OpenKnowledge

FP6-027253

Functionality Description of the OpenKnowledge Kernel

Ronny Siebes[1], Frank van Harmelen[1], Spyros Kotoulas[1],
Dave Dupplaw[5], Dietlind Gerloff[4], Fausto Giunchiglia[2],
Maurizio Marchese[2], Fiona McNeill[4], Andrian Perreau de Pinninck[3],
Dave Robertson[4], Marta Sabou[6], Carles Sierra[3],
Lucia Specia[6], Austin Tate[4], and Mikalai Yatskevich[2]

[1] Faculty of Sciences, Vrije Universiteit Amsterdam, The Netherlands
[2] Dept of Information and Communication Technology, University of Trento, Italy
[3] Artificial Intelligence Research Institute, IIIA-CSIC, Spain
[4] School of Informatics, University of Edinburgh, UK
[5] School of Electronics and Computer Science, University of Southampton, UK
[6] Knowledge Media Institute, The Open University, UK

Report Version: final
Report Preparation Date:
Classification: deliverable D2.1b
Contract Start Date: 1.1.2006 Duration: 36 months
Project Co-ordinator: University of Edinburgh (David Robertson)
Partners: IIIA(CSIC) Barcelona
Vrije Universiteit Amsterdam
University of Edinburgh
KMI, Open University
University of Southampton
University of Trento

The functional description of the Open-Knowledge System

Ronny Siebes[1], Frank van Harmelen[1], Spyros Kotoulas[1]
, Dave Dupplaw[5], Dietlind Geldoff[4], Fausto Giunchiglia[2], Maurizio Marchese[2], Fiona McNeill[4], Andrian Perreau de Pinninck[3], Dave Robertson[4], Marta Sabou[6], Carles Sierra[3], Lucia Specia[6], Austin Tate[4], and Mikalai Yatskevich[2]

[1] Vrije Universiteit Amsterdam, The Netherlands
{ronny,frankh,kot}@few.vu.nl
[2] Department of Information and Communication Technology (DIT),
University of Trento, Povo, Trento, Italy
{fausto|marchese|yatskevi}@dit.unitn.it
[3] Artificial Intelligence Research Institute (IIIA-CSIC), Barcelona, Spain
{adrianp|carles}@iiia.csic.es
[4] The University of Edinburgh, Edinburgh, UK
{f.j.mcneill|dr}@ed.ac.uk, gerloff@staffmail.ed.ac.uk
[5] University of Southampton, UK
davidd@ecs.soton.ac.uk
[6] Knowledge Media Institute, Open University, UK
{r.m.sabou|l.specia}@open.ac.uk

Abstract. This document is the functional description a Peer-to-Peer system that will be developed for the European Funded "Open-Knowledge" project. The purpose of the system is sharing, searching and invoking of services (i.e. pieces of Internet accessible executable code) in a completely distributed manner. We will describe what the system can do, what the goals are, how we evaluate it, which techniques are applied to make it possible and tell what the differences are compared to similar systems described in literature.

1 What is the Open-Knowledge System? An introduction

An open knowledge sharing system is one which places no specific boundary on the number of people who may share knowledge and which can readily be joined by people wishing to share or discover knowledge for complex tasks. We distinguish functional knowledge from content knowledge. Content knowledge is the data that is shared on the network, and that is queried by peers. This may be pictures, music, computational services, etc. Functional knowledge is information about the functionality of services, and the mappings and interactions between them required to use the content knowledge on the OK-system. Open-knowledge will provide mechanisms and tools that will hide the complexity of such functional knowledge. Achieving openness with 'content knowledge' is relatively easy, because all digital data can be shared and other projects **[(DAVE)refs to (semantic) P2P systems]** have already shown that this is feasible, also in a P2P setting. More difficult is openness in the functionality of the system where

the data is stored. We want to build a system where people can also write this functionality in easy readable and executable format. No perfectly open method of sharing knowledge can exist because of well known theoretical obstacles **[(DAVE) please some references to these obstacles]** to maintaining common knowledge in asynchronous environments and practical problems in gaining consensus on semantics for distributed information. In all open knowledge sharing, therefore, some compromise is reached to obtain a particular form or level of openness. Traditional forms of compromise are limiting:

The Worldwide Web : is easy to join but allows only limited forms of knowledge sharing for simple tasks. The content is mostly only readable by humans and also the way people can access this information is rather limited. For example, in most cases there is only a search engine which you can query by boolean connected keywords. Also the WWW is still rather data-oriented, which means that almost only documents (images, music, text etc) can be found and not yet much functionality, like user-friendly web-services. **[(DAVE) What is a user-friendly Web-service? Are these services with an HTML front-end instead of just a WSDL description? please explain]**

Semantic webs :are built through local markup of individual knowledge repositories or large corpora and ontologies built by large consortia or companies to standardize their vocabulary. To be generally useful, local markup or requires broad consensus on what terms mean, but to be confident of consensus in this absolute sense requires knowing who to consult about the ontology used.**(DAVE) Can you spell out what are the problems of opennes of the SWs both in terms of joining them and using them? This text still needs couple of deductions to find out what you mean...**

Web services : can be tuned to complex tasks but are specific to the programs (and people) connected with those tasks. This means that the programmer cannot predict in which combination with other web-services it also can be useful. Currently automatic web-service composition is still very weak and human made compositions are quite expensive because they require much expertise. **(DAVE) Same as before: - hard to join: compositions done by humans - hard to use: different views of the service bw. providers and users**

Grid infrastructures : can improve performance in service delivery but suffer the same scaling problems with respect to aligning vocabularies and functionality for knowledge sharing as Web services.

OpenKnowledge takes a different starting point than these traditional systems **(DAVE) different from?**. We begin with a strictly peer to peer architecture, in which there is no centralised database, overall consensus on ontologies, or pre-connected Web services. This assures a flexible, scalable and open system. Rather than relying on these (non-scaleable) facilitators for knowledge sharing, we rely on models of interaction that are initiated locally by peers and shared through the peer network, simplifying discovery, ontology alignment and service composition because of the narrower interaction context provided by these models. Crucially, the interaction models themselves are the currency of **(DAVE)functional?** knowledge transfer between peers. It turns out that by using this

richer style of communication we can obtain forms of query routing; ontology matching and composition that are dynamic (so adaptive to flexible peer groups) and support complex interactions using comparatively simple computational mechanisms.

The goal of this paper is to give an impression about the system that we will construct in the coming years. We give an overview of what our goals are in the next section. Section 3 gives a literature study providing an impression of related work and where we are different. After that, in Section 4 we take a user-oriented perspective in showing the expected functionality for the users of the system via mockups and task-flow diagrams. In our project we also have expertise on bioinformatics and emergency response. This expertise is used to extract requirements for our system and to evaluate its functionality. We discuss this in section 5. In section 6 we show how we are going to tackle the problems we need to solve in order to fulfil the requirements described in this document. After that, in section 7 we describe how we try to evaluate the quality of the different parts and the requirements of our system.

2 What are our goals of the Open-Knowledge System?

In this section we describe *what* we want to achieve with our system, by giving a generic requirement analysis where many of the requirements come from the case studies described in Section 5.

Our primary goal is to have a working system. This seems trivial, but many systems in research projects are often too ambitious or strand due to lack of communication between members of the project. Therefore, from the beginning of our project we put much stress on a shared vocabulary to align the terms that occur in the different areas of expertise. Also we start with a simple platform where the basic functionalities are simple but work. We have to make sure that the system architecture is flexible enough that it is easy to extend and easy to enhance the basic default functionalities.

We try to reach a large group of potential users, and therefore will pay attention to user friendliness. We will identify different types of users and make our system flexible enough to make it interesting both for dummies and experts. To make our goals specific we give a requirement analysis.

2.1 Requirement analysis

The requirements do not fall out of the blue but are based on discussions with the case-study partners from the Open-knowledge project, i.e. emergency response and bioinformatics, experienced programmers on P2P systems and research papers on P2P and/or Semantic Web systems.

– Large-Scale: in number of human participants. This first requirement points to a user-friendly system. Although the functionality system will not depend on a fixed user-interface, we will provide a default and easy to understand user-interface which supports all three different types of users described later in the system. Besides the user-interface also some basic functionality which should easy the life of the OK-user is part of the core research within the project. To illustrate this we give four examples:

    - *Automatic mapping (or automatic suggestions for mappings)* When the user wants to connect an output of a component to the input of another component, it can happen that the data-types will not have the same identifier. Automatic mapping information will help that user to connect the two components
    - *Automatic query relaxation*. It may happen that no answers can be found for a query posted by a user. In such a case the automatic query relaxation functionality can 'rephrase' the query based on, for example, background information, in order to increase the chance of relevant results
    - *Efficient component and peer discovery* A fast response for user queries is also very important. Therefore, efficient discovery of relevant components and peers able to play roles within those components is a *must have*.
- Large-Scale: in number of nodes. To provide the users with an attractive and therefore large set of components, many computers are needed to host the descriptions and to run parts of the roles described in the components. Therefore, the algorithms written as core functionality for the OK-system should be scalable to a large number of participating 'nodes' (i.e. computers).
- Usuable for different types of users for different types of tasks:
    - Average Web User: This user does not have programming skills, but is able to fill in forms and describe what (s)he wants. Skill similar to the skill required to use Google and to book a hotel online.
    - Visual Developer: This user understands workflows, (s)he is able to design diagrams/dataflows. Usually a domain expert (eg Biologist, Physicist), able to formulate and decompose a complex problem into simpler parts.
    - Programmer: This user understands programming languages, (s)he is able to conceptualize, design and develop complex applications.
- Low integration effort for existing systems: It should be easy to integrate existing services and provide interfaces using legacy protocols.
- Open Participation
    - Incremental Development: Users can build on existing functionality provided by other users.
    - Enabling Reuse: Participants can execute functionality provided by others.
    - Distributed Computation: The functionality can be primitive, meaning running on one computer, but also composed, where more than one computer is involved in the process.
- Means to evaluate functionality offered by others: the system should provide feedback to users about the QOS (quality of service) from the different components and peers. One possibility is that the system can store and provide ratings about quality of the services given by users.

