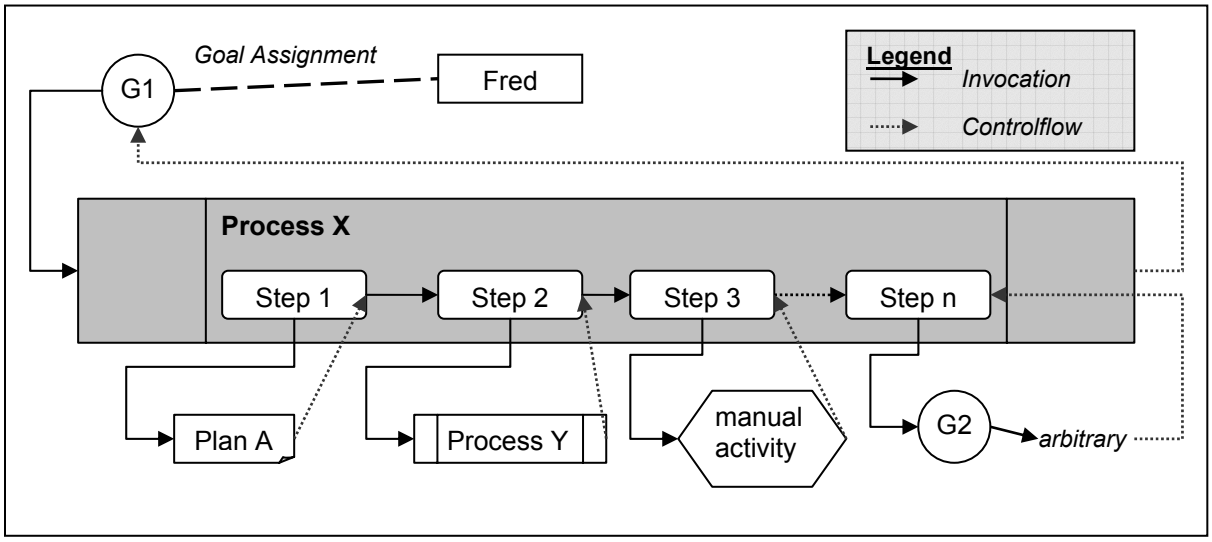

Figure 7: Goals, Plans and Processes for Task-Service-Resolution

This approach allows defining generic fragments for tasks to be solved in a Fred application, thus making use of the benefits of goal-driven systems. The motivation for this approach is that Plans and Processes can be re-used for different scenarios in an application on the one hand and, on the other, that resolution of tasks can be handled by the same technology impartial of whether the task has been assigned to a Fred by its owner or during system runtime. In the remainder of this section we first explain the concepts of Goals, Plans and Processes in more detail before we address the techniques for goal-resolution in section 4.4.

4.2 Goals and Plans

At first we delve into the concept of Goals and Plans as the foundation of the task-service-resolution technique in the FRED system. We explain the structure of Plans and their association to Goals and show how external web services and be invoked as Plans in a FRED application.

Freds determine the activity of a FRED application. This means that meetings and thus system activity can mainly achieved when some Freds have been assigned with Goals so that they will be called into a meeting. The Goals assigned to a Fred are selected from a predefined set of Goals, whereby a Fred may only use goals that are defined in his policy profile because of security reasons. As outlined above, a Goal can be solved by associated Plans. A Plan can consist of one or more **Plan Modules** which hold the concrete problem solving implementations, which can be either a simple JAVA class, an aggregated resource, or a web service. During a meeting, the Goal assigned to a Fred is solved by executing the corresponding Plan, respectively its Plan Modules. For Plan execution, the Meeting Room (see section 2.2.2.1) provides the computational environment and execution control.

4.2.1 Goal, Plan and Plan Modules Description

A pre-requisite for identifying resolving services for a given goal in goal-driven systems is a appropriate description language. The FRED system currently uses a rudimentary XML-based language for this purpose. We record the descriptions elements for Goals, Plans and Plan Modules in Table 6. These descriptions are edited in the Description Editor (see section 2.3). All these descriptions are stored in XML-format because of the ease of use of XML for describing data structures like this. These descriptions can serve a basis for an inference-based extension of the Plan discovery.

Table 6: Description Elements for Goals, Plans, Plan Modules

Component	Description Elements	Explanation
GOAL	*Name*	primary identifier in system
	Version, Creation Date	secondary identifier
	Application	FRED application that Goal is developed for
	Non-functional	incl. portray, keywords
PLAN	*Name*	primary identifier
	Version, Creation Date	secondary identifier
	Link to Goal	name & version of Goal that is solved by this plan
	Link to Plan Modules	name & version of 1..n Plan Modules that hold actual implementation
	Non-functional	incl. portray, keywords
PLAN MODULE	*Name*	primary identifier
	Version, Creation Date	secondary identifier

	Name & Path of JAVA-class	pointer to actual problem solving implementation
	FIPA Protocol Type	FIPA protocol used by Plan Module[8]
	Input & Output	In- & output data of Plan Module, described as SMOs
	Compatible Plan Modules	Plan Modules that can be combined with this one

The description elements listed clarify the coherency between the components. A Goal describes a generic user goal which can be assigned to a Fred. The Plan Module description links to the problem solving resource, provides functional information of a Plan Module by it's in- and outputs (which are described as ontology data), and specifies how the module can be utilized in a meeting between agents. The Plan description realizes the linking between user goals and the problem solving services, i.e. between Goals and Plan Modules. In order to enhance reusability of resources, a Plan Module can be assigned to several Plans and a Goal might be described by different Plans. In a nutshell, a Plan represents the intermediate connection between user goals and the problem solving implementations in the sense of a PSM (see above).

In the current status of implementation, the connection between Goals, Plans and Plan Modules is hardwired. This means that for each Goal, the corresponding Plans and their Plan Modules are specified explicitly at design time and not algorithmically determined during system runtime. In order to automate the detection of suitable Plans for a Goal, only the concept a Plan needs to be replaced by rule-based techniques while the other description components can be kept. Nevertheless, the concept of Goals and Plans allows reuse of existing constructs for different application scenarios in the sense of the goal-driven approach and can be extended to inference-based discovery of suitable Plans for a Goal.

4.2.2 Deployment of External Web Services

After having introduced the concept of Goals and Plans we now can explain how external Web Service can be integrated into a Fred Application. Therefore a special Plan, the so called **WSDLExecutorPlan**, is provided which allows execution of an external web service via its WSDL-description.

---

[8] FIPA-ACL primitives used: FIPAQueryInitiator, FIPAQueryParticipant, FIPARequestInitiator, FIPARequestParticipant, FIPAContractNetInitiator, FIPAContractNetParticipant see: http://www.fipa.org/specifications/index.html

The Web Services Description Language WSDL (Chinnici et al., 2003) is an XML-based language for describing the access interface of a web service. WSDL contains descriptions of the required input and output data of a Web Service, on its technical accessibility and on supported protocols, thus comparable to a Plan Module description in the FRED system. Figure 8 gives a self-explaining overview of WSDL.

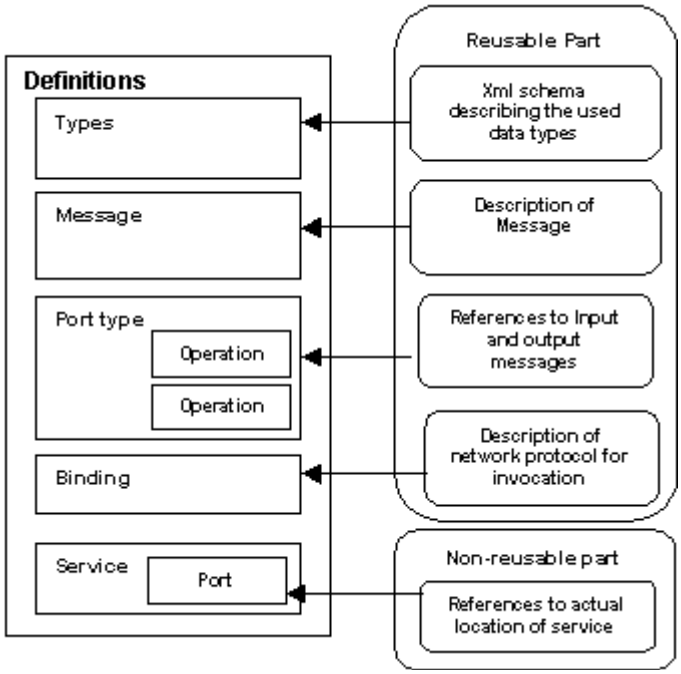

Figure 8: Web Service Description elements in WSDL

As this information is not sufficient for automated discovery of web services, more expressive description models for web services are developed in current efforts in research like DAML-S (DAML-S, 2003) and WSMF (Fensel et al., 2002b). As these technologies are not maturely developed, WSDL is mostly used for describing web services at this point of time. Because of this reason, WSDL is also supported for deploying external web services into the FRED system.

