

## D3.4 Test Scenario

---

**Marc Ehrig, Peter Haase, Steffen Staab, Christoph Tempich**  
**(University of Karlsruhe)**

**with contributions from:**

**Michal Plechawski (Empolis Polska), Jeen Broekstra, Ronny Siebes, Heiner**  
**Stuckenschmidt (VU Amsterdam)**

**Executive Summary.**

SWAP EU IST-2001-34103 Project Deliverable D3.4 (WP3.2)  
Report on test scenarios for evaluating methods.

**Document Id.** SWAP/2003/D3.4/v1.0  
**Project** SWAP EU IST-2001-34103  
**Date** September 12th, 2003  
**Distribution** Restricted

---

## SWAP Consortium

This document is part of a research project partially funded by the IST Programme of the Commission of the European Communities as project number IST-2001-34103. The partners in this project are: Institute AIFB / University of Karlsruhe (coordinator, Germany), Vrije Universiteit Amsterdam VUA (Netherlands), Meta4 (Spain), empolis UK Ltd. (UK), empolis Polska (Poland), Dresdner Bank AG (Germany), and IBIT (Spain).

### **University of Karlsruhe**

Institute AIFB  
Englerstr. 28  
D-76128 Karlsruhe  
Germany  
Tel: +49 721 608 3923, Fax: +49 721 608 6580  
Contactperson: Steffen Staab  
E-mail: staab@aifb.uni-karlsruhe.de

### **Meta4 Spain S.A.**

Rozabella, 8 Centro Europa Empresarial  
28230 Las Rozas (Madrid)  
Spain  
Tel: +34 91 634 85 00, Fax: +34 91 634 86 86  
Contactperson: Adelma Stolbus  
E-mail: adelmas@meta4.com

### **empolis Polska Sp. z o. o.**

Plocka 5a  
01-231 Warsaw  
Poland  
Tel: +48 22 535 88 06, Fax: +48 22 535 88 14  
Contactperson: Mariusz Olko  
E-mail: mariusz.olko@empolis.pl

### **Fundación IBIT**

Reverend Francesc Sitjar 1  
07010 Palma de Mallorca  
Spain  
Tel: +34 971 177 271, Fax: +34 971 177 279  
Contactperson: Esteve Lladó Martí  
E-mail: esteve@ibit.org

### **Vrije Universiteit Amsterdam (VUA)**

Division of Mathematics and Informatics W&I  
De Boelelaan 1081a  
1081 HV Amsterdam  
The Netherlands  
Tel: +31 20 4447786, Fax: +31 20 4447653  
Contactperson: Hans Akkermans  
E-mail: hansakkermans@cs.vu.nl

### **empolis UK ltd.**

Unit B, The Dorcan Complex, Faraday Road  
SN3 5HQ Swindon  
United Kingdom  
Tel: +44 1793 485465, Fax: +44 1793 485451  
Contactperson: John Richards  
E-mail: jer@empolis.co.uk

### **Dresdner Bank AG**

Jürgen-Ponto-Platz 1  
60301 Frankfurt am Main  
Germany  
Tel: +49 69 263 58265, Fax: +49 69 263 81385  
Contactperson: Martin Lorenz  
E-mail: martin.lorenz@dresdner-bank.com

---

# Changes

Version	Date	Author	Changes
0.1	13.05.03	meh	first outline
0.2	21.07.03	cte	test metadata integrator
0.3	14.08.03	meh	adapter integration, merger
0.4	20.08.03	cte	metadata integrator final version
0.5	27.08.03	meh	local repository included
0.6	02.09.03	meh, pha	informer included, corrections, introduction
0.7	06.09.03	cte	implications for SWAP
1.0	12.09.03	cte	final release

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Local Node Repository</b>	<b>3</b>
2.1	Storage . . . . .	3
<b>3</b>	<b>Knowledge Source Integration</b>	<b>7</b>
3.1	Syntactic Extraction . . . . .	7
3.2	Semantic Integration . . . . .	10
3.3	Ontology Integration . . . . .	12
3.4	Metadata Integration . . . . .	14
3.5	Ontology Merging . . . . .	28
<b>4</b>	<b>Informer</b>	<b>37</b>
4.1	Advertisements, Discovery and Peer Selection . . . . .	37
<b>5</b>	<b>User Interface</b>	<b>46</b>
5.1	Visualization Requirements . . . . .	46
<b>6</b>	<b>Adapters</b>	<b>49</b>
6.1	Communication Adapter . . . . .	49
<b>7</b>	<b>Conclusion</b>	<b>56</b>

# Chapter 1

## Introduction

In the previous deliverables of Workpackage 3 “Methods” a lot of research and implementation work has been done. The next step is to enable thoroughly testing and evaluation of the developed methods. The main goal of this deliverable is to fix the evaluation baselines. If already available, the results of the test runs are also presented. Please note that we are not testing the whole SWAP system, but try to evaluate smaller units - the methods for each single component. Final testing of the complete system will be performed through the enterprise case studies.

Just as in deliverables 3.2 and 3.3 the chapters are grouped along functional entities. Each single section represents one component. They are aligned to the following structure:

- test scenario
- used data set
- evaluation criteria and measures
- evaluation process
- different method implementations
- test results (if already available)
- conclusion

Specific requirements of some components lead to slight changes in this structure.

Most of the sections represent a snapshot of ongoing work. Methods are continuously refined and evaluated. Nevertheless the results of the current evaluations give interesting insights as one can see on the next pages.

# Chapter 2

## Local Node Repository

### 2.1 Storage

#### 2.1.1 Requirements

Sesame was brought in to the SWAP project as an existing tool, and has therefore undergone rigorous testing already. However, since the SWAP projects has specific demands and requires additional features, testing these features becomes important.

The SWAP-specific requirements include:

- **Performance** - due to the distributed nature of SWAP, query performance on medium-sized data sets is a main concern.
- **Conformance** - SWAP is built on open standards, like RDF. Conformance to all aspects of the RDF specifications is essential for enabling interaction with other tools.

#### 2.1.2 Performance

For testing Sesame's performance, we have used our own benchmark tool for testing SAIL performance, which can be configured to use different data sets and measure performance of all SAIL operations. We present results of the RDBMS-based repository and the inferencing in-memory repository.

The results presented here are obtained using a dataset generated by the OntoScrape tool. The total test set contains approximately 56,000 statements, which is rather small but gives a fair indication of performance in the average SWAP scenario.

The tests were carried out on a Pentium 3 desktop PC with 256MB RAM.

### RDBMS performance

Performance results on the RDBMS-based repository (with full RDF-MT inferencing are shown in table 2.1 and table 2.2.

total upload time (56,115 triples)	420 s
------------------------------------	-------

Table 2.1: total upload time for RDBMS inferencing repository

method	average/call (ms)
addStatement()	7.5
hasStatement()	20
getStatements()	250
isClass()	2.5
isSubClassOf()	2.9
isProperty()	2.1
isSubPropertyOf()	2.6
getClasses()	96
getProperties()	13

Table 2.2: performance of SAIL methods for RDBMS inferencing repository

### In-memory performance

Performance results on the in-memory repository (with full RDF-MT inferencing are shown in table 2.3 and table 2.4.

total upload time (56,115 triples)	36 s
------------------------------------	------

Table 2.3: total upload time for in-memory inferencing repository

### 2.1.3 Conformance

For testing Sesame's conformance to the RDF specification, several unit tests were implemented to check Sesame's handling of W3C-approved test cases<sup>1</sup> for parsing and for entailment.

<sup>1</sup>See <http://www.w3.org/TR/2003/WD-rdf-testcases-20030123/>

method	average/call (ms)
addStatement()	0.63
hasStatement()	0.09
getStatements()	2.9
isClass()	0.1
isSubClassOf()	0.73
isProperty()	0.01
isSubPropertyOf()	0.2
getClasses()	0.7
getProperties()	0.01

Table 2.4: performance of SAIL methods for in-memory inferencing repository

## Parsing

The Sesame project has developed a suite of parsers and writers for RDF-related document "standards". Currently, it contains parsers for XML-encoded RDF (i.e. the official RDF format, RDF/XML for short) and N-Triples. Writers exist for XML-encoded RDF, N-Triples and Notation 3 (a.k.a. N3).

The parsers and writers for RDF/XML and N-Triples adhere to the specs that were released in January 2003, which can be found at <http://www.w3.org/RDF/>. There is no real specification for Notation 3/N3, but the output generated by the N3 writer should be compatible with other N3 tools.

We have tested the RDF/XML parser thoroughly with the approved test cases that are available from the W3C website (<http://www.w3.org/TR/rdf-testcases/>). The positive test cases from this site are test cases with correct RDF that should generate specific sets of RDF statements. The RDF/XML parser has been tested successfully with all of these.

The negative test cases from this site are test cases which contain incorrect RDF that should result in an error reported by the RDF/XML parser. The RDF/XML parser has been tested successfully with all-but-three of these test cases. The failed test cases concern the character encoding of literals and URIs. According to the specs, URIs and literals should be in Normal Form C<sup>2</sup>, but our parser does not check for this.

## Entailment

Several test cases for entailment were specified, which can be found in the CVS repository of Sesame. These entailment tests deal with the semantics of for example subsumption relations. With the exception that datatype entailment, a feature that is currently not

<sup>2</sup>See <http://www.unicode.org/unicode/reports/tr15/>

supported by Sesame, the inferencer in Sesame is conformant to the Last Call Working Draft of the RDF Model Theory<sup>3</sup>.

---

<sup>3</sup>See <http://www.w3.org/TR/2003/WD-rdf-mt-20030123/>

# Chapter 3

## Knowledge Source Integration

The Knowledge Source Integrator is responsible for the integration of knowledge from local and remote knowledge sources into the Local Node Repository. This process includes the syntactic and semantic extraction of local knowledge sources, the selection of knowledge to be included, the merging of knowledge from different sources and the annotation of the knowledge according to the metadata model. The test scenarios for these methods will be described in the following.

### 3.1 Syntactic Extraction

During the syntactic extraction, an RDF description of the data in the local knowledge sources is generated. The extracted description is a direct representation of the content contained in the knowledge sources.

#### 3.1.1 Scenario

In this scenario, a user in a P2P environment wants to enable a subset of the information in its local knowledge sources for sharing. This process involves the selection of the subset by the user and the automatic extraction of the RDF representation by the extraction tool. The extracted information will then be the input for the subsequent steps of the knowledge source integration process.

#### 3.1.2 Data Set

For the evaluation, we will use two different data sets: As the first data set we will use the information in the file system and mail folders of a typical user of the SWAP system. Secondly, we will use data stored in a database. The database contains bibliographic

metadata from the computer science domain (DBLP) as well as the ACM topic hierarchy.

### 3.1.3 Evaluation Criteria

First of all, the RDF data extracted from the knowledge sources needs to be correct, i.e. it must correspond with the data that was selected for extraction. The data set needs to be complete, but more importantly, subsets that were explicitly excluded from the extraction process (e.g. for security reasons) must not be included in the extracted data.

Another important criteria is performance: It needs to be determined whether the time to extract a large knowledge source is acceptable. Similarly, the scalability of the extraction process needs to be determined.

Also, the usability of the extraction method needs to be evaluated.

### 3.1.4 Evaluation Process

The evaluation will be performed on a single peer with four extensive knowledge sources: a file system, Outlook mail folders, Internet Explorer bookmarks and a relational database. The entire knowledge sources will be extracted. The times to perform the extractions will be measured. If the extraction process does not complete successfully, it will be repeated with smaller data sets until the limitations with respect to the size of the data sources are found.

In another step, a user specified subset of the data sets will be extracted to check whether the extracted information corresponds with the specified subset.

The tests are run on a P3 800Mhz machine with 256MB of RAM.

### 3.1.5 Methods

For the extraction from the different knowledge sources, two methods are used, as described in the following.

#### **OntoScrape**

For the extraction for the first three knowledge sources (File system, Outlook, Bookmarks), the OntoScrape tool will be used. The tool allows to navigate the hierarchical structure of the knowledge sources and to select sets of folders to be extracted.

### Java Database Application

The extraction from the relational database is done using a simple Java application. During the extraction process, this data will be joined using SQL queries and tagged.

#### 3.1.6 Results

First of all it is to state that the extracted syntactic representations of the knowledge sources were correct in all cases. The results regarding the quantitative performance measures and scalability however are far more differentiated. The results for the extraction using the OntoScrape tool are shown in table 3.1 The limitations of the extraction process with OntoScrape were reached during the extraction of a complete file system with 94797 files. After one hour, around 13% of the complete file system had been extracted. This shows, that this method is not feasible for the extraction of complete file systems. In a more typical situation, where a user wants to extract a fairly big folder (1400 files and directories), the time of 7 minutes for the extraction is at the limit of acceptability. The amount of data generated is in a manageable order of magnitude. The results for the

Content	# of Instances	# of Statements	Result file size (kB)	Time (s)
<b>Filesystem</b> Complete	12673	160951	13519	3600 <sup>1</sup>
Selected Folder (big)	1400	18176	1306	420
Selected Folder (small)	132	1668	130	30
<b>Outlook</b> Complete	2224	15320	1929	220
Selected Folder	306	1977	262	30
<b>Bookmarks</b> Complete	32	247	26	< 1
Selected Folder	6	45	5	<< 1

Table 3.1: Results for extraction using OntoScrape

extraction of the Outlook and Bookmark knowledge sources are more encouraging. For these cases, the extraction of the complete sources is manageable.

