An Architecture for Visual Patent Analysis

Von der Fakultät Informatik, Elektrotechnik und Informationstechnik der Universität Stuttgart zur Erlangung der Würde eines Doktors der Naturwissenschaften (Dr. rer. nat.) genehmigte Abhandlung

Vorgelegt von

Mark Oliver Giereth

aus Ludwigsburg

Hauptberichter:	Prof. Dr. Thomas Ertl
Mitberichter:	Prof. Dr. Leo Wanner
Tag der mündlichen Prüfung:	22. Oktober 2012

Institut für Visualisierung und Interaktive Systeme der Universität Stuttgart

2012

Abstract

Patents are of great importance for national and international economies, since they have a high impact on the development of products, trade, and research. Therefore, patents are analyzed by multiple interest groups for different purposes, for example for getting technical details, for observing competitors, or for detecting technological trends. Patent analysis typically includes the following steps: formulating a patent query, submitting it to one or more patent data services, merging and analyzing the results for getting an overview, selecting relevant patents for detailed analysis, continuing with refining the query based on the gained insights until the results are satisfying.

In general, patent analysis is a complex, time and knowledge intensive process, because of multiple application domains, data sources, and legal systems but also because of the intentionally used abstract expressions in patents and the huge amount of patent data. Therefore, this thesis investigates the research question of how patent analysis can be supported by the field of visual analytics which combines visualization, human-computer interaction, data analysis, and data management.

A contribution of this thesis is a general architecture for visual patent analysis which is based on a semantic representation model. The model can be accessed from analysis and presentation components, enriched with analysis results, published and reused. Analysis and presentation components are loosely coupled and exchangeable based a shared terminology defined by ontologies. A second contribution is an ontology for patent metadata. It provides an integrated semantic representation of the major patent metadata aspects and allows metadata restrictions to be added to a semantic search. As a third contribution, this thesis describes new visualization techniques, for example a treemap techniques for visualizing patent portfolios and co-classification relations and a graph overlay based technique for visualizing dependencies between text parts based on semantic relations. Additionally, this thesis contributes to the field of data security by describing a new encryption based method for secure sharing of semantic models to a known group of recipients. The main idea of this method is to only encrypt sensitive information while keeping all non-sensitive information publicly readable (partial encryption). This method allows the usage of open infrastructures for the exchange of sensitive semantic models.

The adequacy of the proposed architecture has been validated by three prototypes for different use cases. It has been shown that the architecture is flexible and extensible. A first evaluation with patent experts suggested that visual analytics is a promising approach for improving patent analysis.

Kurzfassung

Patente sind von großem Interesse für die nationale und internationale Wirtschaft, da sie einen hohen Einfluss auf die Entwicklung von Produkten, Handel und Forschung haben. Daher werden Patente von verschiedenen Interessensgruppen mit unterschiedlicher Zielsetzung analysiert, beispielsweise, um technische Details über Erfindungen zu erhalten, um Forschungsaktivitäten von Mitbewerbern zu beobachten oder um neue Technologietrends zu erkennen.

Patentanalyse beinhaltet für gewöhnlich folgende Schritte: Formulierung einer Patentsuchanfrage, Übermittlung der Anfrage zu ein oder mehreren Patentdatendiensten, Zusammenführen und Analyse der Ergebnisse zur Überblicksgewinnung, Auswahl relevanter Patente zur Detailanalyse, Fortführung mit verfeinerter Suchanfrage auf Basis gewonnener Erkenntnisse bis das Ergebnis zufriedenstellend ist. Auf Grund unterschiedlicher Anwendungsgebiete, Datenquellen, Rechtssysteme sowie der abstrakten Sprache, die in Patenten verwendet wird, ist Patentanalyse im Allgemeinen ein komplexer, zeit- und wissensintensiver Prozess. Diese Arbeit befasst sich mit der Fragestellung, wie Patentanalyse durch das Forschungsgebiet Visual Analytics, welches Visualisierung, Mensch-Maschine Interaktion, Datenanalyse und Datenmanagement kombiniert, unterstützt werden kann.

Ein Beitrag dieser Arbeit ist eine Architektur für visuelle Patentanalyse, die auf einem semantischen Repräsentationsmodel basiert. Auf dieses Model kann sowohl von Analyse- als auch Präsentationskomponenten zugegriffen werden. Es kann um Analyseergebnisse erweitert, veröffentlicht und wiederverwendet werden. Analyse- und Präsentationskomponenten verwenden eine gemeinsame auf Basis von Ontologien definierte Terminologie. Sie sind dadurch lose gekoppelt und austauschbar. Ein zweiter Beitrag der Arbeit ist eine Patentmetadatenontologie. Diese ermöglicht eine integrierte semantische Repräsentation der wichtigsten Patentmetadaten und ermöglicht so die Einbeziehung von Metadatenrestriktionen in semantische Suchanfragen. Ein weiterer Beitrag der Arbeit sind neue Visualisierungstechnologien für Patentmetadaten, beispielsweise Techniken zur Darstellung von Patentportfolios sowie Co-Klassifizierungsbeziehungen auf Basis von Treemaps oder Techniken zur Darstellung semantischer Beziehungen zwischen Textteilen eines Patents basierend auf überlagerten Graphen. Ein letzter Beitrag der Arbeit ist eine Methode zur sicheren Veröffentlichung semantischer Modelle für eine bekannte Gruppe von Empfängern. Die Grundidee ist eine partielle Verschlüsselung, das heißt es werden nur sensitive Informationen verschlüsselt, während nicht-sensitive Information im Klartext lesbar bleiben. Diese Methode erlaubt die Verwendung offener und potentiell unsicherer Infrastrukturen für den Austausch sensitiver semantischer Modelle.

Die Angemessenheit der vorgestellten Architektur und Methoden wurde in drei Prototypen für unterschiedliche Anwendungsfälle validiert. Es wurde dadurch gezeigt, dass die Architektur flexibel und erweiterbar ist. Eine Evaluation mit Patentexperten legt zudem nahe, dass Visual Analytics ein vielversprechender Ansatz zu Unterstützung bei Patentanalysen ist.

Danksagung

Während meiner Zeit am Institut für Visualisierung und interaktive Systeme (VIS) und am Institut für intelligente Systeme (IIS) habe ich viele Menschen kennen lernen dürfen, die zum Gelingen dieser Arbeit beigetragen haben, und bei denen ich mich an dieser Stelle herzlich bedanken möchte. Meinem Doktorvater Prof. Dr. Thomas Ertl danke ich, dass er mir die Möglichkeit gegeben hat an diesem spannenden interdisziplinären Thema zu forschen, für die hervorragende Forschungsumgebung, die er am VIS geschaffen hat, und dafür, dass er meine Forschung stets mit großem Interesse und Geduld begleitet hat. Ganz besonders danke ich Prof. Dr. Leo Wanner für sein großes Engagement über die vielen Jahre hinweg, in denen er mich seit meiner Diplomarbeit als Mentor begleitet und gefördert hat. Er gab mir viele Ideen und Ratschläge für diese Arbeit. Bedanken möchte ich mich auch bei Prof. Dr. Egbert Lehmann für die interessante Zeit am IIS und die philosophischen Einblicke in die Welt der künstlichen Intelligenz.

Ich möchte mich bei allen Kollegen, die mich während meiner Jahre an der Universität Stuttgart begleitet haben, für die angenehme Arbeitsatmosphäre und die vielen anregenden Diskussionen zu den verschiedensten Themengebieten bedanken. Ganz besonders danke ich Dr. Martin Rotard, Steffen Koch und Harald Bosch für die vielen kreativen Sitzungen, die zahlreichen Nacht und Wochenendaktionen, um unsere Prototypen und Veröffentlichungen fertigzustellen sowie für die gute Motivation zum richtigen Zeitpunkt. Mit ihnen zusammen entstanden viele Bausteine dieser Arbeit. Mein Dank gilt auch allen Kollegen aus dem PATExpert Projekt, insbesondere Sören Brügmann, der das Patentthema im Rahmen seiner Diplomarbeit als erster am IIS eingebracht hat. Meinen früheren IIS Kollegen Dr. Bernd Bohnet, Dr. Stefan Klatt und Dr. Dietmar Lippold möchte ich für die spannende Zeit und gute Zusammenarbeit danken. Wir hatten viel Spaß zusammen.

Außerdem möchte ich mich bei allen von mir betreuten Studenten für ihre engagierte Arbeit bedanken, insbesondere bei Steffen Koch, Ying Qian, Ingo Schröck, Achim Stäbler, Dianqiang Yang und Yang Zhou. Mit ihnen zusammen entstanden einige Komponenten dieser Arbeit.

Mein größter Dank jedoch gilt meiner Frau Mandy, die mich über die lange Zeit meiner Promotion unterstützt hat, auch und besonders in den schwierigen Phasen.

Contents

CHAPTER 1

Introduction

1.1 Motivation

Patents are of great importance for national and international economies, since they have a high impact on the development of products, trade, and research. A patent is an exclusive right granted by a legal authority for an invention that fulfills certain conditions. It provides protection for the invention to its owner for a limited period of time in exchange for public disclosure of the invention. In particular it gives the owner the right to prevent others from commercializing the invention.

This combination of disclosure and protection is intended to promote global innovation and technological progress. It also makes patents a valuable source for technological knowledge which, for example, can be used as an indicator for the development of technical fields and markets. This is the reason why patents are analyzed by different interest groups for different purposes. The following list gives some typical examples of patent analysis scenarios.

- Researchers and inventors being interested in disclosed technical details perform deep content analyzes
- Patent examiners analyzing the state-of-the-art in a technical field in order to evaluate the novelty, inventive step and applicability of a patent application during the grant process
- Attorneys searching for infringements of patents for their clients or performing

freedom-to-operate analyses to make sure that a product can be commercialized without infringing other patents

- Managers monitoring the patenting activity of competitors in order to plan their own research and development activities or identifying new entrants in a technology area as future competitors
- Analysts forecasting the development of technical trends, for example by looking for patents that describe new elementary technologies or that combine existing technologies in a new way
- Economists analyzing the patents of a company for asset valuation during mergers and acquisitions

In general, patent documents contain a textual description of the invention which can be supplemented by figures, formulas or charts. Patent documents further define the scope of protection by a list of claims. Since patents are part of a legal system they also contain administrative data which are also referred to as *patent metadata*. Examples are information about dates, applicants, inventors, legal status, citations, etc. All data related to patents are subsumed by the term *patent information*.

In general, there are two primary ways of analyzing patent information: qualitative and quantitative. Qualitative methods focus on the content of patent documents, whereas quantitative methods are based on statistical processing of patent metadata, for example a trend over time, geographic distribution, major players, occurring key terms, etc. In practice both methods are often used together.

Common to most patent analysis tasks is the search for relevant patent documents in a comprehensive patent database. A typical scenario is that users first formulate a query, submit it to a patent database, make quantitative analyses on the results for getting an overview and then select individual patent documents for further qualitative analysis. Both, quantitative and qualitative analysis may gain new insights for refining or expanding the query. Therefore patent analysis can be seen as an iterative process.

1.2 Problem Statement

The previous examples gave an idea of the complexity of patent analyses. The aim therefore is to support patent analysts with appropriate tools that are integrated in a suitable software architecture. Several problems have to be addressed in this regard. One challenge is the large amount of patent data. According to the records of the PATSTAT database, the total number of patent applications worldwide is about 63 million and the total stock of granted patents worldwide is about 7 million [49]. These numbers increase each year. In 2011 for example approximately 1.97 million new patent applications were filed worldwide and the total number of new granted patents was around 777,600 [281, 282]. Data scalability therefore is an important issue.

Since patents are managed by national and regional authorities, their storage is distributed among several patent databases. Additionally, national patent laws and regulations introduce heterogeneous data, in particular concerning information about the legal status of patents. Because of this, the ability of merging heterogeneous data into a unified data model is another important issue.

In order to achieve a maximum coverage for a patent the language used in patents usually is abstract. Also the used terminology used can differ from the technical terms commonly used in the domain. In order to retrieve, classify, interpret or assess patent information, the analyst must hypothesize how patent contents and metadata reflect the semantics. This analysis process can be supported by natural language processing and machine learning methods, for example for text segmentation, information extraction, term association, cluster generation, or information mapping. The results of an analysis – regardless if it has been performed manually or automatically – can be seen as *semantic annotations* that enrich the original information. Annotated patent information can be shared with other experts or it can be published as *added values*.

1.3 Contributions

The contribution of this thesis is in four areas: First, a semantic representation that integrates patent contents and metadata with results of automatic analyses and manual annotations; Second, a new method for secure sharing of sensitive annotations together with public patent information in open environments; Third, interactive visualizations for major patent information aspects; Fourth, a general architecture for visual patent analysis that integrates the developed techniques and models.

1.3.1 Semantic Representation

Since patent metadata are import filter criteria in almost all patent search requests, an integration of patent metadata into a semantic representation formalism is of great importance. A contribution in this area is the *Patent Metadata Ontology (PMO)* [133] as part of an overall representation framework for patent information [130, 274] developed within the PATExpert research project [24]. PMO is divided into several ontology modules, each providing the terminology for a specific patent metadata aspect, such as bibliographic data, patent families, or patent classification. Since there are a large number of patents, data scalability is an important issue. Due to the fact that patents are managed by national and regional authorities based on different patent laws and regulations, heterogeneous metadata formats from multiple data sources had to be considered.

A second contribution is a linking schema for associating semantic annotations, e.g. the results of patent analyses, with patent contents. This linking schema is based on a general path language for locating content parts in XML encoded patent documents.

From a technological point of view, Semantic Web standards are used for a semantic representing of patent metadata and annotations, since they provide a general and formal model for knowledge representation at large scale, basic reasoning capabilities and allow for merging heterogeneous data from diverse sources.

1.3.2 Secure Sharing of Semantic Annotations

Raw patent information is intended to be public on the one hand, but semantic annotations can be sensitive information or part of a business model on the other hand. Therefore, this thesis investigates an approach that combines public information and restricted information in a single semantic model. The contribution is a new method for applying partial encryption on semantic networks. The method is called *Partial RDF Encryption (PRE)* [122, 124]. With this method it is possible to encrypt sensitive fragments of semantic network represented as Resource Description Framework (RDF) graph (see section 2.2) for a set of known recipients while keeping all non-sensitive data publicly readable. The result of this encryption method is an RDF-compliant self-describing semantic network containing encrypted data, encryption metadata, and plain text data.