As pointed above, a WSDL description only contains information on input and output data and on the binding, i.e. the technical accessibility of a web service. As this description does not supply sufficient information on the functionality of a web service to be integrated into a FRED application, the functionality is inspected manually by the application developer for incorporation of a web services as a Plan Module into the system. Then, the web services can be invoked via the Fred Tools Connect (see section 2.2.3). This manual integration of external web services of course is very far away from the overall aim pursuit by research efforts in the Semantic Web Services area, i.e. to discovery web services automatically, to

compose them dynamically, and to conduct necessary mediation during runtime (Fensel et al., 2002b). But at this point of time, there neither is an environment realizing this vision in a mature manner nor are there sophisticated web services in an adequate number which could be used as a functional basis for an application. The WSDLExecutorPlan must thus be comprehended as an intermediate solution that allow integration of web services – as an upcoming species of application component technology – into a Fred application that will be further elaborated in the future.

4.3 Goals and Processes

Apart from Plans, Goals can also be solved by Processes (see Figure 7). Therefore, the FRED system provides a workflow-based technology for specifying more complex problem solving procedures, also referred to as business workflows.

A process defines execution sequences of services for some repeatedly appearing task which is considered as an important requirement for life-size applications. Therein, different kinds of problem have to be solved which are highly interrelated and dependent on each other. First, problem solving services have to be put into a reasonable order regarding the sub-goals solved by each service, second the defined sequence should be easily executable according to the transitions between the invoked services (i.e. the output of the former service should be applicable as input of the following service), and third the execution of a workflow composed of several problem solving services should be transaction-secure.

Generally, there are two oppositional approaches for specifying workflows. The so called procedural approaches define pre-defined, static workflow steps wherein the problems mentioned above can be caught during design time of a system. The drawback of these approaches is that they are static, so that dynamic changes during runtime of the system are not adaptable (at least not to a very high extent), and a workflow has to be defined for every application scenario that shall be supported by the system. The second group, so-called “declarative approaches” provide descriptions of goals and problem solving services in a system and the workflow is determined dynamically through logic-based inferencing on these descriptions. Declarative approaches overcome the static and application scenario dependent drawbacks of procedural approaches, but they request enormous efforts for ensuring the functional requirements of workflow systems (Bussler, 2003).

4.3.1 Process approach in FRED

The Fred Process approach relies on a procedural approach. The reason for this is was detected that a procedural solution accomplishes the requirements of Fred-based applications in a better way since already existing workflows, as there are for many application scenarios, can more easily be transferred into procedural process definitions than into declarative ones. As outlined above, Processes in Fred are highly interconnected with the concept of Goals and Plans and allows recursive definition of processes. Furthermore, a Fred Process can be comprised of different kinds of activities which can be either invocations of implemented functionality, sub-processes, or manual activities.

The foundation of the Fred Process concept is the approach proposed by the Workflow Management Coalition, a union of over 300 industrial members aiming at defining standards for workflow management.[9] The workflow implementation model proposed by the WfMC identifies the components required for a workflow system (WfMC, 1995). Table 7 lists the model components and states the corresponding component in the FRED system.

Table 7: WfMC workflow model components and realization in the FRED system

WfMC model component	Fred tool
Process Definition Tool tool to create the process description in a computer processable form	**Process Composer** *section 4.3.3.1*
Process Definition process modeling primitives, incl.: constituent activities and rules for navigation, possible user tasks, references to invoked applications, definition of any workflow relevant data	**Process Ontology** *section 4.3.2*
Workflow Engine run time execution environment for a workflow instance **Workflow Enactment Service** A software service that may consist of one or more workflow engines in order to create, manage and execute workflow instances. Applications may interface to this service via the Workflow API (WAPI). Control can be realized by a state transition machine or other decision making technique. The Workflow Enactment Service interprets process descriptions and control process instantiation, sequencing of activities, adding work items to user work lists, etc. It also maintains internal control data, links organizational entities and roles with specific participants and may access external applications (document creation etc.). **Workflow Control Data** internal control data (internal state, checkpoints, recovery/restart) **Workflow Relevant Data**	**Process Engine** *section 4.3.2*

[9] See: http://www.wfmc.org

data to determine the state transition of a workflow process instance and data needed for workflow generated by applications	
Application Data application specific data not accessible by the workflow system, i.e. data that will only be manipulated by the application.	**Freds, resp. their Smart Objects** *section 3.1.2*
Worklists Items to be processed in the workflow system	**Fred, reps. their Goals** *section 4.2*
Worklists Handler Managing the interactions with participating elements	**Plans / Plan Modules and Meeting Rooms** *section 4.2*

Further, WfMC has introduced the Workflow Process Definition Language WPDL as a formal language for specifying workflows according to the WfMC Process Reference Model. There is an XML-Serialization of WPDL called XPDL (XML Process Definition Language) which allows exchanging workflow specification data between XML-enabled applications (WfMC, 2002).

4.3.2 Process Ontology and Process Engine

Based on the WfMC Reference Model for Process Definition, a Process Ontology defines the modeling structure for workflow specifications in the FRED system. Figure 9 shows the taxonomic structure of the Process Ontology, further explanations of the concepts are given below. For specifying workflows in the FRED system, XPDL is used.

```
ProcessObject
      Process
            AtomicProcess
            SubProcess
      ProcessInstance
      Activity
            RouteActivity
            BlockActivity
            RegularActivity
                  LoopActivity
                  SubflowActivity
                  NoImplActivity
                  ApplActivity
      ActivityInstance
            RouteActivityInstance
            BlockActivityInstance
            RegularActivityInstance
                  LoopActivityInstance
                  SubflowActivityInstance
                  NoImplActivityInstance
                  ApplActivityInstance
      Transition
      TransitionInstance
      InternalProcessInformation
      InternalActivityInformation
      ProcessInformation
            ControlData
```

```
            CaseData
                  DocumentProxy
                  MediaProxy
                  ApplicationProxy
                  UrlProxy
      Participant
            OrganisationalUnit
            Role
            Person
            VirtualPerson
            Resource
      ParticipantInstance
            OrganisationalUnitInstance
            RoleInstance
            PersonInstance
            VirtualPersonInstance
            ResourceInstanceh
```

Figure 9: Fred Process Ontology Structure

A **ProcessObject** identifies all process objects. An **AtomicProcess** describes the general process while all occurrences of Processes are modeled as a **SubProcess**. The most important concept is an **Activity** which describes the action carried out at a certain step of the process. An Activity is always related to a **Participant** which can be a *Person*, a *VirtualPerson* (i.e. a Fred) in a virtual organization, a *Role* representing a function executed by a Fred or a representative of another organizational unit (*OrganisationalUnit*). This means that Fred Processes are not restricted to definitions for execution orders of technical constructs like Goals, Plans or other Processes, but can contain activities of user-interaction as well. The Activity sub-concepts describe the behavior of the process after execution of an Activity, for instance the *LoopActivity* defines a recursive repetition of an Activity and *SubflowActivity* calls a nested process. A **Transition** comprises conditions for transitions between Activities. When a Process is instantiated, all these concepts become Instance-concepts which hold additional information needed for execution of the process (e.g. a *RegularActivity* which describes a certain state in a process specification and becomes a *RegularActivityInstance* when its corresponding process object get instantiated). Besides the concepts for process description, so-called **ProcessInformation** hold control information for process execution as well as *CaseData* as links to process-external information resources.