Table 3.2 shows the results for the extraction from a database using the Java database application. Obviously, the results are very specific for this data set, especially as most the resources are required for database operations, not for the transformation to the RDF representation. However, the data set and database operations can be considered typical for an extraction scenario. The resources required for the extraction are acceptable, especially considering that the extraction typically is a one time effort. For a repeating extraction, new methods, e.g. based on incremental extraction, would be useful. Regarding the usability it needs to be said that the extraction application needs to be adapted for new data sets, as there currently exists no appropriate generic tool for this task.

Content	# of Instances	# of Statements	Result file size (kB)	Time (s)
<b>Database</b>	126247	378741	37200	1020

Table 3.2: Results for extraction using database application

### 3.1.7 Conclusion

In summary, the methods developed for the syntactic extraction have proven usable and acceptable in performance. However, the limitations of the OntoScrape tool were reached when extracting complete file systems. For typical scenarios, in which users only enable parts of the knowledge sources for sharing, these limitations will not be a major drawback. An incremental extraction to allow would be a “Nice-To-Have” for a possible future version of the OntoScrape tool. For the bibliographic scenario, the extraction of RDF representations from bibtex sources will be an interesting method to evaluate.

## 3.2 Semantic Integration

Whereas the Syntactic Integration only creates RDF representations out of another format, the Semantic Integration adds additional semantic information. This information is normally contained in the data only implicitly. The present step tries to grasp this implicit information and add it to the repository with explicit statements. Unfortunately this information extraction technique can create mistakes.

### 3.2.1 Scenario

In the test scenario the user provides an RDF representation of his structure. Along with this data he can add information of the kind of data e.g. “email” or “file-folders”. After having the data processed, the user is returned a semantically enriched RDF representation with which he can continue using.

### 3.2.2 Data Set

The data set used for evaluating has been generated by the OntoScrape tool. It describes files, folders, emails, and bookmarks on a computer system. A second data set can be extracted from the DBLP database.

### 3.2.3 Evaluation Criteria

Two major evaluation criteria can be thought of:

**implicit knowledge** The amount of implicit semantical knowledge which is eventually modelled and included. This will be evaluated against the actually included implicit knowledge. As this last number is difficult to determine some imprecision occurs with this measure.

**mistake rate** How many wrong semantical statements are within the whole number of statements added?

### 3.2.4 Evaluation Process

The evaluation process is quite straight forward. A method and a data set are chosen. After the process a human has to manually decide on the values for the evaluation criteria.

### 3.2.5 Methods

#### Folders to Concepts

The basic method assumes that the folder hierarchies in either the file or email system also represent concept hierarchies. This is due to the fact that users try to create a hierarchical order when creating such tree structures. The implementation is done by setting folders to concepts and adding the corresponding subclass\_of relations. The new statements are then represented in RDF.

### 3.2.6 Results

The results were difficult to evaluate. Especially to determine the “rate of implicit knowledge” requires expertise of the user about the domain. These experts then provide the corresponding figure. The second measure “mistake rate” can normally be identified more easily. Nevertheless we came up with the following results:

**implicit knowledge** Folders normally represent a hierarchy. No more information is included in this feature. Therefore the rate of identified implicit knowledge is *high*.

**mistake rate** Unfortunately the folders aren’t in a specific hierarchical order. They can represent anything from a *is\_a* hierarchy to a *part\_of* representation. This leads to some inconsistencies. The mistake rate achieves a *medium* level.

### 3.2.7 Conclusion

The tested method seems reasonable for the purpose of SWAP. Semantical knowledge can be easily added. As the hierarchy was created by users themselves, they normally don’t

mind having smaller mistakes in it. Therefore the presented method will be further used, even though some semantical inconsistencies might arise due to the errors in the derived implicit knowledge.

### 3.3 Ontology Integration

In a distributed peer-to-peer system information constantly enters the local peer. One task is to take decisions among the information what to integrate in the local representation and what to “forget”. Only a fraction of the actual knowledge is important for either user or the system (especially for routing purposes). The ontology integrator will be used to fulfill this task.

#### 3.3.1 Scenario

The test scenario is the SWAP system itself. Evaluating Ontology Integration as a stand-alone application or method is difficult and will most probably not return any meaningful results. Users will choose information which will be included into the local repository of the peer. Additionally information is exchanged between the different peers. All this information is integrated. The user can then formulate queries and wait for results.

#### 3.3.2 Data Set

The data set for testing will mainly consist of the extracted file structures. Additionally some ontology structures might be included. This gives the system a more semantical focus, which will hopefully have positive effects on the overall performance.

#### 3.3.3 Evaluation Criteria

Several evaluation criteria have been identified.

**User Satisfaction** The SWAP system is used to exchange queries and find answers. Only the user can determine if the results a method provides are correct.

**Memory Load** A big server can save more information than a small desktop. Memory load can be a critical factor in evaluating the methods.

**Traffic Load** Depending on the saved information on the peers, queries can be routed more or less focused. Network traffic will therefore vary based on the used method.

Unfortunately these measures don't only measure integration, but are influenced by a variety of other factors and methods of the SWAP system. Additionally results can vary if different users apply different methods. For example, using only methods with high memory loads or high traffic loads will probably result in a less positive evaluation, than a combination of some high memory peers (i.e. super-peers) with many small peers.

### **3.3.4 Evaluation Process**

Users simply run the SWAP system. They choose information from the SWAP system, and send queries. The results are monitored and retrieved. Users have to provide feedback on their satisfaction with the SWAP system.

### **3.3.5 Methods**

The most promising methods will be briefly explained.

#### **Full Integration**

An easy algorithm for integration is to add every piece of information passing by into the local node repository.

#### **No Integration**

Alternatively the peer doesn't save any information and performs blind queries to the system. No semantic intelligence as inferencing can be used.

#### **Random Integration**

By integrating and ignoring information at random the peer gets a wide though incomplete view of the existing knowledge in the peer-to-peer network. In general the cross linking gets very high with this approach as presented by Stanley Milgram in his experiments [?].

#### **Integration of Known Data**

Only data of which the peer knows at least something will be added to the knowledge base. I.e. if the peer knows subject, predicate, or object of a statement the statement will be included.

### 3.3.6 Results

As mentioned before testing only works using the whole SWAP system. Therefore the real world tests could not be performed at this time. The following table shows the expected results, which have to be verified at a later point in the project.

Method	User Satisfaction	Memory Load	Traffic Load
All	high	high	low
None	low	zero	high
Random	unknown	medium	low
Known Data	unknown	medium	medium

### 3.3.7 Conclusion

Due to the still missing results a final conclusion can not be given. Still, the test scenarios and measures are now well defined and will be used for the actual test. Integrating known data is expected to perform best. To reduce traffic load an element of random integration will further be added. With these methods one can expect to have a SWAP system behaving as intended and required by the case studies.

## 3.4 Metadata Integration

The metadata integration is performed by the annotator. Metadata integration is the process of adding meta information to incoming statements according to the SWAP metadata model, thus annotating the incoming statements. The metadata model defines a clear semantic to describe the location of external resources and other metadata to handle the ambiguity and decentralization which is inherent in peer-to-peer systems. As in the previous sections we provide in the following a small scenario for the metadata integration process, define the evaluation criteria and present the results of the evaluation.

### 3.4.1 Scenario

The metadata integration is a preprocess within the knowledge integration component. Thus, there is no real world scenario in which the annotator could be used alone. However, we describe the scenarios where it is used and derive from them the evaluation criteria.

The metadata integrator is always involved when it comes to the integration of new knowledge into the local node repository or when existing knowledge is updated. Knowledge is represented in statements and can originate from external applications, other peers or the user herself. In the following we will ignore the last case, because as a manual process it will not generate large amounts of data in short time periods as the other two cases.

The metadata represents technical information. Hence, for the user it is of importance that the information is integrated fast without bothering her too much.

This translates into the following evaluation criteria

- the speed of integration,
- the use of storage resources
- the rate at which information can be retrieved.

### 3.4.2 Data Set

As with other components, we use the DBLP data set to evaluate our methods. The DBLP data set includes roughly 420.000 statements describing 126.000 instances. Each instance is defined by an average of three statements indicating its position in the ontology, the title and its type.

To evaluate the component we assume that a number of peers all share the same knowledge and that one peer integrates the knowledge of all other peers. This is not the case in a real application. However, by deleting information one can achieve a more realistic case, hence the evaluation gives an upper limit for the evaluation criteria.

We expect, that the SWAP system is used predominately, to exchange information about the file and personal communication system. Thus we compare the DBLP data set with the information scraped from the knowledge sources of one of the project participants. The filesystem representation consists of 72.000 statements describing 9.300 instances.

### 3.4.3 Evaluation Criteria

The metadata integration is a service to other components. Hence, the time it consumes for its processes is of importance.

The above described evaluation criteria can be formalized as follows.

We define

$$t_{addition\ type}^{mode} = \text{time}$$

$$S_{statement\ type} = \text{statement}$$

$$|S| = \text{number of statements}$$

$SO$  = swabbi-object

$|SO|$  = number of swabbi-objects

$SP$  = swabbi peer-object

$|SP|$  = number of swabbi peer-objects

$M_{t;Swabbi\ store\ mode}^{mode}$  = Time consumed by the Metadata Integrator or Swabbi-object factory

$R_{t;Swabbi\ store\ mode}^{mode;store\ type}$  = Time consumed by the local node repository

$statement\ type = \begin{cases} C & \text{Content} \\ SO & \text{swabbi-object} \\ T & \text{Total} \end{cases}$

$store\ type = \begin{cases} mem & \text{LNR stores data in memory} \\ database & \text{LNR stores data in database} \end{cases}$

$mode = \begin{cases} store & \text{swabbi-objects are stored} \\ retrieve & \text{swabbi-objects are retrieved} \end{cases}$

$addition\ type = \begin{cases} update & \text{statements are updated} \\ first & \text{statements are added the first time} \end{cases}$

$Swabbi\ store\ mode = \begin{cases} swabbi & \text{local node repository stores SO} \\ no\ swabbi & \text{local node repository does not store SO} \end{cases}$

We have identified six cases to evaluate.

1. The time consumed by the metadata integrator if statements are added the first time.

$$t_{first}^{store}(|S_C|) = M_{t;first;swabbi}^{store}(|S|) + R_{t;first;swabbi}^{store;\{mem;database\}}(|S|) \quad (3.1)$$

The number of Swabbi-objects in dependency of the number of content statements and knowledge providing peers.

$$|SO|(|S_C|; |SP|) = |R_{swabbi}(SO)| \quad (3.2)$$

$$|SO|(|SP|; |S_C|) = |R_{swabbi}(SO)| \quad (3.3)$$

The number of all statements in dependency of the number of content statements and knowledge providing peers.

$$|S|(|S_C|; |SP|) = |R_{swabbi}(S)| \quad (3.4)$$

2. The time consumed by the metadata integrator if statements are added and updated.

$$t_{update}^{store}(|S_C|) = M_{t;update;swabbi}^{store}(|S|) + R_{t;update;swabbi}^{store;\{mem;database\}}(|S|) \quad (3.5)$$

3. The time consumed by the swabbi-object factory to retrieve one swabbi-object given a particular resource.

$$t_{update}^{retrieve}(|S_C|; |SO| = 1) = M_{t;update;swabbi}^{retrieve}(|S|; |SO| = 1) + R_{t;update;swabbi}^{retrieve;\{mem;database\}}(|S|; |SO| = 1) \quad (3.6)$$

4. The time consumed by the swabbi-object factory to retrieve the peer-objects given a particular resource.

$$t_{update}^{retrieve}(|S_C|; |SP| = 1) = M_{t;update;swabbi}^{retrieve}(|S|; |SP| = 1) + R_{t;update;swabbi}^{retrieve;\{mem;database\}}(|S|; |SP| = 1) \quad (3.7)$$

5. The time consumed by the swabbi-object factory to retrieve all peer-objects.

$$t_{update}^{retrieve}(|S_C|; |SP| = n) = M_{t;update;swabbi}^{retrieve}(|S|; |SP| = n) + R_{t;update;swabbi}^{retrieve;\{mem;database\}}(|S|; |SP| = n) \quad (3.8)$$

6. The time consumed by the local node repository to retrieve a statement with and without swabbi-objects included in the local node repository.

$$t_{update}^{retrieve}(|S_C|) = M_{t;update;swabbi}^{retrieve}(|S|) + R_{t;update;swabbi}^{retrieve;\{mem;database\}}(|S|)$$

$$t_{update}^{retrieve}(|S_C|) = M_{t;update;no\ swabbi}^{retrieve}(|S|) + R_{t;update;no\ swabbi}^{retrieve;\{mem;database\}}(|S|) \quad (3.9)$$

### 3.4.4 Evaluation Process

In this section we describe the different assumptions we made evaluating the above defined six cases. The different equations are evaluated with different numbers of peers and statements. Furthermore, not all cases have been evaluated with the entire data set or with the two implementations of the local node repository. However, we provide comparisons, thus the results are transferable to all cases. Table 3.3 gives an overview of the different test cases which are in the following described.

