PRACA DYPLOMOWA MAGISTERSKA

Tomasz Piotr Kogut

Brokering platform for heterogeneous information sources

Opiekun pracy
dr inż. Dominik Ryżko

Ocena: ..........................
Kierunek: Informatyka
Specjalność: Inżynieria Systemów Informatycznych
Data urodzenia: 1987.10.08
Data rozpoczęcia studiów: 2010.10.01

Życiorys


EGZAMIN DYPLOMOWY

Złożył egzamin dyplomowy w dniu: ..........................................................2012 r
z wynikiem: ........................................................................................................
Ogólny wynik studiów: ..........................................................................................
Dodatkowe wnioski i uwagi Komisji: .................................................................

Podpis studenta
SUMMARY

Purpose of this work is to propose a solution that would bring ease to the process of searching data on a particular area of interest in the Internet. The proposed solution concentrates on utilizing already established search engines and databases and providing a unified interface to them. It is designed to be easily adaptable to any knowledge domain. Additional efforts were made to make the solution extensible for further data sources that may be independently attached to the solution in the future.

First part of the work focuses on recognizing widely used means of storing and exchanging information in the Internet. Subsequent chapter concentrates on a concept of a middle-man helper that is the central point of the solution. His role is to integrate and combine all of the analyzed technologies in the previous part. In the following chapters the concept of middle-man is transformed into a framework for knowledge searching.

The provided implementation allows for searching data in some of the most popular databases from the scientific domain. Results are integrated and displayed in a user-friendly form to the end-user.

Keywords: brokering, knowledge source, search engine, agent system

Platforma brokerska dla niejednorodnych źródeł informacji

Celem niniejszej pracy jest zaproponowanie rozwiązania, które ułatwiłoby wyszukiwanie informacji w pewnej określonej domenie w Internecie. Zaproponowana solucja, jako podstawa, używa już utworzone bazy danych oraz wiedzy znajdujące się w Internecie oraz zapewnienia zuniﬁkowany sposób dostępu do nich. Dodatkowo, poczyniono starania, by rozwiązanie pozwalało na dołączanie kolejnych, niezależnych źródeł wiedzy w przyszłości. Pierwsza część pracy skupia się na rozpoznaniu i analizie obecnie powszechnie stosowanych sposobów przechowywania i wymiany informacji w Internecie. Kolejny rozdział opisuje koncepcje pośrednika, który jest centralnym punktem zaproponowanego rozwiązania. Zadaniem pośrednika jest integracja i wymiana danych pomiędzy technologiami opisanymi w poprzednim rozdziale. W dalszej części pracy koncepcja pośrednika jest rozwijana i przekształcona w szkielet aplikacji dla wyszukiwania wiedzy. Dostarczona, przykładowa implementacja, pozwala na wyszukiwanie informacji w najpopularniejszych bazach danych z domeny badawczej i akademickiej. Wyniki są integrowane i wyświetlane w przyjaznej dla końcowego użytkownika formie.

Słowa kluczowe: brokering, źródło wiedzy, silnik wyszukiwawczy, system agentowy
Table of Contents

Introduction ........................................................................................................................................... 6

1 The information-centric world........................................................................................................... 8
   1.1 Computing history ........................................................................................................................ 8
   1.2 Information value ........................................................................................................................ 10
   1.3 Means of storing information ....................................................................................................... 12
      1.3.1 Databases ................................................................................................................................ 13
      1.3.2 Relational databases .............................................................................................................. 14
      1.3.3 Objective databases .............................................................................................................. 14
      1.3.4 NoSQL databases ............................................................................................................... 16
   1.4 Computer networks ...................................................................................................................... 18
      1.4.1 The Internet .......................................................................................................................... 18
      1.4.2 The World Wide Web ......................................................................................................... 19
   1.5 Semi-structured data formats......................................................................................................... 24
      1.5.1 XML ........................................................................................................................................ 24
      1.5.2 JSON ...................................................................................................................................... 25

2 Information brokering ....................................................................................................................... 27
   2.1 Seemingly unified data .................................................................................................................. 27
   2.2 Need for transparent data access ............................................................................................... 29
   2.3 The Middleman ............................................................................................................................ 30
   2.4 Data integration problem ............................................................................................................. 32
   2.5 Types of data sources .................................................................................................................. 35

3 Agent-based solutions for information brokering ........................................................................... 38
   3.1 Rationale ....................................................................................................................................... 38
   3.2 Types of intermediary agents ..................................................................................................... 41
   3.3 Shallow analysis of a middle-agent-less system for information retrieval .................................. 45

4 Implementation of information brokering system for search of scientific data ................................ 47
   4.1 Aim of the project ....................................................................................................................... 47
   4.2 GAIA methodology for agent analysis and design ..................................................................... 48
      4.2.1 Analysis .................................................................................................................................... 49
      4.2.2 Design ..................................................................................................................................... 54
   4.3 Used technologies ....................................................................................................................... 56
   4.4 Agents based on Actors .............................................................................................................. 57
   4.5 Brokering layer ............................................................................................................................. 59
      4.5.1 Task management .................................................................................................................. 59
      4.5.2 The dialect ............................................................................................................................ 61
      4.5.3 Combining search results ..................................................................................................... 62
      4.5.4 Queries .................................................................................................................................. 63
   4.6 Passim layer ................................................................................................................................ 65
      4.6.1 Passim dialect ....................................................................................................................... 65
      4.6.2 Searchers ............................................................................................................................... 69
      4.6.3 Executors ............................................................................................................................. 74
      4.6.4 Resource managing ............................................................................................................. 74

- 4 -
Information nowadays has become one of the most important aspect of the modern societies. Only important information really matters and ability to recognize important information often decides of the success and failure. Because of the computers and the Internet, information can spread and can be searched for easier than ever. Unfortunately, with the increasing power of computing came also growth of, very often unimportant, data into unimaginable size. Processing this data and searching thorough it became a challenge on its own.

Purpose of this work is to propose a solution that would bring ease to the process of searching data on a particular area of interest in the Internet. The proposed solution will concentrate on utilizing already established search engines and databases and providing a unified interface to them. It will be designed to be easily adaptable to any knowledge domain. Also additional efforts will be made to make the solution extensible for further data sources that may be independently attached to the solution in the future.

First part of the work will focus on recognizing presently used means of storing and exchanging information in the Internet. This will give an overview of technologies that may need to be serviced. Subsequent chapter will concentrate on a concept of a middle-man helper that will be the central point of the solution and which main role is
to integrate and combine all of the analyzed technologies in the previous part. In the following chapters the concept of middle-man will be transformed into a framework for knowledge searching.

The example implementation will allow for searching data in some of the most popular databases from the scientific domain. Results will be integrated and displayed in a user-friendly form to the end-user.

This work was financed by the The National Center for Research and Development within the scope of the Strategic Program “Interdisciplinary system for interactive academic and academic technical information”. Agreement number SP/I/1/77065/10.

Praca była finansowana przez Narodowe Centrum Badań i Rozwoju w ramach Programu Strategicznego "Interdyscyplinarny system interaktywnej informacji naukowej i naukowo technicznej", Umowa SP/I/1/77065/10.
1 The information-centric world

1.1 Computing history

In the past three decades we have moved from industrial civilization to the information age. Information has become a value on its own. But what can be called “an information”? Definition from the Cambridge dictionary states - “acts or details about a person, company, product, etc” [1]. It is also possible to use formal theory to define information. In year 1948 Claude Shannon presented his work on theory of information [2]. He defined information in terms of information entropy and uncertainty associated with a random variable. No matter what definition we choose intuitively information is something that gives us new knowledge.

Transition to information age would not be possible without integrating work from many different science disciplines. It was work of mathematician Alan Turing that gave theoretical grounding for creating computers. In the area of electronics and physics intense development could be observed from the late 30s of the 20th century. Research was financed by governments because of the need for machines that would be able to perform very fast computations. Development of new weapons and crypto-
graphic systems for the army during and after World War II had a huge priority. Computers of that time (e.g. American Eniac, German Z1, and English Colossus) were designed to be useful in areas of massive computations – cryptography, nuclear weapon development, weather forecasting and more civil directed computing PI number with high accuracy. Still those machines were unable to process information on a higher than numbers level of abstraction. [3]

Next milestone in area of computers and information exchange was connecting computers into networks. This was a huge improvement because the computing power of machines could be “transmitted” into another place different from the one where the machine was located. First attempt of a remote use of a computers was done in 1940 by George Stibitz. He used a Teletype machine to send data for computations from Dartmouth College to his “Complex Number Calculator” located in New York. Still it was a simple mathematics operation but it took no longer that 20 years for a real computer networks to be created. One of the first were network were SAGE1 and SABRE2. SAGE was a system of connected air-defense radars located in the U.S.A. SABRE was an airline reservation system so it can be considered first commercially used computer network. Few years later in 1965 first WAN3 was established which in turn evolved into ARPANET4 a straight ancestor of today’s Internet [4]. ARPANET gave scientist possibility to use computing power of geographically distant machines in their research. This was a notable change. Now project could be distributed among different universities, companies and government agendas5. Research result cold be easily transported and computing powers fully used. Connecting computers into networks gave unprecedented possibilities that were still to be revealed. Nobody at that time could imagine to what form computers and computer networks will evolve to.

1 Semi-Automatic Ground Environment
2 Semi-Automatic Business Research Environment
3 Wide area network – a network that consist of many network on a broad area (linked across metropolitan, regional or national boundaries)
4 Advanced Research Projects Agency Network
5 ARPANET was financed by United States Department of Defense
1.2 Information value

Because of the computer revolution we witness new challenges arise. The computational power of today’s computers can solve many problems but also generates new ones. Basic personal computer can process data very quickly but on the other hand also can produce large amounts of data. Widespread availability of the Internet and personal computers made it easier then ever to produce information and knowledge. People can put their own website on the Internet, use e-mails, create documents and share them with any number of person they like. Duplication of data is easier then any time before. Lately also social networking became important part of the Internet. On Facebook or Twitter people produces millions of messages everyday, most of them are aimed at specific people or group of people and are irrelevant for the rest. Additionally data from mobile devices (smartphones and tablets) can generate huge traffic of bytes to the servers. Having access to our position thanks to GPS, list of our payments by NFC wallet and all text that we write on IMs, e-mails mobiles are quite a data generator. What is notable is that like in social networks our data from the mobile device will be irrelevant for most people. One could wonder how much data is produced yearly? According to the research done by the IDC company “In 2011, the amount of information created and replicated will surpass 1.8 zettabytes (1.8 trillion gigabytes) - growing by a factor of 9 in just five years” [5]. The data doubles annually. Putting numbers into figures, how much information is that? In 2010 every two days there was generated approximately as much data as from the dawn of man till 2003 [6]. If the data doubled yearly, now in 2012, we are now generating that through half a day.

What is worth noticing is that quality of knowledge is not related to amount of knowledge generated. If it would be, in the last decades there should happen a huge development jump, which was not observed. Although there is much more information available important data remains its value. Professional analysis, confidential information, personal data are desirable by companies and private persons. How much such

---

6 Instant messaging  
7 Zettabyte is $10^{21}$ bytes
information is worth? It strongly depends on the information type and company profile will it be able to cash it. Starting from personal data (e-mails, chats, personal information, birth day and place etc.), Google and Facebook use it to personalize adverts shown on their web pages and in related services. With this useless at first sight knowledge Google grown to be worth 64.72 billions of dollars and Facebook to be worth 13.31 billions of dollars⁸.

Next, documents produced by experts (like articles, PhD. thesis, academic books, journals ect.) are especially valued among professionals and scholars. It can be considered as knowledge in classical sense of that word. Before the information revolution where mostly people working at the universities had access to other scientist works and it was their domain to publish. How much is such a “classical knowledge” worth nowadays when we are flood with other information from the Internet? Approaching this problem from the other side can be interesting. Let's not try to answer how much article or a book can be worth (this can vary based on topic), but let's ask how much a person will be able to spend money on publishing own work in a valued publishing house. Springer is a successful publishing company with recognition among professionals. It delivers specialist information from a wide variety of areas from IT to law. Springer publications are appreciated for its high standards and it gives guaranties that publication if is valuable will find recognition. Springer pricing for option named “Open Access” is US$ 3000/ EUR 2000 plus taxes, depends on country [7]. “Open Access” gives all people online access to published work for free. Of course Springer main aim is to run successful business so it cannot act as a hosting platform for free. If such a publication was published with paid access it would increase potential customers database for Springer but because publication will be available for free Springer has to put a fee on the author. Probably fee high enough to convince him no to use this option to often. Unfortunately Springer does not publish exact data how

⁸ Information from June 30, 2012, source: WolphramAlpha by asking “how much is Google/Facebook worth”
many people used this option but it gives a general idea how much a publication is worth, at least in magnitude.

Two knowledge source types already described were fairly easy to propose a model to evaluate them in terms of money. There are other areas where there is hardly any possibility to assess value of the information. In mid 1960s when computers started to develop very fast a “Pacific Tsunami Warning System” was established. Such a system was also installed in Indian Ocean region so now information are gathered round the clock. Sensors redistributed among many countries in Asia send data to analysis centers. If any threat is detected with means of modern communication like SMS and emails and IM people can be warn with enough timespan for evacuation. This is a good example of an information system that people benefit from in an unmeasurable way [8].