1.3.3 Interactive Patent Visualizations

The aim of interactive patent visualizations is to support users in performing quantitative and qualitative patent analyses by providing more natural and intuitive user interfaces compared to textual presentations, different views on patent metadata, dynamic filters, overviews and details-on-demand.

The contributions of this thesis are new interactive visualizations for the results of quantitative and qualitative patent analyses. The following visualization approaches will be described in detail: graphs for bibliographic data and their relations, timeline and matrix visualizations for temporal data, treemap based visualizations for presenting the classificatory distribution of patent sets and their co-classification relations, and a graph based technique for visualizing the results of semantic analyses at content level, such as extracted is-a or part-of relationships, as overlay on top of the patent content.

Furthermore, this thesis describes a dynamic query-based mapping of instances of a semantic model onto hierarchy and graph visualizations. The mentioned visualization techniques have been published by the author in [131, 125, 134].

1.3.4 Architecture for Visual Patent Analysis

Based on the semantic patent information model, visualization techniques and partial RDF encryption method a general architecture for visual patent analysis has been developed. This architecture is implemented by three prototypes each having different objectives:

- The *PatLens* prototype investigates the integration of patent information into the current analysis context of a user. The main idea of this prototype is to automatically augment a web page with appropriate semantic annotations and to visualize them as graph overlay on demand. PatLens is an application of *semantic lenses* which have been published by Rotard et al. in [211]. In the PatLens prototype the current web page is scanned for patent numbers. If a patent number is found in an underlying semantic patent information model, then the web page is augmented with additional information from the model.

- The *PatWiki* prototype is another visual front-end for the PATExpert services. The focus of this prototype is on manual creation and sharing of semantic annotations.

A wiki system is used for semantic annotation, comments, discussions, notification, and full-text search. The population of the wiki system done automatically based on the patent search requests of the users. This Wiki approach originally has been published in [127].

- The *PatViz* prototype was built as the primary visual front-end for different patent retrieval and analysis services developed in the PATExpert project. The focus of this prototype is on the integrated visualization of different query languages and on the tight integration of result visualizations. The approach has first been published by the author in [128] and was later extended for better supporting iterative query refinement by Koch et al. [169, 170].

1.4 Structure of the Thesis

The thesis is structured as follows. Chapter 2 gives a brief overview of the foundations for this thesis which are the patent domain, Semantic Web standards, data security, information visualization, visual analytics and software architecture. Based on Semantic Web standards chapter 3 describes the *Patent Metadata Ontology (PMO)* for semantic representation of the major patent metadata aspects. Chapter 4 describes the *Partial RDF Encryption (PRE)* method for securing sensitive information contained in semantic models. After having described the model aspects, the chapters 5 and 6 propose new visualization techniques for patent metadata. The focus of chapter 5 is on techniques for visualizing annotations that refer to a patent document (resource level), whereas the focus of chapter 6 is on techniques for visualizing annotations that refer to concrete content parts (content level). Chapter 7 describes semantic lenses which use implicit queries to compare the content of a webpage with an underlying semantic repository. It highlights the matches and augments the website so that the user can interactively view this additional information on demand without losing the current context. In chapter 8 an approach for sharing semantic models using a wiki is described. Chapter 9 discusses the derived architecture for visual patent analysis and reviews the prototypes from an architectural point of view. Finally, chapter 10 summarizes the results of this thesis and gives an outlook to future research.

CHAPTER 2

Foundations

This chapter gives an overview of the state-of-the-art of the fields covered in this thesis. In particular it introduces the patent domain, Semantic Web standards, data security, visual analytics and software architecture.

2.1 Patents

A patent system is founded on national and international patent law and conventions. This section gives an outline of the international patent system based on the information provided by the World Intellectual Property Organization (WIPO) in the *Intellectual Property Handbook* [280]. National patent systems, for example the German or the U.S. patent system can be different in some aspects.

A patent, generally, is an exclusive right granted for an invention. It provides protection for the invention to the owner of the patent for a limited period. In return for patent protection, a patent owner has to publicly disclose information on the invention in order to enrich the total body of technical knowledge in the world. Therefore patents provide valuable information and inspiration for future inventions.

An invention protected by a patent cannot be commercially made, used, distributed or sold without the consent of owner of the patent. Patent rights are enforced in a court. A court holds the authority to stop patent infringement but also can declare a patent invalid upon a successful challenge by a third party. A patent owner has the right to decide who may use the patented invention within the protection period. The patent owner may give permission to, or license, other parties to use the invention on mutually agreed terms. The owner may also sell the right to the invention to someone else, who will then become the new owner of the patent. Once a patent expires the owner no longer holds exclusive rights to the invention. Then the invention enters the public domain and becomes available to commercial exploitation by others.

In general, an invention must fulfill certain conditions to be protected by a patent.

1. It must be of practical use.
2. It must show an element of novelty not known in the body of existing knowledge in its technical field (prior art).
3. The invention must show an inventive step.
4. Its subject matter must be accepted as patentable under law. In many countries, scientific theories, mathematical methods, plant or animal varieties, discoveries of natural substances, commercial methods, or methods for medical treatment are generally not patentable.

A patent is granted by a national patent office or by a regional office that does the work for a number of countries, for example the European Patent Office (EPO) or the African Intellectual Property Organization (OAPI). Further, the Patent Cooperation Treaty (PCT) administered by the WIPO provides for the filing of a single international patent application which has the same effect as national applications filed in the designated countries. An applicant seeking protection may file one application and request protection in as many signatory states as needed.

The patent process, in general, starts with the filing of a patent application at a patent office. As soon as a patent application is received, it is assigned a unique *application number*. After an examination request is received, the patent office will search for relevant prior art documents. A corresponding search report is made available to the applicant after a certain period of time and usually is published together with the patent application. Without a request by the applicant, the result of the examination is published together with the application usually after 18 months. Under certain circumstances, the search report may be published separately. Each patent publication is identified by its *publication number*. Since the patent document may change its state during the patent process, there may be multiple publications of one patent application. The application number always remains the same. This is why the application number can be used to associate different publications of the same invention.

After the publication the applicant has a certain period of time to request examination of the patent in order to obtain a granted patent. For German patents this period is seven years. The office will execute requests with a certain time limit and decide on whether to grant the patent or not. In case of a granted patent, a new document is published that formally discloses the invention and its grant. Granted patents are valid for a certain period, may be prolonged upon request and expire after a maximum of 20 years.

A patent document generally contains the title of the invention, as well as an indication of its technical field. It must include the background and a description of the invention, in clear language and enough detail that an individual with an average understanding of the field could use or reproduce the invention. Such descriptions are usually accompanied by visual materials such as drawings, plans, or diagrams to better describe the invention. The application also contains various claims, that is, information which determines the extent of protection granted by the patent.

The term *patent information* commonly includes three aspects.

1. **Technical information** contained in the content of a patent document.
2. **Bibliographic information** for example, data regarding the applicant, inventor, classification symbols, date of filing, representative, country of filing, document number, etc.
3. **Legal information** related to the legal and procedural status, for example the stages of a patent file, when the next fee will be due, etc.

The next section gives an overview of the major bibliographic data aspects.

2.1.1 Bibliographic Data

Patent Numbers

For published patent documents there exists an internationally agreed standard, the WIPO Standard ST.1 [285], which defines the data elements that have to be available to be able to uniquely identify a patent document. Historically a published patent document was considered to be uniquely identified by the combination of the office code, the publication number and the kind-of-document code, for example *US1234567A*. But with the growing availability of corrected documents this combination is no longer unique [285]. The WIPO standard ST.1 therefore defines that the minimum data elements for uniquely identifying all types of patent documents have to include:

1. Office code according to WIPO Standard ST.3 [289]
2. Publication number according to WIPO Standard ST.6 [286]
3. Kind-of-document code according to WIPO Standard ST.16 [284]
4. Date of publication of the document as provided by the Internationally agreed Numbers for the Identification of bibliographic Data (INID) [288] (40) through (48).

In contrast, there is no standardization for the identification of patent applications. As a consequence there exist various identification schemes for patent application numbers. For example the United States Patent and Trademark Office (USPTO) uses application numbers with a 2-digit series number followed by a slash and then followed by a 6-digit serial number, for example 10/007521. For patents filed at the European Patent Office (EPO) the application number is an identifier with 2 characters and 11-digits where the first 4 digits indicate the filing year of the application, for example EP20010402190.

Further there is the problem that different numbering schemes may be used at the same time to reference the same patent application. This in particular occurs when retrieving data from different sources as shown in the following example.

Example: When retrieving the bibliographic data from the Open Patent Service (OPS) [21] the corresponding response fragment contains two encodings ('epodoc' and 'original'). Both use a different identification scheme which results in two different identifiers `EP20010301662` versus `01301662`.

```xml
<application-reference data-format="epodoc">
        <document-id>
                <doc-number>EP20010301662</doc-number>
                <date>20010223</date>
        </document-id>
</application-reference>
<application-reference data-format="original">
        <document-id>
                <doc-number>01301662</doc-number>
        </document-id>
</application-reference>
```

When retrieving the bibliographic data for the same patent from the European Publication Server [7] the corresponding response fragment contains `01301662.1` as another identifier.

```
<B200>
        <B210>01301662.1</B210>
        <B220><date>20010223</date></B220>
</B200>
```

As described in section 3.4 this causes additional efforts for the unique identification of patent applications.

Kind of Document Codes

The kind-of-document code of a patent document is defined in the WIPO Standard ST.16 as a two-letter code. ST.16 defines eight mutually exclusive groups. Authorities can further supplement these codes with a one digit code from 1 to 7. Digit 8 is used to announce the correction of information represented on the first page of a patent, while digit 9 indicates a correction resulting in a partial or complete reprint of a document.

For example the EPO defines the kind-of-document code A2 as „European Patent Application published without International Search Report", whereas the USPTO defines A2 as „Patent Application publication (republication)".

The WIPO standards ST.9 [288], ST.32 [283] and ST.36 [287] define the bibliographic data contained in patent documents. This section gives a brief overview of the major metadata aspects covered by these standards with the focus on references and legal entities.

WIPO ST.9 defines 58 data entities widely used on the first page of patent documents or in patent gazettes. Each metadata is associated with a unique two-digit INID code (Internationally agreed Numbers for the Identification of bibliographic Data). INID codes are subdivided into eight major groups:

(a) Identification of the patent document (code 1x)

(b) Data concerning the application (code 2x)

(c) Data relating to priority under the Paris Convention (code 3x)

(d) Date(s) of making available to the public (code 4x)

(e) Technical information (code 5x)

(f) References to other legally or procedurally related domestic or previously domestic patent documents (code 6x)

(g) Identification of parties concerned with the patent (code 7x)

(h) Identification of data related to International Conventions other than the Paris Convention (code 8x, 9x)

WIPO ST.32 further refines the INID codes into a set of <Bxxyzz> elements, where xx a the INID code, y a refinement number and zz an optional organization identification (e.g., EP for the EPO). As an example, table 2.1 shows some ST.32 elements for the identification of a patent documents (group 1x).

INID	**ST.32 Code**	**Description**
10	B100	Document Identification
11	B110	Number of the patent document
12	B120	Plain language designation
12	B121	Plain language designation of the kind of document
12	B121EP	Descriptive text for B121 (EPO)
13	B130	Kind of document code (WIPO ST.16)
13	B140	Document date, usually date of publication
19	B190	Publishing country or organization (WIPO ST.3)

Table 2.1: Example of ST.32 Codes

WIPO ST.36 is the current standard of representing patent bibliographic data and content in XML. It defines alternative human-readable names for the ST.32 tags, but also allows using the ST.32 tags for backward compatibility reasons. Beside the bibliographic data ST.36 also defines the markup for representing the content of patent documents in XML. The ST.36 standard is adapted by all important national and regional patent offices. This is why ST.36 has in particular been used as foundation for the naming of classes and properties in the Patent Metadata Ontology (PMO) described in chapter 3.

Priority Claims

A very important concept in patent law is the concept of priority. It is a time-limited right that is defined by the first filing of a patent application. An assignee may claim the priority of an earlier application under certain circumstances. If the claim is valid, the priority date (i.e., the filing date of the first application) is considered to be the effective filing date for all subsequent filings. This is of special importance, since it restricts the prior art relevant for the examination process to be before the claimed priority date.

Citations

Another important type of references are citations. It can generally be distinguished between patent citations on the one hand and non-patent citations on the other. Both types of citations are available in the bibliographic data (INID 56) of an included search report. The following example shows two ST.36 fragments, one for a patent citation ('patcit') and one for a non-patent-literature citation ('nplcit').

```
<patcit dnum="GB2364924" id="sr-pcit0001" num="0001">
  <document-id>
    <country>GB</country>
    <doc-number>2364924</doc-number>
    <kind>A</kind>
    <date>20020213</date>
  </document-id>
  <rel-passage>
    <passage>page 5, line 22 - page 6, line 13; figures 1-4</passage>
    <passage>abstract</passage>
  </rel-passage>
  <category>X</category>
  <rel-claims>1-11</rel-claims>
</patcit>
...
<nplcit id="ncit0001" npl-type="s">
  <text>IBM Technical Disclosure Bulletin, Vol. 30, No. 2, p. 667...</text>
</nplcit>
```

From the example it can be seen that a patent citation is a relation from a patent document to another patent document with additional information, such as a passage, a set of claims the citation is relevant to, or a categorization of the citation. Examples for categorization codes according to WIPO ST.36 are shown in table 2.2.

Code	Description
X	Particularly relevant if taken alone
Y	Particularly relevant if taken combined with another document of the same category
A	Technological background

Table 2.2: Examples of Citation Categorization Codes

Legal Entities

The representation of legal entities, i.e. natural and juristic persons associated with a patent, is another important aspect for the modeling of patent metadata. Each legal entity can have a specific role in association with a patent. Examples for such roles and their corresponding ST.32 codes are:

- Applicant (B710): Legal entity who filed the patent application.
- Inventor (B720): Person who contributes to the concept of an invention.
- Grantee (assignee) (B730): Legal entity to whom the inventor assigns the patent rights.
- Attorney (B740): Legal entity authorized to act for, or in place of the applicant and who is registered to practice before the Patent Office.
- Opponent (B780): Legal entity that filed an opposition against the patent application.
- Licensee (B790): Legal entity who has licensed the patent.

Patent Families

There are different reasons for protecting an invention by multiple patents. The following list contains some of them.

- Since patents are national rights the protection of an invention in different countries requires multiple patent applications filled at the different national patent offices.