**Case 1** We assume that the repository is empty in the beginning and all statements are added the first time. This is a baseline for all following cases. The test was performed for the in-memory version of the repository. We looked at the consumed time when we include all available DBLP data in the repository. We compare the time consumed by the component to generate the additional statements with the time it takes to include all new<sup>2</sup> statements into the repository. The test was not performed with the real data set and the database, because it would have been very similar to the second case (see 3.4.5).

To get an impression how many swabbi-objects are generated in comparison to the number of content statements (eq: 3.4;3.3) we added a smaller amount of statements to the repository and increased the number of peers which know about that knowledge. In this case we store one swabbi-object for each instance and peer. We also calculated the number of statements which are generated by the swabbi-objects for each content statement.

Furthermore, we compared the number of swabbi-objects for the real data set with the DBLP data set to know how meaningful our test data set is.

**Case 2** The update of the content statements will probably be the most common procedure for the single peer, because it is performed it each time the peer updates its representation of the local knowledge. Hence, we compare the time consumed in this case for the DBLP case with the real data set. Furthermore, we examine the time consumed by the update using the database and the in-memory version. We did not update knowledge from external peers, since we already know its complexity from case 1.

**Case 3** The retrieval of a particular swabbi-object given a resource is of importance when the underlying source shall be retrieved, the confidence/trust rating is altered or when other meta information is needed. We observed the time consumed to retrieve 10 swabbi-objects for the in-memory version for the DBLB data set.

**Case 4** To retrieve knowledge which is stored on another peer, we need to find the peer objects for a particular resource. Hence, we compared the time consumed for retrieving all peer objects for one resource varying the amount of peers which have the knowledge.

---

<sup>2</sup>content + metadata

**Case 5** To sent a message to all known peers (broadcast) or to selected one based on its trust value, we need to retrieve all peer objects from the repository. Again, we compared the time consumed varying the number of peers.

**Case 6** Finally it is of importance, if the addition of the swabbi-objects to the local node repository does change the time neede to extract knowledge from the repository. We tested that case for the in-memory version with the DBLP data set.

Equa- tion	Max No. content statements	Max No. statements	Max No. peers	Store type	Data set	Figure
3.1	420.000	1.180.000	1	mem	DBLP	3.1
	18.000	3.180.000	100	mem	DBLP	3.2
3.2	18.000	3.180.000	100	mem	DBLP	3.3
	100.000; 72.000	300.000; 150.000	1	mem	DBLP; real	3.4
3.3	18.000	3.180.000	100	mem	DBLP	3.5
3.4	18.000	3.180.000	100	mem	DBLP	3.6
3.5	450.000	1.180.000	1	mem; database	DBLP;	3.7
	72.000	150.000		mem	real	3.9
3.6	450.000	1.180.000	1	mem	DBLP	3.10
3.7	18.000	3.180.000	100	mem	DBLP	3.11
3.8	18.000	3.180.000	100	mem	DBLP	3.12
3.9	450.000	1.180.000	1	mem	DBLP	3.13

Table 3.3: Overview test cases

### 3.4.5 Results

We have evaluated all equations for the DBLP case. As one can see from results of case 2 the DBLP case is representative for the filesystem case. All test were performed on a 1 GHz machine with 512 Mb RAM.

**Case 1** Figures 3.1;3.2<sup>3</sup> indicates the time used to integrate new statements. It compares the time used for annotation and integrating the annotations into the local node repository. Generating the new statements takes about as much time as integrating the statements into the repository, thus doubling the time needed to integrate the statements. The complexity is  $O(|S| * |P|)$ .

<sup>3</sup>Remember that 1.000ms = 1s; 100.000ms = 1'40"min; 10<sup>6</sup>ms = 16'40"min; 10<sup>7</sup>ms = 2:46'40"std.

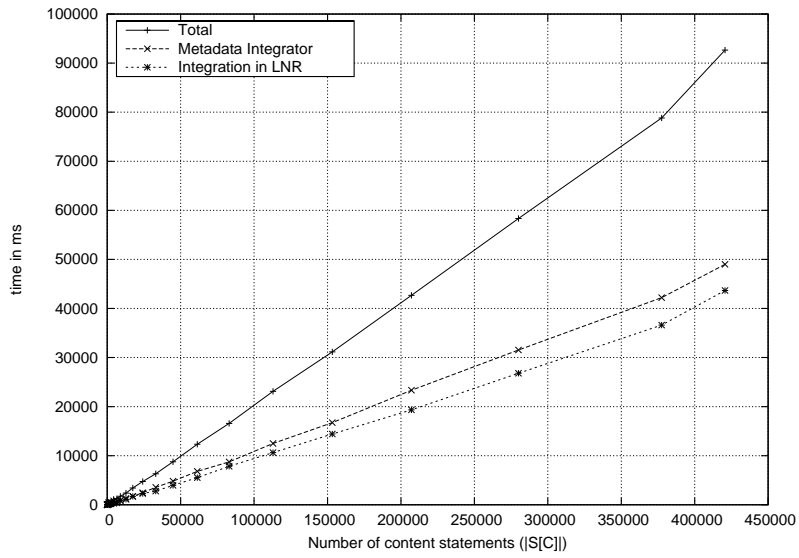


Figure 3.1: First time storage annotation times (see equation 3.1)

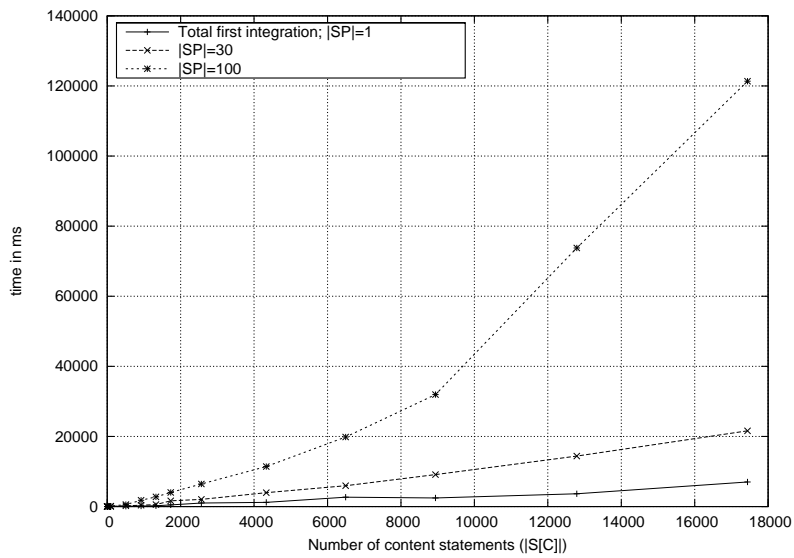


Figure 3.2: First time storage annotation times increasing the number of peers

In Figure 3.3 we plot the number of swabbi-objects against the number of content statements if the number of peers is changing. The bolder line is based on the equation  $|SO| = |S| * 0,3 * |SP| + |SP|$ . Since each instance is defined by approximately three statements the equation represents the expected number of swabbi-objects.

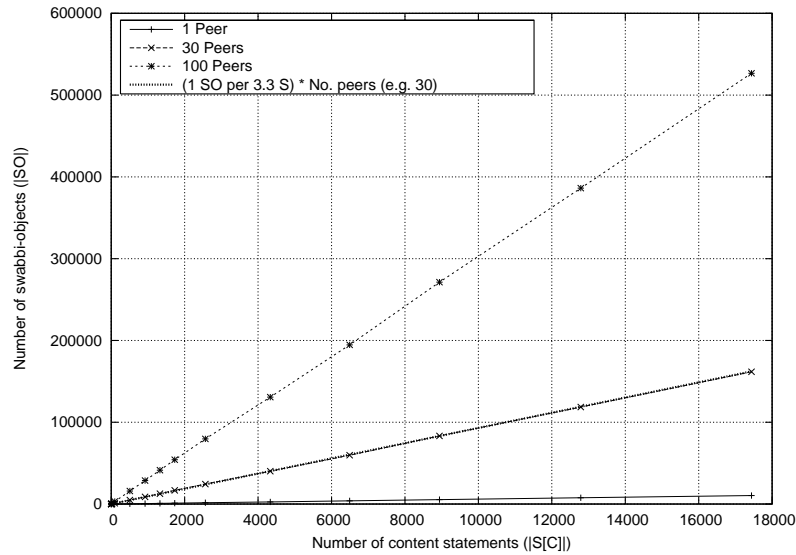


Figure 3.3: Number of swabbi-objects I (see equation 3.2)

In Figure 3.4 we plot the number of swabbi-objects against the number of content statements for the DBLP and filesystem cases. As seen in Figure 3.3 we included one swabbi-object for three statements. In the file system case we just include a swabbi-object for each 8th statement. Hence, our evaluations are an upper limit for the application case and we expect less overhead in a real application.

In Figure 3.5 we plot the number of swabbi-objects against the number of contributing peers if the number of content statements is changing. We assume that all peers contribute all statements. Hence, the figure shows the maximum number of swabbi-objects to expect. Again, the bolder line reflects the expected number of swabbi-objects.

In Figure 3.6 we plot the number of statements against the number of content statements if the number of contributing peers is changing. We assume that all peers contribute all statements. Hence, the figure shows the maximum number of statements to expect.

With 30 peers about 60 times and with 100 peers about 200 times more statements are added. This indicates that it is not feasible to store the knowledge and the references to the knowledge for all peers. As discussed in the previous section, selection must take place in the knowledge integration phase.

**Case 2** We assume that all statements are updated. Updating not all statements will result in lower time consuming. In this way we show the worse performance to be

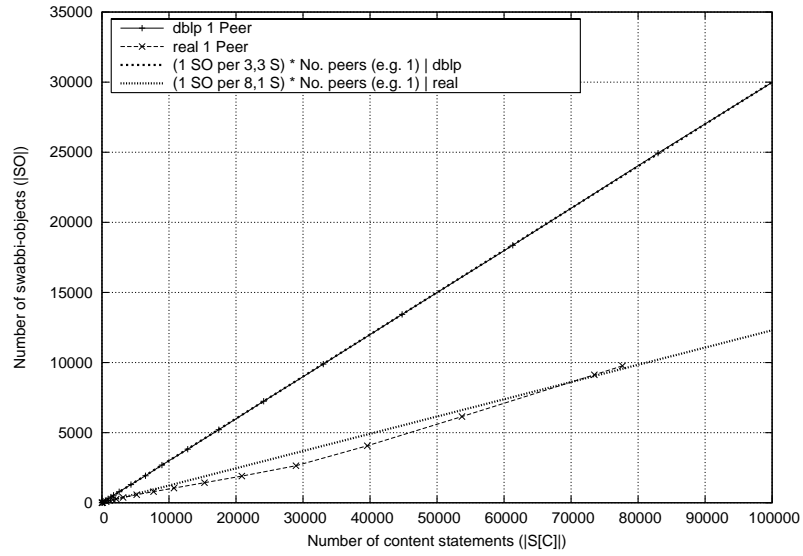


Figure 3.4: DBLP vs. realistic file system, No. of Swabbi-Objects

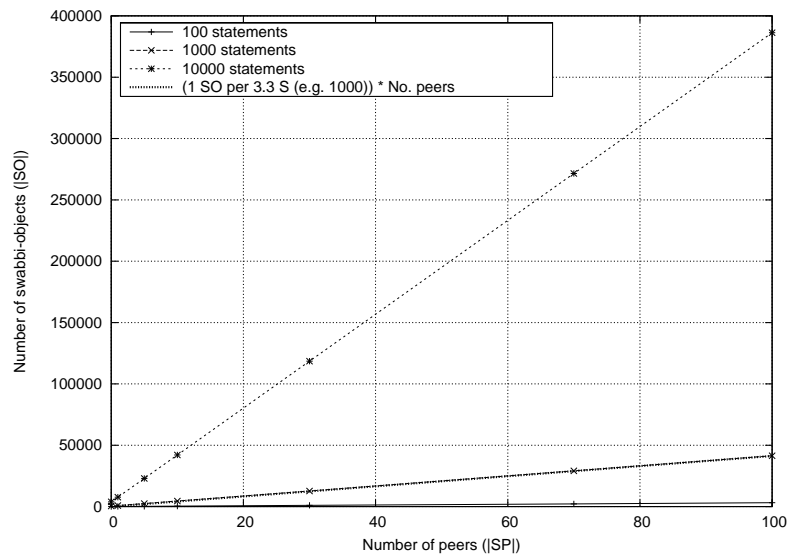


Figure 3.5: Number of swabbi-objects II (see equation 3.3)

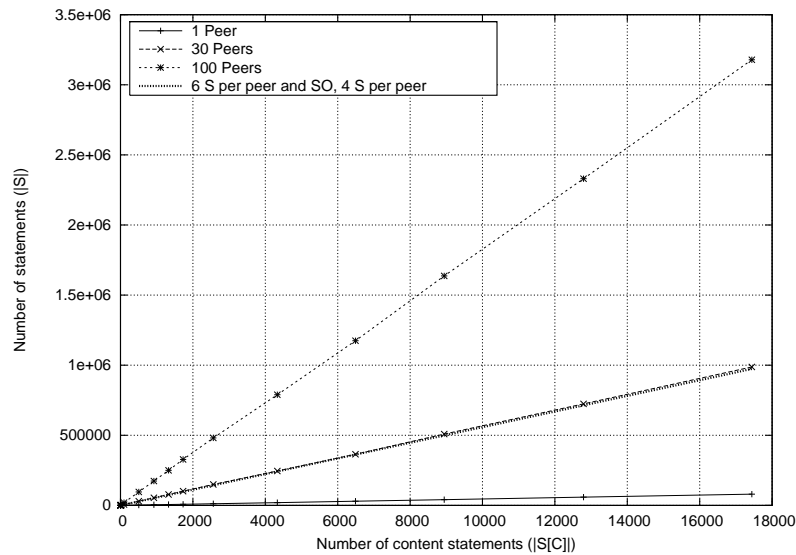


Figure 3.6: Statements and content statements. The bolder line is based on the equation  $|S| = 6 * |P| * |SO| + 4 * |P|$  (see equation 3.4)

expected.