Few examples mentioned above show that information plays very important role in our times. It always had but now when information can be transferred to other end of the world instantly, possibilities are much greater. It is also a fact that “production” of information proceeds. This seeming paradox of increasing value and amount of information leads to an issue of filtering out irrelevant information and gathering different sources of information into one comfortably available and queryable source of information.

### 1.3 Means of storing information

The magnitude of data produced everyday enforced necessity of finding an efficient way of storing and exchanging information. As it was stated before, in a timespan shorter than one day we produce more information than till the end of the last century. For persistent storage of information computers use databases. First databases were not distinguished from other computer software. These systems had no query
language only programming interfaces, so adding new functionality was hard and time consuming since it involved writing new code, testing and debugging. In addition databases were expensive since implemented mostly on mainframes computers. Over the years databases evolved with computers to meet new needs. Today database management systems (DBMS) are one of the most complex systems but in return give many benefits from using them. DBMS take care of the data redundancy what is important because it saves disk space and ensures that data about an entity is logically connected and further updates will be always visible. Next, access to database can be restricted with security policy. Since database can contain sensitive data this is a need so only authorized person can be allowed to view them. Furthermore, data saved in a database can be a mirrored rich data structure from the programming language that in addition may be interrelated. Database will also take care of integrity allowing to put constraints on data. Finally the data can be searched with a specific database query language dialect. Thanks to agreement between companies developing databases those languages are more or less standard and scripts work interchangeable between producers with minor changes.

1.3.1 Databases

Traditional database systems were used only for reaching either textual or numerical information (e.g. flight reservations, bank accounts). Growing power of computers and research in DBMS area in the past few years allowed for extending set of types of saved data [9]. Good example are multimedia databases in which can store pictures, sound and video. This ability is used by popular services like Youtube and Vimeo for storing, searching and streaming videoclip content. Another new type of databases are so called GIS⁹ systems. Designed to support searching for spatial data in maps, processing satellite images and predicting weather conditions. Without it there wouldn't be services like Google Maps and geolocalization in smartphones would be much less

⁹ Geographical information system
useful. Next, special OLAP\textsuperscript{10} systems are used in many companies to support decision making. Such a system has an ability to extract information and rules from VLDB\textsuperscript{11}. Last of the new directions in database area is online and realtime processing which is used in industrial and manufacturing processes.

1.3.2 Relational databases

Traditional databases (i.e. processing textual and numerical data) still play important role. Mostly build on the relational model proposed in 1969 by Edgar F. Codd. which is in turn based on first-order predicate logic ensures great flexibility and performance. This mathematical foundation was proposed to separate the physical storage of data form its conceptual representation. Such aproche allowed for introduction of a universal query language for such databases. Hence it was a lot faster to write queries than to change the database itself. Today SQL\textsuperscript{12} is very popular among programmers and is an ISO\textsuperscript{13} standard implemented by all main DBMS producers. Relational databases now exist on almost every type of devices from small smartpones to large servers, on all major platforms and are dominating. Most popular relational databases are Oracle Database 11g, Microsoft SQL Server, PostresQL and MySQL.

1.3.3 Objective databases

The relational model has been successful in developing the databases for traditional buisness applications mentioned before. However, with time complexity even of traditional applications increase. Therefor in some areas relational design has some shortcomings. When data complexity rise it's better to use pure objective design instead of entity-relation. Applications for engineering design ,manufacturing (e. g.}

\textsuperscript{10} Online analytical processing
\textsuperscript{11} Very large databases, term used when refering to database with extremly big number ot tuplles. The term „extremly big” will change over time as computers will grow faster. Now it is considered 1TB of size and over billion of rows.
\textsuperscript{12} Structured query language
\textsuperscript{13} ISO/IEC 9075-1:2008 http://www.iso.org/iso/catalogue_detail.htm?csnumber=45498
CAD, CAM, CIM)\textsuperscript{14} and scientific experiments data can gain much from changing the paradigm. The object-oriented approach gives much more flexibility not being limited by query language and standard data types. Potential benefits are:

- custom operations on objects,
- objects don't require to be assembled and disassembled what saves coding and execution time,
- concurrency may be much easier since a tree of object can be locked,
- reduced paging.

Although many object-oriented databases have been created, none have found wide acceptance among developers. Drawbacks of using object-oriented databases are:

- lack of official standard,
- still pretty immature comparing to classical ER model,
- late binding may increase access time,
- lower efficiency when data is simple and relations are simple.

That is why most developers still prefer using relational databases. To ease use of the relational databases with object-oriented programming languages additional layer of abstraction the ORM\textsuperscript{15} is constructed. Most programmers prefer to use object instead of tuples, rows and columns and ORM is such an idea that allows changing the point of view without abandoning verified solution. Benefit is that programmer gain an objective interfaces without loosing portability of the database. Most popular examples of such implementations are \textit{Java Persistent API} for \textit{Java Virtual Machine} and \textit{LINQ} for Microsoft's .\textit{net} platform.

\textsuperscript{14} Computer-Aided Desing/Computer-Aided Manifacturing/Computer-Integrated Manufacturing
\textsuperscript{15} Object-relational mapping
1.3.4 NoSQL databases

The term NoSQL was first used by Carlo Strozzi in 1998 for his open source database that did not give a SQL interface. The term was popular again from 2009 and used for conferences of advocates of non-relational databases [10]. Rationale for searching alternatives to already existing databases (especially relational) was the need for processing large amount of data that already established solutions couldn't handle. Advocates of NoSQL databases claim that relational databases are just too expensive and slow to handle some areas of needs. They claim that some scenarios of use cannot be addressed with a traditional approach. Example of such an area is social media services like Facebook and Twitter. Also growing number of internet start-ups avoid using SQL solutions in favor on their own or open source solutions. These companies mainly influenced by Amazon's Dynamo [11] and Google's Bigtable[12] design their solutions to fit well into cloud computing trend. Many of those project become open source like for example first developed at Facebook Cassandra project for new text search feature, become a project under the Apache Foundation. Since NoSQL databases can allocate different needs, they can differ in way of operation. However a set of common traits can be outlined [13]:

- lacking of fixed schemas
- avoiding joins of data
- horizontal scaling.

Lacking of fixed schema allows for easy manipulation of the data. Of course constraints are impossible to implement but it's a trade-off for flexibility. This doesn't mean that data will be inconsistent. It maybe outdated at first by eventually data will propagate through the whole system. In contrast to ACID\textsuperscript{16} systems where operation is performed atomically or has no effect at all we avoid unneeded complexity where it is not needed. Since most of the databases don't use joins (or have joins performing very

\textsuperscript{16} Atomicity, consistency, isolation, durability – set of properties that guaranties that databases transactions are processed reliably.
poorly) data is kept encapsulated. This feature allows for vertical scaling of the solution easily. Additional servers can be added (or removed in case of crash) and there is no need to rely on highly available hardware to avoid sharding in RDBMS clusters. Moreover costly ORM is not needed since most data is keep in format that corresponds to object-oriented programming structures. Objects don't need to bend to fit the relational schema. As it was stated before NoSQL databases found it niche in social media and there is easy rationale why. Since most of the sites are paid for clicking an advert and it is not affect by the fact that data can be outdated or inconsistent (Facebook user won't notice statuses changes). Keeping data consistent all the time may cost more than the advert click is worth (since such a click may happen after thousand of transactions) developers had to look for optimization. Among NoSQL databases we can distinguish three main forms of the database\textsuperscript{17}. One is simple Key-Value database that will act like an associative container. Unique key is used to identify an object. All operations rely also on that key like insert, delete and update. Programmer gets a fault tolerant, distributed and persistent hash table. Although the model is very simple underneath they may be used very sophisticated methods to reach desired performance. Example of such a database is mentioned before Amazon Dynamo. Second type is document-focused databases. The main idea behind this concept is to give user ability to define tuples with any number of attributes of any length without wasting space as none of the attributes is required to insert data into database. There is no schema limiting what information can be stored. Cost of this approach is lack of safety mechanism since data may differ very strongly between tuples and some decrease of performance. What is interesting is that a semi-structured text files (e. g. JSON, XML, YAML) can be used to implement document-focused databases. Semi-structured file formats will be discussed in chapter 1.5. Third type of NoSQL database is column-oriented databases. As the name states it focuses more on the columns rather than rows as in relational database it is done. This kind of database characterizes it easier to pick up single attributes and operate on them rather than getting the whole row. This implies
\textsuperscript{17} http://nosql-database.org/ has a wide catalog of NoSQL databases of all types, retrieved on 2012-08-15
that aggregation on small subset of attributes but with many values will will perform faster. As a results this kind of database will outperform relational in cases where interactive transaction are made such as automatic teller machines.

1.4 Computer networks

Computer networks main task can be generally described as resource sharing. Putting it into other words the aim is to allow anyone in the world to make something available to another person without regard to physical location. An example could be sharing a printer between computers, but more generally it can be describe as sharing information. Even the most sophisticated database system would mean little without computer networks. Ability to process massive amount of data and ability to send it to anywhere in the world makes computers irreplaceable today.

1.4.1 The Internet

Most widely used computers network in civil application is the Internet. Thanks to design organized loosely on the OSI 7 layers (actually has only 4 layers) can easily adjusts to always-changing needs in computing and connect people by wire and air. Thanks to that general design Internet based on the TCP/IP protocol is not tied to a certain technology of data transfer or transportation medium. This allowed finding new applications for it over time. Even more it started to supersede other technologies. If we look at the telecommunication area, where thorough the years a set of complex protocols for sending voice and text were developed, today operators, whether concerning mobile or stationary, turn towards using the Internet. A good example is the SMS-PP\textsuperscript{18} protocol which is a very successful service but there is an effort to replace it with new standard the RCS\textsuperscript{19}. RCS is based on IMS\textsuperscript{20} and it will allow for rich com-

\textsuperscript{18} Short Message Service – Point to Point, globally known by its shor name the SMS
\textsuperscript{19} Rich Communication Services
\textsuperscript{20} IP Multimedia Subsystem
munication with voice, text and video. This makes the Internet even more indispens-
able as the number of mobile devices is significant and these devices now depend on
the Internet heavily. Another example of leap towards the Internet is development of
the OpenAPI by the GSMA\textsuperscript{21} organization. The OpenAPI standard intendeds to ex-
pose telecommunication network to the Internet as webservices\textsuperscript{22}. Not only the tele-
coms industry which primary goal was to communicate people (so was the computer
networks) start to rely on the Internet. Thanks to growing bandwidth of the network
and increasing mobility television not only used the Internet as a medium for its con-
tent but also allowed for interaction and on-the-go usage for the spectators which was
previously impossible. The Internet embraces almost all aspects of our world today
and revolutionaries our outlook. Today over third of world's population uses the Inter-
net which became a universal platform for building communication solutions and con-
vergence with already established.

1.4.2 The World Wide Web

Probably the most important service available in the Internet (certainly most popu-
lar one, widely misstated as the Internet) is the World Wide Web (WWW). It was cre-
ated by Tim Berners-Lee and Robert Cailliau in early 90s of the 20 century in
CERN\textsuperscript{23}. The main idea behind WWW is combining the Internet with hyperlink docu-
ments. Hyperlink document contains links to other documents that can be accessed
from it. Thanks to the Internet a WWW document can point to other documents places
on other servers anywhere in the world.

Dynamical development of the WWW was possible because of two things. Firstly
WWW was freed from fees by CERN shortly after the technology began to raise inter-
est. It allowed third party developers to create unlicensed extensions. Secondly its ar-
chitecture differs from other popular that time hypertext systems in using unidirec-
tional links instead of bidirectional. That strongly simplified development of web

\begin{footnotesize}
\begin{tabular}{ll}
\textsuperscript{21} & GSM Association \\
\textsuperscript{22} & Webservices are being discussed further in chapter 1.4.2 \\
\textsuperscript{23} & Organisation Européenne pour la Recherche Nucléaire
\end{tabular}
\end{footnotesize}
servers, browser and making of new web pages. Today technologies developed at the beginning of the Web evolved and extended their capabilities beyond initial assumptions. Because CERN freed WWW it became open and everyone could develop software for it. That caused problems in compatibility between vendors. That is why World Wide Web Consortium (W3C) was established to coordinate works on development of different aspects of the Web.

Essentially WWW consist of three technologies [14]:

- Hypertext Markup Language (HTML) for meta description of how documents in the WWW should be presented to the user
- Hypertext transfer protocol (HTTP) used for transmission of documents in the web.
- system of global unique identifier for things that could be places in the WWW, first named universal document identifier (UDI), later renamed to URI (uniform resource identifier) or URL (uniform resource locator).

**HTML**

Example of large incompatibilities involved HTML which initially was simple language for formatting webpages. Because many browser implemented incompatible markups there was ongoing effort to standardize it. Companion technology CSS\textsuperscript{24} was developed as a complementary way of adding custom styles to the HTML page. HTML 4.01 was also tried to be published using XML named XHTML but it development was abandon. Now in its fifth version it allows for many advanced options transcending the role of presentation-layer-only language. New features involve video and sound playback, drawing on canvas, geo-localization and offline applications. Also many imprecise definitions from earlier version were tightly specified to avoid browser compatibility issues.

