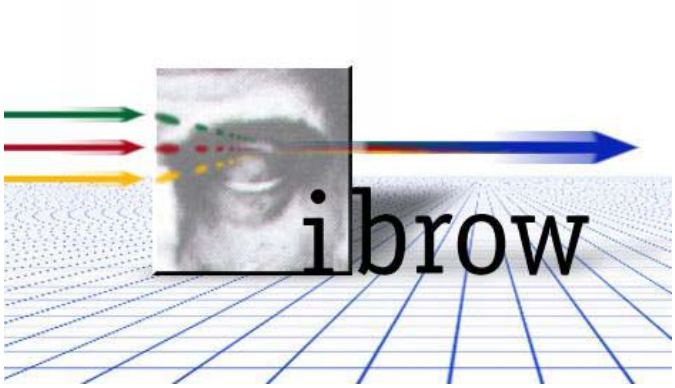


# IBROW FINAL REPORT

B.J. Wielinga, C. van Aart, A.A. Anjewierden, and W.N.H. Jansweijer.

*University of Amsterdam*

**Abstract**

This document is the Final Report of the IBROW project. It summarises the results of the project and provides pointers to the deliverables produced in the project.

| | |
|---|---|
| **Identifier** | **D19** |
| **Class** | **IBROW Final Report** |
| **Version** | **1.0** |
| **Version date** | **29-03-2003** |
| **Status** | **Final** |
| **Distribution** | **Consortium, CE** |
| **Responsible partner** | **University of Amsterdam** |

## IBROW Consortium

This document is part of a research project funded by the IST Programme of the Commission of the European Communities as project number IST-1999-19005. The partners in this project are: the University of Amsterdam, co-ordinator (NL), the University of Karlsruhe (DE), The Open University (UK), The Spanish Council for Scientific Research-IIIA (ES), Stanford University (USA), Intelligent Software Components, S.A. (ES), and the Free University Amsterdam (NL).

**[1]University of Amsterdam**
Department of Social Science Informatics (SWI)
Roetersstraat 15
NL-1018 WB Amsterdam, The Netherlands
Tel: +31 20 525 6789; Fax: +31 20 525 6896
Contact person: B.J. Wielinga
E-mail: wielinga@swi.psy.uva.nl

**[2]University of Karlsruhe**
Institute AIFB
D-76128 Karlsruhe
Germany
Tel: +49 721 608 3923; Fax: +49 721 608 6580
Contact person: R. Studer
E-mail: studer@aifb.uni-karlsruhe.de

**[3]The Open University**
Knowledge Media Institute
Walton Hall
MK7 6AA, Milton Keynes, United Kingdom
Tel: +44 1908 653506; Fax: +44 1908 653169
Contact person: E. Motta
E-mail: e.motta@open.ac.uk

**[4]Spanish Council of Scientific Research (CSIC)**
Artificial Intelligence Research Institute (IIIA)
Campus U.A.B.
08193 Bellaterra, Catalonia (Spain)
Tel: +34 935 809 570; Fax: +34 935 809 661
Contact person: Enric Plaza
E-mail: enric@iiia.csic.es

**[5]Stanford University**
Stanford Medical Informatics
251 Campus Dr., Suite 215
94305-5479, Stanford, CA, USA
Tel: +1 650 723 6979; Fax: +1 650 725 7944
Contact person: M. Musen
E-mail: musen@smi.stanford.edu

**[6]Intelligent Software Components, S.A.**
iSOCO Sant, Ctro.Neg. Can Castanyer
Crta. de Rubí, 88 bj. A
08190 St.Cugat (Barcelona), Spain
Tel +34 93 544 2048; Fax +34 93 589 5669
Contact person: E. Brieva
E-mail: erik@isoco.com

**[7]Vrije Universiteit Amsterdam**
Faculty of Sciences
Division of Mathematics and Computer Science
De Boelelaan 1081a
1081 HV Amsterdam, the Netherlands
Tel: +31 (0)6 51850619; Fax: +31 20 872 2722
Contact person: D. Fensel
E-mail: dieter@cs.vu.nl

**Industrial advisory committee:**

DaimlerChrysler (D)
Deutsche Telekom (D)
Bolesian (NL)
Unilever (NL)
IBM (JP)
Artificial Intelligence Applications Institute (UK)

# Final Report IBROW

Bob Wielinga and Chris van Aart and Anjo Anjewierden, and Wouter Jansweijer

Social Science Informatics, University of Amsterdam
Roetersstraat 15, 1018 WB Amsterdam, The Netherlands
{wielinga,aart,anjo,jansweij}@swi.psy.uva.nl

March 29, 2003

**Abstract**

This document describes the final results of the research done in the IBROW project on intelligent brokering of applications from components that reside in libraries on the World Wide Web.

# Contents

# Executive Summary

The general objective of the IBROW project is to create an Intelligent Brokering Service on the World Wide Web. Such a service should be able to configure applications from existing software components on the basis of a global specification provided by a user. A central idea of IBROW is that of *brokering* [10] between one or more libraries of software components and a user. The user interacts with the broker to specify a task that an application should perform (goal specification), subsequently the broker searches for components in the libraries and -if successful- configures an application that will solve the user's task. Output of the brokering process is a specification of a configuration of components that can be executed.

The broker reasons about the components in terms of competence specifications that are part of the libararies. These competence specifications are based on a competence modelling architecture: the Unified Problem solving method Modelling Language (UPML). The project has developed a number of libraries of software components in various domains: classification, document analysis, information filtering, case based reasoning, and scheduling. Each of these libraries contains a number of executable software components annotated by a competence specification in UPML.

The project has explored several brokering approaches, varying from interactive brokering and simple keyword matching techniques to complex competence reasoning methods based on First Order Logic. Experiments with these techniques show that a middle of the road approach - using semi-formal competence specifications- is the most viable approach. Work on adaptation of IBROW components has resulted in a brokering method based on the Propose-Critique-Modify method.

Issues concerning interoperability were addressed using various forms of FIPA-based agent technology. Also CORBA, SOAP, WSDL and UDDI technologies were explored for inter-component communication.

The ideas developed in IBROW were tested in a number of spin-off applications. These include several classification applications, a Web Information Mediator, a health care service and a conference paper analysis and classification application.

The IBROW project brings together a number of key ideas concerning the automatic configuration of software components in a distributed environment. An elaborate descriptive framework, rich ontologies, distributed component libraries, competence specifications and context dependent brokering architectures and strategies, together form a strong basis for the development of future Semantic Web Services.