The Process Engine executes processes and controls the execution procedure. It deals with *ProcessInstance* data only, i.e. with effectively existing processes. In order to prevent separation of functionality, the Process Engine only controls process execution while handling of application data is left to the meeting rooms where Freds interact (see Figure 10).

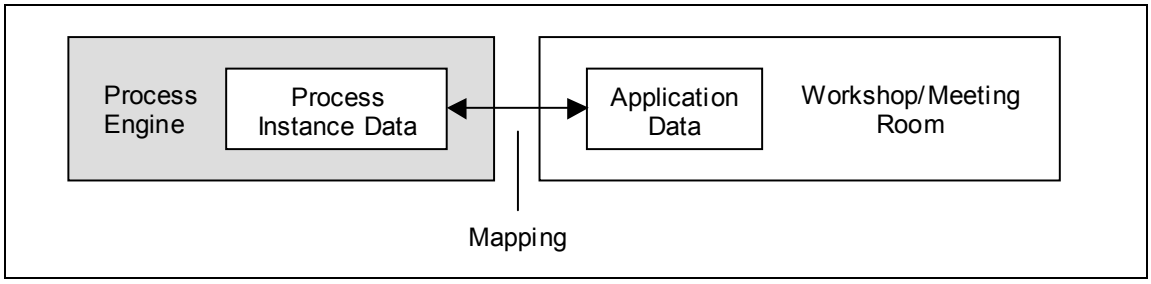

Figure 10: Position of Process Engine in the FRED system

According to the requirements for process management and execution settled by the WfMC (see Table 7), the Process Engine implements a state transition machine for process execution control which relies on a final state model for processes. The management of processes is implemented as an event-driven approach, wherein process execution can be determined by conditions for activities and transitions specified in the XPDL-process-description, by time constraints, and by internal process control restriction. Thus processes can be operated independently, which means they can be invoked and run without unrelated to the state of Freds in a meeting room which offers broader possibilities of automation in an application.

4.3.3 Process Specification and Execution

Fred Processes can be created with the Process Composer. Besides this, the FRED system provides a co-called Activity Manager which allows application users to control their Freds, delegate tasks to them, and provide user-data input if required in a process. We will briefly describe these tools.

4.3.3.1 Process Composer

The Process Composer is the Process Definition tool that an application developer uses for defining a Process. It relies on the concepts defined in the Process Ontology. We will explain the Process Composer by a small example.

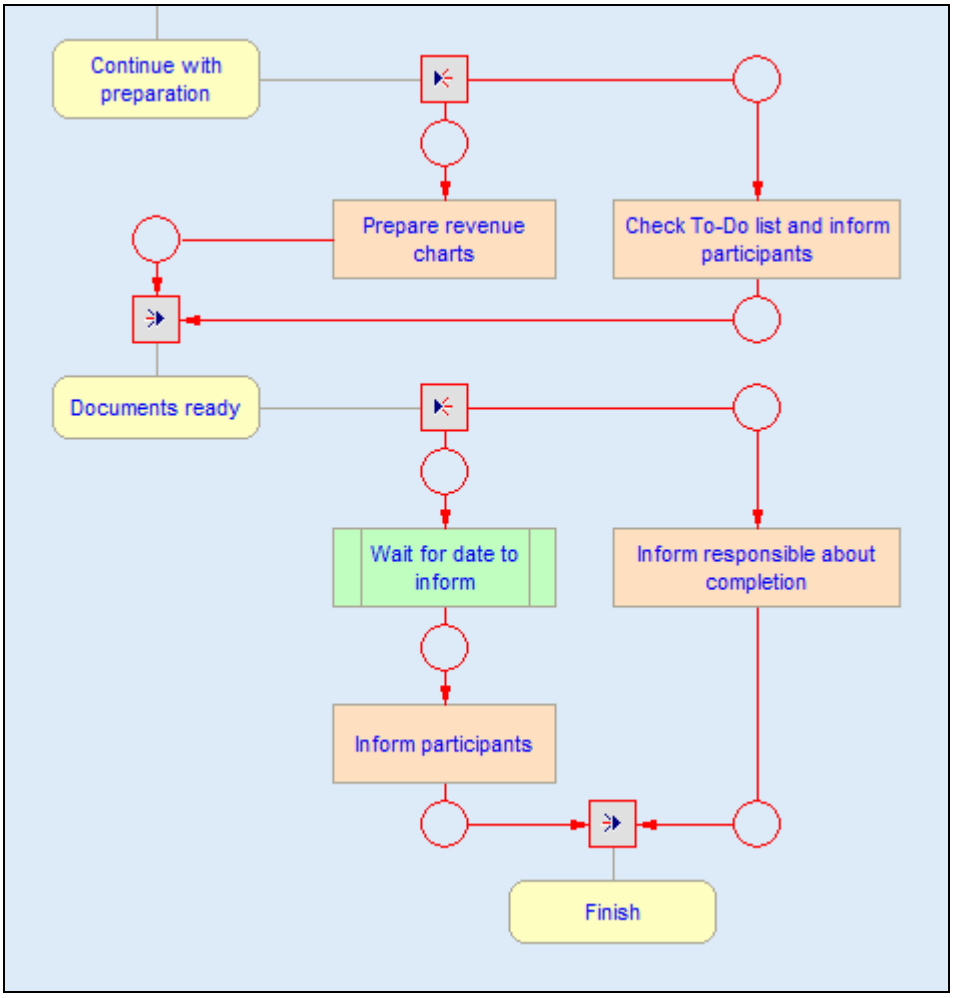

Figure 11: Fred Process Example

The Process shown in Figure 11 describes the procedure of preparing business document. The yellow boxes represent process states, which can be achieved by performing activities. An Activity is denoted by the squared boxes whereby a red box is an automatic activity processable by a Plan, a Sub-Process or an external Web Service via the WSDLExecutorPlan (see section 4.2.2). A green box describes a manual activity, in the example the user has to provide the date for informing the participants, i.e. co-authors of the document.

The strength of the FRED Process technology is that a Process can be comprised by automated activities as well as manual activities. In the example, a second process would be needed in order to allow user interaction if manual activities were not covered in the process specification. Another important feature of the Process technology in the FRED system is that Smart Objects are utilized as the application data. More precisely, OOIs are applied as the input and output data of Activities and OOCs are used to define conditions for Transitions. Thus, Processes are based on the SMOs existing in the system which is

beneficial in several aspects. Firstly, process data are semantically enhanced whereby richer expressiveness is enabled than by using primitive data types. Secondly, the consistency of information is guaranteed as this is an implicit feature of SMO, and thirdly it enables the reuse of information. The usage of ontology objects for workflow specifications is currently not supported by any other process definition tool.

4.3.3.2 Activity Management System

The Activity Management System is a framework for structured communication between a FRED application and its end-users. It consists of tools for user-interaction with agents, which allows assigning tasks to a Fred and performing user-system interaction when required.

The proceeding of a FRED-based system is determined by Freds as they hold the tasks assigned by users and they solve them during meetings. As mentioned above, Freds not only execute implemented services like Plans and Processes for solving a Goal but also allow interaction with humans when required for executing a Fred Process. This allows supporting a broader range of real-world problems as there are many routine procedures which can be automated but require human intervention at some certain points. The major benefit of the FRED system is that such activities are integrated in the system wherein agents maintain and track the co-operation of human and automated activities.

For communication between users and a FRED-based application mainly three different situations are distinguished. The first one is when a user has to fulfill a manual task outside the system, for instance signing a document. The FRED system only needs a notification in order to proceed with the process. The second one is when a human decision is required; therefore the possibilities and constraints have to be presented to the user and the human decision entered determines the further course of the process. The third and most complex one is when a user is requested to feed information from external into the FRED system. Therefore, the requested information has to be determined, an edition facility must be provided, and the information has to be adopted into the Fred system. These procedures further have to be considered when designing a system within user-interaction.