- A continuation extends an existing application with additional claims, has to share at least one inventor and must not already be abandoned or issued.
- A continuation in part provides the opportunity to enhance an existing application that shares at least one common inventor with additional claims yet it repeats certain parts of the parent's specification.
- A divisional application discloses a new and independent invention that may contain parts of the parent. The division and the parent do not need to share a common inventor [3].
- A provisional application offers a way to claim a filing date without having to submit a full patent application. It may later be extended to a full application [26].
- National patent law (as e.g., in Japan or Korea) may require one application for each claim, such that multiple claims are then summarized in a family.

There are various definitions on what constitutes a patent family. Each definition has its own use cases. A good overview of this topic can be found in [183]. In the following, three commonly used definitions are described. Their semantic modeling will be described in more detail in section 3.6.

1. Equivalents: Applications having exactly the same priority or combination of priorities. This definition is used by Esp@cenet [6].
2. Extended families: Applications directly or indirectly linked through priorities. This definition is used by INPADOC [11].
3. Single priority families: Applications originating from a single priority. In the case of multiple priorities, a given subsequent filing is assigned to multiple single-priority families. This definition is used in the WIPO reports [281].

In the following example (table 2.3) there are patent applications a_i with their priorities p_j. Each patent family definition creates a different collection of patent families each having different set of patent applications as members.

In the example the patent families are defined as follows:

- Equivalents: $E_1 = \{a_1, p_1\}$, $E_2 = \{a_2, a_3, p_1, p_2\}$, $E_3 = \{a_4, p_2, p_3\}$, $E_4 = \{a_5, p_3\}$

Applications	Priorities	Equivalents	Extended Families	Single Priority
a_1	p_1	E_1	X_1	S_1
a_2	p_1, p_2	E_2	X_1	S_1, S_2
a_3	p_1, p_2	E_2	X_1	S_1, S_2
a_4	p_2, p_3	E_3	X_1	S_2, S_3
a_5	p_3	E_4	X_1	S_3

Table 2.3: Comparison of Patent Family Definitions

- Extended Families: $X_1 = \{a_1, A_2, a_3, a_4, a_5, p_1, p_2, p_3\}$
- Single Priority: $S_1 = \{a_1, a_2, a_3, p_1\}$, $S_2 = \{a_2, a_3, a_4, p_2\}$, $S_3 = \{a_4, a_5, p_3\}$

2.1.2 Patent Classification

Patent classifications facilitate access to the technological and legal information contained in patent documents. They are also fundamental for selective dissemination of information for investigating the state of the art in given fields of technology and for the preparation of industrial property statistics. Prominent examples of patent classifications are the International Patent Classification (IPC) [12], European Patent Classification (ECLA) [5], US Patent Classification (USPC) [32] and Japanese FI/F-Term classification [8]. The IPC has been developed by the World Intellectual Property Organization (WIPO) for more than 40 years. Most other classification schemes have a mapping to the IPC. This is why the IPC is considered as a prototype for patent classifications and is described in more details below.

The 2011 version of the IPC is organized into the 8 top-level sections, 129 classes, 631 subclasses, 7,392 main-groups, and 62,493 subgroups. Starting with the eighth edition (from January 1, 2006) the classification has been divided into *core* and *advanced* entries. The core entries are updated on a three-yearly basis. The advanced entries are updated more frequently about every three months. With each new edition patent documents might need to be reclassified. The structure of the IPC code is depicted in the figure 2.1.2. Each IPC entry can be seen as a tuple containing a code, label, version, optional 'see-also' references to other entities and optional keywords.

On the top hierarchy level the International Patent Classification (IPC) is divided into eight sections. Each section is designated by one of the capital letters A through H. The

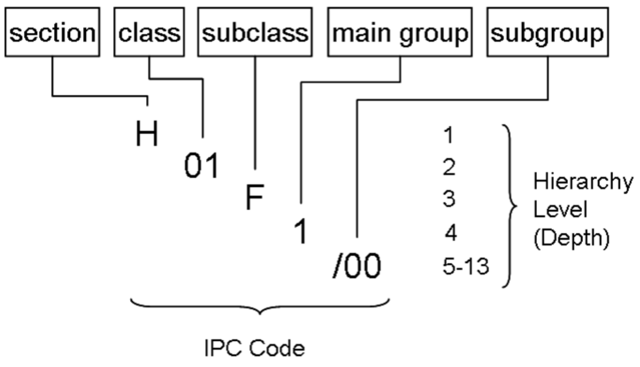


Figure 2.1: Structure of an IPC classification

section titles together with informative headings (titles without classification symbols) broadly indicate the contents of the section. Each section is subdivided into classes as the second hierarchy level. A class symbol consists of the section symbol followed by a two-digit number (e.g., H01). A class further comprises one or more subclasses (the third hierarchical level). Each subclass symbol consists of the class symbol followed by a capital letter (e.g., H01S). Subclasses may have an informative summary giving a broad survey of the content of the subclass.

Each subclass is broken down into main-groups (the fourth hierarchical level) and sub-groups (hierarchical level 5 to 9). Each main-group symbol consists of the subclass symbol followed by a one to three digits, a slash and the digits 00 (e.g., H01S 3/00). Each subgroup symbol consists of the main-group symbol but with at least two digits other than 00 after the slash (e.g., H01S 3/14). A title is preceded by one or more dots indicating the hierarchical position of that group, for example:

```
H           ELECTRICITY
H01         BASIC ELECTRIC ELEMENTS
H01S        DEVICES USING STIMULATED EMISSION

            Note: This subclass covers devices for the generation or amplification,
            by using stimulated emission, of coherent electromagnetic waves or
            other forms of wave energy; such functions as modulating, demodulating,
            controlling, or stabilising such waves.

            Guidance Header: MASERS (H01S 1/00), SEMICONDUCTOR LASERS (H01S 5/00),
            LASERS OTHER THAN SEMICONDUCTOR LASERS (H01S 3/00),
            OTHER DEVICES USING STIMULATED EMISSION (H01S 4/00)
```

```
H01S 3/00    Lasers, i.e. devices for generation, amplification,
             modulation, demodulation, or frequency-changing,
             using stimulated emission, of infra-red, visible,
             or ultra-violet waves (semiconductor lasers H01S 5/00)
H01S 3/09    . Processes or apparatus for excitation, e.g. pumping
H01S 3/091   .. by optical pumping
H01S 3/0915 ... by incoherent light
H01S 3/094   ... by coherent light
H01S 3/0941 .... of a semiconductor laser, e.g. of a laser diode
H01S 3/0943 .... of a gas laser
```

It is important to note that a subgroup defines the field of subject matter within the context of its hierarchical position. The title of subgroup H01S 3/094 therefore has to be read as „Processes or apparatus for excitation of lasers using optical pumping by coherent light". Further, a title may contain a phrase referring to another place in the classification. Such a phrase, called a reference, shows that the subject matter indicated by the reference is covered by the place(s) referred to. The main-group H01S 3/00 for example gives a reference to H01S 5/00 in the context related to semiconductor lasers.

When several successive main groups relate to common subject matter, a guidance heading before the first of such main groups may be provided. The guidance heading is a short statement that indicates the common subject matter found in all of the main groups it is relevant to. Furthermore, there can be notes which define or explain specific words, phrases or the scope of places, or indicate how subject matter is classified. Notes may be associated with any classification entry. For example the subclass H01S has the note „Devices for the generation or amplification, by using stimulated emission, of coherent electromagnetic waves or other forms of wave energy; such functions as modulating, demodulating, controlling, or stabilizing such waves."

Concerning presentation aspects, it is important to allow different ways to order the main groups. Therefore main-groups can also be arranged in a sequence from the most complex or highly specialized subject matter to the least complex subject matter. This order is explicitly stored in an additional index value. Often a residual main-group (e.g., „not otherwise provided for") is placed at the end of the list. Further details about the current IPC version are explained at full length in [37].

2.1.3 Patent Analysis

In general, there are two different types of methods for analyzing patent information. On the one hand there are qualitative methods that focus on the content of patent documents, on the other there are quantitative methods based on statistical processing of patent metadata. In practice both methods are used together [149].

Patent map analysis, or patent mapping, is a common analysis method [72] for patent information. Suzuki defines a patent map as „[p]atent information collected for a specific purpose of use, and assembled, analyzed and depicted in a visual form of presentation such as a chart, graph or table" ([233, p. 2]).

In the literature several patent analysis techniques are described. For example, Trippe [243] provides a survey of common patent analysis tasks, including: Cleanup and grouping of concepts, list generation (histograms), co-occurrency matrices & circle graphs, clustering of structured (fielded) data, mapping document clusters, adding temporal component to cluster map, citation analysis, Subject/action/object (SAO) functions.

Depending on the use case, there are various approaches of searching for patent documents. Hunt et al. [150] gives an overview of different search types which are summarized below.

- Patentability search (novelty search): Find prior art references that may be relevant to the invention's novelty and non-obviousness. Search in Patent documents and non-patent literature.
- Validity search (invalidity search, enforcement readiness search): Ascertain the validity/invalidity of an asserted patent. Search in Patent documents and non-patent literature.
- State-of-the-art search (collection search): Determine existing solutions and potential competitors within a given technological field. Search in Patent documents and non-patent literature. Strategy:
  - Brainstorm keywords related to the purpose, use and composition of the invention.
  - Gather preliminary U.S. classifications by reviewing the abstracts and classification definitions; examine the claims and drawings from the full-text of

patents; examine patents under the links, ’Patent Refs’ and ’Cited by’, for relevancy.

  - Use the U.S. Classifications you found to search by classification number. Review the abstracts, claims and drawings for relevancy. Check all references and note the U.S. Classifications and ’Field of Search’ areas for additional class/subclasses to search.

- Clearance search (freedom-to-operate search, right-to-use search, Infringement Search): Uncover enforceable patents that might block the commercialization of a product or service. Search in live patent documents.

According to a user requirements study conducted by the Fraunhofer Patent Center for German Research [199], in which 17 patent analysts from different interest groups (technology transfer centers, patent attorneys, patent departments, and others) have been interviewed based on a questionnaire, the top five purposes of patent search has been: for filing patent application (94%), licensing (59%), locating cooperation partners (41%), own R&D business (35%) and acquisition of companies (12%). The searched information included: state of the art (100%), legal status (82%), patent family members (76%), observing competitors (47%), technical information (41%), observing the market (35%) and evading patents (12%).

2.2 Semantic Web

The aim of the *Semantic Web* [70] – also called the *Web of Data* – is to achieve a new level of data integration and exchange between heterogeneous systems on a world-wide scale. The standardization of new concepts, models and vocabularies is organized by the World Wide Web Consortium (W3C) within its *Semantic Web Activity* [34]. In this context, ontologies play an important role. They define the formal semantics of terms used for describing the data and by this ensure that the meaning of the data is consistent between and within systems. The next sections briefly describe the major W3C Semantic Web standards.

2.2.1 Resource Description Framework (RDF)

The Resource Description Framework (RDF) [257] is a standard model for data interchange on the Web. It simplifies the merging of data with different underlying schemas and supports the evolution of schemas over time without the need of changing all data consumers [203].

RDF is an assertion language. Each assertion declares that certain information about a resource is true. An assertion is modeled as a $\langle s,p,o\rangle$-triple where s (subject) identifies the resource the assertion is about, p (predicate) is a property of the resource, and o (object) is the value of p. The set $T = (U \cup B) \times U \times (U \cup B \cup L)$ is the set of all RDF triples, U denotes the set of RDF URI references, B denotes the set of blank nodes, and L denotes the set of RDF literals. An RDF URI reference is a Uniform Resource Identifier (URI) [69] that fulfills the restrictions specified in the RDF abstract syntax [257, ch. 6.4].

A triple set can be interpreted as a directed labeled graph $\boxed{\text{s}} \xrightarrow{p} \boxed{\text{o}}$ with the subjects and objects as nodes and node labels, the triples as arcs, and the predicates as arc labels. The set of RDF graphs G is the power set of T. A sub-graph is a subset of the corresponding triple set. In this thesis the term *RDF graph* is used as a synonym for the term *triple set*. A triple set (or any subset) can be serialized in different languages, such as RDF/XML [256], N-Triples [254], or Turtle [270]. The result, an *RDF document*, is a sequence of words over an alphabet defined by the particular RDF serialization language.

An extension to this model is the concept of *named graphs* [92]. It provides a mechanism for making assertions about graphs and relations between them. Applications are, for example keeping track of provenance information, restricting information usage, digital signatures, or versioning. A named graph is a pair $ng = (n, g)$ with n in U and g in G. It is $name(ng) = n$ and $rdfgraph(ng) = g$. It is important to note, that blank nodes cannot be shared between different named graphs, i.e. if ng_1 and ng_2 are different named graphs, then the sets of blank nodes which occur in triples in $rdfgraph(ng_1)$ and in $rdfgraph(ng_2)$ are disjoint. A common way to implement named graphs is to extend RDF triples by a forth component to RDF quads. The forth component associates a URI with a triple. Examples of quad stores are, Virtuoso [33], AllegroGraph [1], or OWLIM [22].

2.2.2 SPARQL

SPARQL [261] is a query language for retrieving data stored in RDF graphs. A SPARQL processor is a query engine that interprets queries in the SPARQL query language, accesses RDF dataset and returns, depending of the variant of the query, either a table, RDF graph or Boolean result. The concept of named graphs has been adopted for the SPARQL query language, by defining an RDF datasets as a set $\{G, (u_1, G_1), \ldots, (u_n, G_n)\}$ where G and each G_i are graphs, and each u_i is a distinct URI. G is a called the *default graph* and the (u_i, G_i) pairs are named graphs. By introducing an unnamed default graph, SPARQL provides backward compatibility with RDF without named graphs and allows the named graphs functionality of SPARQL to be optional [92].

The SPARQL query language specifies four different query variants:

- SELECT queries extract values from an RDF dataset in a table format.
- CONSTRUCT queries extract values from an RDF dataset and transform the results into a target RDF graph based on a query template.
- ASK queries return a Boolean result containment of a sub-graph.
- DESCRIBE queries also extract an RDF graph, but in this query variant the SPARQL processor decides about the content of the returned graph.

A SPARQL endpoint is a web service that implements the SPARQL protocol [260]. The latter defines a request to contain a SPARQL query as payload and a response to contain a SPARQL results document as payload [262]. A SPARQL client is any arbitrary software component that communicates with a SPARQL endpoint via the SPARQL protocol. The components of this infrastructure are depicted in figure 2.2.