3 Why the Open-Knowledge system is a good idea?

Clearly, others have previously identified similar goals to Open Knowledge: sharing, searching and invoking services in a distributed and scaleable way. In this section we discuss some of the relevant other approaches that have aimed for these goals: Web services, Grid services, P2P systems and multi-agent systems. We won't aim to provide

a full-scale literature study here. Instead, we will identify the key ideas behind each of these approaches, and argue why Open-Knowledge occupies a unique niche in this landscape.

Web Services. Perhaps the most closely related effort to Open-Knowledge is the work on Web Services. The aim of Web services is also to enable invoking and executing services in a distributed, scaleable and interoperable manner. The work on *semantic* Web services adds to this the goals to automatically locate and compose such services in an open and heterogeneous environment like the Web.

Both approaches (Web services and Open Knowledge) make the obvious move of achieving these goals by turning services into information objects (Web service *descriptions*), that are then the subject of reasoning tasks for search and composition.

The Open Knowledge approach is in some ways more flexible then the Web services approach, and in other ways more restricted. Open Knowledge is more restricted because semantic Web service work aims at automatic on-line composition of simple services into complex services, by means of intelligent algorithms (e.g. based on configuration or planning). In contrast, Open Knowledge restricts itself to executing predefined "workflows" of services (the "interaction models" to be discussed later in this paper). The only decision that Open Knowledge makes at run-time is which service is executed at which agent (i.e. "recruiting", not composition).

This recruiting aspect of Open Knowledge is more general than the Web service architecture because it separates the advertising of a service from running a service, whereas in the Web service architecture it is generally assumed that advertisements of service functionality are accompanied with the name of the executor of the service. In short: the matching goals of both approaches are the same (finding a service that matches a given functionality), while the composition goals of both approaches are different: Open Knowledge aims to recruit peers to execute predefined workflows, whereas semantic Web services aims to automatically compose complex workflows out of atomic services.

Furthermore, Open Knowledge explicitly acknowledges the need for approximate matching of service requests with advertisements, whereas this is only marginally the case in semantic Web services, and entirely absent in regular Web service work.

Finally, Open Knowledge aims explicitly for a distributed storage model for the workflows and service descriptions, whereas all the dominant Web service architectures (UDDI for regular Web services, WSMX for semantic Web services) assume a centralised architecture.

Grid Services. The general area of "Grid services" is even less well circumscribed than "Web services", hence it is even more difficult to make a crisp comparison. In contrast to Web services, grid-services are typically organised in fixed workflows. This makes them more similar to the Open Knowledge approach. On the other hand, Grid services emphasise various aspects that are ignored in Open Knowledge: long term stability of services, provenance, quality of service and resource monitoring. Similar to Web services, Grid services differ from Open Knowledge by advertising a service functionality together with the identification of the service-provider, whereas Open Knowledge decouples these two (and hence allows for a separate "recruiting" step). Finally, and

perhaps most importantly, most Grid systems provide only a centralised mechanism for advertising services and workflows, while Open Knowledge aims for a fully distributed mechanism.

In particular the myGrid project ¡http://www.mygrid.org.uk/¿ is in many respects close to the goals of Open Knowledge in it's use of preconfigured workflows, and it's approach to manual composition of such workflows. However, it relies on centralised storage of such workflow patterns, which is in sharp contrast with the fully distributed architecture of Open Knowledge.

Peer to peer systems. Obviously, Open Knowledge is close in spirit to the work on peer to peer (P2P) systems. The central P2P ideas of distributed storage, lack of centralised address registers and of symmetric roles of each peer as both provider and requester are fully adopted by Open Knowledge. Nevertheless, Open Knowledge makes two important deviations from most P2P systems. First, most P2P systems aim at data-sharing, whereas Open Knowledge aims at *service sharing*. Of course, data sharing is simply a special case of service sharing (namely sharing a data-access service), making the Open Knowledge system more generic. Secondly, Open Knowledge is in the (small but rapidly growing) family of *semantic* P2P systems, which use rich descriptions of the content that each peer has to offer for purposes of routing queries through the network.

Agents A final class of closely related systems is that of multi-agent systems. In general, there is a superficial similarity between multi-agent and P2P systems: distributed sets of autonomous processes exchanging information. However, on closer inspection, there are rather significant differences. In particular, agent systems have highly structured architectures inside each agent (often relying on cognitive metaphors for their architectural constructs, witness the BDI architecture (Believes, Desires Intentions)), while P2P systems typically treat their peers as atomic. Finally, agent-systems emphasise their pro-active nature (autonomously reacting on their changing environment), while P2P systems in general (and hence Open Knowledge) assume more classical reactive stance.

The differences and correspondences described above are all summarised in the table below. This table shows both that Open Knowledge inherits many aspects from other approaches, but that at the same time it occupies a particular niched of features not yes explored by others.

4 How would the system look like?

This section should give the reader an impression of what happens after downloading the default client. We do this by showing prototype screen-shots and by providing task diagrams that show the expected behavior of the system from a users perspective.

First, we do not assume that each user has the same skills and interests with respect to possibilities with our OK-system. Some people only want to invoke some simple services that run somewhere on the network like a music-sharing service. Some other people try to combine some services on the network to get a more complex service. Programmers on their turn may add new services, for example by writing interaction

Web services	**similarities:**	service-oriented, distributed, automated search based on semantic descriptions	
	differences:	**Web services**	**Open Knowledge**
		composition of atomic services	predefined workflows
		fixed link to executing party	dynamic recruiting
		centralised advertising	distributed
		equivalence matching	approximate matching
Grid Services	**similarities:**	service-oriented, fixed workflows distributed	
	differences:	**Grid services**	**Open Knowledge**
		provenance	absent
		QoS	absent
		resource monitoring	absent
		centralised advertising	distributed
		fixed link to executing party	dynamic recruiting
Peer to Peer Systems	**similarities:**	distributed, scaleable, symmetric roles of each peer	
	differences:	**P2P Systems**	**Open Knowledge**
		aimed at data-sharing	service sharing
		independent of content	exploit semantics
Agent Systems	**similarities:**	distributed, scaleable, symmetric roles of each peer	
	differences:	**P2P Systems**	**Open Knowledge**
		cognitive architecture	none
		pro-active behaviour	reactive

Fig. 1: Open Knowledge compared to other approaches

models for existing web-services. To formalize this matter a bit, we identified three types of users based on increasing the complexities on the things (s)he can do on the OK-system. Every more complex user can do all tasks of the simpler user plus something extra. First, we have the Average Web User that can start a browser and or an easy KaZaa alike client. The only thing that (s)he needs to do is to fill in forms needed as input for a given task, or to find components that match the task that the user has in mind. Second, we have the Visual Composer who understands how components can communicate and (perhaps via an easy user-interface) combines components and/or make more complex taskflows between them. Third, we have the programmer who can (1) wrap legacy components to OK-components, (2)writes the interaction language telling how to communicate with the component and (3) is able to write a semantic description of the component which is needed by the discovery functionality of the OK-system to match user queries with components.

To make the tasks more explicit, we identified five UML use cases shown in Figure 2 which a user can perform:

– **The Execute use case** is the process where a user wants to perform a task on the OK-system. This can be anything, like sharing pictures, finding music etc. Note that the task can be very simple like filling in a field from a webform to something complex like starting an application. There are several ways for a user to do a task, which now will be described in increased order of complexity:

  - **User knows task and the roles for the task are automatically instantiated by peers.** In this case, a user knows what to do. For example, (s)he could know the URL of a website which is actually hosted on a node in the OK-network. Now, the only thing that the user needs to do is, for example, filling in a form and get the results. In this way the user does not even have to be aware that the service is connected to the OK-network.
  - **User needs to find components that fulfill the task but its roles are automatically instantiated.** In this case, a user first searches for the appropriate modules that reflects the task that the user has in mind. It can search via a website (as in the previous example, by just filling in a search field), or a specially developed OK-application/client for this purpose. For example, it could be something similar to the user interface of the KaZaa client, or like the mockup shown in Figure 2ae. Although we mention it here, we make an explicit distinction between the search use case and the execute use case because they are two different activities and involve completely different aspects of the OK-system. Namely, execution needs the *Peer Discovery Service (PDS)*, and searching needs the *Component Discovery Service (CDS)*. Although not sure yet, this PDS could check that peers do what they promise, and change their reputation accordingly. Also when no peers are found, the *Mapping Service* can find components that are similar to the one that has to be instantiated with peers and for which peer are subscribed. This means that peers that are subscribed to other components but are similar in functionality (via mapping) to the one of interest, could fulfill the role. Besides this, when the writer of the component allows it (described in the description of the component), a user