# 1 The IBROW Objectives

A central idea of the IBROW project is that of *brokering* [10] between one or more libraries of software components and a user. The user interacts with the broker to specify a task that an application should perform (goal specification), subsequently the broker searches for components in the libraries and -if successful- configures an application that will solve the user's task. Output of the brokering process is a specification of a configuration of components. The execution of this configuration is controlled by a separate component: the manager. Figure 1 shows a UML Use Case diagram of the brokering process. The broker reasons about the problem of how to find an applicable configuration using competence specifications provided by the libraries. A central role in this reasoning process is played by the ontology that a library subscribes to (the Task Ontology). Such an ontology specifies the knowledge-level vocabulary on which the competence descriptions are based.

# 2 Competence Specifications and Library Organisation

The organisation of the IBROW libraries is based on the Unified Problem Solving Language (UPML) [29, 15, 16]. In spite of its acronym, UPML is a modelling architecture rather than a modelling language. The UPML architecture [30] is summarized in Figure 2. UPML defines the component types that libraries can contain: Task, Problem Solving Method (PSM) [1] and Domain Model. PSMs come in two types: problem decomposers, which decompose a task into subtasks, and reasoning resources, which are primitive operations. Central in the UPML framework is an ontology which acts as the glue between the various elements in the framework. The ontology defines the concepts and relationships necessary to describe the components of a particular library. The Task Ontology defines the conceptualization of the goal specification of the user. The PSM ontology adds terminology required for describing the problem solving methods and the Domain Ontology contains the domain specific terminology. The UPML architecture also contains mappings between the Task, PSM and Domain ontologies in the form of *Bridges*.

In addition to the general architecture, UPML specifies a template for describing the components in a library [30, 13]. The slots in the template that are most important for brokering are:

**input roles** Describes the type of inputs for a task or PSM. Its values are taken from a sortal hierarchy. Usually sorts can be extended using a number of composition operators, such as **list** or **set**.

**output roles** Similar to input roles.

**assumptions** PSMs usually make certain assumptions.

[1] The term PSM is usually used in knowledge engineering to denote knowledge intensive methods. In IBROW we also use PSM to denote components which perform simpler tasks and are not knowledge intensive

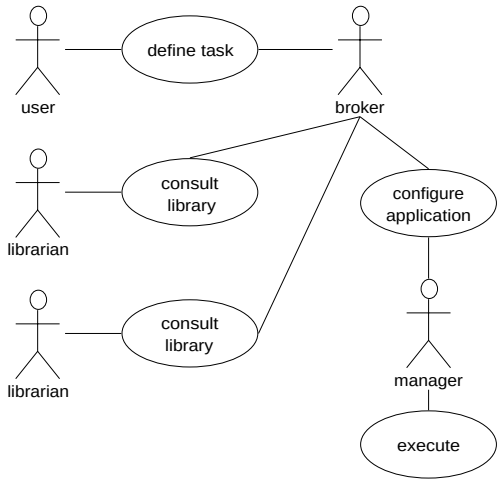


Figure 1: Use case diagram for the brokering process

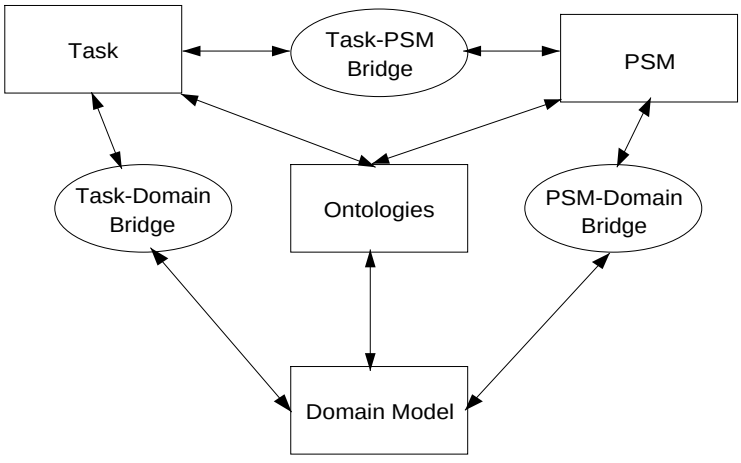


Figure 2: UPML architecture

**preconditions** These have to be satisfied in order for a PSM to be applicable.

**postconditions** Describe the situation that occurs after a PSM has been applied.

**subtasks** For problem decomposers this slot specifies the subtasks.

Although UPML describes the slots required for the competence specification, no language was specified in which the specifications can be written down (the "Object Language" in IBROW terminology). The reason for this is that different component libraries could require languages of different expresiveness and complexity. In the following section we will describe the various formalisms for competence descriptions explored in the project.

## 2.1 Competence Descriptions

There are several approaches to the specification of the competence of a PSM.

### 2.1.1 Propositional

In the Propositional approach the elements of the competence slots (assumptions, preconditions, postconditions) are filled with atomic propositions, which are defined in the PSM ontology. For example, the CBR library described in Deliverable D8 [1] uses a hierarchy of atomic terms (called features in D8) to specify assumptions, preconditions and postconditions. The first version of the Document Analysis library [3] allowed conjunctions of atomic propositions in the competence descriptions, but did not make use of a hierarchy of terms.

The propositional approach to competence specification is relatively straightforward, but requires a strong and well developed ontology of concepts. The ontology should capture the important concepts which characterise problems and methods in a particular Task and PSM domain. The development of the libraries in IBROW has shown that the creation of such an ontology is by no means an easy task. It requires a thorough understanding of the functional and logical properties of the Task and PSM domain. The development of such an understanding is often a research project in itself. The positive side effect of building a competence ontology is a much better insight in the space of problem types and methods for a certain task type. As such it can not only support a (semi-) automated brokering process, but it is both scientific progress and it is useful in the process of hand-configuring components from a library for a particular application (see section 4.1.

If no theoretically well-founded ontology exists or can be constructed, the propositional approach reduces to a simple keyword indexing method. In principle there is nothing wrong with this, but the usability scope of a library indexed by keywords is limited to brokers who understand the relevant set of keywords.

### 2.1.2 First Order Logic

A much more expressive, but also more ambitious approach is to use First Order Logic (FOL) as competence specification formalism. In the initial phase the project had high expectations of the FOL approach. Several classes of PSMs such as classification, hill climbing and parametric design were formalized using FOL theories. The work on Ontolinga libraries of ontologies appeared to be sucessful along similar lines ([20]). Several other formalizations of PSMs were published ([2]. On the basis of these developments attempts were made to represent the IBROW library competences in FOL. This turned out to be a difficult nad time-consuming process.