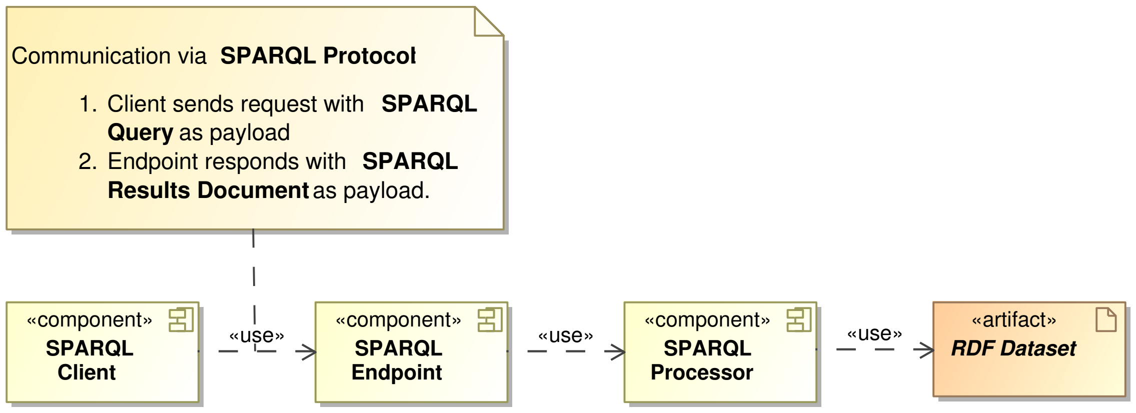

Communication via **SPARQL Protocol**
1. Client sends request with **SPARQL Query** as payload
2. Endpoint responds with **SPARQL Results Document** as payload.
«component» SPARQL Client
«use»
«component» SPARQL Endpoint
«use»
«component» SPARQL Processor
«use»
«artifact» RDF Dataset

Figure 2.2: Components of a SPARQL infrastructure

2.2.3 Ontologies in OWL

According to Gruber,

> „A conceptualization is an abstract, simplified view of the world that we wish to represent for some purpose. [...] An ontology is an explicit specification of a conceptualization.” (Gruber [137, p. 1])

More formally, an ontology consists of a collection of axioms stated in an ontology language and constraints the construction of models satisfying the axioms [248]. In general, it can be distinguished between terminological axioms, individual axioms, and annotations. A terminological axiom can either be a class or property axiom. An individual axiom can either be an instantiation, relation, or attribute axiom. Annotations carry machine and human oriented metadata. Throughout this thesis, we refer to terminological axioms as *terminology* or *T-box* and to individual axioms as *assertional knowledge* or *A-box*.

An ontology language defines the supported types of axioms and their formal semantics. OWL is the ontology language for the Semantic Web. The acronym stands for *Web Ontology Language*. OWL defines three sublanguages: OWL Lite, OWL DL and OWL Full [44]. Each of these sublanguages is a syntactic extension of its simpler predecessor (OWL Lite → OWL DL → OWL Full). OWL Lite was intended to support those use cases in which a classification hierarchy and simple constraints are needed. OWL DL was designed to provide a maximum expressiveness while retaining computational completeness and decidability. OWL DL further has a correspondence with description logic [57]. OWL Full was designed to preserve some compatibility with RDF Schema [255], an initial and simple schema language for RDF. Since OWL Full is undecidable, no reasoning software is able to perform complete reasoning for it.

In October 2009 OWL 2 has become the new Web Ontology Language standard [264]. It addresses some of the problems with OWL described in more details in [136]. OWL 2 defines three profiles: OWL 2 EL, OWL 2 QL, and OWL 2 RL. Each profile is defined as a restriction of the OWL 2 structural specification, i.e. as a subset of the possible structural elements, and trades off different aspects of the expressive power of OWL in return for different computational and implementational benefits (cf. [266]). A detailed explanation can be found in [136] and [265].

The rest of this section describes the most important structural elements of OWL. In order to provide a compact and readable notation, the Unified Modeling Language (UML) [31] is used as proposed by Brockmans et al. [83]. The newly introduced stereotypes and conventions are described below.

1. «**owl:Class**» Each ontology class defines a group of individuals[1] and allows to make statements about this group. Ontology classes are instances of *owl:Class*. In UML notation OWL classes are marked with the stereotype «owl:Class».

2. «**owl:ObjectProperty**» Object properties define relations between individuals and are instances of *owl:ObjectProperty*. In UML notation object properties are represented either as directed associations between two classes. Then the class where the association starts is defined to be the property's domain and the class where the association ends is defined to be the property's the range. An alternative UML representation is to use the stereotype «owl:ObjectProperty» and explicit (*rdfs:domain*) and (*rdfs:range*) associations.

   RDF triples represented by both notations of figure 2.3:

   ```
   pmo:sampleObjProperty a owl:ObjectProperty ;
            rdfs:domain pmo:SampleClass ;
            rdfs:range pmo:SampleClass2 .
   ```

3. «**owl:DatatypeProperty**» Data type properties define relations between individuals and RDF literals. They are instances of *owl:DatatypeProperty*. In UML nota-

[1]Throughout this thesis the terms 'individual', 'instance' and 'resource' are used as synonyms.

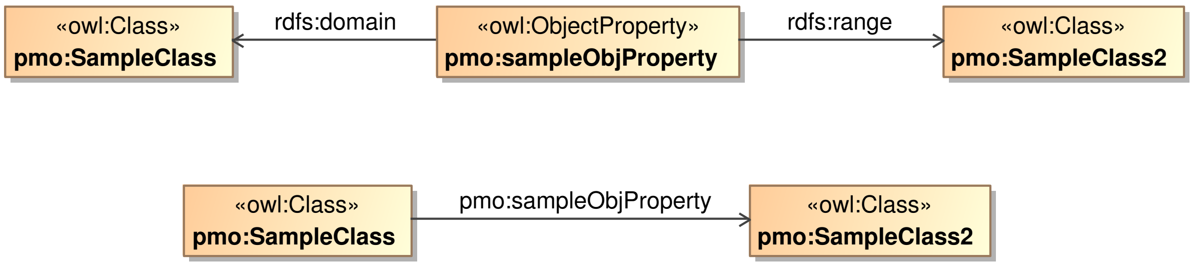


Figure 2.3: Convention for object properties

tion data type properties are represented as attributes of a class. Then the class that defines the attribute is interpreted as the domain and the attribute's type is interpreted as the range. An alternative UML representation is to use the stereotype «owl:DatatypeProperty» and explicit (*rdfs:domain*) and (*rdfs:range*) associations.

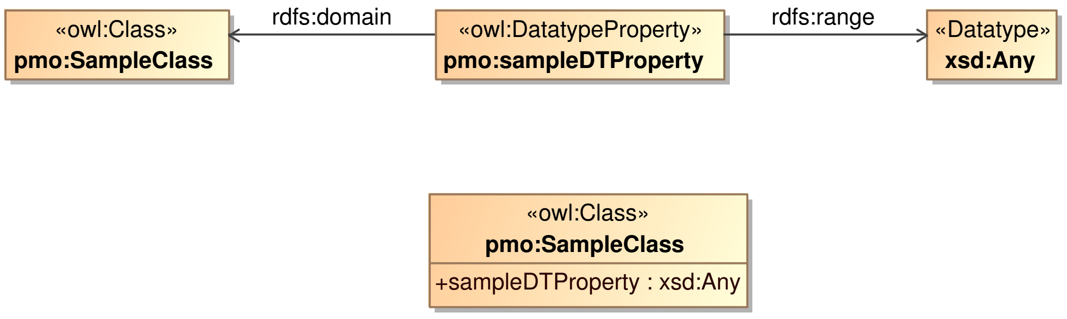


Figure 2.4: Convention for data type properties

RDF triples represented by both notations of figure 2.4:

```
pmo:sampleDTProperty a owl:DatatypeProperty ;
          rdfs:domain pmo:SampleClass ;
          rdfs:range xsd:Date .
```

4. **Inheritance** Inheritance between classes and properties are defined by *rdfs:subPropertyOf* and *rdfs:subClassOf* relations. In UML inheritance relations are used for representation. If an inheritance relations is defined between two «owl:DatatypeProperty» or «owl:ObjectProperty» instances it is interpreted as *rdfs:subPropertyOf* relation. If an inheritance relations is defined between two «owl:Class» instances it is interpreted as *rdfs:subClassOf* relation.

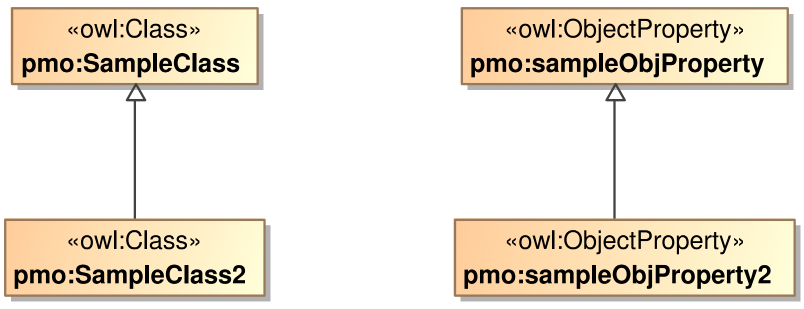


Figure 2.5: Convention for sub-classes and sub-properties

RDF triples represented by figure 2.5:

```
pmo:SampleClass2  rdfs:subClassOf  pmo:SampleClass  .
pmo:sampleObjProperty2  rdfs:subPropertyOf  pmo:sampleObjProperty  .
```

5. **Property Characteristics**: For representing transitive, symmetric, functional and inverse-functional relations in UML, the stereotypes «owl:TransitiveProperty», «owl:SymmetricProperty», «owl:FunctionalProperty» and «owl:InverseFunctional-Property» are used.

A more detailed introduction into Semantic Web standards can be found in Allemang and Hendler [55], Hitzler et al. [147] and Bizer et al. [75].

2.3 Cryptography

The goal of cryptography is to provide a system for keeping information secret by applying mathematical operations one messages. A cryptosystem can formally be described as a tuple (P, C, K, E, D), where P is a set of plaintexts, C is a set of ciphers, K is a set of keys, $E = \{e_k : k \in K\}$ is a family of encryption functions $e_k : P \to C$ and $D = \{d_k : k \in K\}$ is a family of decryption functions $d_k : C \to P$. For all $k_e \in K$ there is a $k_d \in K$ so that $d_{k_d}(e_{k_e}(p)) = p$ holds for all $p \in P$. A cryptosystem is called symmetric if $k_e = k_d$. It is called asymmetric if $k_e \neq k_d$. Examples of symmetric cryptosystems are Triple-DES [39] and AES [40]. An example of an asymmetric cryptosystem is RSA [207].

For secure key transport and in consideration of performance, plaintexts are usually encrypted by using a session-key scheme which combines symmetric and asymmetric encryption (Fig. 2.6). The sender encrypts a plaintext m using a symmetric encryption function f parameterized with a randomly generated session key k. The result is a cipher c_m. To transmit the session key to the recipient in a secure way, k is encrypted with an asymmetric encryption function g parameterized by the public key pub of the recipient. The result is a cipher c_k. Then the ciphers c_m and c_k are transmitted. The recipient recovers the session key k by decrypting c_k using the decryption function g^{-1} parameterized with its private key $priv$. Finally, the recipient computes the plaintext m from c_m using f^{-1} parameterized with k.

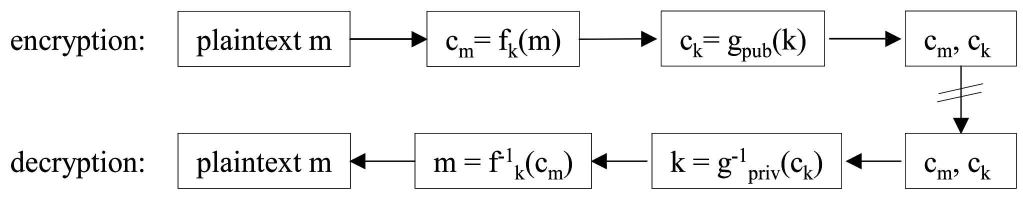


Figure 2.6: Encryption scheme with session-keys

When using cryptosystems, a method to ensure the data integrity is needed. A common approach for this problem is to use one-way hash functions, for example SHA-1 [41] or MD5 [38]. A hash or digest is a sequence of bytes that represents the input in a unique way and usually is smaller than the input. The sender computes the digest d_m of a message m using a one-way hash function h. Both, the digest d_m and the cipher c_m are transferred to the recipient. The recipient decrypts the cipher (let m' be the decrypted cipher) and computes the digest $d_{m'} = h(m')$. If $d_{m'} = d_m$ then $m' = m$ holds.

A more detailed introduction into the field of cryptography and computer security can be found in [74] and [85].

2.4 Information Visualization

In the field of information visualization many visualization and interaction techniques have been developed over the last decades for producing graphical representations of various types of abstract data. This section gives a short overview of this research field.

Information visualization in general, can be seen as a transformation of abstract data into a graphical representation. This transformation can be described by a pipeline. There are similar pipeline models in the literature, the so called visualization pipeline

[138]. A well-known pipeline model is the *information visualization reference model* by Card et al. [87] shown in figure 2.7. Similar models have been proposed by Haber and McNabb [138] and Chi and Riedl [97].

The model of Card et al. describes a process starting with *raw data* as input and generating *views* presented to users as output. Views are basically made from marks and their graphical properties [86]. There is a limited set of marks (e.g., points, lines, areas, surfaces, or volumes) and visual properties (e.g., color, size, shape, orientation, texture, connection, enclosure, or position) [71]. Additionally, two intermediate representations, *data tables* and *visual structures*, are introduced.

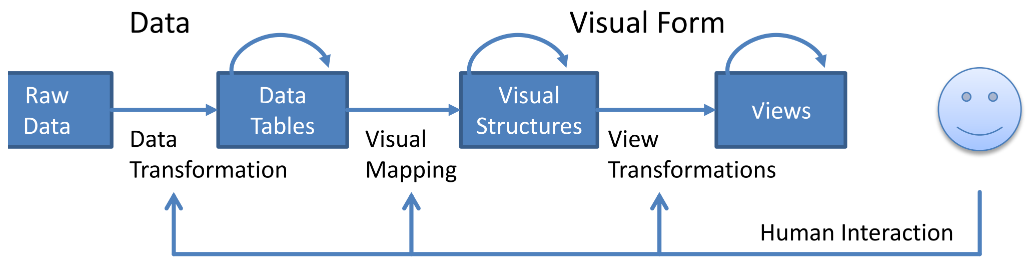


Figure 2.7: Information visualization reference model (adapted from Card et at. [87])

Data tables are relational descriptions of data objects as the result of *data transformations*, for example data aggregation, text parsing, dimension reduction, filtering, normalization, etc. Data tables can be used for the composition of *complex data objects*, i.e. objects that have properties, which can again be complex data objects. This nesting stops with unstructured or atomic properties of a given type. It is also possible to concatenate several transformations by forming transformation chains. Visual structures are the result of a *visual mapping*. They combine (complex) data objects with marks and visual properties and are instantiated in one or more views. Typical *view transformations* are highlighting, scrolling, panning or zooming. Furthermore, users have the possibility to modify these transformations interactively in order to adjust the visualizations for their needs. For instance, data transformations can be controlled by filters, visual transformations by selecting specific layout algorithms, and view transformations by applying location probes [232], viewpoint controls, or distortion methods [179].