In Figure 3.7 we plot the consumed time against the number of content statements for the different components involved. Additionally we compare the time consumed during update with the time consumed if the statements are added the first time<sup>4</sup>. We observe a linear time complexity of  $O(|S| * |P|)$ .

In Figure 3.8 we plot the total consumed time against the number of content statements in the DBLP case and the realistic file system case. As expected it is faster to integrate the real filesystem than the DBLP data, because the number of statements per instance is higher, consequently less swabbi-objects need to be generated. However, the complexity is in the same order.

In Figure 3.9 we plot the total consumed time against the number of content statements if statements are stored in a database and stored in memory (see also 3.7). The in-memory version is about 40 times faster than the database version. This is mainly due to the slower integration procedure and the slower search procedure of the database version which effects the metadata integrator.

**Case 3** In Figure 3.10 we plot the consumed time to retrieve one swabbi-object against the number of content statements. The time represents the average of ten runs. The figure suggests that the retrieval of the swabbi-objects is little influenced by the number of statements and very fast.

<sup>4</sup>Small differences in the observed time can be due to changes in processor time during performance tests.

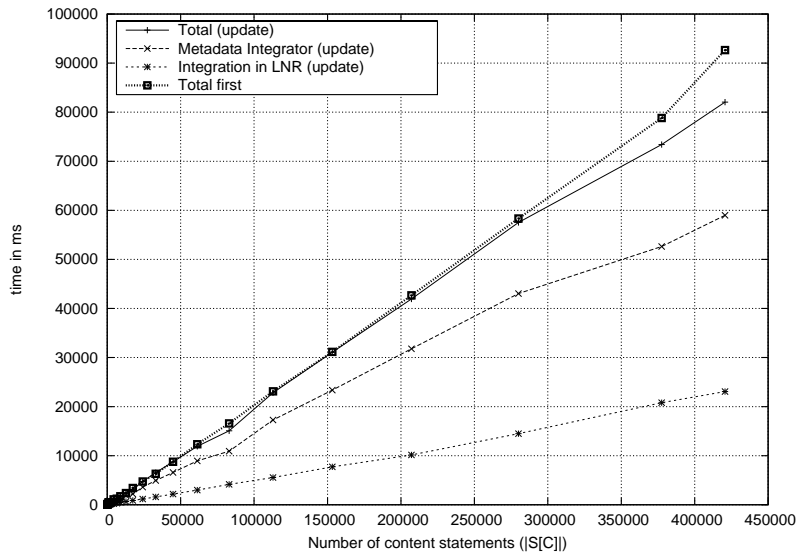


Figure 3.7: Update storage annotation times (see equation 3.5)

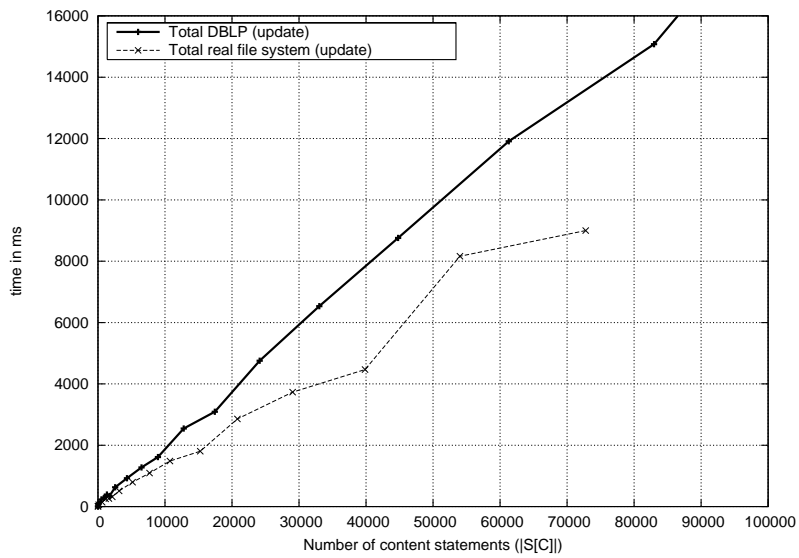


Figure 3.8: DBLP vs. realistic file system (see equation 3.5)

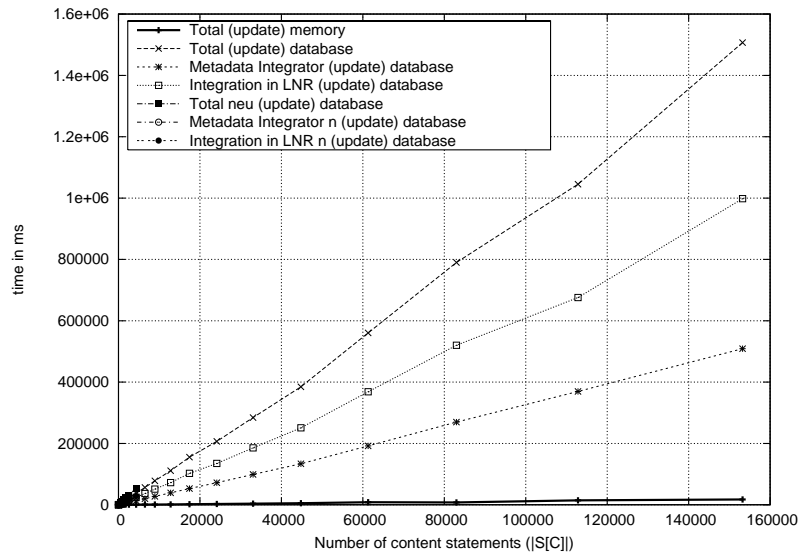


Figure 3.9: Update Memory vs. database (see equation 3.5)

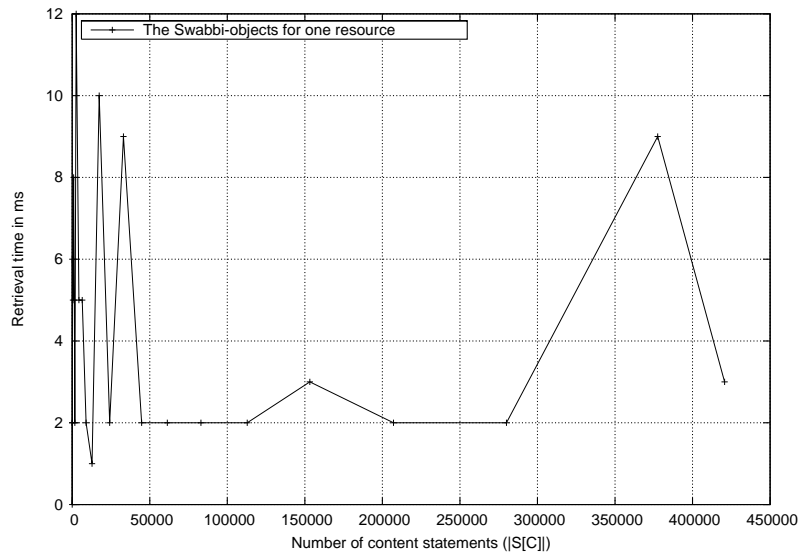


Figure 3.10: Time to retrieve a swabbi-object given a resource (see equation 3.6)

**Case 4** In Figure 3.11 we plot the consumed time to retrieve one swabbi peer-object against the number of know peers when the number of statements is changing. The creation procedure has a complexity  $O(|S| * |P|^2)$ .

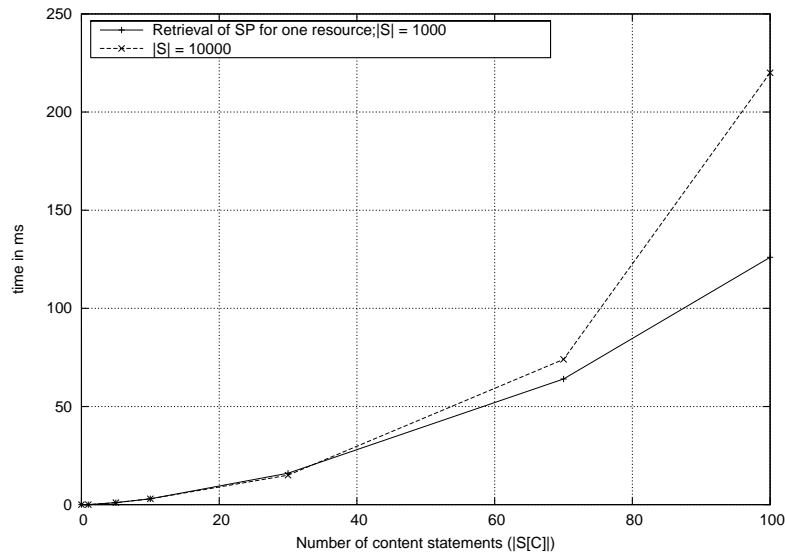


Figure 3.11: Time to retrieve one peer-object given a resource (see equation 3.7)

**Case 5** In Figure 3.12 we plot the consumed time to retrieve all swabbi peer-object against the number of content statements if the number of known peers is changing. Again, the complexity is not linear.

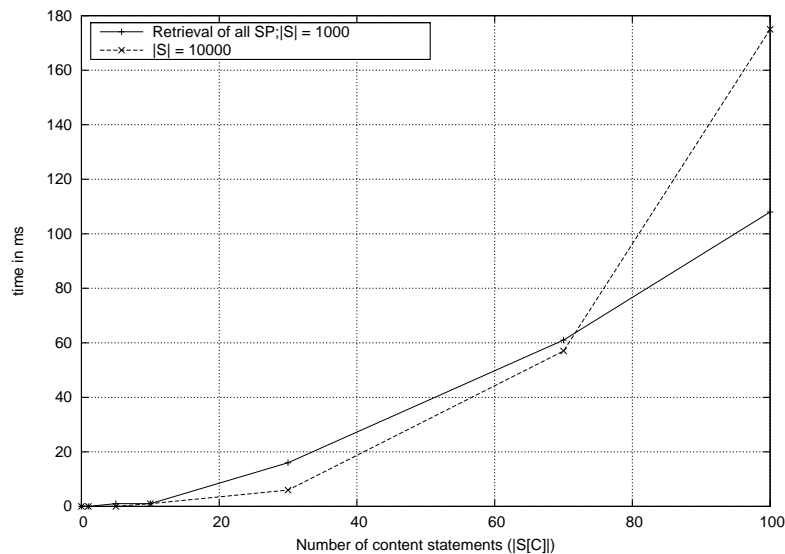


Figure 3.12: Time to retrieve all peers (see equation 3.8)

**Case 6** In Figure 3.13 we plot the consumed time to retrieve one statement against the number of content statements if we include swabbi-statements. We retrieve all `rdf:type` statements (approximately 100.000 with all 450.000 statements included). That means if we include swabbi-objects the size of the repository increases about 1 mio. statements. The evaluation suggests that there is no difference in retrieval times if the swabbi-objects are stored or not stored. At least not for the case that only the knowledge of one peer is completely annotated.