Three tools are currently provided for Activity Management in the FRED system. The Project Assistant Fred which allows developing work plans which can consist of manual activities to be executed manually by human users as well as automated activities. A work plan is assigned to a Fred who controls the execution. The second tool is the Fred Activity

Management Entry (FAMe), a simple web-based Activity Management which allows users to define activities that do not have a personal Fred. Thirdly, the Fred eMail Connector provides a simple eMail based Activity Management for formally integrating any person (who has got eMail capabilities) into a Fred process.

4.4 Goal Resolution

After we have introduced the technological foundations for task-service-resolution, we will now explicate the techniques for solving tasks that have been assigned to a Fred. The FRED system therefore distinguishes three aspects of goal solution. In a chronological order, the first one is the determination of Freds as potential meeting partners according to the Goals that are assigned to Freds, referred to as Meeting Creation. The second one is the detection of appropriate services to solve a specific Goal, referred to as Service Discovery, and the third one the handling of failures during goal resolution due to unavailability of a service, called Execution Handling in the FRED terminology. In the following, we discuss the requirements for each of these resolution problems and present the solution provided by the FRED system.

4.4.1 Meeting Creation

This aspect of Goal Resolution is to determine which Freds are sent into a meeting in order to solve their respective Goals. Following the goal-driven approach, Freds are goal-orientated, which means that a FRED application can only solve tasks when a Goal has been assigned to a Fred. Thus, the question to be addressed here is how to determine corresponding Goals that have been assigned to different Freds and to initiate a meeting between these Freds.

The idea underlying the current solution for meeting creation in the FRED system is Compatible Plans. As enlisted in Table 6, a Plan Module Description contains a field for specifying the Plan Modules compatible to the one described. As Plan Modules, Plans, and Goals are explicitly connected via their descriptions, Compatible Plans and thus Compatible Goals can be determined over the Compatible Plan Modules. If such Compatible Goals are detected to be assigned to different Freds, these Freds are called for a meeting. The compatibility of Plan Modules is defined by the application developer at design time. For instance, a Plan Module that resolves equations of the type "`add(x,y)`" is declared by its designer to be compatible with another Plan Module which contains an invocation for "`(int a) + (int b)`".

Meeting creation in the FRED system is performed by three different machines. The first one is the **Seleng** (short for ´Selection Engine´) which permanently searches all existing Goals Instances for Compatible Goals that are assigned to different Freds and unsolved at that time, and instigates a meeting between the Freds. The second engine is the **Event Manager** which implements an event-driven approach for meeting creation. Therein, concrete meeting partners are pre-specified, that means that it has been derived before during system runtime or it has explicitly defined by the application developer, which is triggered by an event. An event can be timer controlled or an external event coming into the system. The third engine engaged in the meeting creation process is the **Meeting List Manager** which mainly organizes meetings and sends them into a Meeting Room for execution (see section 2.2.2.1). An initiated meeting carries further descriptions specifying its priority and its flavor, which is how the meeting has to be executed as well as conditions for its finalization. The Meeting List Manager checks these conditions, optimizes the meeting list and schedules its execution in the Meeting Room. In addition, the Meeting List Manager can itself initialize a meeting when it is required to solve another Goal from its application context.

The basic technique of Compatible Plans underlying the meeting creation in the FRED system is, in its current implementation status, hard-wired according to the possibilities offered by Goal and Plan descriptions (see section 4.2.1). By extending the descriptive power of these descriptions, enhanced techniques for determining potential meeting partners can be employed in the FRED system. An important finding during the development of the meeting creation engines is that a general engine for permanently checking the complete system for possible meeting partners like the Seleng has proven to be not scalable. Because of this reason, a mixture of engines has been developed which search for meeting candidates in an event-driven manner or in an application context.

4.4.2 Service Discovery

The second aspect for Goal Resolution is the discovery of an appropriate problem solving service for a given goal. As stated above, the aim in research and developed activities in this field is to define a powerful description language for goals and services in order to support enhanced techniques for automatically and dynamically discovering suitable services to solve a goal.

As discussed in the preceding sections, the description language for Goal, Plans, and Plan Modules currently used in the FRED system is very elementary and the determination of a

Plan Module as a problem solving service in hard-wired in the Plan Description (see section 4.2.1). For Fred Processes, there is no declarative description and thus dynamic discovery technique at this point of time and Processes are invoked from the application context. In conclusion, the techniques currently employed in the FRED system are hard-wired as thus to be seen as very rudimentary in terms of automation and dynamic Service Discovery. Nevertheless, the essential technical constructs needed to for advanced discovery techniques are existent and can be extended into more convenient solutions without changing the complete architecture.

4.4.3 Execution Handling

The third aspect to be covered in Goal Resolution assuring the correct and faultless execution of goals or services, respectively. This encompasses solutions for exception handling for Goal execution, especially rollback mechanisms. Appropriate frameworks for Execution Handling are an essential requirement for goal-driven applications because unavailable or not executable services can affect the functionality of the complete system. This problem is multi-leveled as a reliability of a system depends on the low-leveled, technical execution of services has to be assured as well as all other levels up to the application logic from the user perspective. In general, there are two execution handling methods: rollback techniques (also referred to as compensation) and ´building-round-about´ which means to determine a different service execution sequence to solve a goal.

The support for execution handling provided in the Fred System is a compensation solution included into the application development support. In the templates and the API for Plan Module development, the application developer is requested to provide a compensation procedure for the case of unavailability of the Plan Module or any other execution exception. For Process Execution, such a compensation procedure is requested in the invoking application context while the Process Engine provides the rollback mechanism for the complete Process. Although these techniques confer the actual compensation solution to the application developer, they provide an Execution Handling solution for the technical level which ensures the stability of a FRED application.

5. FRED AS MEDIATION PLATFORM FOR THE SEMANTIC WEB

In the antecedent sections we have introduced the technical building blocks of the FRED system. On this basis we now investigate in detail why the FRED system can serve as an

agent-platform for Semantic Web enabled applications with special attention the mediation facilities of the FRED system.

The basic requirements for agent-systems to support Semantic Web applications as proclaimed introductory in this paper are at first a suitable agent runtime environment for automated task processing by agents, secondly support for ontologies as the grounding data model to ensure semantically correct information exchange between involved parties, and thirdly support for invocation of external Web Services and their integration into the system as processing functionalities. While these three requirements are accomplished by the FRED system, i.e. the FredBase provides a stable and scalable agent runtime environment (see section 2.2), the Smart Object technology facilitates employment of ontologies as the data model underlying the complete FRED system (see section 3), and external Web Services can be integrated as problem solving services and be automatically invoked via the WSDLExecutorPlan (see section 4.2.2), the question addressed in this section is which kind of Semantic Web applications can be supported by the FRED system in a expedient manner.

The general functional aim of the FRED system is to enable delegation of tasks to automated representatives, i.e. Freds. In such applications, like, for instance, the Semantic Web scenario referenced in the beginning of this paper, another technical requirement arises: the technological components have to be able to interoperate. In a FRED application, this means that Freds have to ´speak the same language´ and agree on a communicative behavior to be able to interact successfully in a meeting. Technologies that enable interoperability of potential heterogeneous information resources or functional services are commonly referred to as **Mediators**. Such a mediator does not wrap the content or representation format of a resource which is to be integrated into the system, but it provides techniques on a higher level which enable interoperation between heterogeneous resources (Wiederhold, 1992). In the following, we examine this approach in more detail and explicate how the FRED system, apart from supporting ontologies and web services, can serve as a mediation platform for Semantic Web applications.

5.1 Next Generation Mediation Systems

Mediation is a multi-faceted field because the interoperability between resources can be hampered by heterogeneity in different aspects. In order to provide satisfying a means for mediation, a sophisticated system has to provide an integrated solution for different mediation levels. Thereby, a mediator system should provide it services as a central point

accessible by the resources to be mediated. Moreover, it is important to remark that the challenges arising in mediation are of infinite complexity and automated support for these tasks can and should only be provided up to a certain, reasonable extend while human intervention is required for the remaining, more complex mediation issues.