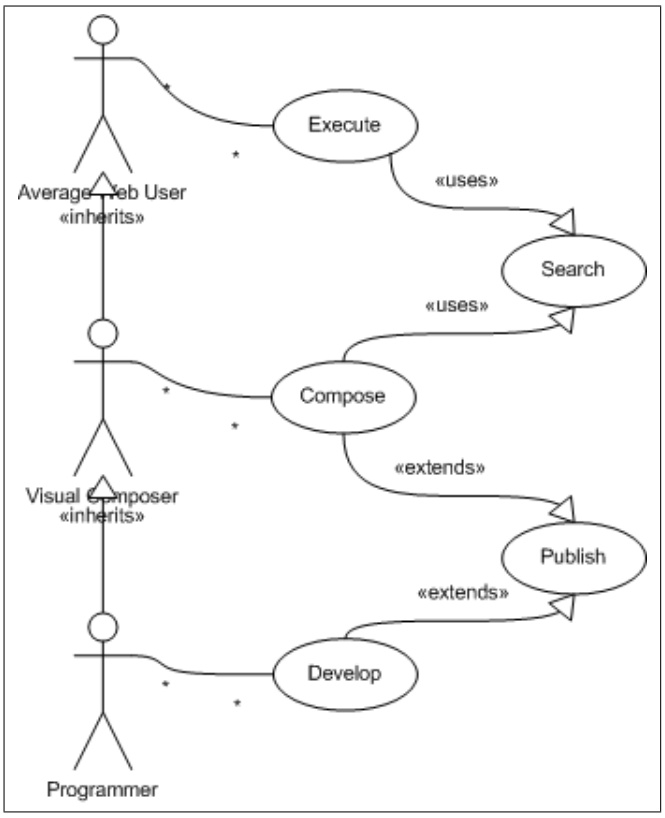

Fig. 2: Scheme of different user types, where each more complex user can do all tasks of the simpler user with something extra. Namely, the *Average Web User* only searches for Components and executes roles described in them, and the *Visual Composer* also can compose and publish new Components. The *Programmer* is the most advanced user who is able to write Components from scratch, for example by wrapping existing web-services into the OK interaction modelling language.

can also download the component so that (s)he has a local copy and can run the component on his/her own machine.

- **The Search use case.** As already said in the previous point, users can search for components by giving queries. The group of Amsterdam will be responsible to add this functionality to the OK-system, namely, matching user queries with modules. It could be that no satisfying components are found, then the user him/herself can rephrase the query or makes use of the *Query Relaxation Service* that automatically tries to rephrase the query into, for example, synonym terms. An example is that a user searches for g̈et a histogram from a pictureänd the relaxation service would rephrase it to g̈et a histogram from an image.
- **The Compose use case.** The second type of users (Composers), are able, for example by a d́rag-and-dropúser interface, to combine two or more components. In this way they make new composed components. For example, a simple g̈et-most-important-keywordsc̈omponent (with as input a text document and as output a set of words) can be made by combining an existing or newly developed component that stems the words from a document and another existing or newly developed component that has as input this set of words and removes the stop words. The interaction model for the new composed component will be made by the user, with help of the system. For example, when a user tries to link the input from one component with the output of another component and the datatypes are different, the Mapping Service can find out if the datatypes are synonyms or not (e.g. INT and INTEGER).
- **The Publish use case.** In this use case, a new component or combined component can be published on the open-knowledge system. The system takes care that the new component is stored in the distributed network of peers that subscribed themselves to the Discovery Service.
- **The Programming use case.** This use case is the most demanding one from a user perspective, namely here the user is able to (1) write interaction models in the OK-interaction language and/or translate other interaction models into the OK-interaction language, (2) write semantic descriptions needed by the discovery services to find appropriate components to user queries, (3) wrap legacy components (e.g. databases or pieces of JAVA or C code) and (4) accessing data that says something on what an invoker may do with the component (for example, if a user may download the whole component or only invoke it) and about the author him/herself (needed to derive trust). We decided that each component at least has an interaction model, and semantic descriptions and accessing data. The developer may keep the code (needed to run the component) private or public by putting it in the code part of the component so that other people can run the component locally.

Now that we have identified the users and the tasks, we show a 'mock-up' of our envisioned system.

Searching components In Figure 3 we show a mockup of how we expect the first default OK-client would look like. In the figure a user has a task in mind, namely the very simple task "sum two numbers". After typing in the search query in the appropriate

field, the query is automatically sent to a ”Component Discovery Service (CDS)” which is completely hidden for the user. The default implementation comes with a default CDS (running on the OK-network), but the user can also specify another one. The CDS is described in Section 6.4. When the query matches with components known by the selected CDS, they will be given back to the querying peer and shown in the user-interface. Each component has some standard information like the name of the component, the author, the trust level, the number of peers that are subscribed to play roles in the component, the datatypes of the input and output, and a description in natural language that explains what the component does. When the user decides to select a component, the only thing (s)he needs to do is to select the row in the table with components. It could be that not enough, if any, results were found for the query of the user. When that occurs, the user could rephrase the query in the hope to get more results. It can also use the ”Query Relaxation Service” that automatically rephrases the query

Running a component When the user selected the component, the system will ask (when the publisher of the component allows it) if (s)he wants to download the code for performing roles (described in the interaction model that comes with the component). Making a local copy has the advantage that there is no network traffic needed, but it will consume more computational resources on the user's machine. For now, we assume that (s)he did that, as shown in Figure 4. When the component expects input from the user, the component will have at least one user-friendly interface like shown in the mockup. The user can fill in the parts expected by the component. After that the user can press the green ”run” button, and the component will do its job. Behind the scenes plays a complex process, namely besides the local peer also other peers can play roles in the interaction model of the component, these peers should be found. Finding these peers is done at the ”Peer Discovery Service (PDS)” described in Section 6.4.

Composing components A more experienced user, which we call the *composer* could decide to make some own components by linking other existing components. It is very similar to visual programming like in the myGrid workflow editor [1]. The things that the user has to do are quite simple. First, a set of components need to be found which will be the building blocks of the composed component. Secondly, the components have to be linked, which means connecting the output of one component to the input of the other. A more advanced user may write complex interaction models between two or more components, in, for example, the LCC language. It could be that the input and output datatypes or instances from these data-types are not compatible. In some cases it is clear that they do not match (for example a binary datastream with a URL), but in other cases the objects are (almost) the same (for example an `INTEGER` with a `NATURAL_NUMBER`). The OK-client has pointer to a ”Mapping Service” that may try to figure out if the data-types can be mapped, and if yes, this mapping is downloaded and added between the links.

[1] http://www.mygrid.org.uk

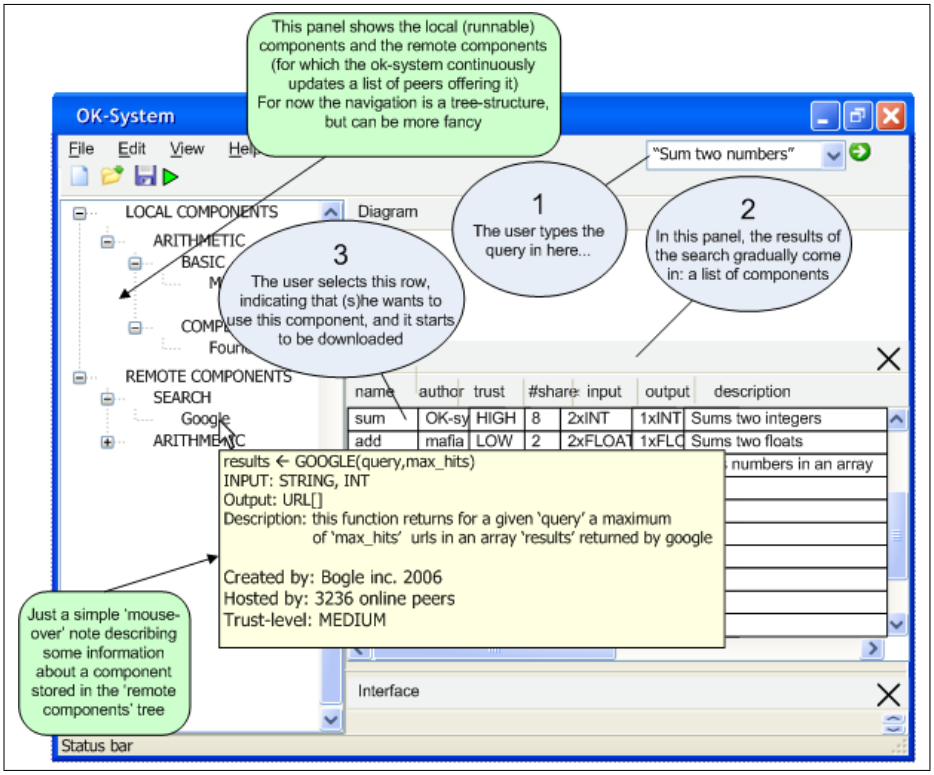

Fig. 3: Mockup of the future OK-system, where a user tries to search for components. (S)he wants to have a component that is able to sum two numbers. First, (s)he types it in the search-field. Second, the results from the network gradually fill the table. Third, the user selects the component of interest.