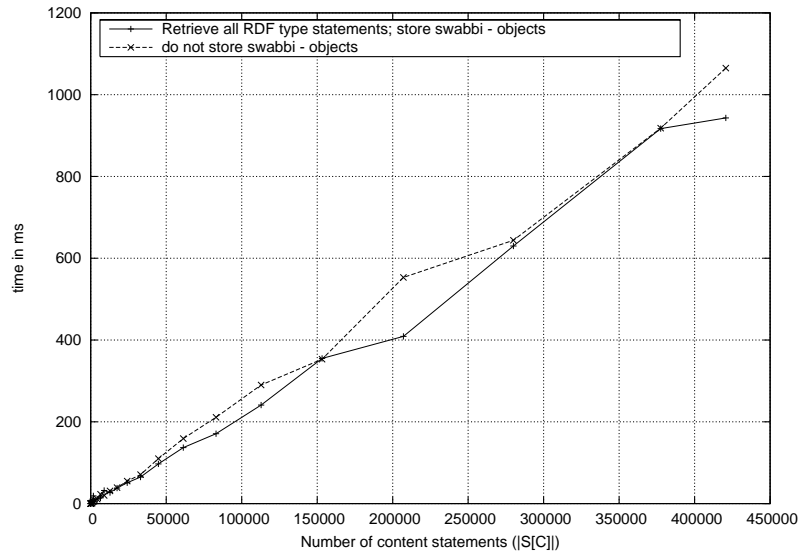


Figure 3.13: Retrieve `rdf:type` statements from repository (see equation 3.9)

### 3.4.6 Conclusion

The inclusion of the swabbi-objects in the repository is reasonable, since it does not alter retrieval times of knowledge. The time to include the statements into the repository doubles. However, it takes for a realistic file system of 100.000 statements only about 40 s for the in-memory case. The integration of the statements into the database takes about 25 min, while 2/3 of the time is spend for the inclusion of all statements and 1/3 for the generation of the swabbi-objects. This indicates, that it should be done as a batch process when using the database.

The increase of statements when more peers contribute knowledge is not feasible. Hence a selection process must take place. The retrieval times for swabbi-objects and peer-objects are well below one seconde, thus not recognizable for a user.

With respect to the case studies the method is applicable. The number of users as well as the number of statements which we expect are well below the limits of this method, hence we do not expect any performance problems due to the presented method.

## 3.5 Ontology Merging

The local repository is constantly confronted with information from other peers or applications. When querying it is necessary to identify semantical equal entities, even if they have different identifiers. The same is true when information is integrated into the local repository. Same entities will be identified, merged and in future actions pose as one entity. All this has to be done automatically, as users don't want to actively participate in the background processes of SWAP. As soon as this is complete, complex inferencing over bigger semantical structures is possible.

The intelligence is included in the step of mapping/identifying equal entities. The actual process of merging is just a syntactical transformation and therefore doesn't need any method testing. The evaluation therefore concentrates on mapping methods.

### 3.5.1 Scenario

A user has an individual understanding of the world. It is archived in his structures, his ontologies. To work together with other people, the user wants to know how his structures map onto the structures of others. After choosing two structures which are supposed to be merged the merging process is started. The results are returned to the user in a table, where he can see which entities have been automatically mapped to each other. Graphical representations can follow at a later stage of the project.

### 3.5.2 Data Sets

Two groups of data sets are planned for evaluation.

- One data set group represents ontologies created and maintained by knowledge engineers. They are of medium size, quite complex in terms of modelling with a about half of the entities being instances. In general this ontology will be rather small (around 500 entities). A set of ontologies describing the tourism domain, specifically for Russia, have been developed at the AIFB and are used for testing. Further a special tool allows to split one ontology in two parts. Explicitly added noise prevents the system to find the mappings to easily. Similar ontologies for testing can be easily created this way.
- The second data sets represent the typical SWAP scenario. Already existing structures of the user are taken i.e. the folders, emails, and bookmarks which have been extracted by the OntoScrape tool. The data is semantically less complex and error prone. On the other hand big amounts of data can be generated this way easily.

### 3.5.3 Evaluation Criteria

A core problem is the evaluation. The goal is to create correct automatic mappings for merging. The created mappings have to be checked against already existing and correct mappings. Typically these will have been created manually. When using the splitting tool the correct mappings can be created along automatically.

Three measures seem promising for our purposes. They are influenced by the standard evaluation measures from the information retrieval community.

$$\text{recall} = \frac{|A \cap B|}{|A|}$$

with A being the set of truly equal entity classes and B being the set of classes identified as equal classes (and merged) by the model.

This measure describes the rate of identified mappings versus the actual number of mappings in the entity set.

$$\text{precision} = \frac{|A \cap B|}{|B|}$$

The precision measure describes the rate of correctly identified equalities versus all identified equalities (including the mistakes).

**calculation complexity** Especially with big ontologies the complexity of similarity calculations can grow exponentially. As SWAP will be processing big amounts of data, this measure is important in our context. The time complexity is put together from several parts:

- Accessing a single entity. This is determined by the used ontology engine (and database).
- Number of (related) entities which have to be accessed by a certain similarity method to do its similarity computations.
- Number of absolute comparisons between two ontologies e.g. each entity of ontology 1 with each entity of ontology 2.

By caching all results to reduce calculation complexity the needed amount of memory rises. But as memory complexity is dependent of the actual implementation of a method the tests focus on time complexity considerations only.

### 3.5.4 Evaluation Process

The actual process again is straight forward. First the mappings between two structures have to be assigned manually. When the two structures are created automatically, this tool can also save the mappings during creation of the structures.

In a second step the defined methods are processed with the two structures and an automatic mapping table is created. For each entity of structure 1 we receive a best mapping entity from structure 2. The mapping also includes a quantitative similarity value. If more

than one method are integrated a weight is assigned to the single methods according to a learned maximizing heuristic over several runs.

A cut-off threshold has to be provided. It defines the similarity value needed to regard two entities as equal, so eventually not every found best mapping will be further processed. The threshold will be defined once for each method. The final list of assumed correct mappings is created.

The last step is to check the manual and the automatic mappings and calculate the evaluation measures described before.

### 3.5.5 Methods

A number of methods have been developed to compare two ontologies. They will only be mentioned briefly as the complete description is presented in a previous deliverable.

**Labels:** Entities are compared based on their labels. If the labels are similar enough, the entities are merged. Similarity of labels is determined on a spelling basis.

**Relations:** Not only the labels are used but all kinds of relations originating and resulting in the given entities. If the contexts are similar enough the two entities are declared as equal. For concepts these would be relations and attributes, for properties domains and ranges, for instances the values of certain properties.

**Hierarchies:** Ontologies have a taxonomy. By interpreting these structures for similarity a mapping can be created.

**Instances:** As soon as instances are known to be equal one can infer that closely involved concepts and properties are also similar.

**Focused Comparison:** To solve the problem of complexity the number of comparisons has to be scaled down considerably. A heuristic to choose among possible comparison candidates is used. A first approach could be to follow some hierarchical paths.

**Relaxed Similarity** describes methods which try to minimize the complexity of similarity calculations. They are based on the previous methods, but relax the number of aspects which are taken into account e.g. only direct parent concepts instead of all predecessors are checked.

**Similarity Paths:** Especially in a peer-to-peer environment we have much more than just two structures. A mapping from one entity to a second one can be possible via a third one (in an extra structure). An example can be a lexicon which knows synonym mappings the previous two structures didn't take into consideration.

The results returned by the methods are a list of entities of one ontology which maps to the best match of the second ontology. Additionally the similarity value between the two entities is saved. For further calculations the similarity values of the individual methods before summarization are also stored.

### 3.5.6 Results

The methods label, relations, hierarchies, and instances have been implemented and tested. As most of them are not meaningful as single methods they were grouped in the following way:

- the labels method alone
- labels, structures and instances together
- labels, structures and instances with assigned optimized weights

The tests were run on three ontologies from the first data set group. Their size was between 200 and 450 entities.

Due to the size of the second data sets the just enumerated methods can not be used. Currently the transformation of the exact methods towards a focused comparison with relaxed similarity is done. As soon as the complexity problem is under control it will also be possible to test the last method across several ontologies. Results will be presented in the next deliverables.

### Mappings

The following table shows the identified mappings. Please note that some of the possible matches might not even be identified by humans.

	test set	matches identified		
	existing matches	labels	labels, structure, instances	optimized
Set 1	278	145	197	201
Set 2	139	62	78	84
Set 3	440	41	206	210

### Precision and Recall

For each test set the precision and the recall are calculated. Starting with a small number of pairs, for which the similarity value is highest (i.e. they are most obviously identical), we add more pairs until every entity has a mapping partner assigned to it. Precision and

recall are plotted against the number of entity pairs. Different colors indicate different merging methods. The red lines always mark the results only using the labels. The blue lines use all methods equally rated. The green lines adjust all methods by giving specific weights to the single elements, i.e. receiving an optimized output.

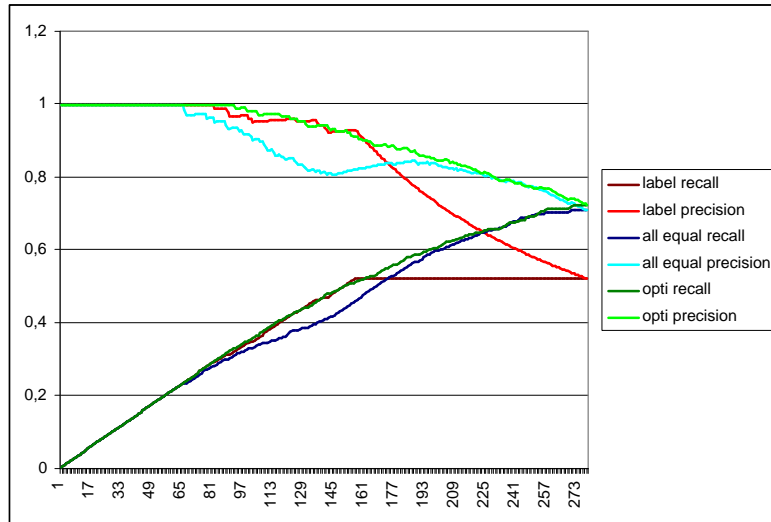


Figure 3.14: Identified equal entities in Set 2

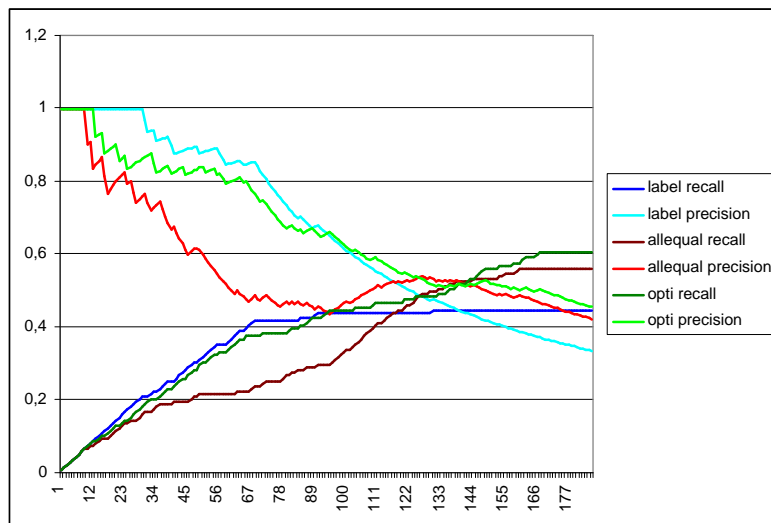


Figure 3.15: Identified equal entities in Set 1

When taking only the best mappings (the left side of the graphs), precision is maximal. On the other hand recall over the whole structure is minimal. By adding more entities the recall rises but at the same time precision loses some value. At the right edge the maximum number of mappings has been found, but quite some false assignments have also

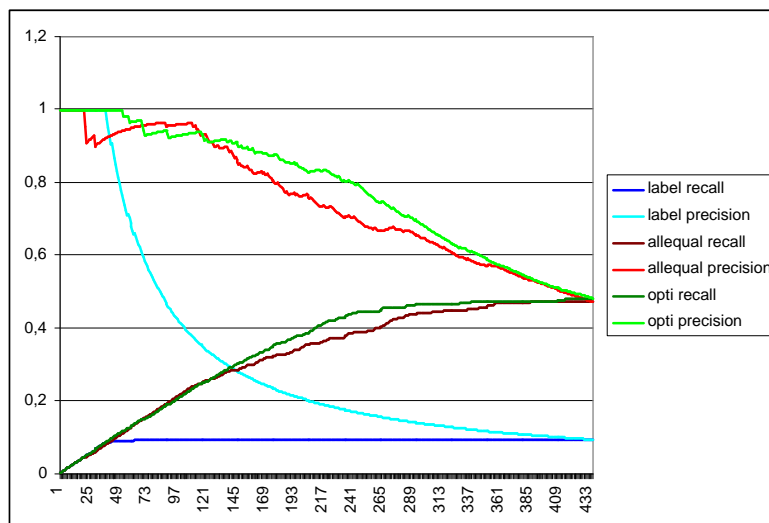


Figure 3.16: Identified equal entities in Set 3

been made.

As one can see from the three test cases checking labels already returns results. Including more structural information (all with the same weight) considerably improves these mappings. Minor advances can be found by adding optimal weights.

As described each best mapping is assigned a similarity value. Above a certain threshold the entities are treated as identical, below they are treated as different. Depending on the chosen threshold value recall and precision vary. The graphs show the recalls and precision for each strategy plotted against the similarity value.

High similarity thresholds mean a high precision, but a low recall. Low thresholds mean a lot of wrong mappings, which is shown in the lower precision, but with higher recall rates. Unfortunately finding the ideal threshold for cut-off is difficult. Depending on the test case the similarities vary considerably. A result from these tests could be that the threshold has to be determined for each ontology pair, rather than using one general threshold.

### Complexity

The complexity of each method was first calculated theoretically and then tested with the given data sets. The methods were again grouped as described above.