### 2.1.3 Frames

While the FOL specification of competences allows for a very expressive and flexible language it is not always the most appropriate represention. The frame representation is a simplified formalism that represents competences as a conjunction of literals. The frame representation heavily relies on the PSM ontology.

### 2.1.4 Factory Model

UPML describes the competence of a PSM (component) in terms of preconditions, postconditions and assumptions. An alternative description is the *factory model* loosely inspired by components in the software engineering community [21].

The main difference compared to the UMPL specification is that the factory model makes the notion of *consumption* explicit. Although consumption can often be seen as the commercial equivalent of the academic transformation or mapping, there are practical advantages when trying to convey what a component does.

A competence specification in terms of the factory model therefore consists of: (1) a set of expressions that describe what is required in the input and reproduced in the output (called *requires*); (2) a set of expressions that is required in the input and not reproduced in the output (called *consumes*); and (3) a set of expressions produced by the PSM (called *produces*).

## 2.2 Library Organisation

Several libraries were developed in the project with varying organisations and sizes. Some other resources were also studied to see how they could fit in with the IBROW library framework. The main forms of library organisation that were identified are:

**Library of homogeneous reasoning resources** The components of a library of this type are primitive components (”reasoning resources” in UPML terminology, ”inferences” in CommonKADS terms). The library does not contain problem decomposing PSMs. An example of this type of library is the document analysis library (version 1, [4]), where all

components have a similar structure, and share a number of assumptions, such as domain knowledge-base representation format and calling conventions. The second version of the document analysis library [5] has a more complex structure in the sense that the structure of the components is more variable. For example, a component can have multiple inputs, or use multiple knowledge bases. Furthermore such type of library can include components which are "canned" configurations that perform combinations of lower level operations. However, the components in the library remain elementary reasoning resources from the broker's point of view.

**Library of heterogeneous PSM's** This was the type of library that was originally envisaged in the IBROW project. It consists of a collection of heterogeneous problem decomposition methods and reasoning resources. These components have in common that they share a common ontology, but otherwise their structure and role in the brokering process is variable. The library of Information Filtering components [18] and the library of Case Based Reasoning IBROW:D8 are of this type. Earlier libraries developed outside the IBROW project, such as the CommonKads PSM libraries [11], the library of diagnosis methods [6], and the library of Generalized Directive Models (GDMs [24]), differ from the type of library described here. The emphasis in earlier work was mainly on task-method decomposition; competences for reasoning resources were usually not explicitly modeled.

**Library of configurable PSMs** This type of library contains one or more generic problem solving components which can be configured by setting parameters or by plugging in certain modules at predefined locations. The classification library [23, 27] is a typical example of this type of library. It contains two PSMs, one for single-solution classification and one for optimal solution classification. Each of these PSMs can be parameterised by specifying critera such as admissibility of solutions, ranking criteria etcetera. The scheduling library [33] takes a similar approach. Outside IBROW, work has been performed on configurable methods for diagnosis [35, 34]. Configurable PSMs are of interest to IBROW since the configuration process is confined to a relatively small search space. Each slot in the skeleton has only few candidates that can serve as fillers.

**Case-based Library** This type of library contains case descriptions of configurations that have been brokered before. Once a broker has found a suitable configurarion, the task specification and the corresponding configuration are stored. Case based reasoning techniques (CBR) can be used for retrieving and adapting configurations for new task specifications. The library and broker developed in Task 3.1 [1] are of this type.

# 3 Broker-Library Relation

There are several high level design choices concerning the relation between a broker and the library, each of which has its own advantages and problems. We can design a library and a broker as closely intertwined modules or we can design a broker that independently operates

in a dynamic world where libraries can appear and disappear. There appears to be a trade-off between the power of a broker to configure complex applications and the scope of components that a broker can reason about. In the following subsections some of these design options are discussed.

## 3.1 Broker closely linked to a Library

In this approach the broker and library are closely linked. Although the broker may be a generic reasoner, it knows the Task Ontology up front and it knows which library it should access and where it is located. The user-broker interaction is determined by the fixed Task Ontology. Choices presented to the user are directly derived from the competence descriptions in the library. Examples of brokers of this type are the WIM broker for the library of information filtering methods [1], IRS-1 [25], and the first broker for the document analysis library.

For brokers that are based on the Propose-Critique-Modify (PCM) method (see section 4.4, additional knowledge about the components in the library may be required. For example, the PCM broker for the classification library requires knowledge about the relative strength of the various PSM configurations (see deliverable D12b [36]). Such brokers are clearly library specific.

## 3.2 Broker and Librarian as separate agencies

If a broker and a library are independent agencies, several processes are needed in addition to the actual configuration process. The broker needs to locate libraries that are relevant for the task specification of the user, needs to request information about components and their UPML descriptions, and needs information about the execution of the configuration. Conventions and standards are required to support these processes. Emerging standards such as XML, RDF, UDDI, WSDL could be used as a basis for such standardisation, but additional standards would be needed to support the complex broker-library communication that IBROW requires. In the interoperability deliverable [37] a protocol and xml encoding for broker-library information exchange is described.

## 3.3 Inter-Library Brokering

If a broker requires components from different libraries, additional machinery is required. The main problem that can occur in inter-library brokering is a mismatch between Task Ontologies. If two libraries use different vocabularies in their competence descriptions the broker will require a mapping between the ontologies. The IBROW project has performed some experiments with ontology mappings based on a common grounding base, but the problem of mapping ontologies is still very much open.

# 4 Broker Approaches

## 4.1 Brokering by hand

Although the goal of the IBROW project is to investigate (semi-) automatic brokering of applications, the experiments with various scenarios have shown that the component oriented approach also supports the quick construction of various applications by hand. Three aspects of the libraries facilitate this:

**ontology** The Task Ontology gives strong support for component-based design. The signatures, operation types and representational vocabulary are specified for the entire library. Thus, the set of components adheres to a coherent ontology, and as such allows for flexible composition. In the software engineering community similar ideas about the use of task ontologies for software development are emerging [40].

**component descriptions** The UPML framework and the description instances for the components, give a uniform and powerful documentation support for the design process. The notion of problem decomposers has many similarities with the idea of design patterns in software engineering [].

**uniform design and interoperability** The components of libraries developed in IBROW generally have a uniform design. Calling conventions, links to knowledge bases and representational formats are uniform throughout the library. This enables an application designer to quickly build an application with a limited amount of coding. This approach is similar to component-based design in the software engineering community [21].