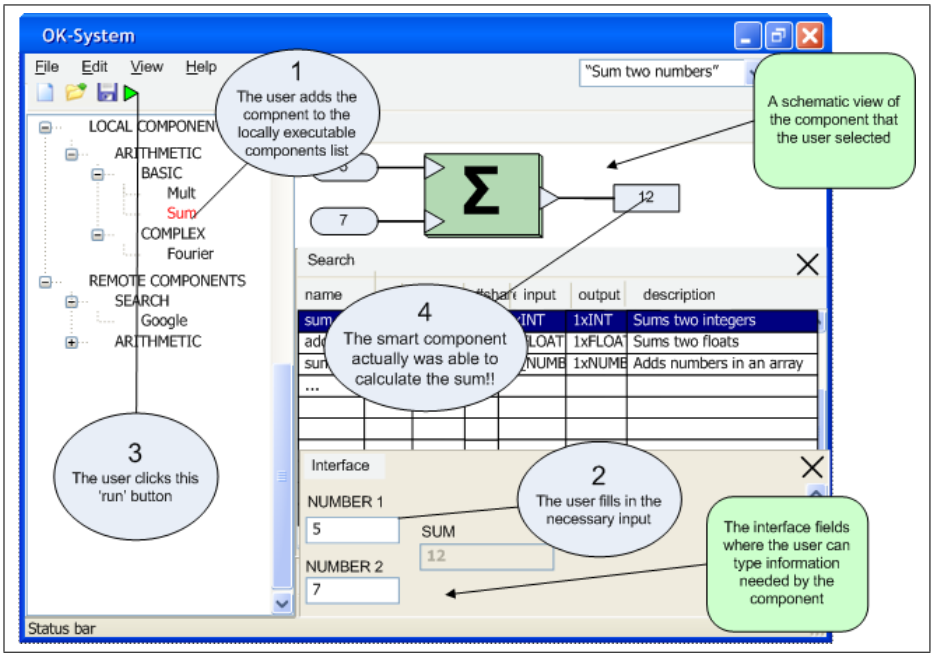

Fig. 4: Mockup of the future OK-system, where a user tries to run a component. First, (s)he decided to copy the component including its code to the ”local components” storage. Second, (s)he fills in the necessary missing information in the user interface which comes with the component. Third, the user clicks the ’run’ button and fourth, in this case the component was able to execute the code.

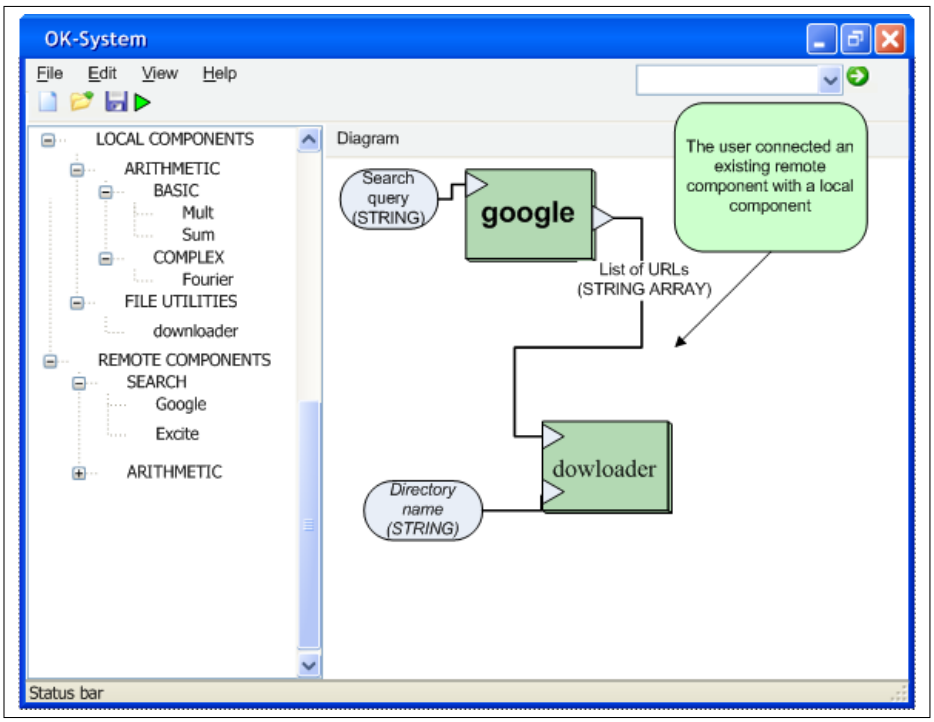

Fig. 5: Mockup of the future OK-system, where a user tries to compose a component out of two other components. First, (s)he selects the two components which will be connected. Second, the output of one component is connected to the other. The user can now decide to publish the composed component to the network, which would not make much sense in this case because its a rather trivial combination. Or the user may actually execute it by first filling in the missing input and second where the system finds a peer to run the remote ”Google” component.

Developing new components The most skilled user, which we call the *programmer* is able to develop components from 'scratch' which means that either (s)he writes code and make it accessible via the Interaction Model language (the standard in OK will be LCC) and give a semantic description of the system. Also existing web-services can be wrapped into the LCC language and be accessed from peers in the OK network.

4.1 Schematic overview of the search and run functionality

In Figure 6 we show the ternal operations in the system for search and run functionality via an UML activity-diagram. In this diagram, the different core services (e.g. Component Discovery Service and the Mapping Service) can be found together with the relations between them.

5 Can you give me some examples where the system will be useful?

In this section we briefly describe two different case-studies, namely Emergency Response and bioinformatics. For each of them we give some typical scenarios in which our system can be of help.

5.1 Bioinformatics *Dietlind*

Brief generic case-study overview: Modern biological and medical research benefits tremendously from the "genomic explosion", i.e. the technological advances that allow the genetic information of individual organisms to be elucidated in a tiny fraction of the time necessary to do this in the past. An important agreement within the research community is that much of this information, and many other experimental results that are based on this information, is made accessible publicly. This is typically accomplished either by offering unrestricted user access to centrally maintained resources containing pre-computed data (hosted by sites receiving research funding to maintain them), or by allowing the user to run specified programs interactively, to search the database or perform more sophisticated analyses, via a WWW-server interface. This form of interaction with the data is primarily aimed at supporting the research of biologist researchers who are "semi-experts" in the field of bioinformatics; their interest is in optimally exploiting the resources and data offered to them but they do are not usually interested (or able) to develop analysis software themselves. In our general description of user profiles this user was described as 'Visual Developer' (Section 2.1). The 'Average Web User' described in the same section can also benefit from bioinformatic resources. However, this latter type of user is anticipated to interact merely to retrieve pre-computed data since his/her lack of expertise would preclude judgment of the confidence levels associated with computational predictions etc.

While the current modes of interaction that are available to the bioinformatics community have proven to accelerate biological research tremendously by allowing access to data produced in experiments elsewhere, some limitations are becoming noticeable (see below). In the OpenKnowledge project this testbed will primarily serve to investigate:

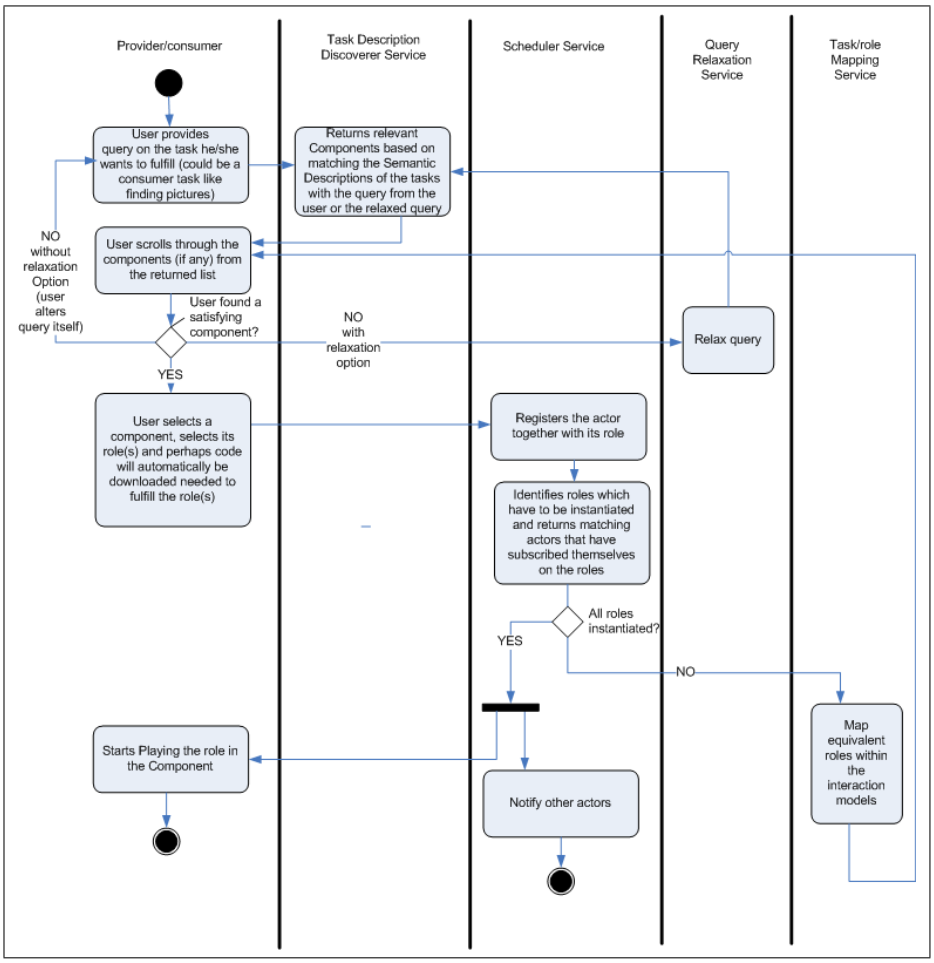

Fig. 6: UML taks diagram, showing the relationships between the user and the (default) services running on the OK-network.