**Labels** The tests returned the following calculation results for the labels method:

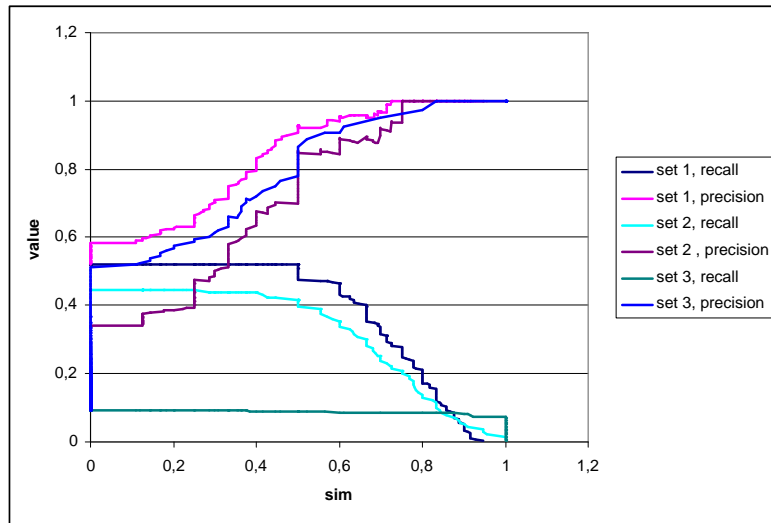


Figure 3.17: Optimized weights of single methods; identified equal entities plotted over similarity value

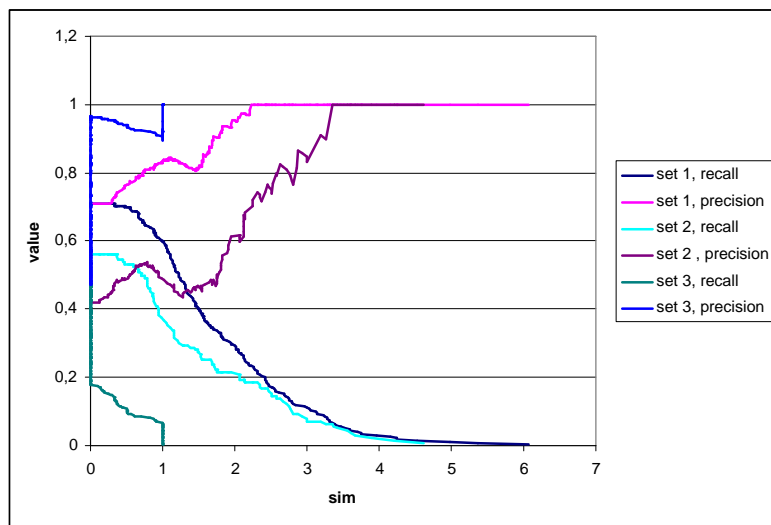


Figure 3.18: Optimized weights of single methods; identified equal entities plotted over similarity value

Test run	No. of entities	Duration
Set A	280	1 min
Set B	190	0.5 min
Set C	440	3.5 min

One can expect a complexity of  $O(\log^2(n) * 1 * n^2)$ . It is derived from:  $O(\log^2 n)$  for entity access,  $O(1)$  for the method complexity, and  $O(n^2)$  for the full comparison of

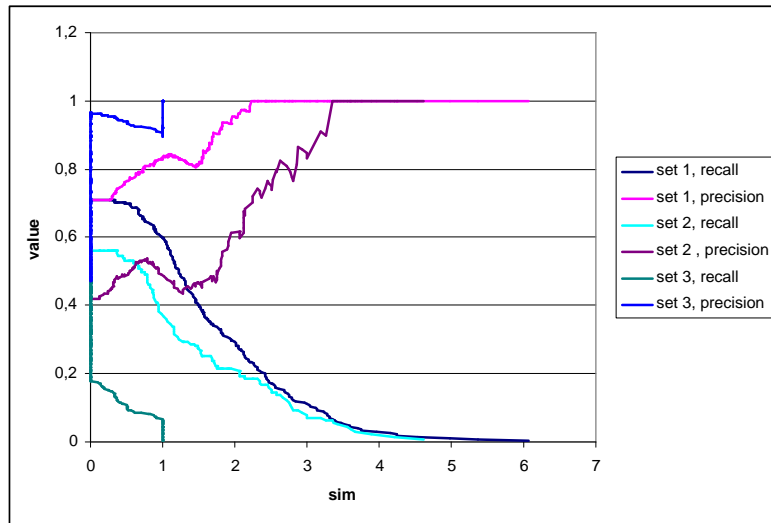


Figure 3.19: Optimized weights of single methods; identified equal entities plotted over similarity value

all possible pairs. The first test results correspond to the expected durations, even though more tests are required.

**Whole structure** The tests returned the following results for the structure methods:

Test run	No. of entities	Duration
Set A	280	6 min
Set B	190	3 min
Set C	440	346 min

Again the results were expected. The total complexity  $O(\log^4 n * n^2)$  is derived from:  $O(\log^2 n)$  for entity access,  $O(\log^2 n)$  for the method complexity, and  $O(n^2)$  for the full comparison of all possible pairs.

This complexity is even worse than when just using labels. Two main points to tackle the complexity problem will be investigated in future: change the methods themselves, or change the number of comparisons.

### 3.5.7 Conclusion

The presented scenario and its measures provide a good basis for testing new methods. Complex mapping algorithms perform better from a semantical point of view. The drawback is the higher complexity. The ideal solution is somewhere in the middle between

complex similarity measures and pruned comparisons. A lot more testing, including for new or refined methods will be necessary.

What one can also see from these results is that semantically enriched methods will considerably improve the merging for the SWAP project. Having the improved mappings in the local repository will eventually lead to better answers to queries - and improve the overall performance of the SWAP system.

The scenarios and data sets used are very typical for the planned SWAP use cases. The results obtained from the tests are therefore especially significant for the case studies and indicate that the developed methods will perform satisfactorily.

# Chapter 4

## Informer

In this chapter we describe test scenarios for the evaluation of the methods related to peer discovery, peer advertisements and peer selection. The reason for covering these methods with one test scenario is the tight coupling of the methods through the expertise model, which the methods are based on, as described in the following:

**Expertise Based Peer Selection** Advertisements are used to inform other peers about the content that is available on the peer. Complementary, discovery requests are used to find other peers that may have the answers to queries a user is interested in. The specifications of the capabilities, or expertises, of the peers is then used as the input for the peer selection algorithm: The peers whose expertises best match the topic of the user's query will be selected for query forwarding.

### 4.1 Advertisements, Discovery and Peer Selection

#### 4.1.1 Scenario

To test the functionality of the components related to peer discovery, peer selection and query routing we are using a real life scenario: In this scenario researchers in a community share bibliographic metadata, e.g. in the form of BibTeX files. A typical user query in this scenario might be: "Find all the publications about Database Management by E.F. Codd". The expertise model captures the topics about which peer can provide publications. These expertises are propagated to other peers as advertisements, e.g. "Peer A has an expertise in Information Systems". Knowing the topic of the query (Database Management), the expertise of a peer (Information Systems) and background information such as a topic hierarchy (in which, for example, Database Management is a subtopic of Information Systems), the peer selection algorithm might select Peer A for query forwarding because its expertise has the smallest semantic distance between expertise and topic of the query.

## 4.1.2 Data Set

To come up with a critical mass of bibliographic data, we have used the DBLP data set (<http://www.informatik.uni-trier.de/ley/db/>), which consists of metadata for 380400 publications in the computer science domain.

As the basis for our expertise model we have used the ACM topic hierarchy (<http://daml.umbc.edu/ontologies/classification.daml>). The ACM topic hierarchy consists of 1287 topics in the computer science domain. Using the relations defined in ACM topic hierarchy, i.e. *SubTopic*, *seeAlso* we can introduce the notion of semantic distance of topics and thus The documents of the DBLP data set have been classified according to the ACM topic hierarchy using a simple classification scheme based on lexical analysis: A publication is said to be about a topic, if the label of the topic is a substring of the title of the publication. For example, a publication with the title “The Capabilities of Relational Database Management Systems.” will be classified to be about the topic “Database Management” (ACMTopic/Information\_Systems/Database\_Management). Topics with labels that are not unique (e.g. ACMTopic/General\_Literature/General, ACMTopic/Hardware/General) have been excluded from the classification, as they typically result in useless classifications, Obviously, this method of classification is not as precise as a sophisticated or manual classification. However, a high precision of the classification is not required for the purpose of our test. It is sufficient to have a more or less realistic classification.

The DBLP data has been enriched with the classification information obtained by the process described above and represented according to the SWRC ontology (<http://ontobroker.semanticweb.org/ontos/swrc.html>). The following example shows a sample publication:

```
<rdf:RDF xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:acm = "http://daml.umbc.edu/ontologies/topic-ont#"
  xmlns="http://www.semanticweb.org/ontologies/swrc-onto-2001-12-11.daml#">
<Publication rdf:about="dblp:persons/Codd81">
  <title>The Capabilities of Relational Database Management Systems.</title>
  <acm:topic rdf:resource="http://daml.umbc.edu/ontologies/classification#
    ACMTopic/Information_Systems/Database_Management"/>
</Publication>
</rdf:RDF>
```

**Statistical information** The following table provides a statistical overview of the DBLP data set and the classification information.

Total number of documents	380440
Classified documents	126247
Number of ACM topics	1287
Number of ACM topics with classified documents	553

### 4.1.3 Evaluation Criteria

We use Information Retrieval measures as evaluation criteria for the methods tested. Here we distinguish measures on the document level (query answering) and on the peer level (peer selection). These measures are defined as follows:

#### Document Level (Query Answering)

$$Precision_{Doc} = \frac{|A \cap B|}{|B|} = \frac{|B|}{|B|} = 1$$

with A being the set of relevant documents and B being the set of returned documents. The precision will always be one, as we work with exact queries and therefore only relevant documents are returned.

$$Recall_{Doc} = \frac{|A \cap B|}{|A|} = \frac{|B|}{|A|}$$

States how many of the relevant documents were returned.

To determine the precision and recall on the document level, we compare with the centralized approach. In the centralized database, the number of documents retrieved will be equal to the number of relevant documents, as work with exact queries and all documents are stored in the central database. Precision and recall will therefore be 1.

#### Peer Level (Peer Selection)

$$Precision_{Peer} = \frac{|A \cap B|}{|B|}$$

For a given query, how many of the peers that were reached had relevant documents. Here A is the set of peers that had relevant documents and B is the set of peers that were reached.

$$Recall_{Peer} = \frac{|A \cap B|}{|A|}$$

For a given query, how many of the peers that had relevant documents were reached.

Other output parameters that might be used as evaluation criteria, but are not considered in the following, are for example:

- **Number of messages** How many messages were generated and sent around the network.

- **Message size** The size of the messages: Combining the number of hops with message size is an indication of the bandwidth usage.
- **Calculation time** The time needed for calculating the set of peers to forward a query to.

#### 4.1.4 Evaluation Process

The evaluation process consists of the following three phases:

(1) Preparation of tests, (2) The execution of the tests, and (3) the analysis of the test results.

##### Preparation of the tests

In the preparation phase, the DBLP data set is distributed across  $n$  peers according to one of the different distribution methods described below. Furthermore, a configuration script that will control the execution of the tests. This file contains information about the initial topology as well as the advertisements and queries to be sent.

##### Execution of the tests

For the execution of the tests we have created a dedicated simulation environment. It is a controlled, configurable environment that supports to run a large number of peers on a single machine. The test environment reuses components of the SWAP environment wherever possible. Other components, such as the Communication Adapter, have been implemented specially for the test environment.

The test runs consist of the following phases:

1. Setup topology: We create the set of  $n$  peers and the topology according to the configuration file.
2. Send Advertisements: Every peer sends an advertisement of its expertise to all other peers it knows. When a peer receives an advertisement, it may decide to store it if the semantic distance of his own expertise is close to that specified in the advertisement. It would be possible to store advertisements, but then we wouldnt see the clusters of semantically close peers.
3. Send Queries: The peers send the actual queries, as specified in the configuration file.

The results of each test run are exported to XML files, e.g.

```

<results>
  <!-- ... -->
  <result query_peer="5" answer_peer="690"
    topic="ACMTopic/Information_Systems/Database_Management"
    hops="1" results="270" />
  <!-- ... -->
</results>

```

This line would be interpreted as follows: Peer 5 has sent a query about Database Management. It has received an answer from peer 166 containing one statement (i.e. one qualifying publication). It took one hop for the query to reach the answering peer.

### Analysis of test results

The result data is parsed and stored in a database. This allows for efficient calculation of the output parameters.

## 4.1.5 Methods

In the process of expertise based advertising and peer selection there are several methods to be evaluated. These methods will be the input variables for the evaluation tests:

1. **Document Distribution** data among peers. The distribution may be according to topic, conference, author or other attributes, or it may be a random distribution.
2. **Topology** characteristics of the topology: For now we will use a random topology, where every peer knows  $n$  randomly selected peers. We assume that the topology does not change during the experiment. More important than the topology is knowing about the expertises of other peers.
3. **Expertise Model** How much and what kind of information is captured by the expertise model is a very important aspect. The expertises represent an abstract representation of the knowledge available on a peer. This could simply be a single or multiple topics. Quantitative measures (number of instances, confidence level) could extend the model. The abstractions of the knowledge in the LNRs are typically obtained by aggregate queries. Unfortunately, SeRQL doesn't support numeric aggregations (avg, sum, count) in its current version.
4. **Advertiser** The advertiser has to decide which peers to send advertisements to and which incoming advertisements should be accepted, i.e. included in the model of the peer. The peer could for example accept all advertisements, or it could accept only semantically close advertisements.