Examples are:

**Feature extraction from PDF documents (brains, reptiles)** *TODO Anjo?*

**Hatika** *TODO Anjo?*

**National health KMI**

## 4.2 Interactive Brokering

All brokering involves some form of interaction with the user to specify the users task requirements. In the case of Interactive Brokering, the interaction with the user goes further. The broker interacts with the user in order to guide the brokering process. In the Internet Reasoning Service (IRS-1 [25]), for example, the broker uses a task and PSM skeleton for classification which have to be filled with plug-ins that determine the specifics of the functionality of the method. In this case the burden of knowing which plug-in to choose for a particular slot in the skeleton, lies with the user. The broker can support the user by providing the relevant options for a particular slot

and optionally provide some help explaining what the functionality of the plug-in is in the form of canned text. If the broker is closely linked to the library -as is the case in the IRS-1[25]- the filled template can be executed and the broker can provide immediate feedback to the user about the results. The disadvantage of this form of brokering is that the user has to have a detailed understanding of the components in the library.

In Static Brokering the broker delivers one or more fixed configurations on the basis of the task specification of the user. The broker retrieves relevant components from one or more libraries and searches for a configuration that computes the required relation between input and output.

In the experiments with several broker architectures that we have performed in the project, the output of the broker took the form of a data flow structure.

## 4.3 Dual-level Brokering

The brokering process essentially operates on the knowledge level. The output of the brokers described above is a knowledge or dataflow structure of the reasoning process in the brokered application. This knowledge flow structure does not contain much information about the details of the information exchanged between components and the control structure of the process as a whole. Early in the project these issues were considered to be interoperability problems, but in the course of the investigations it became clear that the construction of an operational configuration of components required more than just solving interfacing problems. Issues concerned with iteration over composite data structures like sets, handling failure of a component process, conversion of formats between components etcetera, are issues at the semantic interoperability level, and as such issues in the concerns of brokering. The knowledge level brokering and the brokering at the interoperational level can be viewed as two brokering processes at different levels of detail.

In IBROW three experiments with a dual-level broker architecture were performed. The architecture described in Deliverable D10 [37] containts a **manager** agent which configures a multi-agent system from the broker output, which in this case is an XML document specifying the components and their input/output data types. The manager maps the brokered components onto agents and instructs the agent about their task and collaboration mode.

A similar approach is described in Deliverables D8 [1] and D16 [19] where a broker configures the components at the knowledge level and subsequently a team of agents is formed that execute the work.

In a third experiment involved two brokers and a manager. The first broker (a frame-based static broker) generates an initial configuration of components. One of the component of the data flow structure is “classify”. The manager detects that “classify” as such is not an executable component, but that the classify method in the library requires configuration of its parameters. A second broker, based on a Propose-Critique-Modify (PCM) method is called. This broker has knowledge about the parameters of the classification method and about the preferred configura-

tion (Propose knowledge). It returns a configuration structure with the required parameters filled in to the manager. A similar architecture is used in dynamic brokering described in the next section.

### 4.4 Dynamic Brokering

Interactive, static and dual-level brokering result in a structure that can be executed in some way or another. In these models no feedback from the results of execution to the brokering process are foreseen. A more challenging approach to brokering is a dynamic approach where the broker inspects the results of executing an initial configuration and decides on changes in the configuration if the results are not optmimal.

A static brokering process produces an initial configuration which is executed, possibly after a further configuration has been done by the Propose part of the PCM component. The results -an answer or a failure report- are submitted to a verification component. The verification process analysis the results and classifies the results either as "ok" or as a failure class. A typical example of a failure class would be "not enough solutions". This failure class is then submitted to the PCM broker which will return a modified configuration. This broker architecture has been implemented for the conference submission classification application.

## 5 Brokering Methods

The approaches to brokering described in the previous sections do not specify what methods can be used to perform the various subtasks of the brokering process. The following sections describe such methods.

### 5.1 Interaction with the User

The broker has to build a representation of the user's goal and environment in which he wants to achieve his goal. The goal of the user is represented as a UPML task specification. The task specification consists of definitions of the input and output roles of the task, the assumptions that can be assumed to hold in the user's environment and the postcondition that should be true after completion of the task. The assumptions and postconditions are represented in the object language for example propositional or FOL expressions. Input and output roles are characterised using a signature. In formulating the task specification the user should adhere to one or more task ontologies that provide the vocabulary for the signature and object language expressions. A typical example of a task specification could contain as input role a set of keywords, as output a set of documents and as postcondition a statement that each document contains an abstract of a scientific article that contains the input keywords. An assumption could be that the user has access to a digital library such as Science Direct. The task ontology for this example would be a

document ontology defining terms like scientific article, abstract etc. In many occasions the user will also provide one or more domain knowledge bases.

Generally one cannot assume that a user has much knowledge about UPML and the object language used in a library. So, interactive elicitation of the task specification is required. First, the task ontology will have to be chosen. The broker can support the user in selecting a task ontology by providing general task types from which the user can select. Subsequently the broker can use the task ontology to present the user with choices for input and output roles. Elicitation of the assumptions and postcondition may be more difficult depending on the complexity of the specification formalism. The use of templates may provide support here. The second DAC-broker broker uses a propositional representation in the user interface and subsequently generates a factory model frame for brokering.

## 5.2 Locating and Selecting Libraries

Given the task specification of the user, the broker has to locate relevant libraries. In IBROW we have explored three methods for locating libraries. Some of the experimental brokers were closely linked to a library. In this case the broker simply knows the location of the library. A second approach is to require libraries to register at a central place. In the multi agent system architecture described in Deliverable D10 [37] this is done through registration on the Agent Management System (AMS) which is part of the JADE Fipa implementation platform. An alternative registration approach explored in Deliverable D23 [22], is to use the standard Universal Description, Discovery and Integration (UDDI) protocol for Web services.

Once the broker has found a number of libraries, the relevant ones have to be selected. If the user has explicitly specified the task ontologies relevant for the task, the library services can just be queried on the task ontologies they subscribe to. If the task ontologies have not been explicitly specified, the broker can extract terminology from the task specification and libraries can be selected on the basis of this terminology.

## 5.3 Selecting PSMs

After identification of suitable libraries, the broker can retrieve PSM specifications from the library. After retrieval the broker can apply a number of matching and search strategies. These are described in the following subsections.