– its potential for reproducing the range of bioinformatic analyses, and ”customer satisfaction”, that is achieved presently by the centralised resource model.
– its potential for addressing some of the limitations of the centralised resource model that are affecting biologist researchers in practice.

To this end prototype systems are being developed that are targeted at specific bioinformatic problems (or ”scenarios”). In the first instance we hope to demonstrate the potential of OpenKnowledge relating to specific case examples. More generally, the interaction models underpinning the prototypes are conceptually applicable to a large variety of other bioinformatic problems, and could be modified for use in a great variety of current biological research.

Examples of bioinformatic case-studies: For the purpose of this functional requirements document, two bioinformatic problems that are currently investigated in the OpenKnowledge project are briefly described below:

– **Structural models for yeast proteins scenario: quality control through comparison:** a number of computational biology research groups specialize in producing structural models for proteins based on the specific sequence of amino acid building blocks that makes up a protein molecule of interest. Where they are accurate, predictions of this kind can provide highly valuable clues for biological research relating to these proteins. However, the confidence in each individual prediction is not easily estimated and many of the proposed models are completely wrong. In practice, users of the databases providing access to the predictions attempt to ”validate” the predictions by comparing the models proposed by different groups. Where the models (which are distributed as lists of 3-D coordinates for each atom in the protein molecule) agree, over the whole or a part of the protein sequence, the model is deemed an approximately correct representation of the actual 3-D molecular structure of said protein. While this approach is obviously only useful if different methodology was applied to produce the models used in the comparison, the community of expert groups in protein structure prediction consists of a respectable number of research groups that employ different, or partially different, methods, which makes this scenario interesting to investigate in its own right. However, in the first instance, it is treated here as a prototype minimum scenario that depends critically on only very few functionality requirements. In this prototype, the interaction models basically aim to retrieve pre-computed data from resources that do not (intentionally) participate as peers in the network. The prototype requirements also apply to many equivalent sub-scenarios in larger, more complex problems, and extensions to the prototype (e.g. where the comparison of data becomes is carried out as an interaction itself, while it is performed by the user in the prototype scenario). The functionalities expected of an Open Knowledge prototype of this kind are:
  • *database/webserver ”wrapping”* the system should support the inclusion of resources as peers that participate without their knowledge. This is important to enable currently centrally maintained resources (e.g. sequence and/or structure databases) to be linked into the peer-to-peer network and will be accomplished by converting the relevant web sites into web services via a wrapper that enables them to ”participate” in OpenKnowledge interactions

- *interaction with the user interface* the system should allow a degree of interaction with the user, particularly in the selection of peers consulted to tackle a problem. This is important here because this scenario is targeted at the intermediate user ('Visual Developer' type).
- *availability of provenance (and possibly peer reputation) data* the system should enable the user to ascertain what methodology was applied (and/or which other peers have been consulted) by a peer to arrive at the models offered.

– **Proteomics scenario:** data sharing and quality control The number of different proteins that are present in a certain tissue (e.g. human liver cells) under certain conditions (e.g. after intake of alcoholic beverages) can easily reach several hundreds, or more. Characterizing this contingent of proteins, i.e. identifying as many as possible of the proteins present and considering other information that is known about each of them, is crucial for biologists trying to understand the underlying regulation and adaptation of the respective biological system (here the human liver). A technologically advanced strategy to characterize proteins at a large scale involves fragmenting the proteins and the use of mass spectrometric analysis to determine the amino acid sequence of each fragment. This technique is often referred as "Proteomics" by biologist researchers. To accomplish an actual identification of each protein, the fragments are compared to the sequences stored in centrally maintained databases. This is undertaken either in-house or via WWW-servers and the chances for identification vary depending on the quality of the samples subjected to mass spectrometry. Given this limitation, proteomics experts express a great interest in gaining access to data resulting from their colleagues' research, to help them with their own analyses. Interestingly, even access to data that was of no further use to some researchers and was discarded, could be of interest to others. While this scenario involves various additional complexities compared to the one described above (such as a prospective user community at the 'Average Web User' level (Section 2.1)), it poses an interesting data sharing challenge as a sub-scenario, for which OpenKnowledge may be able to provide a solution. The information to be shared in this sub-scenario is primarily protein sequence information, functional annotation, and mass spectra (which can be represented as lists of numbers specifying peak positions and intensities).

  The functionalities of an OpenKnowledge-oriented system we would like to investigate and test in this case are primarily:

  - *distributed data storage* the system must support shared access to data produced by other peers. The benefits of peer-to-peer data sharing should become apparent, both with respect to the accuracy of research results users obtain using the OpenKnowledge system and with respect to the effectiveness of the process (by comparison to the current situation where the data from other research groups is not acceessible).
  - *incremental development* the system should support the passing on of interaction models (or sequences of interaction models) used by different peers to interpret the mass spectra and identify the protein to which the fragment may belong. To an extent this is addressed by the provenance requirement listed above but we would expect this form of knowledge to play a more interesting

role in this scenario, to allow some peers to consult the network for expert advice about how to analyze data as an alternative to simply perusing the peer's data.

- *query routing and constraint/semantic matching* by contrast to the first example, the targeted user is predominantly at the 'Average Web User' level and will expect that decisions are made for them, e.g. regarding which peers to consult. Besides a more autonomous interface this is likely to pose challenges with respect to matching descriptions of user interests and offered expertise and services by other peers.

5.2 Emergency Response

Brief generic case-study overview.

Emergency crises - be it a natural, industrial or metropolitan disasters - are complex situations, with large numbers and varieties of mobile agents medical and rescue teams, police, fire fighters and generic people involved and, in the case of emergency personnel, appearing on the spot at short notice. These different teams generally have incomplete or even contradictory knowledge of the crisis situation. Realtime access to information and knowledge is clearly essential since all emergency response protocols are developed and implemented through the analysis of information. Information comes usually from a variety of distributed sources and a great part of the information in emergency response is spatial and needs to be mapped and appropriately visualized. During an emergency it is critical to gather the right data, at the right time, display them logically and contextually in order to respond and take appropriate action.

Moreover, emergency response activities usually involve a range of different teams from various organizations with their own systems and services. At present, most of the sharing of relevant information that is required for dealing with emergency is limited to a raw data exchange with all the syntactical and semantical conversion problems.

In the Open-Knowledge project, the challenges of such a testbed are

- the rapidity of formation of emergency coalition often very intense and opportunistic communities of practise - and the need to provide answers in real time;
- the necessity of high quality answers
- the richness of the data involved - geographical data deals potentially with a large number of metadata and layered information

This balances well with the previous bioinformatics scenario in which communities of practise are usually slower to form, longer lasting and the pace of cooperation is not normally under such extreme time pressure.

Examples about typical scenarios in this case-study.

For the purpose of this functional requirements document, let's briefly describe a number of possible scenario that we are currently analyzing:

- **Geo Database management scenario:** in this scenario the focus is supporting the coordination, integration and distribution of GIS data among the agencies responsible to maintain the geodata of the territory (local authorities). This is a "preparedness" scenario for different kind of emergency, in the sense that, it is a prerequisite

for an accurate and appropriate data usage during the actual and specific run-time emergency scenario. The main issue here is that every agency produces a large number of datasets, but, even if they are published as ”public data”, often their metadata are not available by the other agencies. Moreover, the individual agencies could provide some additional services (specific databases, capability to create additional thematic layers, specific expertise etc..) but most often they do not explicitly provide the sequence of operations needed in order to use those functionalities. The functionalities of a OK oriented system, we would like to investigate and test in this use-case are:

- coordination and integration of logically distributed geo databases - the system should support users and system managers in the integration effort of existing systems, while maintaining individual autonomy and responsibility;
- semi-supervised support for semantic searches within the peers and semantic mappings among their offered data and/or services - the system should support users to find appropriate interaction models, peers roles, services as well as navigate/search through the distributed repositories;
- support to different visualization channels - depending on the data, on the devices used and on the purpose of the request, it is important to support versatility in visualization: providing common structures but also allowing customised visualization by both peers and interaction designers;

– **Emergency signaling scenario:** in the not-too-distant future, every citizen, vehicle, in-transit package, and other active device might be treated as a potential sensor or responder. Individuals or vehicles that want to signal an emergency (or simply need help), as well as local, regional, national, and international emergency agencies, could look up specialized capabilities or find local assistance through a much more responsive and effective environment. Such systems will need to interoperate both at the technical and at the semantic level. The functionalities of a OK oriented system, we would like to investigate and simulate in this use-case are:

- support semantic interoperability among the involved peers - each involved peers will use contextual information to provide or process (signaling) information;
- support the scalabilty of the system as the number of potential peers increases radically from a limited number of emergency agencies, teams and personnel to potentially the whole population of a given area.