2.4.1 Visualization Techniques

In the area of information visualization a multitude of visualization techniques have been developed over the last decades. One approach of grouping these techniques is by data types. Shneiderman [223] proposed seven categories: one, two, three, and multi-dimensional data, temporal data, tree and network data. Below we will briefly discuss multi-dimensional, temporal, tree and network data, since they are the foundation of the visualization techniques described in chapter 5 and 6.

Multidimensional data can either be visualized in several views or within a single view [291]. Examples for the first approach are scatter plot matrices [66] or coordinated multiple views [208]. Examples of the latter are techniques like Parallel Coordinates [153, 144], glyph based visualizations [98, 94, 209], or dense pixel displays [164]. Furthermore, clustering and dimension reduction techniques can be employed on large data collections in order to give users a condensed view of the data under inspection.

Data of which at least one dimension is associated with is called *temporal or time-oriented data* [53]. There are different ways to do the visual mapping of data properties containing time values. In general one can distinguish static mappings (time to position, color, angle, line width, containment, labels, symbols, etc.) and dynamic mappings (e.g., slide shows or animations). Most static approaches that implement a time-to-space mapping use one display dimension to represent a time axis. In this context one can distinguish between linear axis arrangements (e.g., point plots and line plots [139], EventViewer [65], or VisuExplore [206]) and spiral or cyclic axis arrangements (e.g., RingMaps [296], SpiralDisplay [88], or enhanced interactive spiral [242]). Examples of dynamic mappings are Trendanalyzer [119] and TimeRider [205].

Network data usually is visualized as graphs. Graph visualizations can be classified into three types: matrix, node-link, and implicit visualizations [217]. Matrix visualizations are a direct graphical interpretation of the adjacency matrix of graph. Examples are Graph Sketches [50], MatrixZoom [51], or MatrixExplorer [145]. Node-link visualizations show nodes as objects in two- or tree-dimensional space and the edges as lines or curves. A detailed overview of this area can be found in [107]. In implicit, also called space-filling, graph visualizations edges are encoded by the relative positioning of vertices, for example by overlap, adjacency, or inclusion. Examples are Treemaps [222, 245, 61], Icicle Plots [175], or Beamtrees [244]. Tree visualizations are a special type of graph visualizations. It is one of the best-studied areas in information visualization with more than 200 visualization and layout techniques. A comprehensive overview can be found in [218].

2.4.2 Interaction Techniques

Interaction is a central aspect in visualization. It involves the dialog between the user and the system as the users try to achieve their tasks. In the literature different taxonomies for user tasks at different levels of granularity have been proposed. Ward et al. [276] distinguish the three high-level tasks: explorative analysis, confirmative analysis, and presentation of analysis results. First, during an explorative analysis no hypotheses about the data are given and the user starts extracting relevant information. Second, in an confirmative analysis visualizations are used to verify the hypotheses. The goal is to ascertain facts about the data and gain new insights. Third, during the presentation of analysis results insights about the data are communicated to others.

At a finer granularity, the following tasks have been proposed by Zhou and Feiner [298]: associate, background, categorize, cluster, compare, correlate, distinguish, emphasize, generalize, identify, locate, rank, reveal, and switch. Additional proposals of tasks include: overview, zoom, filter, details-on-demand, relate, history, extract (cf. Shneiderman [223]); Retrieve value, compute derived value, find extrema, sort, determine range, characterize distribution, find anomalies (cf. Amar et al. [56]).

Interaction techniques support these tasks and allow for dynamic changes of visual structures, visual mapping and the views. Furthermore, they support the relation and combination of multiple independent visualizations, for example by selecting objects in one view and immediately highlighting the associated object representations in others. The connection of multiple views by interaction techniques can provide more insight than using the visual components independently. Examples of well-known interaction techniques that implement some of the user tasks are direct manipulation [151], dynamic queries [52], overview+detail [99], brushing & linking [231, 173], and focus+context [114, 179, 89].

This section could only give a brief overview of the research area of information visualization. More comprehensive introductions can be found in [276, 227, 95, 277, 87]. A detailed overview of visualization techniques for time-oriented data can be found in Aigner et al. [53]. A description of design patterns for information visualization applications has been proposed by Heer et al. [141].

2.5 Visual Analytics

The term visual analytics is attributed to the research and development agenda «Illuminating the Path» by Thomas and Cook [239]. At the beginning visual analytics had a strong focus on protection against terrorist activity in the United States. In the meantime, visual analytics describes a multidisciplinary field that combines various research areas, in particular visualization, human-computer interaction, data analysis, data management, geo-spatial and temporal data processing, and statistics. The goal of visual analytics is the development of techniques and tools to enable people to: (i) Synthesize information and derive insight from massive, dynamic, ambiguous, and often conflicting data; (ii) Detect the expected and discover the unexpected; (iii) Provide timely, defensible, and understandable assessments; (iv) Communicate assessments effectively [166].

Keim et al. [167] propose a visual analytics process as depicted in figure 2.8 which is characterized as a combination of automatic and visual analysis methods with a tight coupling through human interaction in order to gain knowledge from data.

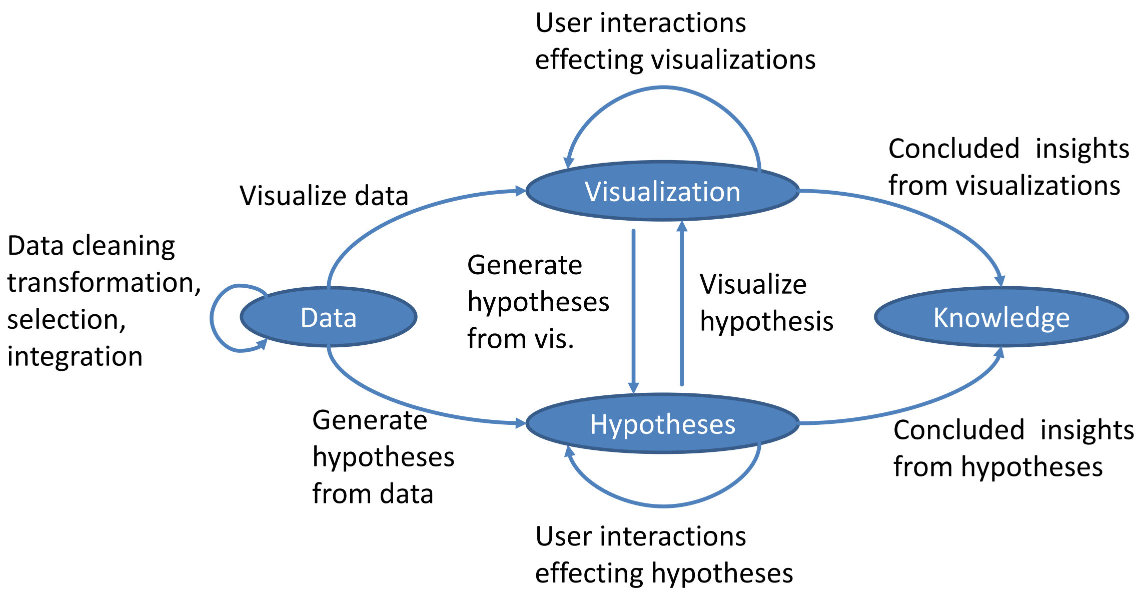

User interactions effecting visualizations
Concluded insights from visualizations
Visualize data
Visualization
Data cleaning transformation, selection, integration
Data
Generate hypotheses from vis.
Visualize hypothesis
Knowledge
Hypotheses
Generate hypotheses from data
Concluded insights from hypotheses
User interactions effecting hypotheses

Figure 2.8: Visual Analytics process (adapted from Keim et al. [167])

In a typical visual analytics scenario, heterogeneous data from multiple sources need to be integrated into a common schema and preprocessed (e.g., data cleaning, normalization, grouping, etc.) in order to extract meaningful units of data for further processing. A human analyst has then the choice between visual or automatic analysis methods. In an initial visualization the analyst may obtain the desired knowledge directly, but it is likely that the initial visualization is not sufficient. User interaction with the visualization can reveal new insight, for example by considering different views on the data. Findings from the visualization can be used for adapting the model for automatic analysis. Models can also be built from the original data by data mining. Once a model is set up the analyst has the ability to modify parameters or select other analysis algorithms. Changing between visual and automatic methods is characteristic for the visual analytics process and leads to a continuous refinement and verification of preliminary results.

Today's challenges in the field include integration of heterogeneous data sources, interpretability and trustworthiness, extraction of semantics, user acceptance, large data streams, and scalability.

2.6 Software Architecture

This section gives an overview of the field of software architecture. First, we provide definitions of the term software architecture. We then discuss architectural styles, software quality characteristics and architecture documentations.

The ISO/IEC/IEEE 42010:2011 standard [156, p. 2] defines the term *architecture* as „[...] fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution." Fielding gives a more detailed definition emphasizing the operational aspects:

> „A software architecture is an abstraction of the run-time elements of a software system [...] defined by a configuration of architectural elements – components, connectors, and data – constrained in their relationships in order to achieve a desired set of architectural properties." (Fielding [111, p. 5])

According to Fielding, a component is „an abstract unit of software instructions and internal state that provides a transformation of data via its interface" (Fielding [111, p. 9]). A connector is an „abstract mechanism that mediates communication, coordination, or cooperation among components" (Fielding [111, p. 10]). Architectural properties include both, functional properties achieved by the system and non-functional properties, often referred to as quality attributes (see section 2.6.2). Since an architecture embodies both functional and non-functional properties, it can be difficult to directly compare architectures for different types of systems. *Architectural styles* are a mechanism for categorizing architectures and for defining their common characteristics.

2.6.1 Architectural Styles

An architectural style defines a pattern of structural organization for a family of systems [219]. A simplified view on an architectural style is a graph in which the nodes represent the components and the arcs represent the connectors. Additionally there can be a set of constraints on how components are to be connected.

Garlan and Shaw [120] describe a collection of architectural styles, including (1) pipes and filters, (2) data abstraction and object-oriented organization, (3) event-based implicit invocation, (4) layered systems, (5) repositories and (6) table-driven interpreters.

Service Oriented Architecture (SOA) is another more recent architectural style. It is a „paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains" [188]. SOA additionally emphasizes on loose coupling between interacting services and their consumers. According to Endrei at al. [109], a *service* is a course-grained, discoverable software entity that exists as a single instance and interacts with applications and other services through a loosely coupled communication model. In this context, an *interface* defines a contract between a consumer and a provider by a set of logically grouped method signatures. Further elements in a SOA are (1) *service provider*, the software entity that implements a service specification, (2) *service consumer*, a software entity that calls a service provider and (3) *service registry*, a specific kind of service provider that acts as a registry and allows for the lookup of other service provider interfaces and service locations [109].

Garlan and Shaw [120] also pointed out that complex software systems typically have a heterogeneous architecture which is a combination of different styles. There are different ways in combining these styles, for example a component of a system organized in one architectural style may have an internal structure that is organized in a completely different style. Another situation is a mixture of architectural connectors, for example a component might access a repository via its interface, but interacts through pipes with other components in the system.

2.6.2 Software Quality Characteristics

The design decisions made for a specific software architecture are often driven by software quality characteristics and their prioritization. The ISO/IEC25010:2011 [155] for example provides a general model for the quality of software products and software intensive computer systems defined by set of characteristics and their relationships. The quality model for software products is composed of 8 characteristics and 31 subcharacteristics as shown in figure 2.9.

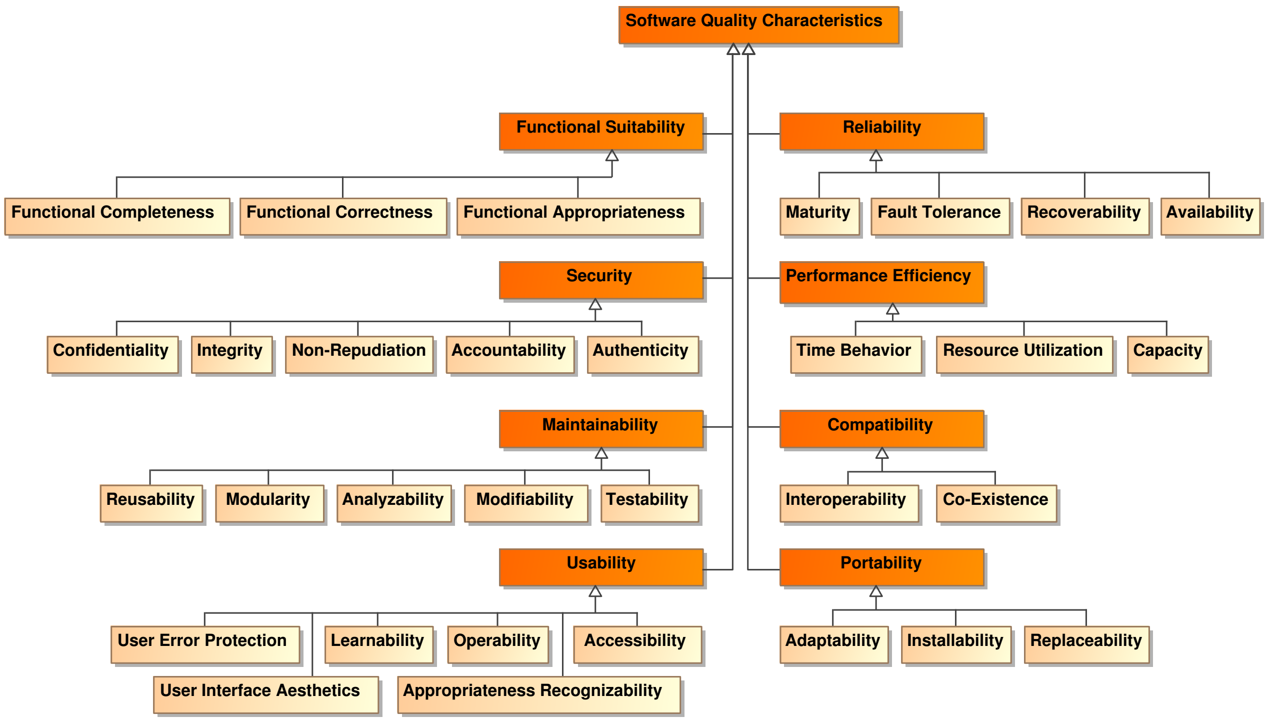


Figure 2.9: Product quality characteristics according to ISO/IEC25010:2011

Because the ISO/IEC25010:2011 provides a conceptual framework and not a ready-to-use model, the framework must be tailored to the specific system under consideration.