\textsuperscript{24} Cascading Style Sheet
HTTP

HTTP protocol is a simple stateless text protocol. It consists of set of methods. Most popular are GET, POST, HEAD, OPTIONS, PUT, DELETE, TRACE and CONNECT. Methods set can be freely extended if needed. If server does not understand some method it can ignore it informing about it. These methods are applied on a resource pointed by a URI. There also exist a safe encrypted version of the protocol named HTTPS. Because protocol is stateless additional efforts must be taken to simulate a session. Tracking user navigating from page to page can be accomplished using cookies, URL rewriting with parameters or hidden variables if forms available.

URI

URI is a special convention of creating identifiers for resources. It subset is an URL. The difference between two come down to fact that URL actually tells you how to get to the resource specifying access protocol, path and other needed parameters. All URIs are URLs but not all URLs are URIs. At first URL main role was linking webpages and realization of the hypertext idea. In general HTTP protocol is not a need. Any other access protocol like FTP or SSH can be used. The same concerning content that can be pointed by URL, not necessary it has to be a web document, but can be virtually anything.

Web service

Widely used applications built on top of the WWW is the Web services. The W3C defines a "Web service" as "a software system designed to support interoperable machine-to-machine interaction over a network" [15]. Two main ways of implementing web services that are currently used are SOAP and REST. One is based on advertising service capabilities and allowing for stateful interaction is build on top of SOAP and WSDL technologies and involve using XML data structures. SOAP is consider as

25 Special text files stored on computer of the client with data that server can read next time visiting.
26 Simple Object Access Protocol,
27 Representational state transfer
a rather complex protocol that implementation takes some effort. In contrast to it REST is quite simple in its design and assumptions. It is constrained to set of operations similar to these in HTTP (i.e. GET, PUT, POST, etc.) and is focused to apply these operations to resources pointed by URIs rather than sending complex messages and maintaining state of the interaction.

**Semantic Web**

On top of these three main technologies other applications were introduced, mainly driven by the W3C. WWW in its initial assumptions was a web of documents that could be linked together and were useful for a human-being. It is mainly focused on the presentation layer. HTML set of markup tags described how the document will look like, but no meta information about the content can be introduced. It is problematic to make a machine processing of the web documents available on the network and expose them to querying in sense of databases. Intensive research in area of NLP\(^28\) is an ongoing effort to make web understandable for machines. These methods rely on probabilistic methods so can and will introduce some errors in results. To overcome these difficulties a “Semantic web” concept was introduced. Main aim of these project is to transform web of documents into web of data such as these available in database [16]. For fulfilling this aim technologies for building custom vocabularies, rules and data stores are developed. Foundation for semantic web are RDF\(^29\), SPARQL\(^30\), OWL\(^31\), and SKOS\(^32\).

**RDF**

RDF is used for conceptual modeling data (in particular data placed on web) in a similar to entity-relation model used in databases. Using so called triples subject-predicate-object data is being placed semantically with other resources. For example to de-

---

\(^28\) Natural language processing
\(^29\) Resource Description Framework
\(^30\) SPARQL Protocol and RDF Query Language, this is a recursive acronym so SPARQL is not extendible.
\(^31\) Web Ontology Language
\(^32\) Simple Knowledge Organization System
note that Warsaw is the capital of Poland we would write a triple Poland-Capital-Warsaw. Here we have two resources Poland and Warsaw tied with a relationship. Of course the relationship could be swapped using other predicate e.g. Warsaw-isACapital-Poland. What is important is that resources in RDF are pointed by its URIs, so its usage has been extended and two applications using RDF are using the same object when referring to the same URI. RDF collections can be visualized as a directed graph and the text representation is not enforced so any suitable form can be used for storing and processing. RDF is very flexible and on top of it were build other ontology languages.

**OWL and SKOS**

Example of such ontology language is OWL and SKOS that gives a set of classes and properties that can be used for building triples. Profit from using standardized language for building applications is that these can easily talk each other and freely exchange information if using the same standards.

**SPARQL**

As it was stated before, semantic web is a form of database, so clients should be able to query the content. For this task SPARQL was proposed. SPARQL is a query language loosely based on the SQL allowing for constructing complex queries using many ontologies. By specifying triple RDF patterns connected with conjunction or disjunction client can send queries to the endpoint using the standard web protocols and getting answer also through it. As it was said queries can indeed be very complex and detailed, example of such a query can be asking “all rivers in Germany that are longer than 50 km”. Building a query like this would involve using many ontologies, and these ontologies are also available through its URI so when asking queries we only send namespaces that we would like to use and not the whole ontology.
1.5 Semi-structured data formats

For the Web be able to connect various systems there is a need for special data format so it can be easily encoded to, send through the network and decoded from. Most traditional software systems use propriety binary format for storage. This is inconvenient when data must be exchanged between different systems, often written in many programming languages. Another drawback of binary format is being unreadable for human-beings. Binary format also need dedicated software to be viewed what often may be very hard to accomplish. That is why most systems that are built with purpose of working in the Web environment use text format. Text format can be easily opened with any text editor, changed and saved what made them preferred by the developers. This fits into the Web philosophy where most popular transport protocol the HTTP is also text-based. Many text-formats were introduced to store information with focus on different purposes. Some try to make common tasks for programmers easier while processing, some try to be more readable for a human-being and that is why they have different expression power. Only some of these formats gain enough attention to have good tools support and have a wide usage. Most popular ones are strongly related to topics described earlier and on those formats this chapter focuses.

1.5.1 XML

XML stands for “Extensible Markup Language”, and is very similar to HTML. In contrast to HTML it was designed to represent data and information, not how the data should be displayed. Main goal for XML was to be easily transferable over the Internet and support wide variety of applications. Set of markups is not predefined but created during creation of the document. XML only defines a set of rules how the data should be encoded. Much flexibility resulted in great number of formats build on the XML. To number just a few:

- RSS/Atom - downloading news headlines
• SOAP - mentioned earlier, protocol for interacting between two machines

• XHTML – an XML compliant version of HTML

• Office Open XML, OpenDocument – file formats for storing files created with word processors. They replaced previously used binary formats.

• XMPP – instant messaging protocol.

• Scalable Vector Graphics – file format for storing two dimensional vector graphics

XML has a strong support for different languages. All aspects of XML like text, comments, tag names are encoded using Unicode. XML besides being “well formatted” (following defined syntax) may also be checked with a schema making it “valid”. For this purpose a “Document Type Definition” is being published. DTD defines set of valid tags and constructs that may be used within a document. What is worth mentioning is a document schema may be also published using XML itself. This type of document is called “XML Schema”. XML was also recognized by databases producers making available type of data intended for storing XML files.

1.5.2 JSON

JSON stands for JavaScript Object Notation. It is based on a subset of the JavaScript Programming Language. It's a lightweight data-interchange format. Comparing to XML is rather less loose in size, and that is why it is easier to send json documents through the internet since they are smaller in size. Although it is completely programming language independent it uses terms that are familiar to programmers using C-like languages (i.e. C, C++, C#, Java, JavaScript, Pear, Python etc.). JSON relies on two structures:

• collection of key-value pairs what is realized in most languages as associative arrays,
• an ordered list of values that is realized in most languages as some kind of sequence.

Using these two kind of structures JSON documents are build from forms:

• objects that consist of unordered set of key-value pairs,

• arrays that is an ordered collection of values,

• values can be a string, number, Boolean value, null, object or another array,

• string is a sequence of Unicode character. Of course usage of Unicode makes it possible to serve difference languages,

• number has the same meaning like in C programming language.

JSON is not extensible but it is not meant to be, and it does not have cyclic data structures. All what can be written in JSON can also be written in XML. What made JSON so popular is it connection to JavaScript language and last years heavy use technology named AJAX. It makes possible dynamical changes and asynchronous operations on web pages.
2 Information brokering

2.1 Seemingly unified data

In chapter 1.3 there was presented how information are stored and shared between computers. On one hand it may seem that there is an ongoing endeavor for making possible unlimited sharing of information and data between computers. Data not only considered as a stream of bytes but also being meaningful and understandable both by human and machine. Construction of computer networks, the Internet and the World Wide Web also indicate that this is true. Also commonly used data formats like XML and JSON aids this aim. On the other hand there can be observed many different approaches to data storing and processing. Considering databases we can see that there is no “one size fits it all” solution and from particular needs NoSQL was born to meet those constrains that classical databases were not able to achieve or the cost was too high. This resulted in variety of database access interfaces. Not only there are problems with SQL dialects between databases but also some databases don't use SQL at all. As for the data formats on top of XML there were build many applications that besides having common data format share no semantics and meaning. Sending data from
one application to another is pointless, although both can communicate using HTTP
protocol, and agree on method invocation, used data remains unintelligible. Also Web
documents, the HTML pages, are rarely created with guidelines of the semantic web
committee. Not only may a web page author have insufficient willingness to follow
the strict rules (much extra work has to be done to meet the requirements) but also
may lack knowledge about ontologies and W3C standards. Even if he is aware of the
semantic web there can be no proper ontology available to describe topics of his inter-
est. And if such an ontology exists author may be unknowing of it. What more one can
imagine many ontologies covering the same facts and areas that are incompatible with
each other and it is not possible to reconcile two different source using different on-
tologies. Another problem is that huge amounts of knowledge were created before
computers were common. The problem is not all of documents are digitized. More-
over these documents that were digitized not always contains sufficient meta-data.

Another problem coming from rather historical custom is that most of the really
valuable documents like scientific publications are published in “Portable Document
Format”³³ format. Unfortunately like HTML its a rather presentation-focused technol-
yogy than a semantic wrap. There is even problem for automatic recognition of para-
graph order since PDF uses 2-dimensional coordinate system rather than linear text
stream. Of course NLP methods can be applied but still this is not very handy and er-
ror prone. Another thing that has to be taken into consideration is that many compa-
nies are making transition into Web with their customer offers and workplace. Major-
ity of them already had some kind of an internal computer system not connected to
network up to now. Such system probably won't be adjusted to meet semantic web re-
quirements (high costs) and already used internal structures will be available maybe
with some small changes to the Web (e. g. by a HTML document or web service de-
pends).

³³ Better known from its acronym: PDF
2.2 Need for transparent data access

The Internet together with the Web create a great platform for unconstrained data exchange. One can expect increasing bandwidth and extending accessibility in future. What probably won't happen (at least in nearest future) is unification of data semantics. It is not truth that this cannot be technically achieved but probably will fail because of the human factor, it involves too much effort and time. History knows projects that failed because of being too complex. A good example which is connected to described topics is the “Xanadu” project, it can be described as older relative of WWW with bigger functionality. However Xanadu was started much before the WWW in 1960, it haven't been released until 1998. In the meantime WWW was created, implementing only a subset of features that Xanadu was aiming at. But it was the WWW that gained popularity because of loosening constraints put on the document creator, giving less possibilities and power in favor of ease of use.

Taking problem with semantics into consideration there arise a need for system that could allow for exchanging data between emerging systems in the WWW. With availability of more and more systems in the WWW one will have to face also problem of finding data sources that meet one's needs and problem of visiting all of them to have the most comprehensive results.

Possible software solution in context of mentioned problem should have the following characteristics:

• it should be easy to add already developed knowledge source (KS) to the solution without modifications of both,

• it should be possible to use common protocols like HTTP for data transfer,

• service should have a standardized access method for clients,
• access to the KS should be completely transparent for the client and any special knowledge about used syntax and semantics of the KS should not be needed,

• system shouldn't be complex to avoid “Xanadu” problem.

2.3 The Middleman

If using data from the Web is to be easy an additional layer of abstraction that hides all complexity and allows to focus on using the knowledge instead of searching for the knowledge is needed. A solution archetype to stated problem can be found in real world occupying many different areas. At this point let's discuss profession of a broker. A broker is a person who arranges transaction between two persons. In most cases one person is seeking for a product (thus can be called a buyer) and the other is offering products (a seller) [17]. Therefore broker is the third person that takes part in the transaction. Broker would normally offer his services for a commission but this aspect is not elaborated anywhere further in this work. Example of a broker from real life is person who facilitates a real estate transactions.

Broker is used extensively in some industries and what makes his help so valuable, especially for buyer, is his wide knowledge about market he specializes in. Broker spends most of the time contacting sellers and scanning theirs offers so he has considerable amount of knowledge about prices and goods. Therefore, has he enough resources to get to a wider spectrum of sellers than any regular buyer would be able to. Cooperation with a specified buyer is incidental and in most case will be limited to a moment when certain good is sold. That's why most brokers can offer their services to a wide range of people making the service inclusive. As for the sellers they also benefit from brokers especially entering the market when they have limited access to customers. Finally brokers can introduce much savings especially when large group of
people seek for the same good and broker can try to optimize the transaction. This phenomena is know as group or collective buying. As it was already said brokers operate on many markets. In context of this work one should get more attention than the others. The information broker (sometimes called an Independent Information Professional) is a person who does information and knowledge searching, management and application. This profession is relatively young but continues its growth and it seems that market need such services [18]. It seems that this kind of activity can be an inspiration for designing software being middle-layer between information seekers and providers. Table 2 presents in the left column description of certain aspects of a broker job and in the right column how those aspect could be mapped to a knowledge sources proxy – the information brokering system. Making use of an example from real-world scenario gives a good point to start the design of the information brokering systems and helps to cover certain use cases during implementation that are likely to show up and be required in the end system.