5.3 SW Tool development

Despite a conscious effort made by the Semantic Web community to migrate and apply its semantic techniques in open, large-scale, distributed and heterogeneous Web environments, existing Semantic Web tools rely on a set of simplifying assumptions mostly considering a controlled, small-scale, centralized and homogeneous setting. Indeed, these first generation tools can only be considered as a proof of concept of the potential of semantic technologies. For example, most tools rely on a single ontology and therefore function only within the domain defined by this ontology. This is the case of two of the Semantic Web tools developed at KMi. The first tool, AquaLog

is an ontology-based question answering system that interprets a question asked using natural language and uses the structure and instances of an ontology to answer the question. While AquaLog is portable from one domain to another, it is constrained to answer questions phrased using concepts defined by the ontology that it incorporates at a given moment. Our second tool, Magpie, is a semantic web browser that provides new mechanisms for browsing and making sense of information on the semantic Web. This tool makes use of the ontology-based semantic annotation associated with a Web page to help the user get a quicker and better understanding of the information on that Web page. Magpie is portable from one domain to another as it allows the user to choose the appropriate ontology from a list of ontologies that are known to the tool. However, similarly to AquaLog, the current version relies on a single ontology active at any moment in time. This limits the scope of the sense making support to the content of the current ontology.

To fully prove the potential of the Semantic Web, existing tools should be reimplemented by dropping the assumptions taken in the early years of the Semantic Web. Concretely, this could mean that the new generation of tools will be capable of dynamically selecting and combining appropriate semantic data directly from the Web, i.e., to perform a self-adaptation to the current domain of interest of the user. For example, AquaLog should be able to answer questions in any domain by automatically selecting and combining appropriate metadata available on the Web. Magpie should be capable to provide appropriate semantic guidance for Web pages in any domain thus allowing the user to freely navigate the Web without the need to install a new ontology every time the topic domain of the viewed page changes. Scaling up semantic web tools to deal with the large amount of dynamically changing online available metadata is not an isolated phenomenon. Indeed, generally speaking about the success factor that differentiates Web 1.0 from Web 2.0, Tim O'Reilly derives the following success criterion in a recent article "the value of the software is proportional to the scale and dynamism of the data it helps to manage".

From the perspective of semantic web tool development in general, and the KMi specific tools in particular, we identify three potential roles to be played by the OpenKnowledge project:

– **Provider of new technology that supports automatic resource discovery.** In particular the ontology mapping research will focus on developing dynamic ontology mapping techniques, i.e., robust and efficient mapping techniques that can be used at run-time by other tools. These techniques are essential in the case of both KMi tools and more widely for any tool that relies on combining distributed, heterogeneous semantic data. *2***Data provider.** The OpenKnowledge system could be a primary data source for SW tools. On one hand, it could offer access to several data discovery services that would be registered with the system. On the other hand, the OK system itself could exploit the wealth of semantic data (and not just services) brought in by its peers and the various mappings between these data sources to offer its own data discovery service (which could probably rely roughly on the same principles as the discovery of interaction models).

– **Development platform for SW tools.** We can also think of the OK system as a development platform for SW tools oriented towards modularity and resource sharing. Indeed, we envision that our own SW tools could be split in several components and published to the OK system as interaction models that capture the invocation logic between these components. This strategy would allow SW tools to share and exchange generic modules such as for example, ontology stores, ontology selection and mapping algorithms. It is likely that this style of component based programming will encouraged the development of standard interfaces for several types of generic SW tools (e.g., such as the DIG interface for reasoners).

6 Which techniques will be applicable?

This section links the technology that we expect to exploit and develop in Open Knowledge to the functionality of the system.

6.1 Interaction models

An interaction model is a set of conventions that regulate dialogues between peers. These conventions include atomic actions over knowledge components (through their WSDL-like descriptions), their combination into tasks, and the flow of tasks to dynamically accomplish objectives of the peers. Examples of atomic actions could be 'copying a document', 'searching for content within a document', or 'activating a service'. Examples of tasks could be 'classify a picture in a class' by combining atomic actions like 'computing a histogram' and 'classify the histogram', that may be performed by different peers. An example of flow could be, 'search for x-ray images of a certain type of patient', then 'classify them according to graphical features', and finally perform 'statistical analysis of the resulting classes'.

A language for interaction modelling must allow the description of:

– Conventions on norms, that is, the consequences of atomic actions of a plug-in over knowledge components (*e.g.* copying a picture might include commitments of paying a certain copyright) or of messages between plug-ins (*e.g.* agreeing on delivering a service might make a plug-in incur in a penalty if the service is not available by a deadline).
– Conventions on the task flow, that is, what joint (multiple peer) activities the plug-ins engage in and in which order.
– Conventions on how each task in the task flow is actually accomplished by the corresponding set of peers through a joint plan of action.

Peers thus need to be able to understand interaction models expressed in a formal language. These models drive the dialogues among peers to accomplish knowledge management tasks. Moreover, given the openness of our approach, peers must include in their basic protocol the capability of establishing dynamic agreements upon concrete interaction models, and the capability of talking about other pragmatic elements like trust, reputation or quality of service. We use as a vehicle for this a form of process calculus that provides a declarative specification of the required interaction: and interaction model. This has the following three advantages:

Describing interaction independently from services In OpenKnowledge the interaction is portable between services (rather than associated with individual services) so it must be described independently of specific services but, once it reaches a service, it must be possible for that service to connect precisely to it.

Separating state of service from state of interaction All interactions have state but it is impossible to synchronise the state of all services in an open, distributed system. The OpenKnowledge solution to this problem is to separate the state essential for an interaction to proceed from the overall state of the distributed system. Typically the specifics of state essential to an interaction are simple and compact, and therefore may be maintained as a form of "contract" between participating services.

Making interaction state portable and inspectable Key to OpenKnowledge is the idea that interactions themselves can be treated as objects to be analysed and transported between appropriately enabled services. This is what allows us to connect facilities (described later) of context maintenance, discovery, ontology matching, *etc*.

The essentials of our interaction modelling language, with a description of a mechanism for making it operational, are in [**?**].

6.2 Mapping

The matching problem within OK framework arises due to inherent heterogeneity of terms in the peers' service descriptions. Thus, one peer can have $expert\_finder$ as its service role while the others could have $person\_finder$, $expertICT\_finder$, $expert\_broker$. While the first and the fourth role denote services which are essentially equivalent, the second is more general then $expert\_finder$, while the third is less general. In fact these terms are used in the context of local, a priori defined, often left implicit ontological description of the world. The solution we propose is to construct semantic mappings (e.g., less and more general then, equivalent to and disjoint from) existing between the terms used during the interaction. One example of such mapping is $< person\_finder, expert\_finder, LG >$, stating that $expert\_finder$ is less general than $person\_finder$. These mappings are those defined and used in C-OWL [3]. We discover semantic mappings exploiting semantic matching approach [4, 5]. This is applied at least in three phases:

Role matching aligns the different ways in which role are described when initiating or joining interaction. An example is the mapping $< person\_finder, expert\_finder, LG >$.

Term matching aligns (structured) terms within the clause defining a role in order to undertake an interaction. An example of matching is $get\_address$, which in one peer can get 2 arguments (e.g., name of the person and his/her address) and in another three arguments, where the third argument (e.g., $Type\ of\ Comm$) could discriminate whether we need an address for personal or work communication.

Query/Answer matching takes place when running an interaction model and deals with the semantic heterogeneity arising from the statement of a query and in the values returned in its answers. For example, an interaction model specifying that the $address\_finder$ needs to look up the address for $Stephen\ Salter$ by invoking the

$get\_address(\text{"}Stephen\ Salter\text{"}, A)$ operation is not guaranteed to match perfectly to the operation the peer actually can perform. Perhaps the operation used by the peer is $find\_address$ and the surname is expected first $Salter,\ Stephen$.

6.3 Query relaxation

The query relaxation techniques are exploited within OK framework in IM and service discovery processes. Thus, the user has to obtain "good enough" IM while queering P2P network and the peers recruited for given IM should be "good enough" performers of their roles. The notion of "good enough answer" is defined in literature [6] as "an answer to a user query which serves its purpose given the amount of effort made in computing it". There are two key points in the definition:

1. A query answer should serve its purpose. Users submit queries in a specific context, giving the queries a specific purpose. The purpose of the query can be expressed as a vector (W) of answers quality features (e.g, relevance, precision, trust, etc.).
2. A query answer should be parametric on the initial effort. Users are likely to be willing to invest more resources with the purpose of getting higher quality answers. How much resources user should invest to receive answers with required quality W can be specified by vector of recourses H (e.g., effort, CPU time, network bandwidth, etc.).

Therefore, in the process of query relaxation the user trades answer quality parameters W on the amount of resources H he is willing to spend on the search process. The trade off between them is defined by query relaxation function F_H. F_H can be either defined analytically or learned from user interactions with the system.