Examples for specific quality models are given by Polillo for Web 2.0 sites [197] and by Goeb and Lochmann for service-oriented architectures [135].

2.6.3 Architecture Description

The term *architecture description* refers to artifacts used to express and document an architecture. An architecture description contains multiple views of a system of interest, for example a structural view showing the components, their interfaces, their dependencies and inheritance relationships. The basic idea of a view is to address specific concerns about a system in the context of its environment and stakeholders. Examples of system concerns are business goals, structure, system features, and quality characteristics described in the previous section.

For architecture descriptions often well-defined notations and models are used, such the Unified Modeling Language (UML) [31]. Additionally, architecture description frameworks, such as Arc42 [229], implement best practices for structuring and presenting the most common system concerns. An overview and explanation of requirements for an architecture description is given in the ISO/IEC/IEEE 42010 standard [156].

CHAPTER 3

Semantic Representation of Patent Metadata

Patent analysis today requires high efforts by human specialists. Reasons are, for example, the inherent complexity of the described inventions, the abstract language used for describing the inventions and formulating the claims, and the complex national and international legal systems with different rules and procedures. The latter in particular results in multiple data sources and heterogeneous data.

One approach to reduce these efforts is a content-oriented processing of patent information [274] by using natural language processing (NLP) and machine learning techniques for information extraction [102, 190], automated classification [68], or similarity computation [93, 152]. Also collaborative methods could be applied to patent information, for example by using wiki systems for reviewing new patents and linking them to related work [36, 25].

The fundamental idea is to integrate the outcome of such patent analyses with a *semantic representation* of patent information. Regardless if the analyses are performed manually by human specialists, automatically by analysis tools or within a community process, the outcome can be seen as *semantic annotations* that enrich the original representation. Such a holistic representation allows to flexibly perform a semantic search on any patent information and analysis aspects.

The focus of this chapter is on semantic representation of patent metadata. It describes the *Patent Metadata Ontology (PMO)* that has been developed as part of the PATExpert ontology framework which is introduced in the next section.

3.1 PATExpert Ontology Framework

The PATExpert ontology framework defines a terminology for semantic representation of patent information in terms of OWL ontologies (see section 2.2.3). Figure 3.1 provides an overview of the framework. The ontologies are briefly described below.

1. To capture common sense knowledge the *Suggested Upper Merged Ontology (SUMO)* [186, 192] is used within the PATExpert framework. SUMO is linked (cf. Niles and Pease [187]) to the English lexical ontology *WordNet* [110] which is widely used for many NPL tasks.

2. The terminology for describing the domain of the invention, for example chemistry, engineering, or physics is specified by domain ontologies. In PATExpert two domain ontologies for the optical recording domain have been developed: the manually engineered *ORDO* ontology and the automatically extracted *AUTO* ontology.

3. The patent domain specific terminology, for example, for legal procedures, patent metadata, and document structure is defined in the *Patent Upper Level Ontology (PULO), Patent Structure Ontology (PSO), Patent Drawing Ontology (PDO) and Patent Metadata Ontology (PMO)*. The latter defines the terminology for patent metadata and is further divided into four ontology modules: *Patent Identification Module (PMO-I)*, *Bibliographic Data Module (PMO-B)*, *Patent Family Module (PMO-F)*, and *Patent Classification Module (PMO-C)*.

For further details on the PATExpert ontology framework the reader is referred to Wanner et al. [275, 274] and Giereth et al. [130]. The construction of the ORDO and AUTO domain ontologies is explained by Potrich and Pianta [198], whereas the application of the PDO for image retrieval is explained by Vrochidis et al. [249].

3.2 Related Work

Semantic representation has a long tradition in Artificial Intelligence. An overview of classical representation approaches can be found, for example, in Sowa [226], Russell and Norvig [212], Luger [182], and Reimer [204].

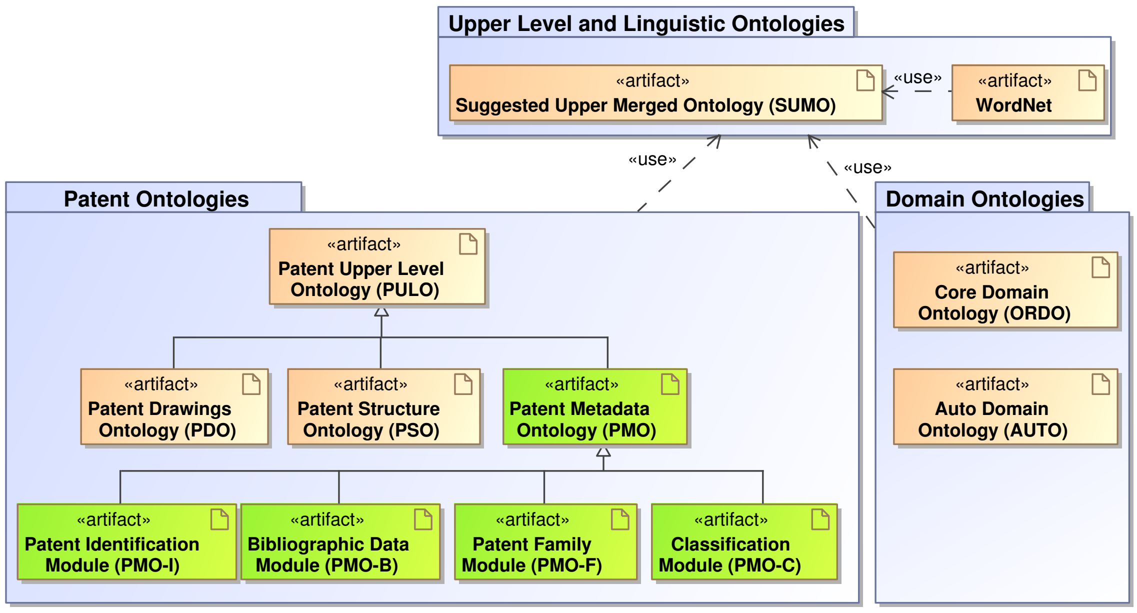

Upper Level and Linguistic Ontologies
«artifact»
Suggested Upper Merged Ontology (SUMO)
«use»
«artifact»
WordNet
«use»
«use»
Patent Ontologies
«artifact»
Patent Upper Level Ontology (PULO)
«artifact»
Patent Drawings Ontology (PDO)
«artifact»
Patent Structure Ontology (PSO)
«artifact»
Patent Metadata Ontology (PMO)
«artifact»
Patent Identification Module (PMO-I)
«artifact»
Bibliographic Data Module (PMO-B)
«artifact»
Patent Family Module (PMO-F)
«artifact»
Classification Module (PMO-C)
Domain Ontologies
«artifact»
Core Domain Ontology (ORDO)
«artifact»
Auto Domain Ontology (AUTO)

Figure 3.1: PATExpert ontology framework with PMO modules

Previous research on semantic representation of intellectual property rights (IPR) have been accomplished by Delgado et al. [105, 106]. They proposed an ontology for digital rights management called IPROnto in order to facilitate the development of e-commerce applications that need to be aware of rights associated to specific multimedia content. Although IPROnto has a different focus as PMO and is much broader, it can be considered as one of the first research projects that uses Semantic Web standards (see section 2.2) for an IPR domain. Similar to the PATExpert ontology framework Delgado et al. reused general concepts for time, space, part hood, etc. from upper level ontologies, such as SUMO [192], DOLCE [118] and LRI-Core [80].

A patent ontology has been proposed by Taduri et al. [234, 235] for linking patents with court cases. If terms are searched in the court case database, then the associated patents and their metadata are also returned. The proposed ontology only contains a limited subset of the vocabulary defined in PMO. Additionally, Ghoula et al. [121] proposed an ontology based system for automatically annotating patents. But in contrast to PMO, the focus is on content and structure of patents rather than patent metadata. Shih and Liu [221] apply social network analysis to patent networks in order to improve the classification of patents. Their proposed patent ontology network contains four node types: *patent*, *UPC-class*, *inventor* and *assignee*, which is a very limited subset of PMO. Furthermore, Mukherjea et al. [185] describe a system for facilitating information retrieval and knowledge discovery from patents. The system identifies biological terms and relations from the patents and combines this information with knowledge from biomedical ontologies to create a semantic network. Their special focus is on the biomedical domain rather than on general patent metadata.

In addition to the mentioned IPR and patent related ontologies, there are also more general vocabularies for describing metadata properties of common resources. Examples are *DCMI Metadata Terms* [47] of the Dublin Core Metadata Initiative or Adobe's *Extensible Metadata Platform (XMP)* [42]. Both provide a collection of terms such as title, creator, date, rights, language, etc. Although they do not define any hierarchy and therefore are no ontologies in the stricter sense, an alignment with DCMI Metadata Terms or XMP can be useful, since they are widely used in the Web [108].

In comparison with the mentioned ontologies and vocabularies, PMO (i) specifically models patent metadata, (ii) is more detailed than the mentioned ontologies, (iii) uses OWL as ontology language, and (iv) considers the standardization efforts of the World Intellectual Property Organization (WIPO). For most of the PMO classes and properties a mapping to the WIPO ST.32 and ST.36 standards (section 2.1) is defined.

3.3 Patent Metadata Ontology

The *Patent Metadata Ontology (PMO)* [133, 130] has been developed as part of the PATExpert ontology framework described in the previous chapter. The work described in this chapter is partly based on the diploma thesis of Stäbler [230].

Starting points for modeling and engineering of the PMO have been the patent standards described in section 2.1 as well as best practices for knowledge representation [226, 204] and ontology design patterns [116]. For the population of the ontology a defined set of about eight thousand sample patents from the optical recording and the machine tools domains has been used. The design goals for the construction of PMO have been:

1. **Extensibility:** Since patents are part of complex legal systems based on national and international laws and conventions that further are subject to change, it is not feasible to cover all patent metadata aspects in PMO. Therefore PMO defines a core

terminology separated into different modules. Each module covers a specific patent metadata aspect. New aspects can be added to PMO by defining new modules or extending existing modules with new concepts, properties or instances. It is also important to note that PMO can be aligned with other ontologies by using standard OWL capabilities: *owl:equivalentClass* at class level, *owl:equivalentProperty* at property level and *owl:sameAs* at instance level. Alignment with other ontologies is not part of the PMO definition and therefore is not discussed here.

2. **Practicality**: When performing a semantic search based on terms defined in PMO, it is important that these terms are intuitively understood by patent searchers. One approach therefore is to use well-known terms of the patent domain on the one hand and to reduce the layers of abstraction to a minimum on the other.

3. **Efficiency**: Due to the massive amount of patent data PMO have to be efficient with regard to inferences. Since inferences on large triples sets are expensive on the one hand and since only a limited expressiveness of the ontology layer is required on the other hand, PMO is specified using a subset of OWL called *OWL RDFS/Plus*. This subset has been introduced by Allemang and Hendler [55] for very efficient but limited reasoning with full merging capabilities.

4. **Merging**: It should be able to merge assertions from different PMO based metadata knowledge bases by simply building the unit of the triple sets. This allows building a large network of linked patent metadata. The merging of triple sets with blank nodes requires the OWL constructs *owl:functionalProperty* and *owl:inverseFunctionalProperty*, which create additional *owl:sameAs* assertions between individuals. One approach to reduce the amount of inferences of this kind is the use of URI references instead of blank nodes. Two triple sets without blank nodes can be merged by simply building the set union. But to abstain from blank nodes is not always possible, for example it is difficult to find an appropriate URI that uniquely defines a person. Using names in URIs may be ambiguous. Other information, such as the social security number, is not appropriate for publishing on the Web. That is why the Friend of a Friend (FOAF) vocabulary [81] uses inverse functional properties, like the email address or homepage, to reason about the identity of blank nodes.

5. **Alignment**: Patent metadata are intended to be published on the Web and therefore have to be linked with other data, e.g. with data published in the Linking Open Data (LOD) project [103]. External ontologies can be aligned by using *owl:equivalentClass* on class level, *owl:equivalentProperty* on property level and *owl:sameAs* on instance level. These alignment statements are placed into an alignment module, so that the system that uses PMO can choose whether to align PMO with external ontologies or not. When an alignment module is used, it is further up to the underlying inference system, whether the external ontologies are added to the terminological assertions (T-Box) of the reasoner or not. If external ontologies are not added to the reasoner's T-Box, PMO remains of type OWL RDFS/Plus (see section 2.2).

6. **Modularization:** The PMO is organized in several modules. Each module models a certain patent metadata aspect. In the next sections the PMO modules are described in more details.

3.4 Identification Module (PMO-I)

The PMO identification module (PMO-I) provides core properties and classes for the identification of patent documents (see section 2.1.1) and a scheme for mapping patent document identifiers to RDF URI references (section 2.2.1).

In PMO-I each patent document is represented by an instance of the class *pmo:PatentDocument*. Patent applications are modeled as instances of the *pmo:PatentApplication* class. Other documents, such as cited non-patent literature, oppositions, etc. are represented by instances of *pmo:NonPatentDocument*. The classes *pmo:PatentDocument* and *pmo:NonPatentDocument* are disjoint. PMO-I defines two relations between patent documents and patent applications. One is *pmo:applicationReference*, which associates a patent application with one of its publications, and the inverse relation *pmo:publicationReference*. Each patent document is filed at or published by a certain patent authority and has a kind-of-document type associated. Figure 3.2 gives an overview of the PMO-I vocabulary.