<table>
<thead>
<tr>
<th>Real world Broker</th>
<th>Software Broker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cooperates with people looking for certain good called buyers or clients.</td>
<td>Should make available its interface for other software, the clients. Should service multiple clients at once.</td>
</tr>
<tr>
<td>Cooperates with people that run their business and provide goods called sellers.</td>
<td>Should cooperate with other software, the knowledge sources, that share some interface that broker can possibly use to find interesting information.</td>
</tr>
<tr>
<td>Brokers has some kind of market specialization where it can act effectively.</td>
<td>Broker should be able to be connected to a set of knowledge sources that share common area of interest</td>
</tr>
<tr>
<td>Broker tries to understands both the buyer and the seller, even if they use different language to describe the same things</td>
<td>Broker should be able to make an on-the-fly translations of used data-structures both ways client-service and service-client</td>
</tr>
</tbody>
</table>
Broker, because of his wide knowledge about the market, will only present offer from sellers that may meet needs of buyer. Broker should be able to choose knowledge sources that are able to supply with information that is desired, and should NOT contact those whose data will be abandoned for bandwidth saving.

The fact that businesses show up on and disappear from the market is transparent for client as long as he is using broker services. Only that concerns him are information that he is looking for. It should be possible for software broker to add and remove knowledge sources dynamically and this should NOT concern connected clients.

Broker adjust its activities to fit existing market and interact with business. Broker should NOT require already established knowledge sources to bent its access method, data format and any aspect to needs of the broker.

Broker can gather many request and offers and match them to create best situation both for buyers and sellers. Broker should gather queries from clients and try to cache results to improve overall flow performance.

<table>
<thead>
<tr>
<th>Real world Broker</th>
<th>Software Broker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Broker, because of his wide knowledge about the market, will only present offer from sellers that may meet needs of buyer</td>
<td>Broker should be able to choose knowledge sources that are able to supply with information that is desired, and should NOT contact those whose data will be abandoned for bandwidth saving.</td>
</tr>
<tr>
<td>The fact that businesses show up on and disappear from the market is transparent for client as long as he is using broker services. Only that concerns him are information that he is looking for.</td>
<td>It should be possible for software broker to add and remove knowledge sources dynamically and this should NOT concern connected clients.</td>
</tr>
<tr>
<td>Broker adjust its activities to fit existing market and interact with business.</td>
<td>Broker should NOT require already established knowledge sources to bent its access method, data format and any aspect to needs of the broker.</td>
</tr>
<tr>
<td>Broker can gather many request and offers and match them to create best situation both for buyers and sellers</td>
<td>Broker should gather queries from clients and try to cache results to improve overall flow performance.</td>
</tr>
</tbody>
</table>

Table 1: Mapping between roles of real and software broker

2.4 Data integration problem

Data integration is the problem of combining data residing at different sources and providing client with concise view of these data. Data integration system relies on a global schema and a set of knowledge resources. Knowledge resources contain real data and the schema is a reconciled model of underlying sources. There can be two approaches to the problem of defining a schema. First called global-as-view requires schema to be defined in terms of the knowledge sources. The other approach called local-as-view allows for defining the schema independently from the knowledge sources. However this also evolves defining local source-to-schema mappings for every knowledge source. The data exchange problem is related to the data integration problem in the sense that both problems are concerned with management of data.
stored in heterogeneous formats. Data exchange problem can be defined as taking information given in some sort of schema from the knowledge source and creating an instance of information in another scheme as accurately as it can be done. For a certain scheme and information there can exists many possible solutions, all satisfying constraints but some of them carrying excessive information. Efforts concentrate on finding a solution that has a homomorphism into any other solution and hence is the most general but also is the smallest substructure among all homomorphic images. This kind of problem is deliberated more in [19]. However for the need of Information Brokering this problem can be relaxed and redefined where there exists only one mapping between the source and target schemes. This kind of problem is closer to the problem of data integration on which we focus and define using formalism.

Obviously the main task when creating a data integration system is the definition of mappings between the data sources and global schema and this task can be defined in a formalized way. Also it is clear that data integration system main parts are:

- data sources, mappings and schema.

Let's define a data integration system $I$ in terms of a mentioned triple $\langle G, S, M \rangle$ using definition from [20].

- $G$ is the global schema expressed in language $L_G$ over an alphabet $A_G$. The alphabet assigns a symbol of specified type (i.e. relation if $G$ is relation, class if $G$ is object-oriented etc.)

- $S$ is the source schema expressed in its own language $L_S$ over an alphabet $A_G$

- $M$ is the mapping between $G$ and $S$ defined with a set of assertions: $q_S \rightarrow q_G \land q_S \leftarrow q_G$ where $q$ is a query with defined arity over specified schema. Queries are expressed in query languages build upon correspond-
ing alphabets. Intuitively an assertion denotes then a knowledge represented by a query in source query language can be also expressed in a query in language of the global schema and vice versa.

Provided definition is as general as it can be to catch all possible aspects of relations from those simple to very complex constraint-based. E.g. in semantics of a database schema corresponds to schema from the database. There is a simplifying assumption that all the data sources act in corresponding domains but in terms of Information Brokering this is not a need and indeed should not be assumed. This topic will be elaborated further in following chapters.

One of the most important aspect is the definition of correspondence between data sources and scheme. This definitions determine how the queries are processed in the system. As it was stated there are two difference approaches to defining schema: local-as-view (LAV) and the global-as-view (GAV). The LAV approach defines that all elements from the sources schema \( S \) has a corresponding query in global schema of the form: \( S \rightarrow q_G \). In other words LAV states that content of each source should be directly expressed in global schema. Idea favors those schemes that are well-established and stable in time. Also expandability of the system gains much from this approach as it is easy to append new source because the global schema does not need to be changes. Because of that the LAV in the contrary to GAV is well-suited for usage in information brokering system.

For mappings between sources and global schemes we can also define three different means of constraining between schemes:

- **Sound views** where we have an assertion is true that \( \forall x, s(x) \rightarrow q_G(x) \)
- For **complete views** we have: \( \forall x, q_G(x) \rightarrow s(x) \)
- For **exact views** \( \forall x, q_G(x) \leftrightarrow s(x) \)
For the information brokering sound view constrains is desirable as it allow us to express anything that can be found in the information source in terms of global schema. Any information tuple extension from the knowledge source satisfies the global schema.

2.5 Types of data sources

When building an application that will have functionality of a data integrator we need to build a schema and corresponding mappings for the information sources. It's worth noticing that data sources will differ in their access method, meta-description for data they provide, data structures they will use. Generally all knowledge sources can be put into one of three groups [21]:

- structured,
- semi-structured and
- unstructured data.

The main division was established with means of easiness for parsing received data and how queryable the data-source is. When designing such a system we have to face the fact, that besides our global schema some information sources may not be easily queryable using attributes from the global schema, in extreme case it may be even impossible to use them. The only benefit from this kind of sources is that gained knowledge may be applied to the mentioned schema. Additionally we can face another situation where the result is hardly parseable and applying the scheme is a difficult task. Worst case when both of the cases mentioned above occur.

Structured information sources with meta description, and can be queried with a specially designed language. Examples of such systems are relational databases with
its SQL, object-oriented databases and knowledge bases. Those were mentioned in the chapter 1.3.1.

Half-structured data that are available using forms on web pages. These allows to query in a constrained manner without getting the full information. E. g. of such application would be querying a travel agency database via its WWW homepage. Although many parameters can be provided (like for example country, number of stars of a hotel etc.) user don’t have a full freedom like in normal database and they cannot ask sophisticated queries as no dedicated query language is available.

Last type of information sources are unstructured data sources like the HTML webpages or UseNet groups which gives access to data understandable only by a human-being and that are less useful when it comes to machine processing. What is worth noticing is that not all knowledge sources returning results in HTML will be unstructured. If the site presents results in a concise way then using them can be manageable. Problem arises when we use search engines like Google or Bing that returns list of mainly not connected URL that share no common layout so NLP and machine learning techniques has to be used to extract any valuable information.

All of those types of source information can contain valuable information that we could be interested in. That is why none of them can be omitted when looking for certain information. A person that is looking for the information needs to spend considerable amount of time searching for proper sources of information that possibly can be queried. Afterward querying them takes also time because different information sources can have varying access interfaces, which user must first be familiar with. Even if we query only structured databases an understating of its meta-description model is needed. Moreover when gathering data from various knowledge sources following problems has to be faced:

- Duplication of data
- Information conflicts
• Partial information
• Uncertain information

Data can differ in meta-description but have the same meaning which has to be detected as this is different case from the data conflict when one have to decide which information is proper. When there is a large number of knowledge sources we can mark an information as proper if overwhelming majority points to a certain information. If the number of knowledge sources is little or the distribution is regular a piece of information is uncertain. One also has to be able to gather chunks of information and merge them with already established models. Generally the problem is to integrate different sources of information (in both access and data presentation), supplying general access interface to them and consistent form of merged results.
3 Agent-based solutions for information brokering

3.1 Rationale

The information brokering can be accomplished by a wide variety of methodologies but one of them seems to fit more than anything other. Distributed intelligence systems or by another name distributed multiagent systems fit into this task very good. At this point it seems that it would be beneficial to define what is an agent? Referring to literature canon [22] we will find that there is no common agreement on the definition of Agent. For our needs let us enumerate what characterizes agents and try to refer it to the brokering need and point out the benefits and potential problems:
• Agents are autonomous units that cannot be forced to do anything but may be willing to accomplish certain task – this in general is a pro for the system as agents can potentially do their best to find most appropriate information in the knowledge source they service. On the other hand since agents can be appended from outside and can work on their own they can also by a type of malicious software that will ruin search results.

• Agents are organized into societies and have a built int way of communicating with each other – ease when creating the brokering system. Agents are computational entities and all of the result have to be send to one central agent for aggregation so communication is a must and here it is included.

• Agents natural environment is a distributed environment and they have a mobility feature – not rarely interesting things are distributed in difference services in the Internet, thus knowledge sources also have distributed nature (if they were not distributed they would probably be grouped under one single scheme). Of course the bandwidth will grow but there still be applications where doing computations directly not remotely will make them much faster. Here the mobility feature will increase the applicability of the project.

• System almost never rely on a single agent or such agent can be quickly exchanged to a new one – when one of the agent representing certain knowledge source will be down the whole system will be working as long as at least one agent will be operational. This means that search results will be inferior but any result is better than no result at all.

Agents seems to be the most natural way of implementing discussed earlier techniques of brokering and data integration. Level of abstraction allows to discuss systems in the same terms as within the project documents creation phase and in the code development. Concepts from the agent-oriented programming especially can be useful when applying AI techniques. Agents may have an AI but it is not essential. This allows for great scalability and flexibility. When designing knowledge source agents one
has option of adopting agent to needs of operating the information source. The less the
information source if structured, the more agent will have to be intelligent and use
more sophisticated methods of text processing and information retrieval to full in the
global scheme with information. Additionally agents have option for advertising their
capabilities. This options is realized as a the YelloPages agent, a type of system agent
that main task is to register other agents and answer queries about agents. This option
allows broker to pick only those agents that are capable of fulfilling requested query
what allows for saving processing power. What more agents architecture allow for
easy QoS\textsuperscript{34}. The middle-agent assures that data that he stores follows the desired qual-
ity and standard policies. Data policy in this context is overall intention and direction
that agent follows in terms of data he receives from knowledge source agents. Com-
mon metrics used in data warehouses could be applied here. Additionally usage of the
broker gives additional layer of privacy and anonymity as all the data from external
services flows through the broker. What more broker can realize some security policy
and it is the only type of agent that client can trust since other knowledge source
agents should be free to append anytime they want not always being trustworthy. Bro-
ker agent has also the possibility to use knowledge from one source in cooperation
from other. This can happen by merging results and completing attributes or by send-
ing additional queries using information from certain knowledge sources to other KS-
s.

In the following chapters there will be presented analysis how an information bro-
kering system should be build upon the agent framework and what type of decision
still has to be made. Also already established solutions will be presented.

\textsuperscript{34} Quality of service
3.2 Types of intermediary agents

As already stated foundation of the computer-aided search system will be agents. In open environments when there is a need to connect services with customers we can distinguish three main types of agents [23]:

1. The service providers (P-Agents)
2. The service requester (R-Agents)
3. Middle-agent described earlier as the broker but generally middle-agent can be something else from the broker, what is discussed further.

The basic process of communication is as follows: the P-Agents advertise their abilities to the middle-agent which after registration will keep-track of them and their services. The R-Agent connects to middle-agent and requests for a certain service. The Middle-agent with its best knowledge searches through registered capabilities of P-Agents and connects the two. Connection can be done virtually by only exchanging information between the two, and middle-agent can be a service-added proxy in act of communication. Connection can also occur by giving list of addresses of P-Agents that meet the constraints to the R-Agent that initiated the search.

The middle-agent mediation service should allow for provisioning of the distributed services and need to have the following characteristics:

1. processing of agents capabilities
2. semantic interoperability between R and P agents
3. management of data if any acquired
4. distribution of queries and transaction to optimize system operation.

Processing of the capabilities can be done using already established capability description languages like LARKS\textsuperscript{35}, CDL\textsuperscript{36} or XML-based RDF(s) discussed earlier.

\textsuperscript{35} http://www.cs.cmu.edu/~softagents/larks.html
\textsuperscript{36} http://www.ai.aiai.ac.uk/project/oplan/cdl/index.html
Capabilities processing has to be done in real-time to find the most appropriate sources. Selection of any particular method of representing capabilities should also be somehow reconciled with the desired query interface. Simple keyword search will have other features than using type interference of complex data structures or reasoning engine for constraint matchmaking.