6.4 Discovery

As already mentioned previously, in our system a there are two moments that a search is issued, namely first when a user types in a query the system needs to return a set of relevant components, second when the user selected the component that fulfils the need, it needs to find peers that can play the roles described in the components. Each default client in the open-knowledge will have a default search interaction model and a pointer to both a default Component Discovery service and a Peer Discovery service. For this moment we assume that both component descriptions and peer descriptions are a set of terms (i.e. keywords) which describe the functionality and expertise respectively. Therefore, discovery systems found in literature and the ones that we will develop in the first phase of the project will be identical, because in principle they both have the same functionality, i.e. matching term sets (queries) with term sets (component descriptions and peer descriptions). As we will discuss, we take the approach of combining DHT's with Semantic Overlay Networks (SONs). With the SON approach peers are going to know some peers not only by address but also by their expertise which means a 'semantic' description of the data. The word semantic means something like a human and machine readable description of some domain of interest. The big advantage of having knowledge about the content of other peers is that queries can be forwarded

to the best matching ones (instead of random), resulting in a more efficient forwarding process, i.e. less bandwidth usage. DHTs are currently seen as a good competitor with the SON approach. The generic idea of DHTs is that each item that is shared on the network is hashed to a unique hash-key. This key serves as message identifier, and the message is efficiently routed to the peer whose network identifier lies closest to the key. With efficient, we mean that only O(logN) messages are needed to route the message to that peer, where N is the number of peers in the network. This means that each peer is responsible for a key-space and therefore becomes a kind of yellow-page for content for this key-space. The items (and the appurtenant content) can also efficiently be retrieved if the requester knows the key. Current discovery systems either are not aware of term semantics, and therefore cannot support approximate query answering ([13][9][1][8])[12], or if they do [11][10][7], updating the (shared) structures that capture term semantics is expensive. Peers can gather descriptions using either a pull or a pushed based approach. Most systems use a push based protocol [10][7][8]: Peers advertise their data (or a summary of their data), and other peers decide to remember them or not. Alternatively, it could be more efficient for peers to pro-actively pursue data that will enhance their local knowledge. We are proposing a method where peers muster descriptions relevant to their own. Once calculating which are the most important terms for them, they pull descriptions with that terms using the DHT. For example, a peer responsible for RDF can easily detect that OWL and reasoning are important terms for it. Then, it can ask for descriptions with these terms from the peers responsible for them. Now, it will have a broader view on topics close to RDF. We can apply such a process recursively. Peers compute the most important terms in their local data, ask for more descriptions with that terms and repeat, therefore improving semantic locality.We expect that this technique will improve the performance of language processing techniques that are applied locally (eg LSI), in turn improving support for approximate answers. Furthermore, it should improve the efficiency of the discovery system, since now data are semantically instead of lexicographically clustered. In pRoute, we have seen that semantic clustering of peers can increase recall by an order of magnitude compared to a random overlay[10]. We expect that clustering data instead of peers will have a similar effect, since, in both cases, we are clustering sets of related terms. Besides the semantic relatedness measures to match queries with descriptions, also mapping techniques described in the previous subsection can be used by the discovery services. Therefore we also will have a component- and peer discovery service that combines the semantic similarity measures together with the mapping functionalities.

6.5 Security and Trust

The research on trust and reputation will be based on refining the existing model by Sierra and Debenham [**?**, **?**] **CARLES PLEASE GIVE THE BIBTEX** that uses principles of information theory to analyze the dialogues among agents. The information interchanged in dialogues is used to asses the probabilities of a number of predicates that are supporting the notion of trust, relationship building, or honour. Maximum entropy inference and ontology semantic distance are the basic tools that permit the best use of every single piece of information.

In this model, the basic trust predicate $T(\alpha, \beta, \delta)$ represents the trust agent α has on agent β on deals of type δ. Being the trust an *expectation* that α has on *deviations* of behaviour of agent β on deals of type δ.

From the perspective of peer-to-peer interactions the notion of trust is somehow conditioned by the interaction model being used. An agreement will be more or less easily broken, or deviations might be expected depending on *how* that agreement has been reached. In this way, interaction models that provide room for long discussions and explanations would perhaps lead to more solid contracts that interaction models that force reaching agreements in a less flexible way. From this initial thought, and taking into account the architecture being proposed in the project we will study the notion of trust with respect to two different entities:

- **Agents** Peers will make available agents (or services) to participate in interaction models, i.e. to incarnate roles and 'run interaction models. This process is call 'peer recruiting' and it is done with the help of the *peer registry* component of the peer architecture. These agents can perform well or not according to the expectations. As said before, they might have a different behaviour from the one expected, and therefore they are more or less trustworthy. Agents might be private, so you can only execute at the peer site, or they might be public and you can copy them, run them in your machine, and even make them available from your peer to the rest of the network. Trust will definitely influence in the decision of whether make an agent available to others or not. Trustworthy agents will probably spread in the network and untrustworthy one will probably die out.
- **Interaction models** When trying to solve a problem peers will need to find interaction models that drive the execution of services or the interaction among agents. The architecture deals with that using the discovery service. Interaction models will guarantee different properties of the interaction/dialogue between agents and therefore they might allow for different levels of fraud/wrong behaviour. Therefore it makes sense that interaction models have different levels of trust depending on what is the goal of the interaction.

In this way we are aiming at defining two trust measures: $T(\alpha, \beta, \delta)$ the trust of agent or peer α on β for interactions of type δ, and $T(\alpha, IM, \delta)$ the trust of agent or peer α on the interaction model IM for interactions of type δ.

Reputation, as the opinion of a group on some matter, will also be studied within the project along the same two dimension of trust: Reputation of agents and reputation of interaction models. We consider centralised models as not being appropriate given the P2P approach of the project and we'll aim at a distributed (i.e. social) reputation model.

6.6 media2Semantics

Automatic Media Analysis When a user instantiates an interaction model that uses some multimedia, it may be necessary for some automatic media analysis to take place, to avoid the user having to provide input into the system. Clearly, this is only going to be appropriate in certain situations where the current techniques in media analysis are able to provide solutions.

Let us consider a user who fits the visual composer use case. This user publishes holiday brochures that attract tourists to pretty locations. To help them create these brochures they are creating an interaction model that, when executed, will deliver a set of images of natural scenes from a given place.

At first they searched the network and used a Google Images component to retrieve images of the place they were interested in, e.g. 'England'. They found the results were poor, so they concatenated "natural" to the search (e.g. 'England natural'), and the results now contained the type of images they wanted, but also contains many others that were not relevant.

They searched on the network for components that would take an image (or set of images) and return whether it is a landscape-type of photo. They composed this component with the Google Images component to filter the results of the Google Images component to retrieve only those images that were of pretty landscapes of the place they chose.

Now, the component that takes each image and looks at it to decide whether it is a landscape-type of photo uses automatic analysis of the media content to achieve this; it is the only way to do this kind of analysis. Research in content-based image analysis has provided a means for achieving this specific kind of functionality, as well as other similar content-based techniques.

Detecting landscape-type photos can be achieved by various combinations of edge direction coherence analysis and colour analysis, for example. Face detectors can be built using forms of colour and shape filtering to increase the precision of Google Image results for 'Group' or 'Face'. There are many other content-based detectors available and, like those described above, are all specific to some domain of image or object. Other research being undertaken is taking a Web 2.0 approach to classifier generation, taking the community view of some concept from Flickr and generating classifiers using those photos as training data.

Such media analysis often returns some conceptual classification from some image, allowing various flow control selections, or filtering to take place within the interaction model. However, it is possible that the media analysis will return a vector of numbers that will require a separate component to provide some extra meaning to. This can also be readily achieved using OpenKnowledge, by simply sharing the relevant components.

Once our visual composer has created their interaction model, that takes the name of a place and returns pretty photographs from around that place, they may share their model, so that other users may also search for such pictures.

Automatic Media Annotation We see automatic media annotation as an important part of the OpenKnowledge system. Although in section 6.6 we have described a very specific use case, where a user takes a landscape-type photo detector and uses it within their composition of components (interaction model), such media classification will be very important in very specific cases, such as the bioinformatics scenario. Although the media analysis components that would be useful for this domain will have little or no use in other domains, by making those components reusable in an OpenKnowledge scenario, new services that inhabit that domain will automatically have use of those analysers.

As new techniques are developed for detecting and analysing media more generally, new interaction models can be built that provide ever-better functionality on general photos; perhaps more importantly, opening the classifiers to the network means that new and unforseen uses for them will become available as new interaction models (a Web 2.0 view of the world).

6.7 Semantic Web Tools

The Open University will explore the innovations provided by OpenKnowledge to enhance two tools aiming to support user tasks on the web: a question answering system (Aqualog) and a semantic web browser (Magpie).

Currently, both systems can use only one ontology at any time to support the tasks of query answering or semantic browsing. This limits particularly the coverage of the answers in Aqualog and the sensemaking support in Magpie. The reason for this limitation is that in order to support question answering or semantic browsing on the semantic web in large scale, important technical issues, which are key to the OpenKnowledge project, to do particularly with dynamic ontology mapping and the ability to reason about the quality of information coming from different sources, have to be solved.

In the enhanced versions of these systems we intend to extract knowledge coming from multiple ontologies, in principle, from any ontology available on the web. As result, we expect Aqualog to become a general-purpose ontology-based question answering system on the web, and Magpie to provide semantic support for general-purpose web navigation and sense-making. Therefore, we will rely mainly on the dynamic ontology mapping method to be provided by OpenKnowledge.