### 5.3.1 Propositional Matching

If the task and competence specification are propositional then matching is simply the comparison of the current goal and the postconditions of the PSMs. If a problem decomposer is found new subgoals will be generated. If a suitable reasoning resource is found, its precondition is set up as the new goal. This is a straightforward backward goal regression search process. This

method was used in the WIM application [1, 19]. A forward search process from the task input roles was used in de first DAC broker.

### 5.3.2 Frame-based Matching

Frames can be viewed as conjunctions of atomic formulae. Matching two frames is generally straightforward. In order to match a goal frame G with a competence frame C, each atomic formula in G should be derivable from C. Derivation means either equality or derivable using inheritance. In the experiments with frame-based brokering no variables in the formulae were used. If variables are required unification techniques can be used for matching. The search method can be either forward from the task input or backward from the task competence specification. Deliverable D10 [37] describes the details of the frame matching used in the second document analysis broker.

### 5.3.3 Theorem Proving based Matching

The Theorem proving based match strategy starts with the selection of a candidate PSM that has an output role that matches the output role of the task. The next step is to attempt to prove the post condition of the task from the background theory and the following formula.

$(PSMPrecondition \rightarrow PSMPostcondition) \wedge TaskAssumption$

Several situations can arise in this step:

1. *perfect match* The postcondition of the task can be proven and the precondition of the PSM is either trivially true or can be derived from the task assumption. In this case the PSM is sufficient to achieve the task.

2. *match* The postcondition of the task can be proven, but the precondition of the PSM is not provable. In this case the precondition of the PSM is set as a new goal and the brokering process continues in a backward chaining fashion.

3. *no match* other PSM's will be tried

4. *all PSMs fail* If all PSMs fail to satisfy the task specification, an attempt will be made to generate subgoals. If the task's postcondition contains conjunctions an and-split strategy is used to generate subgoals of the original goal. These subgoals are then handled in the same way as the original goal.

After a succesful matching process the broker will produce a dataflow structure describing the sequence of PSMs to be used.

### 5.3.4 Case Based Reasoning

Case Based Reasoning (CBR) techniques for brokering have been explored in Task 3.1 [1]. CBR techniques can be applied succesfully when a set of instantiated configurations is available. Such configurations can be created by a broker as was the case in the CBR broker in Deliverable D8 [1], or can be hand-brokered. In the latter case applications created by human engineers can be augmented with a UPML specification that can be used in the CBR brokering process.

## 5.4 Composition and Configuration

Most of the match and search methods deliver a chain of components representing the calling order from input to output. The broker has to check whether the input and output signatures of components that are connected in the chain, are equal. If not, the broker has to construct a **bridge**. The only bridges that have been considered sofar are bridges that convert sets to single elements and vice versa. Currently none of the libraries built in IBROW contains bridges.

If one or more of the PSMs in the brokered configuration is configurable, a configuration method has to be applied. This configuration process can be done in interaction with the user, as is the case in the IRS-1 interactive broker [25]. Automatic configuration can be achieved by methods for parametric design. Such methods require knowledge that goes beyond the competence specifications discussed sofar. If a Propose-Critique-Modify (PCM) method is used for the configuration, knowledge about the skeleton method, possible components, requirements operationalisation, constraints, preferences, violations and fixes is required. Deliverables D4 [18] and D10 [37] describe an implementation of such a broker.

# 6 Interoperability

Once a broker has identified relevant components that can be combined to solve a particular task, a configuration needs to be designed where the components can exchange data and where the control regime between components is regulated. Several options exist to achieve this. One approach investigated in the IBROW project was to (inter)connect the components and services using standards and technology from the agent community. Agents communicate conforming to the FIPA standards (such as ACL and SL0) and use message content ontologies as vehicles to give meaning to the information and instructions exchanged [37]. The communicating part of the agents was implemented on top of the JADE library. The PSMs and the operator-agents representing the library components were built in SWI-Prolog. We applied a great deal of the technology stack for enabling PSM composition via the Web, as illustrated in the IBROW prototype.

A second approach [22, 26] that was investigated in IBROW is based on emerging standards such as SOAP, WSDL and UDDI. These standards provide some basic support for solving the interoperability problem, but only to a certain extent. Many interoperability problems remain,

both at a detailed symbol level as on a higher level of inter-component communication.

The problem of library providers who want to publish a service was investigated using automatic wrapper generators based on CORBA and SOAP [26]. The IRS-II system provides mechanisms with which a service developer can specify a service at the knowledge level in UPML. Subsequently wrappers are generated that hide the low-level details of SOAP.

# 7 Experimental Implementations and Spin-off Applications

In this section we briefly describe a number of experimental implementations and spin-off applications. Details are provided in Deliverables D8, D9, D12b, D13, D14, D15 and D16.

A number of experimental implementations of brokers were built. One broker implementation is based on a FOL competence specification and theorem proving technology. Two other brokers were implemented that make heavy use of the Document Analysis Component (DAC) library [5].

The first broker considers the competence of DA components as a set of propositions in the domain of document analysis (e.g. this component adds the notion of dehyphenation) and had principle DA notions implicitely represented (e.g. that documents have representations). We call this a *propositional* broker.

The second broker is *frame-based*: all knowledge is explicitely represented in ontologies and frames. The broker has access to ontologies that represent what documents can contain and frames representing the DAC library itself (i.e. components, representations, options, operations, knowledge bases). In addition, the user can add application specific ontologies.

The library of Document Analysis Components (DACs) is a homogeneous library of software components related to document analysis in the broadest sense. Several applications in IBROW are based on the DAC library, ranging from "simple" document conversion (e.g. PDF to HTML) via extracting research department citation summaries from CiteSeer (HATIKA), to analysing HTML forms to classify conference submissions.

Two implementations were built of the Internet Reasoning System (IRS). IRS-1 is an interactive broker which supports a user in the application configuration process. IRS-I was used to configure a number of classification applications. IRS-II provides generic mechanisms for publishing web services on different platforms. IRS-II was used to develop a distributed application in the health-care domain.

The WIM system was implemented as a configurable application consisting of a collection of agents with particular PSMs (registered to a UPML library) that have access to the knowledge of a particular domain; in the WIM application of Task 6.4 the PSMs are about intelligent information retrieval and integration over the internet, the task domain is medical bibliography, and the domain knowledge models used are Thesauri, Evidence-based Medicine categories, etc. The WIM system was also used as the basis for a prototype web service which can be incorporated in existing applications such as MsWord.