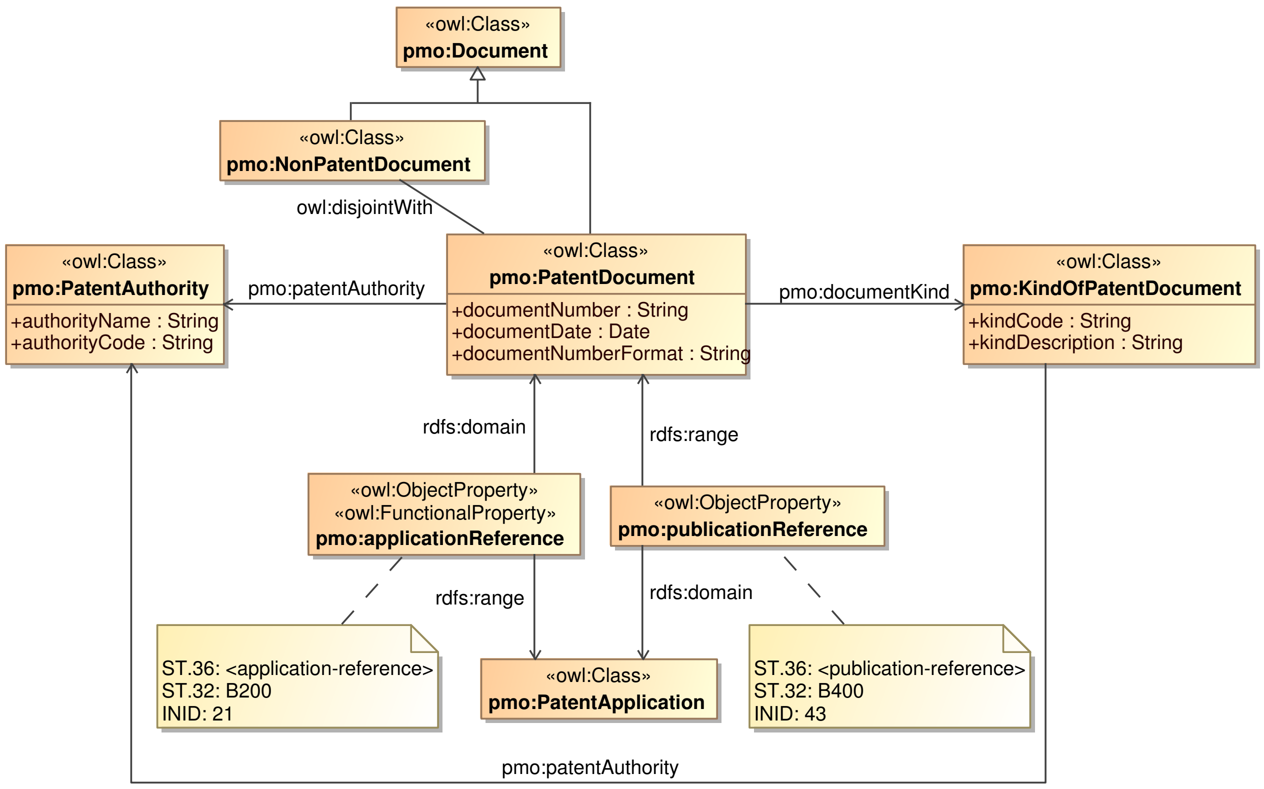


Figure 3.2: PMO-I vocabulary

3.4.1 Mapping of Patent Documents to URIs

Published Patent Documents

Each *pmo:PatentDocument* individual is encoded by a URI reference. In the following, it is specified how URI references for published patent documents and patent applications are constructed. This is essential for a correct and efficient population of the ontology. Based on the standards described in section 2.1.1 PMO-I defines the URI mapping as the concatenation of the following elements:

1. The namespace URI *http://pat.patexpert.org/* (prefixed as *pat*). It is important to note that as prefix separator the slash character (/) is used rather than the hash (#). There is an ongoing discussion of how to separate namespaces in RDF URI references (cf. [140]). The reason why the slash has been used is because it allows appending an XPath expression as shown in section 6.2.2.

2. Office code
3. Publication number
4. Kind-of-document code

Example: A patent document with office code *EP*, publication number *01301662* and kind-of-document code *B1* is mapped to the following URI:

http://pat.patexpert.org/EP01301662B1 or *pat:EP01301662B1* for short.

This mapping has the advantage that all ST.1 conform patent identifiers can be directly mapped to RDF URI references to be used as subjects or objects in RDF triples. This mapping is also the basis for an efficient merging of patent metadata from diverse sources. By using RDF URI references instead of RDF blank nodes the merging is significantly faster, since it can be realized as a simple set union operation on the underlying triple sets rather than depending on inferences.

Patent Applications

RDF URI references for patent applications are constructed the same way by appending the application number to the *http://pat.patexpert.org/* namespace. Per definition each patent publication has exactly one associated patent application. But depending on the data source a patent application can have different numbers which will cause different patent application instances. Therefore a mechanism for indicating the equality of different patent application instances is needed. A common approach in OWL is to introduce *owl:sameAs* assertions between instances that are defined as equal. The following RDF triples state that the instances *pat:01301662.1*, *pat:01301662* and *pat:EP20010301662* are pairwise equal.

```
pat:01301662.1  owl:sameAs  pat:01301662  .
pat:01301662  owl:sameAs  pat:EP20010301662  .
pat:EP20010301662  owl:sameAs  pat:01301662.1  .
```

But when merging patent applications from different sources, this would require additional knowledge that is not always contained in the particular data fragment. A solution for this merging problem is to logically infer *owl:sameAs* assertions by using a reasoner. This is reached by defining *pmo:applicationReference* as functional property (by making it a sub-property of *owl:FunctionalProperty*). Together with the defined URI mapping and numbering scheme for patent publications it is now possible to infer *owl:sameAs* assertions between patent application individuals automatically.

Example: Based on the PMO-I terminology and the two assertions:

```
pat:EP01301662B1  pmo:applicationReference  pat:01301662  .
pat:EP01301662B1  pmo:applicationReference  pat:EP20010301662  .
```

the following assertions will be inferred by an OWL reasoner:

```
pat:01301662  a  pmo:PatentApplication  ;
          pmo:publicationReference  pat:EP01301662B1  ;
          owl:sameAs  pat:EP20010301662  .
pat:EP20010301662  a  pmo:PatentApplication  ;
          pmo:publicationReference  pat:EP01301662B1  ;
          owl:sameAs  pat:01301662  .
```

For requesting all publications of a specific patent application also all applications that are connected by *owl:sameAs* should be considered, especially when they are linked with different patent publications. As an example for a corresponding SPARQL query the following query binds all publications associated with an application *pat:EP20010301662* to the variable *?pat*.

```
SELECT  DISTINCT  ?pat
WHERE  ?app  owl:sameAs  pat:EP20010301662  ;  pmo:publicationReference  ?pat  .
```

3.4.2 Kind of Document Codes

As described in section 2.1.1 the semantics of kind-of-document codes depend on the authority which has published the patent document. Therefore, they always have to be interpreted together with the corresponding authority. This is the reason why in PMO-I *pmo:KindOfPatentDocument* instances have the three properties *pmo:patentAuthority*, *pmo:kindCode*, and *pmo:kindDescription*. The latter is a human readable form of the meaning of the code. The machine readable form is defined by the following URI mapping:

1. The namespace URI *http://pmo.patexpert.org/* prefixed as *pmo*

2. Office code
3. Kind-of-document code separated by a dash (-)

Example: The URI mapping of a kind-of-document code *A2* defined by the European Patent Office *EP* would be:

> *http://pmo.patexpert.org/EP-A2* or *pmo:EP-A2* for short.

3.4.3 Identification Properties

A URI per definition does not encode any domain specific information. Therefore PMO-I introduces the following properties and classes to explicitly model the information for identifying patent documents:

- *pmo:patentAuthority* a reference to the national, regional or international patent organization at which the patent document has been filed or published. Each patent authority is modeled as an instance of the *pmo:PatentAuthority* class and has a name and code property. The 'European Patent Office', for example, has the code 'EP'.
- *pmo:documentKind* a reference to a *pmo:KindOfDocument* instance that represents the kind-of-document code of the patent document.
- *pmo:documentDate* the date when the document has been filed or published
- *pmo:documentNumber* the application or publication number
- *pmo:documentNumberFormat* an optional information about the used format for document numbers, e.g. 'epodoc', 'docdb', or 'original'

3.5 Bibliographic Data Module (PMO-B)

The PMO Bibliographic Data Module (PMO-B) defines the terminology for representing bibliographic data as shown on a patent's front page. PMO-B has been constructed based on the WIPO Standards ST.9 [288], ST.32 [283] and ST.36 [287] which have been described in section 2.1.1.

PMO-B categorizes the different bibliographic metadata elements into the following five groups. This categorization is basically motivated by semantic similarity of the contained elements. The groups are described in the following sections.

1. References: This group defines references between patent documents and other documents. Most important references are priority claims and citations.

2. Dates: This group contains a set of dates each having a distinct semantic. An overview is given in table 3.2.

3. Persons or Organizations: This group defines the terminology for modeling parties involved in the patent process, for example applicants, inventors, attorneys, opponents, examiners, or licensees.

4. Text Contents and Languages: This group defines properties for linking the plain textual contents, such as titles and abstracts, in various languages. It also provides terms for modeling in which languages an application has been filed or which translations are available.

5. Countries: The terminology for associating information about the countries involved in the patent process, for example filing country, or designated contracting states.

3.5.1 References

This group defines the terminology for references between patent documents and between patent documents and other documents. Most important references are priority claims and citations. Figure 3.3 gives an overview.

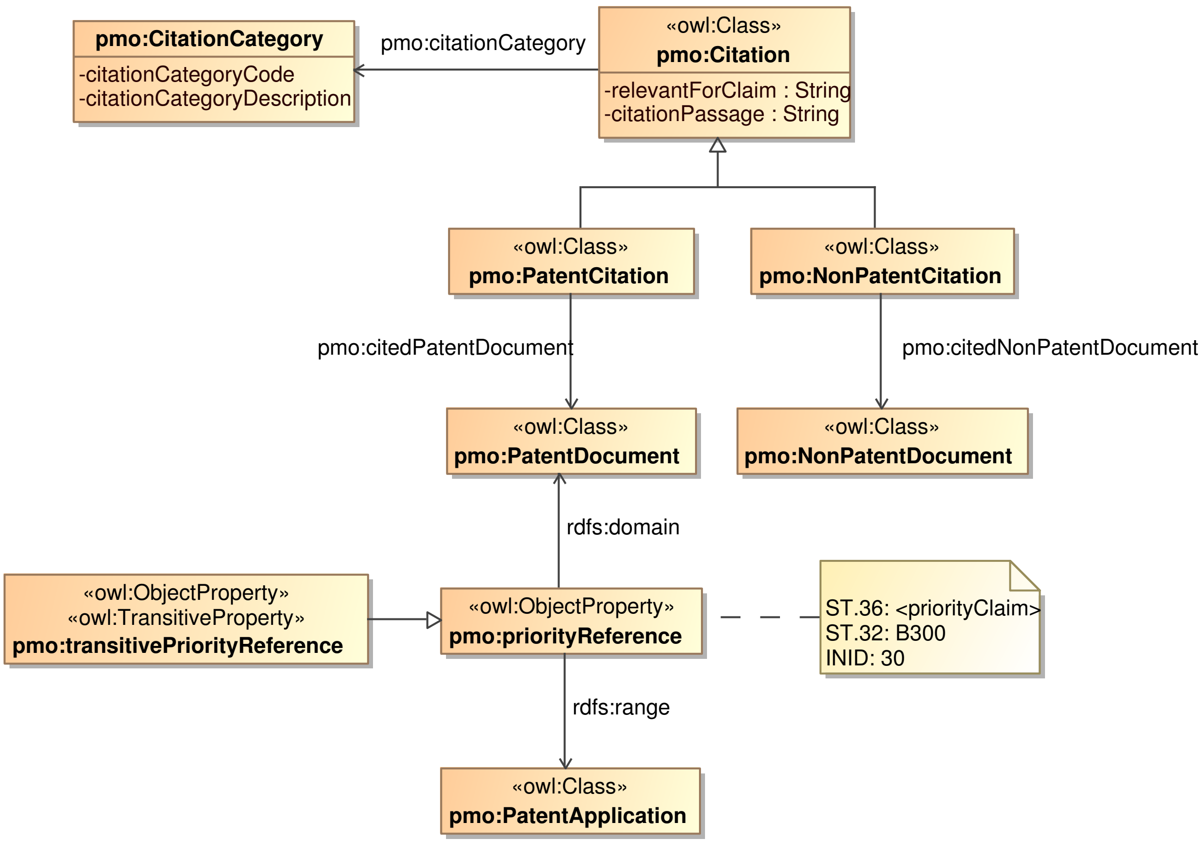


Figure 3.3: PMO-B vocabulary for references

Priority Claims

In the PMO-B priority claims are modeled as *pmo:priorityReference* object properties which indicate a correspondence to an earlier patent application. As such a *pmo:priority-Reference* relation is similar to a *pmo:applicationReference* relation. But in contrast to the latter *pmo:priorityReference* is not functional since there can be multiple priority claims for one application. It is also important to note that priority claims in general are transitive (i.e., if a is priority of b and b is priority of c then also a is priority of c). For this purpose OWL provides the *owl:TransitiveProperty* construct which indicates a reasoner to compute the transitive closure. In order to distinguish between direct priority claims and the inferred transitive closure a second relation called *pmo:transitivePriorityReference* is introduced and defined as sub-property of *pmo:priorityReference* and instance of *owl:TransitiveProperty*.

Citations

Another important type of references are citations. It can generally be distinguished between patent citations on the one hand and non-patent citations on the other. As shown in figure 3.3 citations are modeled as instances of *pmo:Citation*. The two mentioned citation types are represented by the subclasses *pmo:PatentCitation* and *pmo:NonPatentCitation* which are disjoint with each other.

Concerning the modeling of patent citation relations, there are two options. One is to use the RDF reification vocabulary (section 2.2) in order to attach additional information to the citation relation. The other is to represent the citation relation as a class. Common to both approaches is the use of blank nodes for grouping data. The interesting part is the merging of different RDF-graphs defining the same citations. For the merging based on the class-based approach inferencing is required to pairwise introduce *owl:sameAs* relations between two corresponding bank nodes. For example, this can be done by declaring the *pmo:citatedPatentDocument* relation as *owl:inverseFunctionalProperty*.

The reification approach in contrast does not need any inference because the reified triple is contained in both graphs. But as a result of the merging the reification quads are doubled. Besides this doubling of data, the reification approach has other shortcomings summarized by Sahoo et al. [213]. For example, the RDF formal semantics does not extend to the reification vocabulary. Further, the entailment rules do not hold between an RDF triple and its reification. The use of blank nodes required for reification makes it difficult to use reasoning and increases the complexity of query patterns since the queries have to explicitly take into account an extra entity. Because of this, the class based approach is used in PMO, which is also proposed as best practice for n-ary relations by Hayes and Welty [271].