5. **Selection Function** The selection function determines which peers should be selected for query forwarding. One approach to select all peers with distance below a certain threshold, another approach would be to selecting the best  $n$  peers.
6. **Max number of hops** What is the maximum number of hops for forwarding a message? Typically, the recall will increase with the number of hops, but so will the number of generated messages.

### 4.1.6 Results

In the following we exemplarily present results for a subset of the methods described above. First, we will describe the input parameters that have been held fixed:

- **Document Distribution** The documents have been distributed according to topics, i.e. we have 1287 peers, every peer containing all documents for one topic.
- **Forwarding vs. Peer Set** Queries are forwarded directly to the selected peers.
- **Expertise Model** The expertise model simply contain a single topic of the ACM topic hierarchy, according to the distribution of the documents.
- **Advertiser** The advertiser sends advertisements to all peers it knows from the initial topology, but it only accepts the advertisements that are semantically close to the own expertise (semantic similarity greater than 0.2).
- **Selection Function** The best 2 peers are selected for forwarding. In the perfect topology case (see below) it is sufficient to forward to only one peer, as the queries will always be forwarded along the shortest path.

The following parameters are input variables:

- **Topology** We have compared two cases: In the first case the topology is formed – using global knowledge – according to the ACM topic hierarchy. That means, a peer knows all the peers with neighboring topics, i.e. the topics that are in direct subtopic or supertopic relation. We call this case “perfect topology”. Of course, in typical real life scenarios such a perfect topology cannot be realized, e.g. due to the lack of global knowledge. In the second case we generate a random initial topology. By accepting only advertisements which are semantically close to the expertise of the receiving peer, ideally a topology similar to the perfect topology emerges in the absence of global control.
- **Max number of hops** The tests have been run with a maximum number of 0 (no forwarding, i.e. only local queries) to 8 hops.

The diagrams show results for the Information Retrieval evaluation criteria for both input variables.

The first diagram 4.2 shows the recall of documents as a function of the maximum number of hops for both the perfect topology and the random topology case. For the

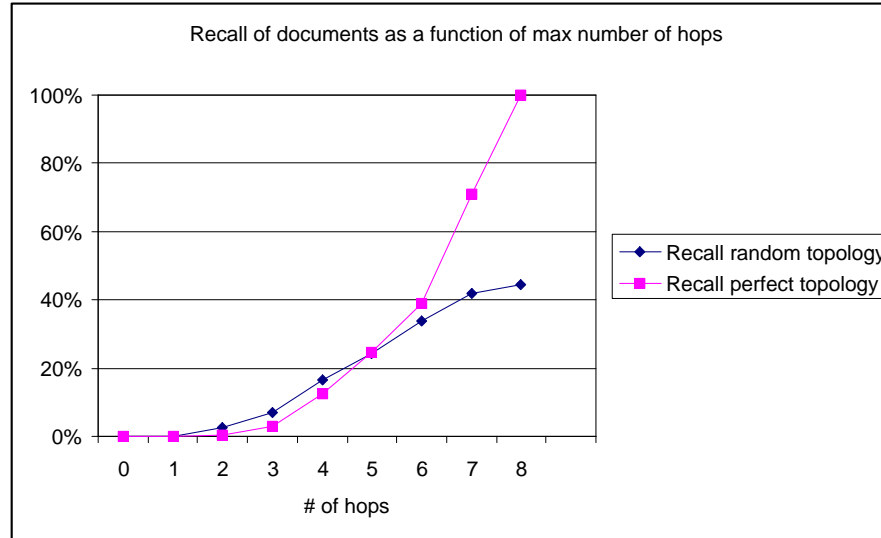


Figure 4.1: Recall of documents

perfect topology case we observe that the recall increases with the number of hops until it reaches 1 with 8 hops. This is what one would expect, as the queries are routed along the shortest path in the topology. The longest possible shortest path in the ACM topology has the length 8, we therefore need at most 8 hops to reach the peer that has the relevant answers. For the random topology case we also observe an increasing recall, however, it does not approach 1. The reason for this is that the peer with the relevant document may not be reachable with the given topology and peer selection algorithm.

A similar result can be observed for the recall of the peer selection, as shown in figure 4.2. For the perfect topology case we will reach the relevant peers with at most 8 hops. Again, in the random topology case we reach a recall of around 0.5 with 8 hops.

The last figure 4.3 shows how many of the reached peers actually had relevant documents. As one expects, in the perfect topology case, this value is considerably higher than for the random topology. A precision of 0.015 for the random topology case with 8 hops might seem very low, but one must consider that only 1 out of 1287 peers (0.0008) actually provides the relevant documents. Therefore, the expertise based peer selection outperforms a totally random selection by around a factor of 20.

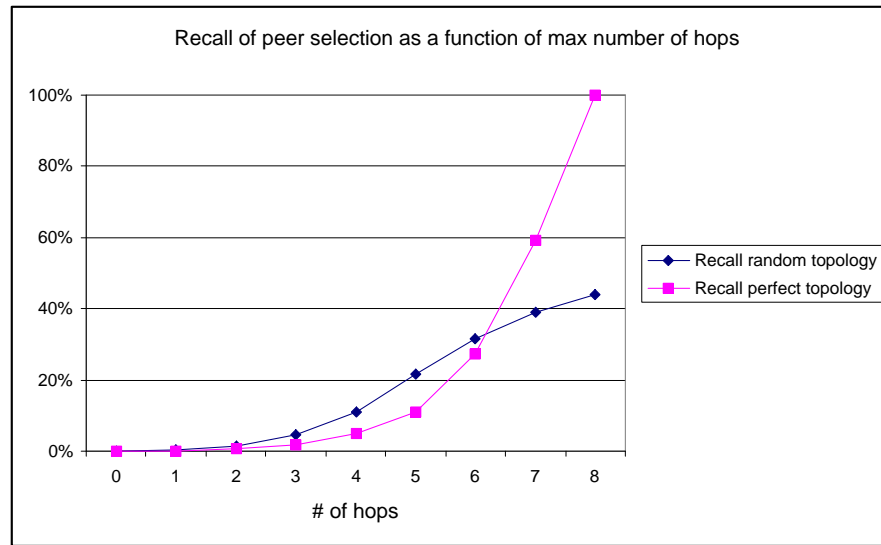


Figure 4.2: Recall of peer selection

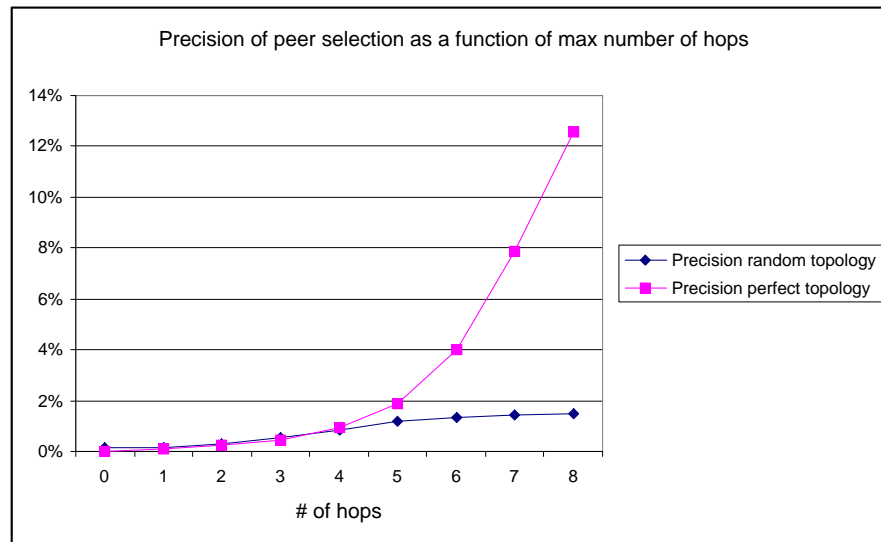


Figure 4.3: Precision of peer selection

### 4.1.7 Conclusion

Test evaluation results have been shown exemplarily for two input variables. (To evaluate and test other methods, the process described could be exercised in a similar manner.) The first major result is that using a topic hierarchy to impose a topology on the P2P network and using the knowledge about the hierarchy for query routing can yield a perfect precision and recall (for the chosen document distribution). Of course it is not always possible to impose this topology. But even with a random topology and intelligent advertisement

and peer selection algorithms based on semantic distance, we are able to achieve a fairly high precision and recall.

It is important to state that the results are very domain specific. The results obtained here are especially valuable for the proposed bibliographic case study. For a different domain one would need to adapt the expertise model. In our model for expertises and capabilities we are currently limited by the expressiveness of SeRQL. For the case study in the tourism domain, with a fairly small number of peers, the role of the informer will be relatively small, as the peer selector can fairly easily operate with a broadcast approach without running into scalability problems.

# Chapter 5

## User Interface

### 5.1 Visualization Requirements

The quality of a visualization depends on many factors. Some of them are linked to human cognition, our way of thinking and our mental limitations. We will not discuss these general requirements that hold for many kinds of visualizations, but rather concentrate on the specific requirements that can be derived from the application in the context of the SWAP project. Here we identify two main tasks that have to be supported. Both tasks are linked to different ways of accessing information. The first one is query formulation: here the user has to be guided in the process of choosing the right things to ask for in the right way. The second task is knowledge exploration with a more interactive process of accessing information in the network. We will discuss the requirements connected to these two tasks in the following.

#### 5.1.1 Query Formulation

The SWAP system takes advantage of the knowledge of the whole network in order to answer the queries of users. Visualization techniques should support the following query-related tasks:

**Query formulation:** Query interfaces that contain a graphical presentation of the available knowledge (ontology), make the query formulation task much easier as the user is already acquainted with the used terminology. The ideal situation would be that queries can be formulated visually.

**Query Result presentation:** Inadequate presentation of query answers shadows the performance of many search engines as the user is overwhelmed with results without being able to pick the part that is of interest for him. Graphical presentations can explain the relation of the answer to the original terms of the query.

**Query reformulation:** Very often the user is not satisfied by the returned answer: there are either too many items or no items at all. Graphical presentation of results allows a user to perform:

- Query relaxation: in case of empty sets, by giving up some of the search terms and choosing the results that come closer to the original query.
- Query broadening: in the case of an empty answer set some terms can be replaced by their super-classes. thereby broadening the scope of the query.
- Query narrowing: if there are too many items, the query can be repeated with more specialised sub-classes of certain query terms, narrowing the scope of the query.

Visualizations should help the user in finding relevant information and in addition should make the formulation of queries easier by giving visual clues.

### 5.1.2 Network Exploration

The main feature of a peer-to-peer network is the ability to access not only the information that is stored locally, but also to take advantage of information in the whole network. In an ideal case, the user of a peer-to-peer system will not even notice the difference between local and remote information. In the presence of conceptual knowledge as a basis for searching and querying, however, we have to deal with the fact that different peers will use different conceptualizations to describe their information. This means that if we still want to provide the kind of query formulation support described above, we need a proper specification of how the different conceptual models relate to each other. A common solution is to define mappings between different models that formulate semantic relations between classes and properties in the different models. A number of methods have been proposed to generate these mappings, however, none of these methods is elaborated enough to produce mappings that do not require human inspection and improvement. Here, again, visualization techniques can play a vital role to support the user in creating and inspecting mappings between the models of different peers in the system. In order to support the exploration of the network, we formulate the following requirements on the supporting visualization.

**Distribution of Knowledge:** The visualization should display the distribution of knowledge in the network. It should link peers to thematic areas based on the information they host. On a higher level, the visualization should clearly show groups of peers specialized in a specific thematic area.

**Personal Views on External Data:** In order to make the distribution of knowledge clear, it has to be possible to explore information in the whole network based on a single conceptual model. More specifically, this requires to be able to create personal views on external data in terms of a local conceptual model.

**Semantic Relations between Conceptual Models:** The creation of personal views on external information requires semantic mappings between the models of different peers. In order to better understand and to maintain these mappings, a visualization of existing mappings between two conceptual models is a further requirement.

**Comparison of Conceptual Models:** In order to judge the quality of existing mappings or to create new ones visualizations that help to compare different conceptual models are required. In especially approaches that highlight differences between models are useful in this context.

In summary, the wish to explore a whole network of information rather than a single peer poses new requirements on supporting visualizations. Here, visualizations that display more than one model a time are required.

# Chapter 6

## Adapters

### 6.1 Communication Adapter

A reliable and well-performing network communication layer is one of the fundamental success-factors for the SWAP project. This is one of the "basic infrastructure" components, which is crucial for the performance of the whole system.