The mediation to be covered in open and distributed systems of potentially heterogeneous problem solving services, like web service enabled applications, can be differentiated into three aspects. At first, *data mediation* which is concerned with enabling syntactically and semantically correct information interchange between information resources, secondly *process mediation* which covers mediations between dissimilar business logics that are used by different services, and thirdly *protocol mediation* for handling divergent communication behaviours (Fensel et al., 2002a). For data mediation, techniques are needed for enabling mediation between different syntactical representations of information, for mediation between different semantics of data, and for mediation between different conceptual models that underlie the information structures. Therefore, ontologies and especially ontology integration techniques are considered to be a sophisticated mediation technology because they enable these mediation tasks to a reasonable extent (Wiederhold, 1994). The challenge in process mediation is that mismatches in process sequences have to be resolved. Therefore, business processes have to be described in a higher-leveled language which allows detecting mismatches and determining the degree of interoperability between heterogeneous processes. For protocol mediation, the communication behavior of interacting services has to be rendered interoperable, wherefore semantic descriptions of the message exchange are needed to allow detection and resolution of mismatches.

In conclusion, the essential requirements for a mediator system are to provide high-level description facilities for data, processes, and protocols to allocate techniques for detection and resolution of mismatches in these descriptions. Thus, the technological building blocks of a mediator system are a data integration facilities, a process model along with a process engine, and message protocol processing facilities. Recent systems that comprise these building blocks are so-called Enterprise Application Integration systems (EAI) which mostly provide monolithic solutions for integrating various information resources and thereby enabling interoperability between heterogeneous resources (Bussler, 2003). For integrating an information or service resource into such an EAI-system, it has to be aligned to the system's data, process and protocol models in order to achieve interoperability with other resources connected to the system. This means that the integration in these systems is not performed

inside the system but during the preliminary alignment of the resource to the system structure, thus these systems do not supply mediation technology but information processing facilities with rigid integration support.

In contrast to this, the architecture of the FRED system allows mediation-oriented integration. Therein, Smart Objects represent the data model, Goals, Plans and Processes are the processing facilities and protocol mediation is performed by the agent-communication protocols. The integration is achieved by the Goal Resolution technologies and the Meeting Room Control so that integration is performed inside the system during runtime. Thereby, the mediation techniques enable interoperability by higher-leveled descriptions of data and service resources without interfering the concrete application logic. Further benefits of the FRED system are at first that it allows automated task processing via agents and secondly that it is flexible in terms of supportable application fields because no application specific functionality is predetermined in the system. Systems that provide such kind of mediation facilities are also referred to as next generation mediation systems because they overcome the limitations of recent integrations technologies in terms of mediation during system runtime without domain restrictions for applicability (McCann and Manning, 1997). In the following, we explicate the mediation facilities for Semantic Web applications of the FRED platform in more detail.

5.1.1 Plan Level Mediation

Resources are only interoperable when they are interoperable in all three mediation aspects mentioned, i.e. when they are interoperable according to their data structure and the business logic and their communication behavior. In the FRED system, only Freds are called into a meeting whose interoperability is secured. The mediation facilities are provided by the Meeting Manager and its preliminary technologies for Goal Resolution. We explain how interoperability is achieved and assured for Goals that are resolved by Plans, thus called plan level mediation.

The resolvability of a Goal at the plan level requires interoperability at three positions: firstly, the ontologies applied have to be interoperable as a common data structure, secondly the Plans that are to be processed have to be compatible as a reasonable application logic, and thirdly the communication behaviors of the Freds have to be intermateable in order to ensure successful information interchange during a meeting. Therefore, various elements of the FRED system need to cooperate. The interoperability of ontologies is mainly controlled by

the application developer who either ensures that interoperable ontologies are used in an application scenario or employs the functionalities of Mapping or Morphing, respectively, to transform Smart Objects with respect to the required data structure (see section 3.4.3). The compatibility of Plans is also established at design time; while it is automatically checked by the Selection Engines in the Goal Resolution process during system runtime (see sections 2.2.2.2 and 4.4.1). The communication behavior of agents is defined in the applied agent communication protocols specified in the Plan Module description (see section 4.2.1). Before a meeting between Freds is executed in the Meeting Room, the Meeting Room Manager checks the interoperability of used ontologies and the correspondence of the respective communication protocols (see section 2.2.2.1). Thus, the complete interoperability is assured and the application functionality, i.e. solving a Goal assigned to a Fred, is performed in a meeting.

So, a Meeting Room in the FRED system is the place where mediated resources are executed. The quality of the mediation support depends on the mediative power of the incorporated FRED components which is to be graded as relatively limited in the current implementation status. Nevertheless, the approach realized seems to be promising: there are separated components for different mediation tasks whose output is tested for interoperability and the Meeting Room Manager determines the processability of resources according to their overall interoperability. This architectural concept determines complete interoperability required for automated processing of resources, it allows mediation independent of a concrete application context, it incorporates technologies for runtime mediation based of higher-level descriptions, and it is flexible in terms of extendibility of the applied mediation technologies.

5.1.2 Process Level Mediation

For mediation of business procedures, here referred to as process level mediation, further mediation technologies are needed. Apart form the interoperability of ontologies and the correspondence of communication protocols, the compatibility of processes has to be determined. Similar to other mediation aspects, a suitable description language for business processes is needed which allows detection of mismatches and their resolvability between different processes as well as a technique for processing these descriptions (Kayed and Colomb, 2002).

At this point of time, there is no suitable description language for process mediation employed in the FRED system. By extending the process description capabilities sufficient

for process mediation, such functionality can easily be added to the mediation framework existing for plan level mediation.

5.2 FRED as Mediation Platform in DIP

Net Dynamics is a partner in the DIP-Project (Data, Information, and Process Integration with Semantic Web Services), an 6th Framework Integrated Project funded by the European Commission which starts in January 2004. The aim of DIP is to develop next generation integration technologies for Semantic Web Service driven applications.

The outcome of the project will be, among others, the development of an architecture for mediation between Semantic Web Services. Therein, the FRED system will be used as the platform for integration and automatic execution of the mediation technologies which will be development in the project. As explained above, the mediation architecture of the FRED system ensures that only completely interoperable resources will be further processed during meetings between agents. In combination, the mediation system developed in the DIP project aims at providing high quality mediation technologies for Semantic Web Services which can be executed automatically by the FRED system. As a benefit for the further development of the FRED system, the mediation technologies developed in the DIP project can be used to extend the currently existing mediation facilities of the FRED system which will enhance its usability as an agent-based mediation platform for Semantic Web applications.

6. RELATED WORK

Working areas related to the FRED system is agent platforms on the one hand and, on the other, ontology-based systems as well as Semantic Web Service technologies. In order to position the FRED system in these areas, we investigate current research and development activities and allocate the technical solutions of the FRED system in these areas.

6.1 Related Agent Technologies

Agent-technology for automated task processing has become an exceedingly explored area of research and development in the last years due to functional benefits expected from the agent paradigm (autonomous, pro-active, and social agents for resolving task) in many application areas. Of special interest for positioning the FRED system are development platforms for multi-agent systems. We review agent platforms that have been deeply investigated for the

FRED system design and which are frequently referred to as state of the art agent systems (Nguyen et al., 2002).

The first platform to investigate is JADE (Java Agent DEvelopment Framework)[10] which is a development environment for multi agent systems. The main objective of JADE is to provide a middleware for development of multi-agent-systems in compliance to the FIPA-Specifications for agent-based systems.[11] The major functional concern is to provide a system for secure and reliable agent communication whereupon systems for concrete application scenarios can be built. The design of the JADE platform implements the FIPA reference model for agent platforms completely: the Agent Communication Language ACL is used for agent interaction and ontologies are applied for semantic descriptions of the message content. For system control, specialized agents manage access to the system and provide a communication channel for meetings, and the management of agents relies on a ´yellow pages´ model (Bellifemine et al.; 1999). The JADE platform is an open source that has been developed since 1999 and it is applied in several research and development activities.