Semantic interoperability derives from defining mappings between data sources from the data integration problem. The difference is that because agents can possibly have injected intelligence and sometimes we cannot know the source schema from the very beginning AI methods can be used to deduct this mappings. Many techniques are available and very helpful can be description using ontologies since this can be a mutual hook. Unfortunately we are discussing here the vastness of Internet so the magnitude of heterogeneity is so big that this possibly is the hardest implementation part to achieve good results when building such a system.

Data management is connected to operation of the system which evolves efficient persistent storage and processing of capabilities and auxiliary data that can help to build up internal services for monitoring actions and overall performance of the service. Also the restrictions and security policy has to be preserved having in mind the delicate nature of data that is processed and involved privacy. Also if any AI is used historical data and possible feedback from the R-Agents will supply machine learning algorithms with essential data for further enhancement of the service.

The distribution of queries can help archive better performance. Because of the distributed nature some long-running task will be performed outside the build system and still reactiveness for clients should be preserved. Techniques like caching popular queries and merging some of them can bring substantial upturn in system operation.

When having a platform for exchanging data using agents there are certain further aspects that has to be taken into consideration. A policy for agent naming needs to be established. There is possibility that middle-agent will take part in the connecting dif-
ferent societies of agents and cross-registration is essential to achieve wider access to information.

This also causes that middle-agent should be able to agree on communication protocol with other agents that may differ. Of course there are some standards [24] (e.g. FIPA ACL, KQML) but as it was said everything is going to work in the Internet environment where richness of “standards” is significant and not always easy to reconcile.

The same may happen if middle-agent has a direct access to some kind of database where it need to understand underlying API. Again wide spectrum of standards exists: JDBC/ODBC, Java RMI and CORBA to name just a few.

Middle-agent may not only interact with other agents but also with a human-being by receiving queries and sending results (possibly using additional personal agents layer). Then a special manmachine interfaces has to be provided. This involves regular users that may want to browse the available domains and ontologies that may be covered by the system or processed data and all of this has to be restricted to applied security policies. Also system administrators may be interested in browsing the system. He may like to scan statistics and performance data for system tuning.

As it was stated before not always the middle-agent has to be the Broker. When considering privacy we can distinguish many roles that the middle-agent personate [25].

<table>
<thead>
<tr>
<th>Preferences initially known by</th>
<th>Capabilities initially known by</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P-Agent only</td>
</tr>
<tr>
<td>R-Agent</td>
<td>(Broadcaster)</td>
</tr>
<tr>
<td>R-Agent + middle-agent</td>
<td>Anonymizer</td>
</tr>
<tr>
<td>R-Agent + middle-agent + P-Agent</td>
<td>Blackboard</td>
</tr>
</tbody>
</table>

*Table 2: Collation of different roles that the middle-agent personate*
Table 2 shows that depending on how wide is the knowledge of a certain agent about other agents, the middle-agent plays a different role. We are only interested in the broker because he is suited best to fit into the Internet environment. Broker can proactively interface both P-Agent and R-Agent. Connection between them and all communication has to go through the broker. The broker is not restricted to any type of agents and this is particularly useful in an environment where all knowledge sources can enter and leave the society. This leaves much space for flexibility of communication and providing new resources to the clients. Moreover, the responsibility to integrate with the system lies on the clients and services. This may seem as a deny for earlier mentioned problem of convincing authors of web documents to follow the semantic web standards. The difference lies in the fact that if particular knowledge source seems interesting to be explored for the system P-Agents can be created as a layer on top of those services by the brokering system maintainer without affecting original author of the source.

Figure 1 depicts how a single transaction or request is realized in the system with a broker. As it's clear to see there is no direct connection between the requester and the provider. The action “0: (Un)Advertise service” can happen anytime while the system is running while other transaction is going on. The broker's task is to keep request served consistent. Here also outlines a benefit for the clients, its the broker thing to manage new sources append and removal.
3.3 Shallow analysis of a middle-agent-less system for information retrieval

So far all analysis were done having in mind that there exists a single or more distinguished points where all agents can turn to and ask for needed information or advertise that they share information. It is worth to mention that there are examples of systems that share the same goal as the brokering system (i.e. finding information from different sources) but try to pursue the goal using distinct assumptions. Still using the agent-oriented methodology but adding to it a P2P idea. The main idea is based on assumption that all nodes that take part in the system are equal. Without the global mediator that can forward queries further agents must cooperate to distribute queries among themselves. Agent assumes that it will eventually reach a point where the searched information can be found and it will be returned to the initiator. Effectiveness of such a system strongly relies on how good is the communication protocol that agents use. There are reports [26] where this approach combined with context-sensitive distributed search algorithms brings positive influence on system perfor-

---

37 Peer-to-Peer
mance. However there are drawbacks of such an approach. Preserving any type of security policy seems to be impossible. Anonymity of sent queries also cannot be introduced since when the query is propagating any agent on the path needs to have access to the query to be able to tell if he has the needed information. Also this approach involves that a knowledge source maintainer to develop corresponding agent that can be used in the system. This is a kind of problem discussed earlier why semantic-web is still not popular.
4 Implementation of information brokering system for search of scientific data.

4.1 Aim of the project

Purpose of this chapter is creating an agent-based brokering framework with purpose of real services for clients searching for scientific information. In the past few years, mainly thanks to the Internet and WWW, scientific community gained possibility to reach a wide audience with their research results. With ease of publishing comes noise and difficulty of finding the proper information. Additionally with increasing complexity of projects where many of them exceed boundaries of a single discipline comes need of interdisciplinary cooperation and knowledge sharing. Ability to find an expert for a certain area (not rarely other than the one we work in) may be the key for new discoveries and problems definition. For the purpose of gathering data about scientist many project are run e. g. Google Scholar, ArnetMiner, CiteSeerX. DBLP to
name just a few. Their databases are rich and mostly kept up-to-date but it does not meant that they coincide. Some keep data that others do not and so to have the most accurate information many services should be visited. The main goal is to provide a client with a transparent access to the chosen knowledge sources with shallow erasure of duplicates and attributes merging. Similar project were conducted lately [27] but it focused on using single knowledge source (i. e. DBLP) for building researcher profiles.

4.2 **GAIA methodology for agent analysis and design**

Agents can be viewed as another layer of abstraction on top of other programming techniques like OOP\(^{38}\). Thus it requires different approach than the classical methods of designing software like for example classes and sequence diagrams. For this purpose GAIA methodology was created [28], which has been tailored to the analysis and design of agent-based systems. It is meant for real-world big scale applications. Methodology assumes that agents are heterogeneous and work together for the common goal. No assumptions were made about the hosting platform and environment. Those assumptions make GAIA fit very well into created project.

GAIA uses a top-down approach for analysis and design. Starting with a general idea of the system breaking it down into pieces to consider the implementation detailed enough to begin implementation. Basic GAIA concepts are divided into two groups: abstract and concrete. Abstract entities are used to conceptually model the system, in contrast concrete entities will have a corresponding counterparts in the implementation.

\(^{38}\) Object-oriented programming
4.2.1 Analysis

The main role of the analysis phase is to create understanding of the system. GAIA approaches the topic by trying to specify an organization view of modeled problem. Set of roles with relationships and interaction patterns is base of the GAIA analysis.

The Roles models

First step in analysis is finding roles in the system. A role should correspond to some entity that may act independently in a form of an individual, a department within organization or an organization itself. In the brokering system there can be enumerated three clear roles:

- Broker who acts as a middle man and is the center point of the system,
- Searcher who prospects assigned knowledge sources,
- Client is an entity seeking for a certain information.

Every roles consists of four attributes: responsibilities, permissions, activities and protocols.

Responsibilities are divided into liveness properties that describe what is the actual task of the role and safety properties are invariants describing things that role should maintain. Identifying responsibilities for the brokering system:

- Client
  ◦ whenever there is a request it should gather input data and process it to the system
  ◦ after every request client should wait for the response and present the data
  ◦ when response was presented to the user, client should repeat the process

Livness expression:

\[ \text{Client} = (\text{AwaitQuery} \circ \text{SendToBroker} \circ \text{AwaitResponse} \circ \text{ReceiveResults} \circ \text{PresentResponse})^\omega \]
Where $A \circ B$ means that $A$ precedes $B$ and $\omega$ means infinite repetition. Activities are underlined and the other are protocols (activities that involved contacting with other

### Safety:

- Broker
  - at any time should be able to receive query
  - after receiving query should process it to appropriate search agents
  - should wait for one or more response
  - if more than one response was received should merge them
  - after receiving all responses should send results to the Client

Livness expression:

**Broker** = ($\text{WaitForQuery} \ || \ \text{HandlesQuery})^\omega$

**HandlesQuery** = ($\text{ReceiveQuery} \circ \text{RedistributeQuery} \circ ((\text{ReceiveResponse}^+) \circ [\text{MergeResults}] \circ \text{SendBackResults})$).

Where $A^+$ means that $A$ occurs one or more times, $[A]$ means that $A$ is optional, $A \ || \ B$ means $A$ and $B$ interleaved.

### Safety:

- 50 -

---

**Figure 2: Class diagram for Dialect concept**

---

[Diagram of Dialect concept with class notation and attributes.]

- **Dialect**
  - Entity
    - multiValue : Boolean
  - Attribute
    - type : AttributeType

- <<enumeration>>
  - AttributeType
    - NAME
    - PHONE_NUMBER
    - ADDRESS
    - GRAD
    - URL
    - EMAIL
    - DATE_TIME
    - INTEGER
    - FLOAT
    - STRING

---
searchQuery = empty → searchResult = nil
len(searchResults) → len(partialSearchResults)

• Searcher
  ○ should receive process queries all time
  ○ should send back results when finished

Livness expression:
Searcher = (WaitForQuery || (ReceiveQuery \o MakeSearch \o SendResponse))\o

Safety:
isAdvertisedWithItsCapabilities = true

Associated set of permissions are basically resources that can be used to fulfill its
responsibilities. In terms of brokering system resources are abilities to read and gener-
ate information. Identifying permissions for the brokering system roles:

• Client
  ○ WRITES searchQuery – for purpose of introducing new search task to the
    system given by a human-being
  ○ READS searchResults – for purpose of presenting the results to the end
    user

• Searcher
  ○创造了 partialSearchResults – for purpose of transferring results from
    external knowledge source to the system.
  ○ READS searchQuery – for purpose of generating appropriate query to the
    external resource

• Broker
  ○ READS searcherAbilities – for purpose of choosing appropriate searchers
  ○ READ partialSearchResults – for gathering results from Searchers
  ○ UPDATES searchResults – for purpose of combining results.
**BROKER**

<table>
<thead>
<tr>
<th>Description</th>
<th>The role is responsible for forwarding queries to appropriate knowledge sources searchers, merging and returning results.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocols</td>
<td>ReceiveQuery, ReceiveResponse, SendBackResults, RedistributeQuery</td>
</tr>
<tr>
<td>Activities</td>
<td>MergeResults, WaitForQuery</td>
</tr>
<tr>
<td>Permissions</td>
<td>READS searcherAbilities, partialSearchResults, UPDATES searchResults</td>
</tr>
<tr>
<td>Responsibilities</td>
<td>Broker = (WaitForQuery</td>
</tr>
<tr>
<td>Safety</td>
<td>searchQuery = empty → searchResult = nil, len(searchResults) → len(partialSearchResults)</td>
</tr>
</tbody>
</table>

*Table 3: Schema for role Broker*

**CLIENT**

<table>
<thead>
<tr>
<th>Description</th>
<th>Initiates search task in the system and visualizes results.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocols</td>
<td>SendToBroker, ReceiveResults</td>
</tr>
<tr>
<td>Activities</td>
<td>AwaitQuery, AwaitResponse, PresentResponse</td>
</tr>
<tr>
<td>Permissions</td>
<td>WRITES searchQuery, READS searchResults</td>
</tr>
<tr>
<td>Responsibilities</td>
<td>(AwaitQuery * SendToBroker * AwaitResponse * ReceiveResults * PresentResponse)(^*)</td>
</tr>
<tr>
<td>Safety</td>
<td>true</td>
</tr>
</tbody>
</table>

*Table 4: Schema for role Client*

**SEARCHER**

<table>
<thead>
<tr>
<th>Description</th>
<th>Performs search in associated knowledge source on behalf on client</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocols</td>
<td>ReceiveQuery, SendResponse</td>
</tr>
<tr>
<td>Activities</td>
<td>WaitForQuery, MakeSearch</td>
</tr>
<tr>
<td>Permissions</td>
<td>CREATES partialSearchResults, READS searchQuery</td>
</tr>
<tr>
<td>Responsibilities</td>
<td>(WaitForQuery</td>
</tr>
<tr>
<td>Safety</td>
<td>isAdvertisedWithItsCapabilities = true</td>
</tr>
</tbody>
</table>

*Table 5: Schema for role Searcher*
The Interaction model

A detailed description of protocols defined in previous chapter with focus on exchange of information:

- **SendToBroker → ReceiveQuery**
  - purpose: sending description of searched data from user
  - initiator: Client
  - responder: Broker
  - inputs: Query
  - outputs: None
  - processing: None

- **RedistributeQuery → ReceiveQuery → SendResponse**
  - purpose: giving Searcher a task
  - initiator: Broker
  - responder: Searcher
  - inputs: Query
  - outputs: List of results
  - processing: Merging results from different Searchers

- **SendBackResults → ReceiveResults**
  - purpose: Delivery of results to Client
  - initiator: Broker
  - responder: Client
  - inputs: List of merged results
4.2.2 Design

Objective of the design in most software engineering techniques leads to a transformation of the formal specification created during analysis to form which can be easily implemented in the source code. In GAIA however this rule is not applicable. GAIA tries to create a design model with enough low-level details to which traditional software techniques (especially including OOP) can be applied to.