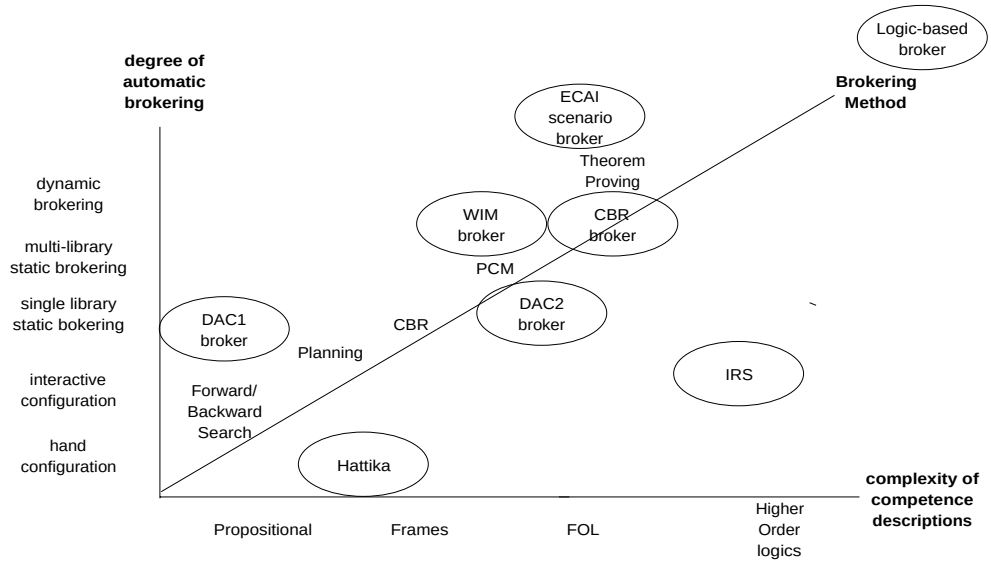


Figure 3: The Broker Space and the points investigated in IBROW

# 8 Conclusions

## 8.1 The Brokering Space

Figure 3 is an attempt to visualise the points in the broker space that IBROW has investigated. The three dimensions of the broker space are: complexity of the competence specification formalisms, brokering methods and the complexity of the brokering approach. Although this figure should be viewed as an "artistic impression" rather than a formal representation, it should give an idea of the scope of the investigations of the brokering problem that have been performed during the course of the project. The various explorations of the broker space have led to the following principles and lessons.

## 8.2 Principles

**Principle: Framework** Brokering has to rely on a standardised framework for describing components in a libary. The UPML architecture [30] has proven to be a stable and powerful means to support library construction and brokering.

**Principle: Object Language** The formalism to be used for specifying the UPML instances for components should be chosen on the basis of the understanding of the Task/PSM domain. Logic (first-order or higher-order) is a powerful formalism for specifying competences, but requires a thorough understanding of the tasks and methods as well as extensive formal background. Its scope is therefore limited to academic research environments. A much more realistic approach is the frame-based competence description. This still requires a

good understanding of the domain and a well-engineered ontology. Once such an ontology is in place, frame descriptions of components should be achievable by library designers without much formal background. Propositional competence specifications should be used when the TASK/PSM domain is understood in terms of a hierarchy of terms for methods and techniques that are based on a consensus in the field.

**Principle: Task Ontology** The Task Ontology that a library adheres to is an essential prerequisite for brokering. It serves as a basis for communicating with the user about the goals to be achieved, it is the basis for the specification of the competence of components and it supports the coherence and interoperability of the library components. A mapping between the vocabulary of the Task Ontology and terms or concepts that are (better) understood by a user is a useful vehicle for eliciting the task specification from the user.

**Principle: Method Ontology** The Method Ontology adds method specific concepts and relations to the Task Ontology. It provides the vocabulary for the competence specifications. A rich Method Ontology greatly enhances the efficiency of the brokering search process. The richer the competence descriptions are, the more focussed the PSM selection process can be.

**Principle: Broker Architecture and Methods** There is a close relation between the complexity of the broker, its power and scope, and the competence specification formalism chosen for a library. Given the preference for propositional and frame-based competence specification formalisms, a broker architecture based on backward search appears a straightforward and feasible approach. Brokers of this type can be designed to be independent of a particular library.

**Principle: Dynamic Brokering** For libraries of configurable components PCM brokering offers the possibility of dynamically adapting the task and method on the basis of an assessment of the results of executing an earlier configuration. PCM brokers require knowledge that goes beyond competence specifications and that is library specific. In spite of the the extra effort that a library provider has to invest in dynamic brokering, this approach may be fruitful in environments where data are uncertain.

## 8.3 Lessons Learned

The general approach of brokering that IBROW has investigated appears to be feasible. Moving from the purely logic competence specification to less powerful, but more realistic formalisms made both library development and broker design a do-able task.

In the course of the project we have realised that we were not investigating just one or two methods for brokering, but that there is a large multi-dimensional brokering space. Investigating a number of points in this space has been a very fruitful exercise that has provided insights about how future Semantic Web Services can be designed on a larger scale.

Developing libraries with components within the UPML framework has been a time consuming, but rewarding exercise. All library builders report a much better understanding of the tasks and methods gained through building the library and modelling the competences of the components.

Throughout the project it was found that interoperability problems were far larger than expected. When brokering an application from components selected from libraries residing on different platforms and implemented in different languages, many low-level interoperability problems occur. Standards like HTTP, XML, XSD, ISO*, RDF(S) help, but many low-level problems remain. This problem is not unique for IBROW, it also occurs in browsing the current web. The solutions that current browsers provide -such as handling illegal syntax, different character sets, different representational formats etcetera- could very well be incorporated in a broker/manager and as such solve some of the problems that the project had to cope with. However, the type of inter-library communication is much more varied in brokered applications than current communications between browsers and web servers.

A major lesson learned in the IBROW project is that ontologies play an essential role in the brokering process. Without well-designed ontologies for the Task/PSM domain of a library no generic broker can fullfil the goals of distributed Semantic Web Services. It is a major challenge for the Semantic Web Services community to develop such ontologies in a structured manner and to develop means to relate these ontologies to each other.

In conclusion the IBROW has achieved its goal to develop a framework, design principles and methods for the automatic configuration of software components in a distributed environment. The feasibility of these ideas has been proven in a number of prototype applications.