The second platform to be mentioned is the ZEUS[12], also a development platform for multi agent system. Similar to the JADE approach, ZEUS utilizes ACL for agent interaction along with ontologies for semantic message content description. In contrast to the FIPA reference model, the control of the ZEUS system realizes a goal-driven approach for task-service-resolution which is performed by a coordination engine that determines reasonable sequences of goals for an agent and a Planner / Scheduler that constructs service execution sequences to solve a goal using AI-planning techniques (Collis and van Buskirk, 2001).

Another initiative related to agent technologies for the Internet is the Agentcities project which aims at creating a worldwide network of agent platforms in order to provide intelligent service composition and execution remotely over the Internet.[13] The outcome of the project is a word-wide network of currently 38 agent platforms as well as research results concerning the interweavement of agent technology with Web Technology, especially with Semantic Web and Semantic Web Service technologies.

---

[10] JADE homepage: http://jade.cselt.it

[11] Foundation for Intelligent Physical Agent. Specifications 1997, 1998, 2000. FIPA homepage: www.fipa.org

[12] ZEUS homepage: http://more.btexact.com/projects/agents/zeus/

[13] Agentcities homepage: http://www.agentcities.org/

Similar to the JADE and ZEUS system, the FRED system is a development framework for agent-based applications. It provides a reliable agent runtime environment that complies with the FIPA reference model for agent interaction by using ACL and ontologies. In contrast to the other systems, it further provides the Smart Objects technology that allows using ontologies as the grounding data model throughout the entire system and not only for semantic descriptions of message content. The FRED technology for task-service-resolution realizes a goal-driven approach like the ZEUS system, but the underlying technologies are interlaced with the other FRED components to a higher extent than in ZEUS which enables the FRED system to serve as a mediator platform for Semantic Web applications. Finally, the FRED system allows integrating external Web Services into the system which is not supported by any other system yet and still considered as a research aspect in the Agentcities project.

6.2 Comparison with Semantic Web technologies

The second group of technologies related to the FRED system is Semantic Web technologies, more precisely technologies for handling ontology data and technologies for automatic discovery, composition, and execution of Web Services, referred to as Semantic Web Service technologies. We point out the state of the art in these fields regarding the underlying approaches and the maturity of existing technologies and compare it to the technologies utilized in the FRED System.

6.2.1 Ontology Technology

The challenge arising for making ontologies usable as the underlying data model in ontology-based systems is the development to provide appropriate means for handling ontology data. This includes secure and stable storage and retrieval technologies, techniques of maintaining evolving ontologies and ontology integration techniques for enabling interoperability of heterogeneous ontologies. Although there are various research and development activities working in this field, no sophisticated system for ontology management exists at this point of time. Thus, we examine the state of the art in integrated ontology management systems and position the FRED technology of Smart Objects (see section 3) to this.

Research and development activities concerned with ontology management are separated and not assimilated at this point of time. Although there are different efforts that work on ontology editors, on storage and retrieval systems, on versioning mechanisms and on

ontology integration techniques, only very few systems incorporate these technologies into a coherent ontology management environment. One approach that aims at creating such an integrated ontology management system is the WebODE, developed at the Universidad Politécnica de Madrid (Corcho et al., 2002). WebODE is comprised of an ontology editor, an ontology library facility including storage for ontology data in conventional RDBs and retrieving them as well as browsing and documentation facilities, a versioning mechanism for evolving ontologies, support for merging ontologies and for import and export of ontology representation standards. All features of WebODE have been developed exclusively for the platform and thus lack in quality in some aspects, for example the versioning mechanism is very basic and the storage technology is insufficient in terms of security and scalability, and the system is still under development. As discussed extensively in section 3, the Smart Objects technology of the FRED platform contains all functionalities required for ontology management. Although there also are some shortcomings, particularly in import / export facilities and ontology integration, the FRED technology can be considered as confirming to the state of the art in ontology-based systems since no sophisticated environment for ontology management is existent.

A special feature of the Smart Objects technology is that ontologies are compiled into Java Objects which allows using conventional Java-technologies for handling ontology data. A similar approach is realized by OntoJava[14] wherein ontology schema definitions created with PROTÉGÉ are compiled into Java Objects. The expressiveness of these Java Objects is limited to taxonomic structures with simple constraint definitions since RDF(S) is the used as the source representation format for ontologies. For rule-based querying of ontologies, logic expressions are transformed into Java methods which then check the repository (Eberhart, 2001). Compared to the Smart Objects technology, OntoJava objects are not as expressive as OOIs and the rule definitions are not as reusable as OOCs. Furthermore, no additional ontology management technologies are provided around the OntoJava approach since it is an academic effort. Besides, the benefits of the Smart Objects technology, i.e. usage of conventional Java technologies for ontology handling, as well as the hazards, i.e. the need of recompiling ontologies when updated and re-generation of ontologies for export, have also been observed in the OntoJava approach.

---

[14] see: http://www.i-u.de/schools/eberhart/ontojava/

6.2.2 Semantic Web Service technologies

Semantic Web Service technologies cover technologies for automated discovery, composition, and execution of Web Services in order to enable remote exchange and reuse of computational functionality over the Internet. The problems and the technological approaches are similar to the task-service-resolution techniques in the FRED system, thus we outline the state of the art in Semantic Web Service technologies and position the FRED techniques to this.

For discovery and composition of Web Services, goal-driven approaches similar to the concepts of Goals, Plans and Processes are considered in the leading research efforts. Therein, the basic idea is to provide declarative descriptions on capabilities, i.e. generic descriptions of user goals solvable by existing web services, as well as on the web services and utilize these descriptions to determine a suitable web service for a requested capability by reasoning on these descriptions (Arroyo et al., to appear). The most recent approaches in Semantic Web Services are DAML-S which provides an ontology for semantic descriptions of web services (DAML-S, 2003) and the Web Service Modeling Framework WSMF which provides a more comprehensive approach for Semantic Web Services (Fensel et al., 2002b). When transferring the FRED approach into this terminology, a Goal would be a capability, a Plan Module a web service available in the system, and the Plan-concept is the intermediate glue that brings capabilities and web services together. This similarity emphasizes that the approach followed for task-service-resolution in the FRED system confirm to the state of the art, although the current status of implementation has to be seen as very rudimentary. But also the technologies for discovery and composition of Semantic Web Services are currently under development and only prototypical academic tools exist that present possible technical solutions.

Also, in the area of Semantic Web Services workflow or process technologies are considered for specifying reasonable execution sequences as a procedural approach for Semantic Web Service composition. The current trend leaves the dichotomised view of procedural and declarative techniques and proclaims a combination of both to realize workflow specifications. Procedural implementations shall be used for defining the control flow on a higher level of abstraction while declarative mechanisms shall be employed for discovery and dynamic composition of Web Services which actually solve the problem (Fensel et al., 2002b). At this point of time, the most recommended process-based approach for Semantic Web Services is BPEL4WS which emerged form industrial efforts (Curbera et al., 2002).

BPEL4WS provides a reference model for workflow specification in web service based system which is in general terms of structure and expressiveness comparable to the WfMC reference model applied in the FRED system. A process engine for BPEL4WS processes along with a validity check for process specifications is provided by IBM, called BPWSJ.[15] Also DAML-S employs a process-approach since web services are understand as processes. Therefore, the DAML-S Process Model is defined as a basic building block of the DAML-S ontology which consists of two ontologies. A Process Ontology that describes the structure of a process and a Process Control Ontology which aims at indicate the control structures needed for process execution (DAML-S, 2003). A process editor comparable to the FRED Process Composer is currently under construction.[16] This approach is not mature yet because the disposition of its components has not been elaborated efficiently, thus DAML-S is heavily criticised throughout the Semantic Web Services research community (Lara et al., 2003). It is important to remark that these process approaches only support automated activities in a process while, in the FRED system, manual activities can be included in a process specification along with a activity management system which allows support for a broader range of real world business processes.

7. APPLICATIONS AND FUTURE DIRECTIONS

In the final section of this paper we briefly portray existing FRED applications and present current issues and future plans concerning the ongoing development of the FRED system. Thereby we exemplify the broad variety of application scenarios for the FRED system and to expose the ongoing development efforts to enhance the system's functionality.