The agent model

In the agent model there should be created a mapping of roles to agents. In most perfect case there is one-to-one mapping, but it is possible for an agent to include more than one role. In case of the project system there actually exists a one-to-one mapping because of rather low complexity of interactions and well defined needs.

<table>
<thead>
<tr>
<th>Role name</th>
<th>Agent name</th>
<th>Multiplicity</th>
<th>Creation time</th>
<th>Termination time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Broker</td>
<td>Broker</td>
<td>1</td>
<td>System start</td>
<td>System termination</td>
</tr>
<tr>
<td>Client</td>
<td>ClientAgent</td>
<td>*</td>
<td>At client request</td>
<td>When search results are received</td>
</tr>
<tr>
<td>Searcher</td>
<td>SearchAgent</td>
<td>1+</td>
<td>Anytime when system is up</td>
<td>Anytime</td>
</tr>
</tbody>
</table>

*Table 6: Agent model information*
The services model

The services model is build to identify duties that are assigned to each agent and what are the properties

<table>
<thead>
<tr>
<th>Service</th>
<th>Input</th>
<th>Output</th>
<th>Pre-Conditions</th>
<th>Post-Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BROKER</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>To take a task</td>
<td>Query text</td>
<td>Task</td>
<td>Query not nil</td>
<td>Task queue not empty</td>
</tr>
<tr>
<td>Assign a task</td>
<td>Task</td>
<td>None</td>
<td>Task queue not empty and number of Searchers not empty</td>
<td>Task queue^{n-1} is task queue^{n-1}</td>
</tr>
<tr>
<td>Receive results</td>
<td>Results</td>
<td>None</td>
<td>Task for results</td>
<td>Task.results not empty and amount incremented by 1</td>
</tr>
<tr>
<td>Send back results</td>
<td>Task</td>
<td>List of results</td>
<td>All results ready</td>
<td>Removed task</td>
</tr>
<tr>
<td>Results merging</td>
<td>List of list of results</td>
<td>List of result</td>
<td>List has more then 1 list</td>
<td>Number of results from list of list smaller or equal to merged results</td>
</tr>
<tr>
<td><strong>CLIENT</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Receive query from user</td>
<td>String of chars</td>
<td>A query</td>
<td>Text not empty</td>
<td>None</td>
</tr>
<tr>
<td>Pass query</td>
<td>Query</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Receive list of results</td>
<td>List of results</td>
<td>Visualization of the results</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td><strong>SEARCHER</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Receive task</td>
<td>Task</td>
<td>None</td>
<td>None</td>
<td>Task queue not empty</td>
</tr>
<tr>
<td>Send back results</td>
<td>List of results</td>
<td>None</td>
<td>None</td>
<td>Task queue length decreased by 1</td>
</tr>
</tbody>
</table>

*Table 7: Services model for brokering system*
The acquaintance model

The acquaintance model is basically a directed graph of agents as vertices and communications lines as arcs. In this case no changes from the model presented in [25] is observed. Thus as the acquaintance model the Figure 1 can be used.

4.3 Used technologies

The choice of technology plays important role in this project. On the one hand system is agent-based and there exists standards of inter data exchange (e. g. FIPA-ACL) so it should be fairly easy to extend it with new knowledge source searchers. On the other hand agents are still a niche for most programmers and extorting this paradigm may be a problem. System will be only valuable if new sources can be added with ease. Also as it was shown in chapter 1.3 the variety of used technologies is huge and it would be very beneficial to pick a technology that can be easily integrated with existing databases and have set of libraries that will work with the Internet standards. Taking it all into consideration Scala programming language was chosen with its integrated Actors framework. Main rationale for this choice is that Scala's primary implementation is JVM\(^{39}\) based. Thus it is fully operational with Java programs and libraries. Since Java is one of the most popular programming languages with many libraries this fulfills mentioned needs. Additionally from the programmers point of view Scala is superior to Java because it has much more expression power in its syntax which results in much shorter code. Scala also comes with integrated Actors framework which was borrowed from Erlang programming language. Actors are alternative to mutexes and monitors approach to concurrency. It's main idea is that every computational entity (in most cases a thread) uses messaging system to communicate with other entities what fits very well into agent-based approach.

\(^{39}\) Java Virtual Machine
4.4 **Agents based on Actors**

The system built has three logical layer:

1. the actors framework as a foundation for the whole platform,
2. the brokering layer which be a skeleton for creating data-integration services.
3. the Passim\textsuperscript{40} layer where using brokering skeleton a few already established Internet knowledge sources about scientists works will be exposed to clients.

As it was mentioned, Scala allows for using the Actors paradigm when approaching concurrency topics. Basically an actor has the ability to send messages to other actors and receive messages which can be processed at any time. Those two functionalities allows Actors to be a base for the lowest layer number one.

From a technical point of view Actor is represented a java thread (sometimes shared with other agents) and a synchronized queue where messages are put. Handling a messages involves taking a message from the queue and matching it to a set of registered types. If match is found an associated routine is executed. Idea of and actor is very similar to the concept of an agent and thus can be used as a base for implementing actors.

What actors lack comparing to agents but is needed in this project are:

- a globally unique identifier,
- possibility of advertising

For identifying an agent a name of simple type of string of characters will be used. For purpose of advertising an YellowPages agent is created with the start of agent platform. He will register agents under their name and allow for querying for other agents with a custom predicate. Besides mentioned problems other features are already to be used. Especially important for agents, a communication channel in form of messages. Messages here are just custom java objects with one constraint put for sake of the work will be integrated.
agents. All of them must derive from common Message class. This class has an automatically generated UUID\textsuperscript{41} which will allow for recognizing conversations. Whenever a complex messages interchange occurs messages has to contains the initially generated UUID. This will allow for completing different intertwined long running tasks. Although agent should be able to work in an open network environment (preferably using plain-text to communicate) java objects are used instead. This creates two major problems:

1. How to communicate Agents that are located on different machines?
2. How to communicate agents that do not use the JVM?

Answer for question number one is already here, since agents subsystem contains interfaces for remote agents that uses TCP/IP for communication. The only thing that is needed is the IP address and the port of the machine where desired agent is located. Answer to question number two is strongly related to the first question. Because an agent infrastructure is used it is quite easy to create additional agent that will act as a proxy and will translate Java objects to whatever format is needed. With remote agents in front of any society of agents all limitations are gone.

Layer one is very simple because Actors framework does most of the work and hides complexity of servicing a communication channel.

A base algorithm for all agents (here and build further on top of this) is:

1. Initialize all resources
2. Send message to platform manager that initialization is done.
3. Send message to Yellow-pages, registering itself
4. Act

\textsuperscript{41} Universally unique identifier
In the process of developing layer number one (the agent base) following messages were indemnified and created:

<table>
<thead>
<tr>
<th>Message</th>
<th>Content</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InitDone</td>
<td>Agent source</td>
<td>Send to management platform when agent is ready to enter the environment</td>
</tr>
<tr>
<td>AgentQueryMsg</td>
<td>Agent source; filtering predicate</td>
<td>Sent to Yellow-pages agent when looking for a certain agent. Contains a predicate used when filtering all agents.</td>
</tr>
<tr>
<td>AgentListMsg</td>
<td>Token; list of agents</td>
<td>Send in response to AgentQueryMsg, list of agents that meet the predicate.</td>
</tr>
<tr>
<td>AgentRegistrationRequest</td>
<td>Agent to be registered;</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sent to Yellow-pages after init, to register and be queryable for other agents.</td>
</tr>
<tr>
<td>AgentRegisteredACK</td>
<td>Token</td>
<td>Sent by Yellow-pages agent in response to AgentRegistrationRequest when the Agent was registered.</td>
</tr>
</tbody>
</table>

*Table 8: List of layer one messages*

### 4.5 Brokering layer

#### 4.5.1 Task management

The brokering layer is designed to allow for easy creation for data-integration-like services. The main aim was to create such an architecture that would allow for strong separation between the semantic content of the exchanged data and the process of bro-
kering. Thus the central middle-agent broker is created in a generic manner and is not to be changed.

To fulfill this aim the broker is supported by two abstract notions:

1. the task manager

2. the dialect.

Task manager allows for implementing a policy of execution for any type of work (in particular the search task). With interface consisting of four methods: createNewTask, cancelTask, startTaskExecution, getTaskContact user of the brokering framework is freed from supplying methods of communication with clients as all of this is done by generic broker behavior and usage of Yellow-pages. This generic behavior is an almost one-to-one mapping of some of the protocols and activities from the analysis described in chapter 4.2.1. After being registered in the Yellow-pages an infinite loop of receiving queries and passing results to the clients occurs. One thing that was moved from a Broker to another Agent is the task execution. Logically an Executor is part of the broker, on behalf of which contacts search agents and merges the results. How it merges the results is up to the user of the layer. This movement was decided to allow different policies of getting the answers. As only the broker can contact the client user can possibly assign many executors to one task. If any executor is finished then partial results can be obtained. This allows for searching in live mode when if anything is available then it is shown to the client and end user. Searching through the internet can be a time-consuming task so results can be available after a day or two.

Another enhancement, this time for the knowledge sources maintainers, are how the SearchAgent base class is build. After registering with YellowPages as usually it starts to receive queries. If any query is received an asynchronous call to search routine is made that should be averted by the maintainer. The query is expressed as free text and an additional category (discussed further in next paragraphs). This seems to be very poor in terms of restricting one's query but there is a rationale for such decision. An alternative implementation of query interface was done at first using conjunc-
tion of key-values attributes. Results were very miserable. Most of the time getting none or very few result made using the system very unsatisfactory so this idea was dropped.

Leaving as much space for interpretation for search agents as it is possible allows him to do his best while searching. Behind this idea stands a strong belief that unnecessary results can always be filtered out before sending to clients by the broker, but if some valuable results are not shown at all is an unacceptable situation. It's in the best faith in the searcher that he will use this query as good as it can be. In exchange for this constraints are put on the format of results where it has to comply to the used dialect (in chapter 2.4 called a global schema). This allows client to pick up information that he is interested in even if the query might been too loose.

**4.5.2 The dialect**

The dialect role is to establish a common structure for exchange of data between different modules and agents. The main assumption is that all parts of the system share common semantic understanding. This is vital as data can be processed, changed, merged and filtered out in many points before it reaches a client. If comprehension is not covering on different levels the results may be handicapped. Problems with results include not true statements about entities coming from different knowledge sources. Problem faced here comes from the assumption used by most non-relational databases (as those are possibly types of some knowledge sources underneath) that presented information may be outdated or sometimes incomplete or even wrong. As it was described in chapter 1.3.4 about NoSQL databases the data eventually will be correct but this has to be taken into consideration. To avoid those shortcomings merging step requires to filter out data that may be irrelevant and leave only those that has a higher probability of being true. Techniques of NLP and statistics can come with help. Sometimes simple comparison of frequency of an attribute can decide what should be filter out. For example if we identify that from three different knowledge sources we have
information about the same entity but one of them send a discordant attribute comparing to other two we may believe that it is wrong. If such a situation occurs another question arise what about the other attributes? Should they be abandoned also believing that the whole entity is corrupted. There is no universal answer. Unfortunately incompatible attributes problem is not always that the knowledge source is sending corrupted data. Sometimes the discordant attribute may be another form of the same thing e. g. name and surname written in reverse order. Clearly there is no universal solution and relevant algorithms must be used to discover meaning of the attribute one-by-one.

The model of dialect used in this project in some way corresponds to the relational model. Dialects classes were depicted in figure 2. It consists of set of Entities. Entity role is to aggregate linked attributes that pertain to the same object. This is mostly the same as tables in relational databases. Difference lies in removal of primary and foreign keys. This allows for creating instanced of entities only partially filled in and not bound to other entities. Rationale for this is that although every knowledge source operate the same schema it may not be able to deliver some of the attributes at all. Actually dialect should be designed that way to be as wide as it can be covering more attributes that any of the knowledge sources attached in future will be capable of delivering. As a result search outcome should be more complete than using any of the sources alone.

4.5.3 Combining search results

Disabling keys would decrease usefulness of the system as there is at first sight no option for associating entities that may somehow be related. As it was stated in previous paragraphs knowledge sources yield information (mostly in plain-text format) that can have diversity of forms. Liability of connecting information from different entities rely on the merge algorithm that should be adjusted to the type of information that are processed. For example university names that may come in different languages need
different algorithms than mentioned name-surname problem where multilingualism does occur very rarely.