The Communication Adapter was designed to fulfill these needs. The goal was to have a simple encapsulation of network communication, that is easy to use, reliable, and at the same time flexible - which means not constraining users of the layer (programmers in this case) in their choices.

One of the project goals was to operate in a peer-to-peer environment. We have decided to use JXTA as a peer-to-peer platform. At the moment we do not plan to evaluate any other platform, as we see no real alternative to JXTA in the Java world. However, by defining appropriate evaluation criteria, we make this evaluation and comparison with our implementation possible in the future when the opportunity arises.

#### 6.1.1 Scenario

The basic test scenario is very simple, and derived from SWAP platform use cases:

- user opens a SWAP application,
- he finds which peers are connected,
- he poses a query to one of these peers,
- he watches the results, and selects a file to download.

During these activities, the measurements will be performed and their results reported.

The test will be performed using several other SWAP components, like Local Node Repository. This will allow the whole evaluation to be more SWAP context-oriented.

The basic scenario will be repeated in several different configurations. Generally, the communication will be evaluated in two completely different environments - LAN and WAN. Both are important, and both should be evaluated as they are completely incompatible by their nature. For the WAN scenario we will assume the most typical configuration, ie. both peers will be hidden behind firewalls and NATs. The LAN scenario will be tested in an Ethernet network.

### 6.1.2 Data Set

The data set will contain:

- contents of local node repositories (RDF files) of several peers,
- a number of SWAP-specific SeRQL queries to be asked,
- a set of files that will be downloaded.

The contents of local node repositories will be the result of translation of file structure into SWAP RDF description of them, using an OntoScrape tool. This will make the whole test scenario more similar to the real usage.

### 6.1.3 Evaluation Criteria

There are two important aspects of communication for the end user:

- performance
  - time to first response - the time passed from user query until first answer is received (does not make sense in a download),
  - time to download - the time passed from user download request until the entire file is actually downloaded and stored locally (it could be called 'time to last response', but it does not make sense in a querying scenario, as there is no possibility to recognize that all answers are back, and timeout mechanism is used instead),
  - discovery time - the average time in which the peer can discover existence of a single connected peer,
  - availability time - the average time in which the peer can be able to contact other connect peer,

- maximum transfer - how much data (in terms of kilobytes in the case of download, and in terms of statements in the case of replying to a query) may be transferred during a fixed amount of time,
- reliability
  - approximate probability of losing once established connection to other peer,
  - approximate probability of transfer error.

### 6.1.4 Evaluation Process

For the performance evaluation, the set of little benchmarking tools has been implemented. They are able to measure all interesting for us performance numbers automatically. The initialization of measurements, like posing a query or initiating a file download, will be done by hand.

The reliability will be evaluated using SWAP platform UI by hand, as it is harder to measure it automatically.

All the tests will be performed in two different configurations:

**LAN configuration** - consisting of 3 different peers installed on 2 different PCs in the same local 100Mbit Ethernet network,

- all peers have JXTA configured in an ad-hoc manner (no rdv/relay, TCP on, HTTP off, TCP multicast on),

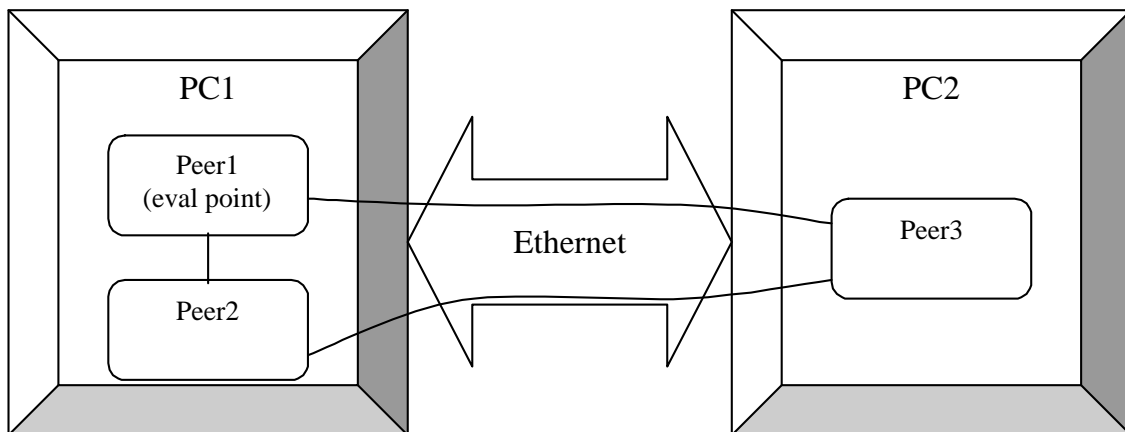


Figure 6.1: LAN configuration

**WAN configuration** - using 3 different peers installed on 3 different machines, where two of them reside in the same LAN and the third one is accessible only through Internet, both LANs are guarded by firewalls and use NATs,

- there is an additional rdv/relay peer making the communication through firewalls/NATs possible,
- all tested SWAP peers have JXTA configured the same way - TCP off, HTTP on, and a single HTTP rdv/relay configured.

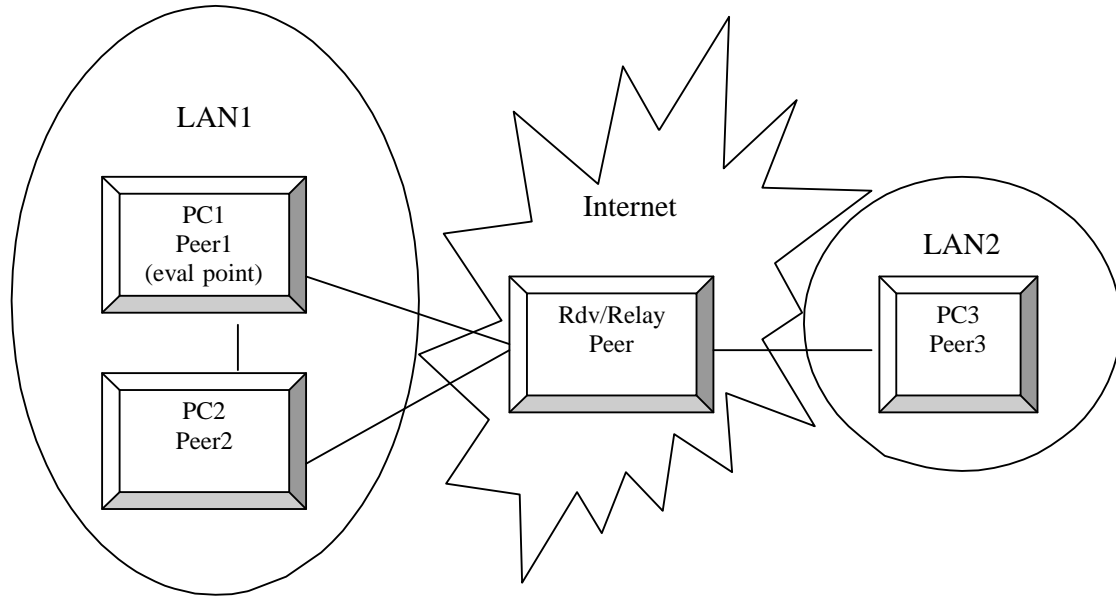


Figure 6.2: WAN configuration

### 6.1.5 Methods

**Functional Model- and JXTA-based Implementation** This method was developed on a base of more general approach to peer-to-peer communication developed as a part of SWAP project, called SWAP Functional Model. This approach is an effort to make it easier for developers to cope with peer-to-peer networks nature. The most common programming language paradigm now is a paradigm of a synchronous method call. However, this is very badly suited for network communication.

Functional Model delivers a very useful abstraction for that - Asynchronous Result. This is an object, that is a container for results of all network communication results. It brings:

- the flexibility of using of any of programming paradigms like: answer polling, synchronous wait for answer, or registering an answer listener on that,
- the ability of holding more than one result (in a network communication, it is rather the rule not an exception to get more than one result of an operation),

- the possibility of controlling the timeout behavior.

The current implementation of Functional Model in our system uses JXTA platform. For the motivation of this choice, please refer to [1].

## 6.1.6 Results

### Discovery & Availability Time

In this test, we have measured the time which takes to:

- discover another peer connected,
- be able to connect with this peer.

For each of two configurations, we made 6 runs of a test, simulating different scenarios - like peer starting order, manual activity of a user etc.

We have measured average and max time. In WAN configuration, we have also measured average time relative to rendezvous discovery, as we found a rendezvous discovery the most time-consuming factor (it took ca. 41 s without JXTA cache and ca. 52 s with cache to discover a rendezvous peer).

Time measured		LAN configuration		WAN configuration	
		peer on same PC	peer in same LAN	peer in same LAN	peer in WAN
Discovery	with JXTA cache	avg. 19,1s max 36,4 s	avg. 21,6 s max 36,4 s	avg. 60 s avg.-rdv 7,5s max 97,2 s	avg. 60 s avg.-rdv 7,5s max 36,4 s
	w/out JXTA cache	avg. 25,2s max 37,5 s	avg. 31,3 s max 37,9 s	avg. 58,1 s avg.-rdv 17 s max 80,4 s	avg. 60,1 s avg.-rdv 20 s max 80,4 s
Availability	with JXTA cache	avg. 34,1 s max 47,4 s	avg. 47,4 s max 76,6 s	avg. 102,9 s avg.-rdv 50 s max 222,4 s	avg. 108,4 s avg.-rdv 56 s max 22,4 s
	w/out JXTA cache	avg. 28,7 s max 38,5 s	avg. 34,8 s max 50,1 s	avg. 58,2 s avg.-rdv 17 s max 80,4 s	avg. 65,1 s avg.-rdv 24 s max 80,4 s

The most surprising conclusion is that using JXTA cache brings usually worse results. The differences between LAN and WAN scenarios are mainly caused by the time spent to find and communicate with intermediate rendezvous peer.

### Time To First Reply

In this test, every peer's local node repository was loaded with data extracted from a SWAP deliverables file directory, using SWAP system. The resulting repository contained ca. 870 statements.

Then, several queries were posed against each peer. The query was a simple "select all" one (we measure transfer time, and not performance of query engine). The measured value was the time it took before the first query reply came back (in seconds). In the WAN scenario, two variants of statement package size were evaluated - in one of them, statements were sent in packages max 100 together, in the other max. package size was set to 1000.

All results were compared to the results of querying the local peer (ie. the scenario of peer sending a query to itself).

		LAN configuration		WAN configuration			
	local peer	peer on same PC	peer in same LAN	peer on same PC	peer in same LAN	peer in WAN (1000 stmt/pkg)	peer in WAN (100 stmt/pkg)
average (s)	0,046	3,84	2,584	4,068	3,422	37,39	9,771
min (s)	0,031	2,516	1,562	2,734	2,031	48,922	8,687
max (s)	0,078	6,062	3,422	5,703	4	44,887	11,516

### Transfer Rate (Download & Query Replying)

The same scenarios were tested for query replying and file download transfer rates. The WAN tests were conducted in heavy real-life situation - the network was under some substantial load. Measured 'ping' time from the remote peer to the rendezvous machine was >300ms on average, and 3700 ms max. (60 tries of a 'ping' operating system command).

		LAN configuration		WAN configuration			
	local peer	peer on same PC	peer in same LAN	peer on same PC	peer in same LAN	peer in WAN (1000 stmt/pkg)	peer in WAN (100 stmt/pkg)
Statements (stmt/sec.)	20589,74	216,38	337,87	214,62	255,17	19,44	16,3
Download (KB/sec)	18738	434	350,2	210,9	205,2	7,1	7,1

## **Reliability**

The reliability of service was measured in the very simple way - using the system for 2 hours. No established connection between peers was broken. In one case of 6, the result of query to the remote WAN peer was returned incomplete, due to system timeouts. In the LAN scenario, nothing was lost.

### **6.1.7 Conclusion**

The conclusion from the evaluation performed is simple - the Communication Adapter satisfies the requirements for normal SWAP system usage planned. Even in the very hard network situation, the peers are still able to communicate. File transfers are enough for transferring ordinary documents. There are also two minor conclusions for JXTA configuration: surprisingly, using JXTA cache brings worse results overall. Secondly, LAN configuration is slightly better from the WAN one between two local peers, so the idea of a somehow mixed configuration as a synergy of the two should be investigated.

Regarding the originally planned case studies, the tests of the communication adapter suggest that discovery times and transmissions rates are sufficient. The touristic case study is completely covered with our test scenarios. However the scalability case study poses new requirements which could not be tested yet.

# Chapter 7

## Conclusion

In this document we have presented test scenarios for the testing and evaluation of the methods which have been designed and developed.

For most of the methods, the tests have already been practically performed. These first tests have shown promising results and have proven that the developed methods are working properly, are performant and usable. The results of this deliverable will also be used for the development of new and refinement of the existing methods with deliverable D3.6, Methods Refinement. In the same way as the methods, the test scenarios have been developed with the use cases in mind. The overall positive results of the performed tests suggest that the developed methods particularly meet the requirements of the use cases. Some of the test scenarios have been obtained using simulated environments, e.g. the expertise based peer selection described in chapter 4. Especially for these scenarios it will be interesting and important to validate them in real life with the case studies.