7.1 Existing Applications

At this point of time, the following FRED applications are existing which have been created in collaborative projects between Net Dynamics and the end users.

eCoach - Skill Management at Wiener Stadtwerke

According to EU market liberalisation rules Wien Energie, Austria's largest utility company (www.wien-energie.at), was faced with the problem of changing from a state owned

[15] http://www.alphaworks.ibm.com/tech/bpws4j

[16] http://www.ksl.stanford.edu/projects/DAML/damls.shtml

company into a customer oriented service company. This change requires a lot of training effort for thousands of employees over a long period of time. To support this, eCoach was introduced. eCoach is a FRED application wherein each employee gets a personal Fred that permanently monitors education offerings, tracks the personal education path, and requests feedback from employees. Thereby eCoach provides a means for management that supports strategic decision about future education development. eCoach is in successful production since more than one year (Smolle and Sure, 2002).

ORGA-FRED - Automation of Business Knowledge Processes at AVL List

AVL List is an Austrian based Motor Research Company which strongly relies on high quality in every business process. [17] Quality in business processes is basically defined as having the knowledge of a process reusable and available in time. Two business processes were selected to be supported by the FRED System with the objective to enable stepwise automation of them. The first business process, called ORGA is to automate the design review process for motor development, where a number of highly specialized project members are supported by an ORGA-FRED with the necessary material and knowledge about the project status to achieve short and effective review meetings. The second business process called ORGDB verifies that semantic misspelling of identifications or names from research resources of supportive companies are detected at the most accurate time, though preventing parallel or wrong activities at AVL.

U-Haul-FRED - Wien Energie Relocation Service

Almost 7% of the Vienna population is moving from one place to the other within one year. Due to the fact that this event has a high likelihood for a mover to change his energy provider, To provide a unique service, Wien Energie is providing a FRED based application U-Haul-FRED which will support a customer in a way that this FRED manages and automates the necessary address and similar changes with only minimal involvement by the users. Since up to 300 independent partners, like Post or Telekom, may be concerned by U-Haul the integration-service of the respective partner processes will be a mayor task done by the U-Haul-FRED.

[17] see: www.avl.com

BA-FRED - Business Process Activity Management Tool

The tracking of Business Processes in an collaborative environment in general, and especially within project processes, is a major task which can be very effective supported by the FRED system. The BA-FRED is an application which allows the delegation of activities to an Activity Manager, a FRED, who based on the type and attributes of the activity (due date, performer, escalation rules, business process to follow etc.) can perform the necessary tracking automatically. This tracking process for general business processes can also be done for members in a project by supporting everyone with his personalized FRED who then acts like a project office specialist and reducing so the time spent for administration tasks. When a BA-FRED performs a business or project process, it can capture this process immediately as an experience and enable though someone less know ledged about this process to reuse this experience by CSR (case based reasoning) technologies applied.

7.2 Future Developments

Throughout the paper we have presented the current status of technical solutions provided by the FRED system. Therein, it was repeatedly stated that the current implementations are only intermediate solutions which in fact follow an appropriate technological approach but are not maturely developed at this point of time. Here, we summarize these points and point out current working areas and future plans for further development of the FRED system.

7.2.1 Smart Objects

The Smart Object technologies are in an appropriate state regarding the support for using ontology data as the underlying data model in the FRED system. The further development considered for this are consolidation and enhancement of certain aspects. In particular, the OXMLCompiler will be replaced by and RDF / OWL-compiler in order to support ontology representation standards. To enhance the facilities for incorporation of ontologies into a FRED application, the Ontology Tower is planned to be extended with import and export features for ontology schema and instance data. Eventually, the technologies for mediation between ontologies and especially between SMOs will be extended in order to improve interoperability support between Freds that use different ontologies.

7.2.2 Task-Service Resolution

The task-service-resolution techniques of the FRED system realize a promising approach for automated and dynamic goal resolution, although the technological realization is very rudimentary at this point of time. The essential pre-requisite for enhanced solutions is a powerful description language for goals and problem resolving services upon which more sophisticated technologies for goal resolution can be developed. The focus for future development in this area is to incorporate emerging standards for goal, service, and process description languages into the FRED system and not to develop an own, proprietary solution for this. Especially techniques developed in the area of Semantic Web Services research will be explored in order to support integration of external Web Services as well as interoperability with other Semantic Web based applications in a better way.

7.2.3 Mediation Technology

The third major area for future development is the enhancement of the mediation facilities of the FRED system. As outlined in section 5, the strength of the FRED system is that it can be used as a mediation platform for agent-based Semantic Web applications whereby the concept of Meeting Rooms in cooperation with its surrounding technologies enables automated execution of mediated resources. The further development of the FRED system concentrates on enhancing its mediation facilities and, as the approach seems to be promising for mediation at the plan level, it will be extended towards mediation at the process level.

8. CONCLUSIONS

In this paper we have presented the FRED system as an agent platform with special capabilities as a mediator system for Semantic Web driven applications. The aim of the FRED system is to serve as a development platform for agent-based systems that support automatic execution of tasks delegated to an electronic representative and which utilize the Internet, more precisely the Semantic Web, as a resource for information and functionality. The requirement for such a system is, apart from an adequate agent runtime environment, means for integration of Semantic Web resources. Therefore, techniques for integration of ontologies and Semantic Web Services as well as mediation facilities for enabling interoperability between potentially heterogeneous resources are required.

In order to explicate the usability of the FRED system as a agent platform for Semantic Web applications and its mediation facilities, we thoroughly presented the architecture and the

technical solutions for handling ontologies and task-service-resolution. The FredBase is the agent runtime environment of the FRED system which consists of the FredBase Server for executes and control of meetings between agents, the Fred Database for storing all relevant application data, and interfaces for system control and user interaction. The interaction between agents, called Freds in the system, takes place in a Meeting Room whereby only Freds are called for a meeting that can and solve a goal according to the application context and which are interoperable according to the data model and functionalities to be used. The FRED platform supplies integrated interfaces for connection to external systems and other Fred applications, as well as for central management of user interfaces.

The complete FRED system employs ontologies as the underlying data model, thereby exploiting the benefits of ontologies for information processing and enabling integration of external ontologies into the FRED system that are needed for a Semantic Web application. For managing ontology data, the Smart Object technology has been developed which transforms ontology definitions into Java Objects so that ontology instances can be handled by conventional Java technologies which are more mature in terms of security and performance than currently existing ontology technologies. Regarding the expressiveness as a ontology representation language, Smart Objects are comparable to OWL Lite for ontology instance descriptions. Additionally, logical expressions can be defined on a concept which can be used as generic selection mechanisms on instances. The techniques for Smart Object management comprise all aspects needed for ontology-based systems, that is performant and scalable ontology data storage and retrieval, ontology versioning to enable evolvability of ontologies, and transformation techniques to make heterogeneous ontology instances interoperable. Although these technologies can not be considered to be mature, the Smart Objects technology confirms to the state of the art in ontology-based systems since no sophisticated integrated environment for ontology management exists at this point of time.

For the task-service-resolution, i.e. how a task assigned to an agent is solved by adequate problem solving services available in the system, the FRED system realizes a goal-driven approach. Therein a Goal represents a generic user goal that can be assigned as a task to a Fred-agent. Goals can either be solved by a Plan as a simple problem solving implementation or by a Process which is a more complex workflow specification. Thereby, external Web Services can be integrated into the FRED system via a special Plan which allows to invoke a Web Service according to its WSDL-description. Upon these constructs, Goal Resolution techniques are applied which are differentiated into determination of appropriate partners for

a meeting (meeting creation), discovery of suitable problem solving services for a given goal (service discovery) and exception handling during service execution (exception handling). Although the description language for Goals and Plans in the FRED system is rudimentary in its current implementation status and thus the Goal Resolution techniques are very basic, we emphasize that architectural concept allows enhancing the Goal Resolution technologies by extending the technical building blocks without changing the overall architecture.