A dialect is complemented by specially defined enumeration named AttributeType and a multiValaue flag telling if the attribute has many values (e. g. workplaces of a person). It's purpose is to give a hint to the merging algorithm in what kind of form the information should be expected from the knowledge sources. It is also a constraint for the searcher agent how should he format information received from the served knowledge service. Besides normal types derived straight from programming languages like integer, float and string also some types that generate additional information about how the information may be formatted. Some of them are general: name (in sense that someone would use it with this-is), address, phone number, e-mail or URL. Those specific types allow for using NLP algorithms e. g. merging name and surname. Some knowledge sources may give it in the format of a tuple (Name, Surname), some may reverse it and other can only give first letter of the name followed by surname. Whatever format is used there can be made an effort to create an efficient algorithm for this task. Similarly a URL can contain parameters, queries, paths etc. With mentioned problem of multilingual university names a homepage URL is a great place to start merging but only after leaving only the host part. Many languages has ready methods for extracting parts of the URL including the host part. Without this type information but for instance using only String qualifier it would be hard to develop methods of merging.

4.5.4 Queries

Queries are constructed using the dialect entities and a text sentence (or keyword). Such query should be interpreted as intention of getting any information about given entity that has anything to do with the associated text sentence. As it was already mentioned this gives search agent possibility to interpret the sequence whatever he wants to get the widest results (maybe sometimes too rich). There is another rationale for
constructing queries like that. Many of the available search engines on the WWW has a free-text search interfaces so giving too many constraints in query could possibly results in impossibility to get any information from the knowledge source. Even if the underlying search source if more like a relational database with rich querying interface there is high possibility that given sequence describes a database attribute that has an index on and will allow for fast search.

Dialect has also another function. It help identifying agents that are capable of performing requested search. Search agents when registering in the Yellow-pages they expose what kind of entities they are capable of searching. It is a simple list, that is used by executors when receiving a query. Since query consists one entity it is fairly easy to pick those agents

During designing the second layer following messages are sent between agents:

<table>
<thead>
<tr>
<th>Message</th>
<th>Content</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QueryMessage</td>
<td>Source agent; token; query text; entity described by the query</td>
<td>Is used for sending queries to the broker and further to search agents</td>
</tr>
<tr>
<td>SearchResultMessage</td>
<td>Source agent; token; list of SingleResults</td>
<td>Is sent to the broker and to client with search results expressed in the dialect</td>
</tr>
<tr>
<td>InformationMessage</td>
<td>Source; token; list of InformationSnips</td>
<td>Messages used for visualization purposes.</td>
</tr>
<tr>
<td>InformationSnip</td>
<td>None, is abstract</td>
<td>Base class for all results returned for the visualization</td>
</tr>
<tr>
<td>ResultDescription</td>
<td>Source; list of SingleResults</td>
<td>Used for holding results by executors</td>
</tr>
<tr>
<td>SingleResult</td>
<td>Attributes with values</td>
<td>Storing single search entry. Value may be a sequence.</td>
</tr>
</tbody>
</table>

Table 9: List of messages in layer two
4.6 \textbf{Passim layer}

Passim layer is built upon two previously described layers. That's why most work concentrates on assuring concrete implementations of already defined abstract notions. This chapter is also a guide on how to create a brokering system for another area of interest. Here focus is aimed at supplying scientific knowledge.

For creating a top-layer brokering system four things has to be initiated:

1. dialect i. e. set of entities with attributes that will have corresponding types. The types enumeration may be extended and changed to meet requirements of the area of interest freely,

2. the search executors that will on behalf of the broker perform communication with the searcher agents, gather the results and apply merging and filtering algorithms,

3. task management object for the broker that will comply underneath hardware used. In case of machine clustering additional remote agents will be needed. This will also allow for optimizing number of agents in the system due to number of cores availability on the CPU.

4. at least one search agent which basically involves writing capabilities i. e. a list of entities that such agent will service and one search method that will return list of SingleResult. Rest will be taken care by the rest of the framework.

4.6.1 \textbf{Passim dialect}

Best place to start the implementation is the dialect part. Defining dialect requires an in-depth knowledge about the described area. Let us gather some facts about the researchers society:

- In the area of research and development a researcher and his publications are placed in center.
• The more citations researcher has the more recognized he is.

• Researcher besides the number of citations are often classified by the h-index\(^{42}\) which classifies productivity and impact on others of the researcher.

• Researchers often works in groups publishing together. Information about co-workers can be particularly interesting.

• Value of publications are also measured by number of citations.

• Year of publication is important as only few publications published are defining a new area and are timeless. Majority of publications concerns topics that are at that time popular.

• Universities are still the place where most scientists perform research. Although in past decades the wealthiest companies has been building very successful R&D centers around the world still best researchers are working in universities.

• Information about universities is also important for the companies that would like to establish a cooperation, thus they should be included in this project.

<table>
<thead>
<tr>
<th>Entity name</th>
<th>Attribute name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PERSON</td>
<td>personName</td>
<td>NAME</td>
<td>Describes name and surname of the person.</td>
</tr>
<tr>
<td></td>
<td>personYearsOld</td>
<td>INTEGER</td>
<td>How old the person is.</td>
</tr>
<tr>
<td></td>
<td>personPhoneNumber</td>
<td>PHONE_NUMBER</td>
<td>Contact phone number.</td>
</tr>
<tr>
<td></td>
<td>personEmail</td>
<td>EMAIL</td>
<td>Contact e-mail.</td>
</tr>
<tr>
<td></td>
<td>personHomepage</td>
<td>URL</td>
<td>WWW homepage of the person.</td>
</tr>
<tr>
<td></td>
<td>personHomeUniversity</td>
<td>NAME</td>
<td>Name of the university that the person is associated with.</td>
</tr>
</tbody>
</table>

\(^{42}\) Letter h comes from the name of the index author Jorge E. Hirsch
<table>
<thead>
<tr>
<th>Entity name</th>
<th>Attribute name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>personHIndex</td>
<td>INTEGER</td>
<td>Hirsch index of the person</td>
<td></td>
</tr>
<tr>
<td>personPicture</td>
<td>URL</td>
<td>Picture of the person located in the internet</td>
<td></td>
</tr>
<tr>
<td>PersonCitationNum</td>
<td>INTEGER</td>
<td>Number of citations by a person.</td>
<td></td>
</tr>
<tr>
<td>publicationAuthor</td>
<td>NAME, multiValue</td>
<td>Publications authors list.</td>
<td></td>
</tr>
<tr>
<td>publicationTitle</td>
<td>STRING</td>
<td>A name of the publication.</td>
<td></td>
</tr>
<tr>
<td>publicationArea</td>
<td>STRING, multiValue</td>
<td>List of areas this publication in placed in.</td>
<td></td>
</tr>
<tr>
<td>publicationYear</td>
<td>DATE_TIME</td>
<td>When publication was published.</td>
<td></td>
</tr>
<tr>
<td>publicationIsCitedBy</td>
<td>STRING, multiValue</td>
<td>Other publications that cites this publication.</td>
<td></td>
</tr>
<tr>
<td>publicationCitations Number</td>
<td>INTEGER</td>
<td>Sum of citations from other publications.</td>
<td></td>
</tr>
<tr>
<td>universityName</td>
<td>STRING</td>
<td>Name of the university.</td>
<td></td>
</tr>
<tr>
<td>universityFoundation Year</td>
<td>DATE_TIME</td>
<td>When university was established.</td>
<td></td>
</tr>
<tr>
<td>universityCountry</td>
<td>STRING</td>
<td>Name of the country where university is located.</td>
<td></td>
</tr>
<tr>
<td>universityHomeCity</td>
<td>ADDRESS</td>
<td>Address of the university more or less precise</td>
<td></td>
</tr>
<tr>
<td>universityHomepage</td>
<td>URL</td>
<td>URL address of the university homepage</td>
<td></td>
</tr>
</tbody>
</table>
In the third layer four entities were defined that are listen in the table number 10. All of the entities are related to each other in that sens that for every concrete entity other three could also be initiated. E. g. for PERSON we could have list of publications that this person is author of creating PUBLICATION entity. Probably this PERSON has a home university so we could initiate UNIVERSITY entity. Both UNIVERSITY and the PERSON can have their HOMEPAGE. It is easy to see that this gives much possibilities when searching for certain information. We could ask for all publications about “Datamining”. Gather all the authors names and send queries about their data to find out what is the average age of researcher in topics of datamining. Of course this is not much more than using JOIN on tables in the relational databases using keys. As it was said before entity has no keys but possibly some attributes may be used in a key-like way. Figure 3 show what are the relations between entities in Passim dialect. It's pretty clear to see that the center entity and probably the most important one is the PERSON entity. This information should probably be utilized while designing the user interface.
4.6.2 Searchers

After defining the dialect, second thing that should be analyzed are the knowledge sources that are available for us. This is the moment to rethink the dialect again. There may be some additional attributes that were not taken into consideration in first iteration but are available to get from the knowledge source and are useful for our aims. The same for the attributes types that may need to be extended to allow for optimized analysis of the data. This is also a place to analyze what kind of interfaces are available from the knowledge sources. Are those free text search, or maybe structured SQL-like queries? In the initial stage of design of this project, the main assumption was that the queries will be send to the search agents using a set of attribute-value

![Diagram](image-url)

*Figure 3: Entities and attributes with "JOIN" ability*
pairs. This assumption was made before available knowledge sources were found and analyzed. Result were rather poor, while most queries were delivering little or none answers it was a real problem to put those queries through the knowledge source. The real cause of this was how search agent communicated with the knowledge source. Most of them were using a free-text sentence query. This imposed on the search agent to query many times the knowledge source and then filter out some parts. This generated a lot of network traffic and because search agent had little knowledge about what really is the person looking for most of the search results were very bad. This caused a huge change in the way queries are asked to already described free-text search. Of course now many of the presented results are useless but in sense of user experience this is a better way of system to respond. Good example of similar problems is when comparing two search engines like Google and Wolfram Alpha. If we're looking for something that can be found in both search engines in Google our task is to look through many snippets that it yielded and find the interesting information. In Wolfram Alpha this will come down to entering the proper query. System tries to find out what user meant and filters out information. Probably presuming from looking at popularity of both engines it is harder and much more frustrating for the user to try to modify the query than to view hundreds of snippets.

DBLP

First knowledge source that is integrated with the system is the DBLP Computer Science Bibliography\(^{43}\) database. It is a joint project of Schloss Dagstuhl-Leibniz Center for Informatics and the University of Trier. DBLP simply allows for searching people and their publications. The whole database is available for download as a compressed XML file with a given DTD schema. Using obtained from the Passim-Project code repository script\(^{44}\) allowed to push DBLP content into MongoDB database. MongoDB is representative of the NoSQL type of databases and has no constraining

\(^{43}\) Homepage address: http://www.informatik.uni-trier.de/~ley/db/

\(^{44}\) Script author is Marek Kozłowski from Institute of Computer Science from Warsaw University of Technology

- 70 -
schema. Although because data is inserted by script there is logical grouping of the knowledge. Two important collections from the project are “Documents” and “People”. Collection “People” consists of author's names and unique id's. Corresponding id's are placed in the “Documents” collections what allows for associating data from the two collections. What is very convenient is the fact that if there is more than one author id's are placed in a list. No need for breaking the text and trying to match surnames and names. What else can be found in the DBLP is the year of publications and sometimes ISBN and URL pointing to some additional information about the publication. To sum up information from DBLP consists of:

1. List of authors
2. Publication title
3. ISBN
4. Publication year

ARNETMINER

Arnetminer⁴⁵ is an academic researcher social network search engine that is extracting information from the WWW and using NLP techniques try to build profiles of people from the scientific world. Arnetminer has a RESTful service for searching experts, publications and conferences. The number of attributes is quite big and only some were chooses to be used because for needs of experiments this would overcome all other knowledge sources limiting this project to be a dumb pipe. Answers from the RESTful service comes in the JSON format. Attributes delivered by the Arnetminer platform:

1. personName,
2. personEmail,
3. personPhoneNumber,
4. personHIndex,
5. personCitationNum,
6. personHomepage,
7. personPicture,
8. publicationTitle,
9. publicationAuthor,
10. publicationCitationsNumber

**CiteSeerX**

CiteSeerX[^46] is an evolving scientific literature digital library and search engine that focuses primarily on the literature in computer and information science. Unfortunately data is available for download but not for performing remote search. Fortunately the HTML code of the webpage is very regular and enough simple to parse it. Using regular expressions and mocking a HTTP server by the search agent (simple GET does not work) we can use web fronted programmatically. Small obstacle is that the page has a pagination and multiple calls to the server are needed to get sufficient number of results. Parts of the page consist of tags for:

1. publication year
2. snippet
3. authors
4. abstract

which are translated into search results.

[^46]: [http://citeseerx.ist.psu.edu](http://citeseerx.ist.psu.edu)
Dbpedia

Dbpedia is a community project to extract structured information from Wikipedia and to surround it with ontologies. This allow for asking sophisticated queries against information that was placed on Wikipedia. This information can be extracted by connecting to a SPARQL endpoint and send queries. SPARQL is a similar to SQL languages for asking about facts that are classified using ontologies. Dbpedia is added for a special reason. It does not contain information about the publications and people (at least contains very little) that it is not going to list PUBLICATION and PERSON it's capabilities. Although Wikipedia and this implies that also Dbpedia has a great base of universities. This will be used as a use case when we have two different groups of search agents that can search completely different things. Based on a given country name all universities from that country will be listed and send back. Information that can be extracted from the results:

1. university name
2. established year
3. home city
4. homepage
5. country (this is obvious).