# 9 Future Outlook

The IBROW enterprise was well ahead of its time. It envisaged Semantic Web Services as early as 1996, albeit mainly in knowledge-intensive application domains. During the project the scope of tasks, libraries and applications has widened to include other, less knowledge intensive, but more commonly required tasks. In the Semantic Web community Web services only recently have become a major research topic. The scope of emerging standards such as UDDI, WSDL and WSFL are much more limited than the IBROW approach. In addition the type of services that have been explored in IBROW are much more complex than the services envisaged by current web service developers. Moreover, the diversity of the future Semantic Web will require a variety of approaches and methods to flexibly configure applications that users require. Although many problems remain to be solved, the IBROW project has set the scene for the next generation of configurable Semantic Web Services. It brings together a number of key ideas concerning the automatic configuration of software components in a distributed environment. An elaborate descriptive framework, rich ontologies, distributed component libraries, competence specifications and context dependent brokering architectures and strategies, together form a rich basis for the development of future Semantic Web Services.

# References


[1] Chema Abasolo, Josep-Lluis Arcos, Eva Armengol, Mario Gomez, Ramon Lopes de Mantaras, and Enric Plaza. Component selection. IBROW Deliverable D8, IIIA, Barcelona, January 2003.

[2] J. M. Akkermans, B. J. Wielinga, and A. Th. Schreiber. Steps in constructing problem-solving methods. In B. R. Gaines and M. A. Musen, editors, *Proceedings of the 8th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop. Volume 2: Shareable and Reusable Problem-Solving Methods*, pages 29–1 – 29–21, Alberta, Canada, January 30 – February 4 1994. SRDG Publications, University of Calgary.

[3] Anjo Anjewierden. Document analysis components. IBROW Deliverable D2b (part 2)[2], SWI, University of Amsterdam, February 2002.

[4] Anjo Anjewierden. A library of document analysis components. IBROW Deliverable D2b (part 1), SWI, University of Amsterdam, February 2002.

[5] Anjo Anjewierden. A library of document analysis components. IBROW Deliverable D2b Version 2, SWI, University of Amsterdam, February 2003.

[6] V.R. Benjamins. *Problem Solving Methods For Diagnosis*. PhD thesis, University of Amsterdam, 1993.

[7] V.R. Benjamins. Dissemination and use plan. IBROW Deliverable D18, SWI, University of Amsterdam, August 2000.

[8] V.R. Benjamins. Project presentation. IBROW Deliverable D17, SWI, University of Amsterdam, March 2000.

[9] V.R. Benjamins. Web workbench. IBROW Deliverable D7, iSOCO, Madrid, January 2001.

[10] V.R. Benjamins, E. Plaza, Motta E., D. Fensel, R. Studer, B. Wielinga, G. Schreiber, and Z. Zdrahal. An Intelligent Brokering Service for Knowledge-Component Reuse on the World-WideWeb. In *Proceedings of the 11th Workshop on Knowledge Acquisition, Modeling and Management (KAW 98), Banff, Canada*, 1998.

[11] J. Breuker and W. van de Velde, editors. *CommonKADS Library for Expertise Modeling*. IOS Press, Amsterdam, The Netherlands, 1994.

[12] IBROW Consortium. Technology implementation plan. IBROW Deliverable D20, SWI, University of Amsterdam, March 2003.


[2] See `www.swi.psy.uva.nl/projects/ibrow/deliverables/dac-library.pdf` for the most recent version.

[13] Monica Crubezy, Mark Musen, and Guus Schreiber. Upml validation and tool support. IBROW Deliverable D6, Stanford University, January 2001.

[14] A.J. Duineveld, B.J. Wielinga, V.R. Benjamins, and N. Christoph. Knowledge management library. IBROW Deliverable D2a, SWI, University of Amsterdam, February 2001.

[15] D. Fensel, V. R. Benjamins, S. Decker, M. Gaspari, R. Groenboom, W. Grosso, M. Musen, E. Motta, E. Plaza, A. Th. Schreiber, R. Studer, and B. J. Wielinga. The component model of UPML in a nutshell. In *Proceedings of the First Working IFIP Conference on Software Architecture (WICSA1)*, San Antonio, Texas, 1999.

[16] D. Fensel, V. R. Benjamins, E. Motta, and B. Wielinga. UPML: A framework for knowledge system reuse. In *Proceedings of the 16th International Joint Conference on AI (IJCAI-99)*, pages 16–21, Sweden, 1999.

[17] Dieter Fensel and Christoph Bussler. Upml as a standard. IBROW Deliverable D22, Vrije Universiteit, January 2003.

[18] Mario Gomez, Enric Plaza, et al. Libraries for information agents. IBROW Deliverable D4, IIIA, Barcelona, January 2002.

[19] Mario Gomez, Enric Plaza, et al. Wim application. IBROW Deliverable D16, IIIA, Barcelona, January 2003.

[20] T. R. Gruber. Towards principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies*, 43:907–928, 1995.

[21] G. Heineman and W. Councill. *Component-based software engineering*. Addison Wesley, Boston, May 2001.

[22] Maite López-S'anchez, Juan-Antonio Prieto, and Richard Benjamins. Web services. IBROW Deliverable D23, iSOCO, January 2003.

[23] E. Motta and W. Lu. A library of Components for Classification Problem Solving. In *PKAW: The 2000 Pacific Knowledge Acquisition Workshop*, 2000.

[24] E. Motta, K. O'Hara, and N. Shadbolt. Grounding GDMs: A structured case study. *International Journal of Human-Computer Studies*, 40:315–347, 1994.

[25] Enrico Motta, Monica Crubezy, Wenjin Lu, , and Mark Musen. Web interface. IBROW Deliverable D9/14, KMI, Open University, January 2003.

[26] Enrico Motta, John Domingue, Liliana Cabral, and Mauro Gaspari. Reasoning service. IBROW Deliverable D14, KMI, Open University, January 2003.

[27] Enrico Motta and Wenjin Lu. Classification library. IBROW Deliverable D1, KMI, Open University, January 2001.

[28] Boris Omalayenko and Frank van Harmelen. Standardization awareness. IBROW Deliverable D21, Vrije Universiteit, January 2003.

[29] Boris Omelayenko, Monica Crubezy, Dieter Fensel, Richard Benjamins, Enrico Motta, Mark Musen, and Ying Ding. *Spinning the Semantic Web*.

[30] Borys Omelayenko. Upml v2. IBROW Deliverable D5, Free University of Amsterdam, January 2003.

[31] Borys Omelayenko and Dieter Fensel. Component adaptation. IBROW Deliverable D12a, Free University of Amsterdam, January 2003.

[32] Dnyanesh Rajpathak, Enrico Motta, and Arthur Stutt. Applications of scheduling library. IBROW Deliverable D13, KMI, Open University, January 2003.