Finally, we have outlined why the FRED system can serve as platform for Semantic Web applications with special attention to its usability as an automatic mediator system. The primary support for Semantic Web applications is the possibility of employing ontologies and of integrating external Web Services into the system. As incorporated resources might be heterogeneous, especially if derived form an open and ambiguous environment like the Internet, an agent platform for Semantic Web has to assure that used resources are completely interoperable before they are sent into a meeting for automated task processing. In the FRED system, the Meeting Room Manager and its preliminary technologies ensure that only Freds are called into a meeting which firstly apply the same or interoperable ontologies, secondly have compatible Goals, and thirdly have intermateable communication behaviors. The mediation facilities rely on various technical constructs of the FRED system and, due to their currently rudimentary implementation status, are limited in the mediative power provided. Nevertheless, the interoperability is derived and checked during by means of runtime without interfering the application context, the FRED system can be seen as next generation mediator system for agent-based Semantic Web applications.

In conclusion, the FRED system provides a development environment for agent-based with satisfactory support for Semantic Web applications. Its most important characteristic is that the overall architecture confirms to the approaches currently persecuted in research and development efforts for Semantic Web and Semantic Web Services driven applications. Thus, the functional quality of the FRED system for support of ontologies and integration of Semantic Web Services, its Goal Resolutions techniques, and, in consequence, its mediation facilities can be easily extended by applying more sophisticated technologies for specific FRED components without changing the whole system.

Acknowledgements

The work presented in this paper is based on a collaborative project between Net Dynamics and DERI Austria in the summer of 2003. We like to thank the members of Net Dynamics and of the DERI Austria Institute for support and fruitful input to this work.

REFERENCES

Arroyo, S.; Lara, R.; Gómez, J.; Berka, D.; Ding, Y.; Fensel, D. (2004): Semantic Aspects of Web Services. In Munindar. P. Singh (Ed.), *Practical Handbook of Internet Computing.* Baton Rouge: Chapman & Hall and CRC Press, to appear.

Bellifemine, F.; Rimassa, G.; Poggi, A. (1999). 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.

Berners-Lee, T.; Hendler, J.; Lassila, O. (2001): *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.

Bussler, C. (2003): *B2B Integration.* Berlin, Heidelberg: Springer.

Chaudhri, V. K.; Farqhuar, A.; Fikes, R.; Karp, P. D.; Rice, J. P. (1998): *Open Knowledge Base Connectivity 2.0.3*. Knowledge Systems Laboratory, Stanford University.

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

Collis, J.; Ndumu, D.; van Buskrik, C. (2001). *The ZEUS Technical Manual*. Release 1.04. available at: http://zeusagent.sourceforge.net/docs/techmanual/TOC.html

Corcho, O.; Fernández-López, M.; Gómez-Pérez, A.; Vicente, O. (2002). *WebODE: an integrated workbench for ontology representation, reasoning and exchange.* Lecture Notes on Artificial Intelligence Vol 2473. 13th International Conference on Knowledge Engineering an Knowledge Management (EKAW'02). Berlin: Springer, pp. 138-153.

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.

The DAML Services Coalition (2003): DAML-S: *Semantic Markup for Web Services, version 0.9.* http://www.daml.org/services/daml-s/0.9/damls.pdf, 2003.

Eberhart, A. (2001). *OntoJava – Applying Mainstream Technology to the Semantic Web*, Workshop on Semantic Web-based E-Commerce and Rules Markup Languages at the ICEC, Vienna (Austria).

Erdmann, M. (2003): *OXML 2.0. Reference manual for users and developers of OXML - the XML-based Ontology Representation language for OntoEdit.* Karlsruhe: Ontoprise GmbH.

Fensel, D.; Decker, S.; Erdmann, M.; Studer, R.(1998). Ontobroker: The Very High Idea. In: *Proceedings of the 11th International Flairs Conference* (FLAIRS-98), Sanibal Island, Florida,.

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

Fensel, D.; Wahlster, W.; Lieberman, H.; Hendler, J. (Ed.) (2002b): *Spinning the Semantic Web. Brining the World Wide Web to Its Full Potential.* Boston: MIT Press.

Fensel, D. (2003). *Ontologies. A Silver Bullet for Knowledge Management and E-Commerce*. 2nd Edition. Berlin, Heidelberg: Springer.

Gómez-Pérez, A.; Fernández-López, M.; Corcho, O. (2003): *Ontological Engineering.* Series in Advanced Information and Knowledge Processing. London: Springer Ltd.

Horrocks; I.; Patel-Schneider, P.F. (2003): *Three Theses of Representation in the Semantic Web,* In Proceedings of the WWW2003, Bulgaria.

John, B. E. and Kieras, D. E., *The GOMS family of analysis techniques: Tools for design and evaluation,* Carnegie Mellon University School of Computer Science Technical Report No. CMU-CS-94-181, 1994.

Kayed, A.; Colomb, R.M. (2002). Business to Business Electronic Commerce: The Electronic Tendering. In M. Warkentin (Ed.): *Business to Business Electronic Commerce, Challenges & Solutions*. Hershey (USA): IDEA GROUP PUBLISHING.

Kifer, M.; Lausen, G.; Wu, J. (1995): *Logical Foundations of Object-Oriented and Frame-Based Languages*, Journal of the ACM, 42:741-843.

Kiryakov, A.; Simov, K.; Ognyanov, D. (2002): Ontology Middleware and Reasoning. In Davis, J.; Fensel, D.; v. Harmelen, F. (Ed.), *Towards the Semantic Web. Ontology-Driven Knowledge Management.* London: Wiley.

Klein, M.; Fensel, D. (2001): *Ontology versioning on the Semantic Web*. In Proceedings of the 1st Semantic Web Working Symposium (SWWS), pages 75-91, Stanford University.

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, PA, September 30, 2003

Magkanaraki, A.; Karvounarakis, G.; Anh, T. T.; Christophides, V.; Plexousakis, D. (2002): *Ontology Storage and Querying.* Information Systems Laboratory, Foundation for Research and Technology Hellas: Technical Report 308.

McCann J.A.; Manning K.J. (1997). Performance Management Tool for Interoperable Environments. In K. Stone (ed.): *Proc. of the 4th Annual Conference of the British Computer Society Client/Server Group, CSG '97, Cambridge, UK*, pp 60-70.

McGuinness, D.; v. Harmelen, F.: *OWL Web Ontology Language Overview.* W3C Candidate Recommendation, 18. Aug. 2003.

Mitra, N.: *SOAP Version 1.2. Part 0: Primer.* W3C Recommendation 2003.

Nguyen G. T., Dang T. T., Hluchy L., Balogh Z., Laclavik M., Budinska I. (2002). *Agent Platform Evaluation and Comparison.* Technical report for Pellucid 5FP IST-2001-34519. Bratislava, Slovakia.

Noy, N. F.; Musen, M. A (2003): *Ontology Versioning as an Element of an Ontology-Management Framework.* To be published in IEEE Intelligent Systems.

Smolle, P.; Sure, Y. (2002). FRED: *Ontology-based Agents for enabling E-Coaching Support in a large Company*. 2nd International Workshop on Ontologies in Agent Systems (OAS 2002), at the 1st International Conference on Autonomous Agents & Multiagent Systems, Bologna, Italy.

Sure, Y.; Erdmann, M.; Angele J.; Staab, S.; Studer, R.; Wenke, D. (2002): OntoEdit: Collaborative ontology development for the Semantic Web. In *First International Semantic Web Conference (ISWC 2002)*, volume 2342 of *LNCS*, pages 221–235. Springer.

Ullman, J.D. (1988): *Principles of Databases and Knowledge Based Systems, Vol. I*. Computer Science Press, New York.

Wache, H. et al.. (2001): *Ontology-Based Information Integration – A Survey of Existing Approaches*. In Proceedings of the Workshop Ontologies and Information Sharing, IJCAI 2001.

WfMC (1995): *The Workflow Reference Model.* The Workflow Management Coalition. WFMC-TC00-1013.

WfMC (2002): *Workflow Process Definition Language – XML Process Definition Language.* The Workflow Management Coalition. WFMC-TC00-1025.

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

Wiederhold, G. (1994): *Interoperation, Mediation, and Ontologies.* In Proceedings of the International Symposium for Fifth Generation Computer Systems, Workshop on Heterogeneous Cooperative Knowledge Bases, Tokyo.