Because SPARQL is very much as SQL we can image doing very similar operations like in relational database. Since we can write a query that will find universities by country we can also do it by entering home city or any other attribute that is available. This only involves writing another query and sending it. Unfortunately we don't have possibility to guest what keyword was enter by the user (i.e. city or country) so the more queries we will write the more connections to the endpoint will have to be done.
4.6.3 Executors

Executors are nothing more than an alter ego of Broker specialized in one kind of Entity searching. Executor after being created by the Broker will take the query and search for search agents, connect to them, gather the results and pass the back. Since in this project there are three meaningful entities (WEBSITE is excluded) there are three executors.

The simplest executor is the one in charge of the UNIVERSITY search. The gathered data is merged using homepage addresses by using only the host part. If any of the attributes is missing then it is skipped. The only one that cannot be skipped is a name of the University. Executor also tries to find if data is corrupted. Especially year which can be passed in different forms and the one desirable is containing four digits.

Second executor is responsible for searching PUBLICATIONS. All of the results are put into a map using title as the key trimmed and lowercase. If publication with certain title is already in the map then only attributes are added to the created result object.

Third executor takes care of PERSON search. This is the most complicated scenario because answers are also gathered by the key which is person name. But because it may contain many abbreviations it makes it harder to guess that this is the same person. Another thing is that when search for a certain person we also get his co-workers so results should be sorted to get surnames closes to the keyword at the top.

4.6.4 Resource managing

Since there is no specific platform nor specific requirements, the task manager simply creates a task and assigns a new executor based on the query type. When the results are ready executor is garbage collected. Every new search task involves creating a thread for the client and an executor task. Also a thread per knowledge source is created. Because those are agent-based threads we can leave it to the Scala platform to
maintain thread pools. If we would have any special needs it would be possible to control it like for example need of clustering.

### 4.7 User front-end

For demonstration purpose a web interface was created. The presentation layer is independent from the underlying agent framework. It was created using the Play Framework\(^7\) that is dedicated for creating lightweight MVC web applications using Java or Scala programming languages.

![Passim search](image)

**Figure 4: The main view**

The GUI is run locally under port 9000 invoked from the Internet browser. Figure 4 shows the main view where queries can be entered.

In the main view there is a text box where queries should be entered and a drop-down list with all entities that were defined in the program's dialect. Clicking on the search button submits entered data to the server and the search begins.

\(^7\) [http://www.playframework.org/](http://www.playframework.org/)
Search process can be observed in the terminal. Figure 5 shows sample output while querying for publications about java. Also process of registering in YellowPages and agents interactions are visible.

Figure 5: Server console while performing search
Figure 6: Sample publications search

Search for "Divesh Srivastava" (Results number: 207)

Divesh Srivastava

E-mail: divesh-srivastava@research.att.com

List of publications:

- "Rule Ordering in Bottom-Up Fingerprint Evaluation of Logic Programs" from year 1990 by Raghu Ramakrishnan, Divesh, Raghu Ramakrishnan, Divesh Srivastava, S. Sudarshan
- "Space Optimization in the Bottom-Up Evaluation of Logic Programs" from year 1990 by S. Sudarshan, Divesh Srivastava, Raghu Ramakrishnan, Jeffrey P. Naugle
- "Rule ordering in bottom-up fingerprint evaluation of logic programs" from year 1990 by Raghu Ramakrishnan, Divesh Srivastava, S. Sudarshan
- "Magic Sets and Bottom-Up Evaluation of Well-Founded Models" from year 1991 by David B. Knap, Peter J. Stuckey, Divesh Srivastava
- "Date Model and Query Evaluation in Global Information Systems" from year 1991 by Alot Y. Levy, Divesh Srivastava, Thomas Kirk
- "Patching Constraint Systems" from year 1992 by Divesh Srivastava, Raghu Ramakrishnan
- "Correcting the Search in Bottom-Up Evaluation" from year 1992 by Raghu Ramakrishnan, Divesh Srivastava, S. Sudarshan
- "CORAL: Correct, Rollbacks and Logic" from year 1992 by Raghu Ramakrishnan, Divesh Srivastava, S. Sudarshan
- "Query Restricted Bottom-up Evaluation of Normal Logic Programs" from year 1992 by David Kemp, Peter J. Stuckey, Divesh Srivastava
- "Efficient Bottom-Up Evaluation Of Logic Programs" from year 1992 by Raghu Ramakrishnan, Divesh Srivastava, S. Sudarshan
- "Exploring Program Execution in Deductive Systems" from year 1993 by Tian An, Raghu Ramakrishnan, William G. Roth, Pavlos Seshia, Divesh Srivastava
- "A Framework and Induction in Context-Free Query Languages with Linear Arithmetic Constraints" from year 1993 by Divesh Srivastava

Figure 7: Sample person search top results

- 77 -
Figure 6 shows result of mentioned query for publications on java. Results are sorted using citations number. If citations are not available result is put to the end. Names of the publications authors can be clicked to invoke a person search.

Authors search contains authors list that are related to the query on the top and all co-publishing authors below. If there are available also H-index and citations number is given. At the bottom of each entry list of knowledge sources is given. Figures 7 and 8 depicts person top and bottom search results.

![Sample person search bottom results](image)

The last available option is searching for universities by country. Figure 9 depicts results for searching universities in France. Single results may contain names of the university (in different languages if given), address of the university homepage and
year of establishment. Results are sorted using given year starting from the oldest. If it is not available result is put to the end.

Figure 9: Sample university search results
Summary and Conclusion

The purpose of this work was to propose a solution for searching information from distributed knowledge sources. An analysis of current technologies used for storing and exchanging information was made. This gave an in-depth image of use cases for the developed system and let to avoid some of the problems when constructing a general solution for information exchange. Analysis of the brokering concept was made as a base for proposing a data integration problem solution corresponding to the stated problem. Literature for already proposed solutions was analyzed. The brokering idea was implemented using agent-oriented methodology which gave the assumed flexibility in attaching additional knowledge sources. This, build on top of the agents, brokering solution makes it easier to create a domain-focused application that exposes different knowledge sources to its clients with a unified interface. As an example application, the scientific domain model for brokering was build and implemented using the alternative Actors approach to multitasking. This allows the solution to be scalable horizontally and be very efficient in terms of use of the computational power of today's computers. A concrete implementation has been provided for the scientific domain utilizes most widely used databases like ArnetMiner, CiteSeerX and DBLP. Also as an example of the system abilities, DBPedia was used to demonstrate a knowledge
source choosing scenario. This gave in outcome a possibility to query for information about people, their publications and universities. For the purpose of results presentation a simple user web interface was created which allows for querying and browsing the data.

Results delivered by the application correspond to needs of a researcher searching for potential co-workers or information on publications from a given area. Publication search gives results that take into consideration the importance of the work (i.e., citations number) which allows for finding the essential literature in given area. Using authors names from the found publications an application user can find other people working on the same topic and other publications from the author. As an addition a university search by country was added to test search using agents having different abilities.

Proposed solution could strongly benefit from introducing machine learning techniques and end-user feedback. There are cases where an improper data is provided what is obvious for a human-being but not for the computer. Some problems like detecting different forms of second name need non-trivial algorithms. This would allow for evaluation of the data and keeping it coherent across single results. Also creating a dedicated database for building user profiles, as all other part are somehow connected, could bring additional benefits. It would result in less connections to the Internet, allows for caching of the query results and evaluating results in terms of truth. This would also decrease problem of being dependent from the external sources. This can be a real problem since most of the services tries to limit traffic from single addresses.

The project complements the theoretical elaboration included in this work. It is a good starting point for creating a whole spectrum of knowledge brokering applications in different domains. Those that are already created will benefit from new knowledge sources attached that may be created by external developers. Some of the important aspect of such a system were not covered in this work like for example security. But because of the agent base all already established solutions in areas that were not de-
scribed here can be incorporated easily having in mind the flexibility of agent-oriented approach and the rich spectrum of publications on this topic.
Appendix A – User guide

Installation

Compilation and execution of the program is controlled by the SBT\textsuperscript{48} program. SBT can be downloaded from the Internet\textsuperscript{49} and used for free. It takes care about downloading all dependencies from the Internet, including the Scala compiler, using the same repositories as Maven build system for Java. SBT is controlled from the terminal. Java and Javac has to be put into the system PATH for the SBT to run. SBT should also be put into the PATH for ease of use.

SBT run script (that is packed with the sbt.jar execution file) should be edited and the value for “Xmx” should be set at least to “1500M” to avoid the “Java.lang.OutOfMemoryError: PermGen space” error occurring too often.

Running

While being in the root directory of the project sbt command should be run. It will load the project files and start the SBT console. All dependencies will be resolved and can take a while when run for the first time. Afterward play command should be run that will invoke web container libraries. Next run command will invoke the server to

\textsuperscript{48} Simple build tool
\textsuperscript{49} https://github.com/harrah/xsbt/wiki/Getting-Started-Setup
listen on port 9000. Entering address localhost:9000 in the web browser will invoke
compilation process and redirect to the search page http://localhost:9000/search
where queries can be entered.

CAUTION
When platform is started before any interaction is made user MUST invoke the localhost:9000 address as it initializes the agent platform.

Configuring MongoDB

One of the agents uses stored locally DBLP knowledge source. DBLP was copied
into MongoDB database. The database image can be found on the enclosed DVD disc
in the data/db folder. Before running any query the MongoDB daemon should be run
with the parameters “--dbpath={path_to_database_image}/data/db. If standard config-
uration is used it should start listening on port 27017. The same port will be used by
the client agent to connect.
### Tables index

Table 1: Mapping between roles of real and software broker ............................................. 33
Table 2: Collation of different roles that the middle-agent personate .......................... 44
Table 3: Schema for role Broker .................................................................................. 53
Table 4: Schema for role Client .................................................................................. 53
Table 5: Schema for role Searcher ............................................................................. 53
Table 6: Agent model information .............................................................................. 55
Table 7: Services model for brokering system ............................................................. 56
Table 8: List of layer one messages ............................................................................ 60
Table 9: List of messages in layer two ....................................................................... 66
Table 10: List of entities and attributes defined in the third layer ............................. 69
Figures Index

Figure 1: Message flow in single transaction.................................................................46
Figure 2: Class diagram for Dialect concept.................................................................50
Figure 3: Entities and attributes with "JOIN" ability......................................................70
Figure 4: The main view...............................................................................................76
Figure 5: Server console while performing search......................................................77
Figure 6: Sample publications search..........................................................................78
Figure 7: Sample person search top results.................................................................78
Figure 8: Sample person search bottom results............................................................79
Figure 9: Sample university search results...................................................................80
Bibliography

2: Shannon, Claude, A mathematical theory of communications, 1948
5: Gantz, John; Reinsel, David , Extracting Value from Chaos, EMC Corporation, 2011
6: Siegler, MG, Eric Schmidt: Every 2 Days We Create As Much Information As We Did Up To 2003, http://techcrunch.com/2010/08/04/schmidt-data/,
7: Springer, Springer Open Access - How to order, http://www.springer.com/open+access/open+choice?SGWID=0-40359-12-683307-0,
10: Strauch, Christof, NoSQL Databases,
11: DeCandia, Giuseppe; Hastorun, Deniz; Jampani, Madan; Kakulapati, Gunavardhan; Lakshman, Avinash; Pilchin, Alex; Sivasubramanian, Swaminathan; Vosshall, Peter; Vogels, Werner, Dynamo: Amazon’s Highly Available Key-value Store, , 2007
12: Chang, Fay; Dean, Jeffrey; Ghemawat, Sanjay; Hsieh, Wilson C.; Wallach, Deborah A.; Burrows, Mike; Chandra, Tushar; Fikes, Andrew; Gruber, Robert E., Bigtable: A Distributed Storage System for Structured Data, , 2006
13: Lith, Adam; Mattsso, Jakob, Investigating storage solutions for large data, CHALMERS UNIVERSITY OF TECHNOLOGY, Department of Computer Science and Engineering, 2010
15: W3C, Web services glossary, http://www.w3.org/TR/ws-gloss/,
16: W3C, Semantic Web Standards, http://www.w3.org/2001/sw/wiki/Main_Page,
18: Association of Independent Information Professionals, Knowledge Delivered,
http://www.aiip.org/
19: Fagin, Ronald; Koalitis, Phokion G.; Popa, Lucian, Data Exchange: Getting to the Core, IBM Almaden Research Center, 2005
20: Lenzerini, Maurizio, Data Integration: A Theoretical Perspective, , 2002
21: Martin, David; Oohama, Hiroki; Moran, Douglas; Cheyer, Adam, Information Brokering in an Agent Architecture, , 1997
23: Klusch, Matthias; Sycara, Katia , Brokering and matchmaking for coordination of agent societies: a survey, Springer-Verlag, 2001
25: Decker, Keith; Sycara, Katia; Williamson, Mike, Middle-Agents for the Internet, , 1997
26: Zhang, Haizheng; Croft, Bruce; Levine, Brian; Lesser, Victor, A Multi-agent Approach for Peer-to-Peer based Information Retrieval System, , 2004
27: Adnan, Sadaf; Tahir, Amal; Basharat, Amna; Cesare, Sergio de, An Agent-Oriented Architecture for Researcher Profiling and Association Using Semantic Web Technologies, Springer-Verlag, 2010