[33] Dnyanesh Rajpathak, Enrico Motta, Zdenek Zdrahal, and Arthur Stutt. Scheduling library. IBROW Deliverable D3, KMI, Open University, January 2003.

[34] A. ten Teije. *Automated Configuration of Problem Solving Methods in Diagnosis*. PhD thesis, University of Amsterdam, Amsterdam, The Netherlands, 1997.

[35] A. ten Teije and F. van Harmelen. An extended spectrum of logical definitions for diagnostic systems. In *Proceedings of DX-94 Fifth International Workshop on Principles of Diagnosis*, 1994.

[36] Annette ten Teije and Frank van Harmelen. Wp4.1, wp4.2: Task and method adaptation. IBROW Deliverable D12b, Free University of Amsterdam, January 2003.

[37] Chris van Aart, Anjo Anjewierden, Wouter Jansweijer, Jan Wielemaker, and Bob Wielinga. Interoperability. IBROW Deliverable D10, SWI, University of Amsterdam, January 2003.

[38] B.J. Wielinga, C. van Aart, A.A. Anjewierden, and W.N.H. Jansweijer. Ibrow final report. IBROW Deliverable D19, SWI, University of Amsterdam, March 2003.

[39] Bob Wielinga, Anjo Anjewierden, Chris van Aart, and Wouter Jansweijer. Brokering in ibrow. IBROW Deliverable D15, SWI, University of Amsterdam, January 2003.

[40] Fabio Zlot, Kathia Maral de Oliveira, and Ana Regina Rocha. Modeling task knowledge to support software development. In *Proceedings of the 14th international conference on Software engineering and Knowledge engineering*, pages 35–42. ACM Press, 2002.

# A Deliverables

Below a list of all deliverables of the IBROW project is given. The Delivarable Summary Sheets in PPR1, PPR2, PPR3 provide a short description of each deliverable.

Note that deliverable D11 was not produced due to funding problems at the U.S. (not funded) partner of the project (Stanford University).


IBROW:D1 Enrico Motta and Wenjin Lu. Classification library. IBROW Deliverable D1, KMI, Open University, January 2003.

IBROW:D2a A.J. Duineveld, B.J. Wielinga, V.R. Benjamins, and N. Christoph. Knowledge management library. IBROW Deliverable D2a, SWI, University of Amsterdam, February 2001.

Anjewierden:dac-overview Anjo Anjewierden. A library of document analysis components. IBROW Deliverable D2b (part 1), SWI, University of Amsterdam, February 2002.

Anjewierden:dac-library Anjo Anjewierden. Document analysis components. IBROW Deliverable D2b (part 2)[3], SWI, University of Amsterdam, February 2002.

IBROW:D2b Anjo Anjewierden. A library of document analysis components. IBROW Deliverable D2b Version 2, SWI, University of Amsterdam, February 2003.

IBROW:D3 Dnyanesh Rajpathak, Enrico Motta, Zdenek Zdrahal, and Arthur Stutt. Scheduling library. IBROW Deliverable D3, KMI, Open University, January 2003.

IBROW:D5 Borys Omelayenko. Upml v2. IBROW Deliverable D5, Free University of Amsterdam, January 2003.

IBROW:D6 Monica Crubezy, Mark Musen, and Guus Schreiber. Upml validation and tool support. IBROW Deliverable D6, Stanford University, January 2001.

IBROW:D7 V.R. Benjamins. Web workbench. IBROW Deliverable D7, iSOCO, Madrid, January 2001.

IBROW:D8 Chema Abasolo, Josep-Lluis Arcos, Eva Armengol, Mario Gomez, Ramon Lopes de Mantaras, and Enric Plaza. Component selection. IBROW Deliverable D8, IIIA, Barcelona, January 2003.

IBROW:D9 Enrico Motta, Monica Crubezy, Wenjin Lu and Mark Musen. Web interface. IBROW Deliverable D9/14, KMI, Open University, January 2003.


[3] See `www.swi.psy.uva.nl/projects/ibrow/deliverables/dac-library.pdf` for the most recent version.

IBROW:D10 Chris van Aart, Anjo Anjewierden, Wouter Jansweijer, Jan Wielemaker, and Bob Wielinga. Interoperability. IBROW Deliverable D10, SWI, University of Amsterdam, January 2003.

IBROW:D12a Borys Omelayenko and Dieter Fensel. Component adaptation. IBROW Deliverable D12a, Free University of Amsterdam, January 2003.

IBROW:D12b Annette ten Teije and Frank van Harmelen. Wp4.1, wp4.2: Task and method adaptation. IBROW Deliverable D12b, Free University of Amsterdam, January 2003.

IBROW:D13 Dnyanesh Rajpathak, Enrico Motta, and Arthur Stutt. Applications of scheduling library. IBROW Deliverable D13, KMI, Open University, January 2003.

IBROW:D14 Enrico Motta, John Domingue, Liliana Cabral, and Mauro Gaspari. Reasoning service. IBROW Deliverable D14, KMI, Open University, January 2003.

IBROW:D15 Bob Wielinga, Anjo Anjewierden, Chris van Aart, and Wouter Jansweijer. Brokering in ibrow. IBROW Deliverable D15, SWI, University of Amsterdam, January 2003

IBROW:D16 Mario Gomez, Enric Plaza, et al. Wim application. IBROW Deliverable D16, IIIA, Barcelona, January 2003.

IBROW:D17 V.R. Benjamins. Project presentation. IBROW Deliverable D17, SWI, University of Amsterdam, March 2000.

IBROW:D18 V.R. Benjamins. Dissemination and use plan. IBROW Deliverable D18, SWI, University of Amsterdam, August 2000.

IBROW:D19 B.J. Wielinga, C. van Aart, A.A. Anjewierden, and W.N.H. Jansweijer. Ibrow final report. IBROW Deliverable D19, SWI, University of Amsterdam, March 2003.

IBROW:D20 IBROW Consortium. Technology implementation plan. IBROW Deliverable D20, SWI, University of Amsterdam, March 2003.

IBROW:D21 Boris Omalayenko and Frank van Harmelen. Standardization awareness. IBROW Deliverable D21, Vrije Universiteit, January 2003.

IBROW:D22 Dieter Fensel and Christoph Bussler. Upml as a standard. IBROW Deliverable D22, Vrije Universiteit, January 2003.

IBROW:D23 Maite López-Sánchez, Juan-Antonio Prieto, and Richard Benjamins. Web services. IBROW Deliverable D23, iSOCO, January 2003.