As a consequence, this will allow us to investigate whether the OpenKnowledge technologies can be integrated with existing technologies and thus enable the development of new, more powerful and flexible user-centered semantic web tools.

7 How will you evaluate your system?

In this section, we tell at least how we are going to evaluate the requirements given in section two, and besides this each partner describes how they evaluate the usefulness or other characteristics of their technology that they bring in (given in the previous section).

7.1 Interaction models

Two empirical questions are crucial for peer to peer use of interaction models: whether our mechanisms scale to systems containing millions of components and whether they can be deployed in a manner that promotes the development of communities of practise. Although these two questions interact, the experimental methods appropriate to answering them are different. In the former case we require series of laboratory experiments using a test rig that controls the environment in which we deploy our methods - the aim being to produce results that hold generally for systems that preserve our experimental assumptions. In the latter case we support people in use of the OpenKnowledge tools -

the aim being to demonstrate by example how nuclei of communities of practise can be fostered in specific domains of application.

Scaling experiments : The reason a peer to peer architecture is attractive is that it offers the prospect of obtaining satisfactory performance via opportunistic interactions between peers at run time, rather than structuring their interaction at design time. The broad experimental issue is whether satisfactory performance in automated knowledge sharing can be achieved by this means. The more specific hypothesis to be tested is that a combination of our mechanisms for interaction model interpretation, ontology matching and query routing exists such that satisfactory performance may be predicted to an acceptable level of accuracy from any starting state of knowledge in constituent components, with performance reaching levels remaining constant as number of components increases. We look for satisfactory rather than perfect or optimal performance because we recognise that the price paid for the scaling benefits of a peer to peer architecture is a reduction in overall quality of answer and predictability of outcome.

Development of communities of practise : To be viable in practise a system must support a community of practise - a group of like-minded people who develop a vested interest in maintaining the system. A peer to peer system that experimentally confirms the scaling hypothesis above may not develop a community of practise. Conversely, a system that disconfirms the scaling hypothesis in general may in practise scale large enough to support substantial communities of practise. Our hypothesis is that an engineering process exists for initiating and maintaining peer to peer networks such that stable communities of practise are established. Since a community of practise is always domain specific it is impossible to confirm this hypothesis in general - the best we can do is confirm it on specific examples so our experimental method is case study.

7.2 Mapping

The quality of matching can be evaluated either (i) directly (i.e., by exploiting the golden standard of the mappings holding between terms and role descriptions) or (ii) indirectly (i.e., by observing the user satisfaction measures when all the other parameters of the system (e.g., number of peers in the network, routing algorithm results, etc) are fixed).

Considerable human effort is necessary in order to construct the golden standard in (i). However, these efforts can be partly automated (see [2] for more details). The results of evaluation on the golden standard further can be reused in computation of query relaxation function F_H. Moreover they will allow to compute the widely used matching quality measures of Precision and Recall.

An indirect evaluation (ii) requires less human effort since the user satisfaction measures can be computed in either explicit or implicit way in the process of user interaction with the system. However matching quality is only one of the factors which may influence on the quality of results presented to the user. The number of peers in the network and quality of semantic routing algorithm results are among the others. Therefore the non trivial task in this case is to distinguish among the influence of various factors on the quality of answer.

7.3 Query relaxation

The quality of the query relaxation techniques can be evaluated by comparison the observed user satisfaction measures in the cases when the query relaxation techniques are exploited and omitted respectively.*PLEASE EXTEND THIS PART*

7.4 Discovery

We will evaluate the discovery algorithms via simulations and by doing field experiments with the 'real' system which is discussed in this paper. We evaluate the influence of different instances of algorithms and parameters like the maximum number of advertisements and query hops in the network via indicating user satisfaction (measured as component and peer description recall), system efficiency (number of messages sent) and robustness.

7.5 Security and Trust

We plan to empirically verify the trust and reputation models in the two case studies of the project. The two case studies are rich enough to explore the possibilities that a trust model can offer. In particular:

- Emergency case study. It is easy to imagine the need for trust on (e.g.) fire brigades and other roles in order to organise and co-ordinate rescue teams. Emergency protocols/co-ordination protocols can give different results, and therefore trust on interaction models makes sense in this case study.
- bioinformatics case study. In proteomics, trust on peers, either laboratories or data bases, seems important. Reliability of information is the essential aspect here, so trust in the previous information-based sense is central. Trust on the interaction models to get the right information is also important (e.g. what processes are applied to the samples or what types of agents (roles) where used in the interaction model)

7.6 media2Semantics

The only way to evaluate the inclusion of such an abstract part of the system is empirically, by building interaction models using classifiers that have been wrapped into OK Components and attempting to utilise them in some interaction models to show their usefulness. We can show how they can be useful in specific cases by leveraging the scenarios to provide use cases for the techniques suggested. *PLEASE EXTEND THIS PART*

7.7 Semantic Web tools

One of the goals of the Open Knowledge project is to adopt technological advances from the project in order to enhance its existing Semantic Web tools and to test the improvement brought by the new technology with respect to traditional techniques. In particular we will evaluate the following aspects:

Answer coverage for PowerAqua. As PowerAqua will have access to an increased number of ontologies and their associated metadata thanks to the dynamic ontology mapping mechanism, we expect that the number of queries that it will be able to answer will also increase. Therefore, we will test the number of queries answered by AquaLog, the traditional version of PowerAqua, and PowerAqua.

Answer quality for PowerAqua. Access to more metadata should allow this tool to correctly answer a larger amount of queries. Here we are not only interested in the number of questions that it is capable to handle (as before) but in the number of questions that it can *answer correctly*.

Extended scope for the Semantic Browser. Our hypothesis is that the Semantic Browser will be able to provide semantic markup for Web pages displaying information from a large variety of domains instead of being limited to provide semantic markup for a single topic domain only as it is currently. We wish to evaluate this extended scope of the tool in terms of number of Web pages that it can handle and the richness of semantic annotation that it can find and display.

Time performance. Both tools will rely OpenKnowledge technology at run-time while being employed by users. Therefore, it is important that they provide a good time performance. Obviously, this requirement is harder to achieve when they rely on automatically selecting and using ontologies than in the case when they were using a single ontology.

Usability is an important requirement for our two tools and it will also be evaluated. In this case we will rely on extended expertise in KMi to conduct user evaluation experiments.

Acknowledgements: This work has been supported by the FP6 OpenKnowledge project[2].

References

1. JXTA project. see http://www.jxta.org.
2. P. Avesani, F. Giunchiglia, and M. Yatskevich. A large scale taxonomy mapping evaluation. In *Proceedings of International Semantic Web Conference (ISWC)*, 2005.
3. Paolo Bouquet, Fausto Giunchiglia, Frank van Harmelen, Luciano Serafini1, and Heiner Stuckenschmidt. Contextualizing ontologies. *Journal of Web Semantics*, 1(4):24, 2004.
4. F. Giunchiglia and P. Shvaiko. Semantic matching. *The Knowledge Engineering Review Journal*, (18(3)):265–280, 2003.
5. F. Giunchiglia, P. Shvaiko, and M. Yatskevich. S-Match: an algorithm and an implementation of semantic matching. In *Proceedings of 1st european semantic web symposium (ESWS'04)*.
6. Fausto Giunchiglia and Ilya Zaihrayeu. Making peer databases interact - a vision for an architecture supporting data coordination. In *CIA '02: Proceedings of the 6th International Workshop on Cooperative Information Agents VI*, pages 18–35, London, UK, 2002. Springer-Verlag.
7. Peter Haase, Jeen Broekstra, Marc Ehrig, Maarten Menken, Peter Mika, Mariusz Olko, Michal Plechawski, Pawel Pyszlak, Björn Schnizler, Ronny Siebes, Steffen Staab, and Christoph Tempich. Bibster - a semantics-based bibliographic peer-to-peer system. In Sheila A. McIlraith, Dimitris Plexousakis, and Frank van Harmelen, editors, *International*

[2] http://www.openk.org/

Semantic Web Conference, volume 3298 of *Lecture Notes in Computer Science*, pages 122–136. Springer, 2004.
8. N. Leibowitz, M. Ripeanu, and A. Wierzbicki. Deconstructing the kazaa network. 3rd IEEE Workshop on Internet Applications (WIAPP'03). Santa Clara, CA., 2003.
9. A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, Heidelberg, Germany, November 2001.
10. R. Siebes and S. Kotoulas. proute: Expertise-based selection using shared term similarity matrices. In K. Verbeeck, K. Tuyls, A. Nowé, B. Manderick, and B. Kuijpers, editors, *Proceedings of the 17th Belgian-Dutch Conference on Artificial Intelligence*, pages 202–208, Brussels, Belgium, October 2005. Contactforum.
11. Chunqiang Tang, Zhichen Xu, and Mallik Mahalingam. pSearch: Information retrieval in structured overlays. In *ACM HotNets-I*, October 2002.
12. Bernard Traversat, Mohamed Abdelaziz, and Eric Pouyoul. A Loosely-Consistent DHT Rendezvous Walker. Technical report, Sun Microsystems, Inc, March 2003.
13. Universal Description, Discovery and Integration of Business for the Web.