To each citation instance properties can be attached, such as the category (*pmo:citationCategory*) or the claim the citation is relevant for (*pmo:relevantForClaim*). The following snippet gives an example.

```
pat:EP01301662B1-20100526
        pmo:patentCitation [
                a pmo:PatentCitation ;
                pmo:citationPassage "page 5, line 22 - page 6, line 13" ;
                pmo:citationCategory pmo:X;
                pmo:citedPatentDocument pat:GB2364924A2-20020213 ] .
```

3.5.2 Dates

The time aspect plays an important role for patent analysis. The Dates group within PMO-B models all time related data as data type properties with the XML Schema type *xsd:date*.

```
pat:EP01301662B1-20100526
        pmo:publicationDate "2010-05-26"^^xsd:date ;
        pmo:filingDate "2004-04-12"^^xsd:date .
```

Some of the dates give information about the status of a patent. This information can be used by rules for deriving the patent's status. For example, it can be derived that

- If there is a 'Date of refusal of application' (B235) the status is 'patent refused'
- If there is a 'Date of withdrawal' (B236) the status is 'patent withdrawn'
- If there is a 'Date of revocation' (B239) the status is 'patent revoked'

The ST.32 elements and their corresponding PMO-B date properties are listed in table 3.2.

3.5.3 Legal Entities

The representation of legal entities, i.e. natural and juristic persons associated with a patent, is another important aspect for the modeling of patent metadata. Each legal entity has a specific role in association with a patent, such as inventor, applicant, attorney, etc. as described in more details in section 2.1.1.

The specific roles are introduced during the ontology population process. Since one legal entity can have different roles, for example, inventors of some patents could be applicants of others, role classes are not disjoint with each other. The alignment with roles

Date related PMO-B property	**Description**	**ST.32**
exhibitionFilingDate	Exhibition filing date	B231
completeSpecificationFilingDate	Complete specification filing date	B232
receiptAtNationalOfficeDate	Receipt date at national office	B233
receiptAtInternationalOfficeDate	Receipt date at international office	B234
refusalOfApplicationDate	Date of refusal of application	B235
withdrawalDate	Date of withdrawal	B236
applicationDeemedWithdrawnDate	Date application deemed withdrawn	B237
applicationRightsReestablishedDate	Date of application rights re-established	B238
revocationDate	Date of revocation	B239
requestForExaminationDate	Date of request for examination	B241
despatchOfFirstExaminationReportDate	Date of despatch of 1st examination report	B242
patentMaintainedAsAmendedDate	Date of patent maintained as amended	B243
requestForConversionToNationalApplDate	Conversion request to nation application	B244
suspensionOrInterruptionOfProceedingsDate	Date of suspension/interrupt. of proceedings	B245
resumptionOfProceedingsDate	Date of resumption of proceedings	B246
notificationRightsAfterAppealDate	Date of notification rights after appeal	B248

Table 3.2: Overview of PMO-B date properties

is modeled in PMO-B as depicted in figure 3.4. Each role is modeled as a subclass of *pmo:LegalEntityRole*. The relation between legal entities and patent documents is defined by the *pmo:legalEntity* object property. Its domain is *pmo:PatentDocument* and its range is *pmo:LegalEntityRole*. Further, a specific sub-property is introduced for each legal entity role. This allows reasoners to infer the roles of legal entities without explicitly defining them during the ontology population phase.

Example: The following SPARQL query selects all legal entities associated with a given patent independent of their roles:

```
SELECT ?entity
WHERE { pat:EP01301662B1-20100526 pmo:legalEntity ?entity . }
```

The next query selects all legal entities that have the role 'inventor' or 'applicant':

```
SELECT ?entity
WHERE {
  { pat:EP01301662B1-20100526 pmo:inventor ?entity . }
  UNION
  { pat:EP01301662B1-20100526 pmo:applicant ?entity . }
}
```

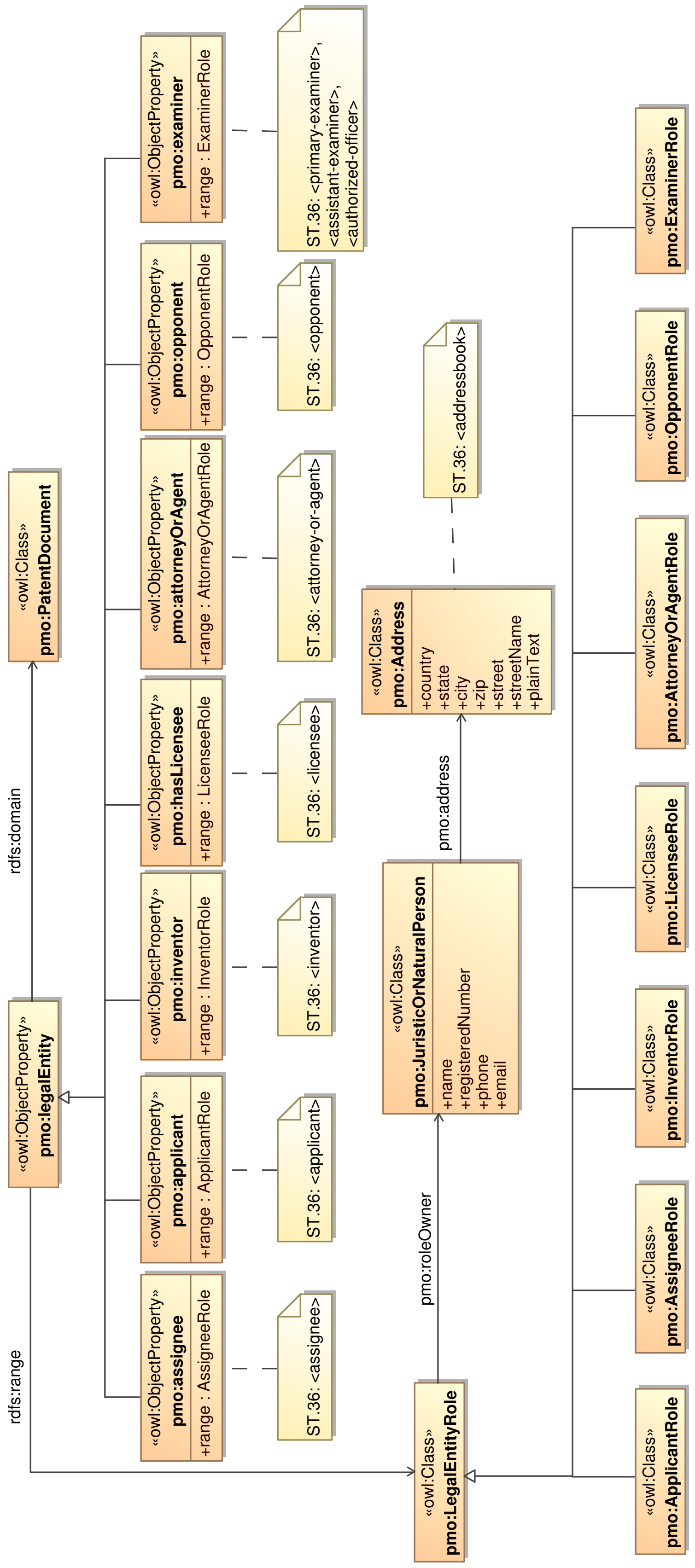

«owl:Class» pmo:PatentDocument
rdfs:domain
«owl:ObjectProperty» pmo:legalEntity
rdfs:range
«owl:ObjectProperty» pmo:examiner +range : ExaminerRole
«owl:ObjectProperty» pmo:opponent +range : OpponentRole
«owl:ObjectProperty» pmo:attorneyOrAgent +range : AttorneyOrAgentRole
«owl:ObjectProperty» pmo:hasLicensee +range : LicenseeRole
«owl:ObjectProperty» pmo:inventor +range : InventorRole
«owl:ObjectProperty» pmo:applicant +range : ApplicantRole
«owl:ObjectProperty» pmo:assignee +range : AssigneeRole
ST.36: <primary-examiner>, <assistant-examiner>, <authorized-officer>
ST.36: <opponent>
ST.36: <attorney-or-agent>
ST.36: <licensee>
ST.36: <inventor>
ST.36: <applicant>
ST.36: <assignee>
ST.36: <addressbook>
«owl:Class» pmo:Address +country +state +city +zip +street +streetName +plainText
pmo:address
«owl:Class» pmo:JuristicOrNaturalPerson +name +registeredNumber +phone +email
pmo:roleOwner
«owl:Class» pmo:LegalEntityRole
«owl:Class» pmo:ExaminerRole
«owl:Class» pmo:OpponentRole
«owl:Class» pmo:AttorneyOrAgentRole
«owl:Class» pmo:LicenseeRole
«owl:Class» pmo:InventorRole
«owl:Class» pmo:AssigneeRole
«owl:Class» pmo:ApplicantRole

Figure 3.4: PMO-B vocabulary for legal entities

3.5.4 Text and Languages

The PMO-B group for text and languages defines properties for linking the plain contents, such as titles, abstracts, etc. in various languages. It also provides terms for modeling in which languages an application has been filed or which translations are available. Table 3.3 gives an overview of the PMO-B text and language properties.

It is important to mention that only the titles are stored in the knowledge base. The content of a patent is not directly included in order to keep the knowledge base lean and scalable. Instead the content is linked using an XPointer as described in section 6.2. For the definition of titles PMO uses RDF literals which allow specifying the language of a literal.

Property	**Description**	**ST.32**
filingLanguage	Language in which the application was originally filed	B250
publicationLanguage	Language in which the application is published	B260
title	Title of the invention	B541, B542
abstract	XPointer reference to the abstract of the invention	-
description	XPointer reference to the description of the invention	-
claims	XPointer reference to the claims of the invention	-

Table 3.3: Overview of PMO-B text and language properties

3.5.5 Countries

The PMO Bibliographic Module group for Countries defines the terminology for associating information about the countries involved in the patent process. This includes the filing country for national patent applications or the designated states for PCT or European patent application. Table 3.4 gives an overview of the PMO-B country properties. The domain of a country property is *pmo:PatentDocument*. The range of a country property is the union of *pmo:Country* and *pmo:PatentOrganization*.

3.6 Patent Family Module (PMO-F)

Concerning the modeling of patent families there are basically two different approaches: relation-based and class-based modeling. In a *relation-based modeling approach* a patent

Property	**Description**	**ST.32**
designatedPCTState	Designated State according to the PCT	B810
electedPCTState	PCT Elected State	B820
designatedContractingState	Designated Contracting State under regional patent conventions	B840
countryOfApplicant	The original country of the applicant in order to distinguish residents from non-resitients	-

Table 3.4: Overview of PMO-B country properties

family is implicitly modeled by relations between family members. Different patent family definitions can be modeled by using different family relations. Depending on the family type logical inferences can derive implicit transitive and symmetric relations. Shortcomings of this approach are that inferences are expensive and equivalent families cannot be defined this way.

In a *class-based modeling approach* a patent family is explicitly modeled as individual with its family members associated. There are two challenges with this approach. First, patent families are dynamic, i.e. they can grow over time. Second, there are no agreed unique identifiers for patent families, so that merging different family definitions can be difficult. As a solution for the latter problem a naming convention can be introduced to define a unique serialization of priorities. As solution for the first problem is the introduction of *owl:sameAs* assertions between changed patent family individual. Both solutions will be described below.

To combine the advantages of both approaches PMO-F uses a combination of class-based and relation-based modeling. Single priority families and extended families are modeled using relations, whereas equivalent families are modeled explicitly as instances of *pmo:PatentFamily*. A typical query on the metadata knowledge base is to find all patents that are in the same family as a given patent. For the three patent family definitions the corresponding SPARQL requests are described in the following.

```
# Get single priority family of pat:EP01301662B1-20100526
SELECT DISTINCT ?pat ?prio
WHERE   {
  ?x pmo:priorityReference ?prio .
  ?pat pmo:priorityReference ?prio .
  FILTER (?x = pat:EP01301662B1-20100526) }
```

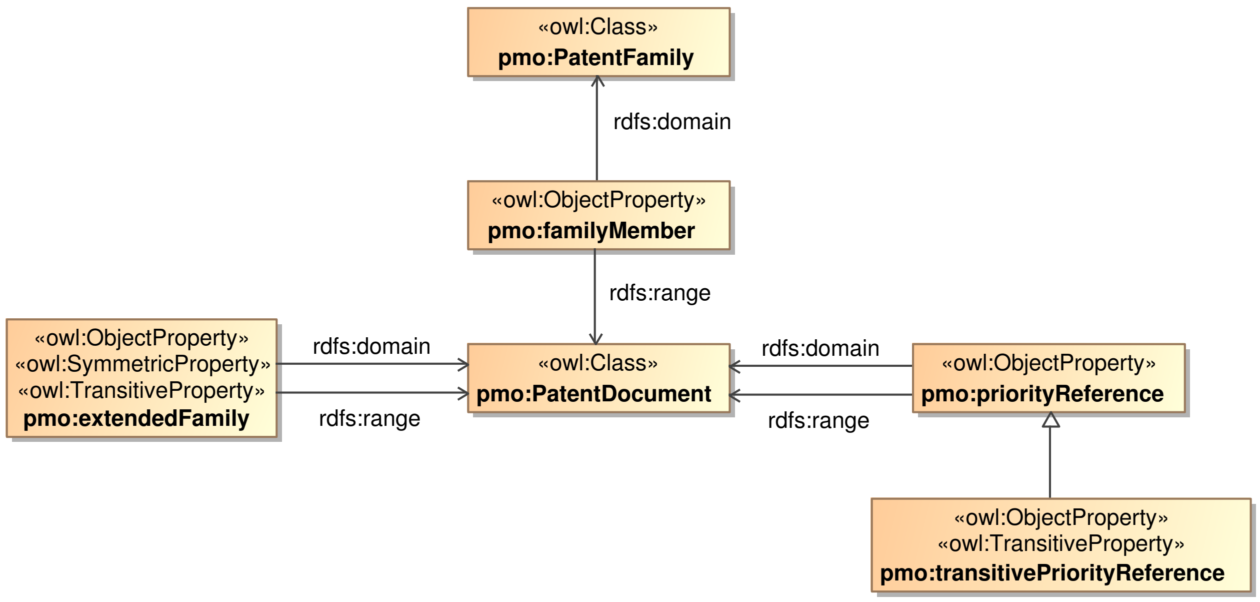


Figure 3.5: PMO-F vocabulary

```
# Get extended family of pat:EP01301662B1-20100526
SELECT  DISTINCT ?pat ?fxprio
WHERE   {
  ?x pmo:priorityReference ?prio .
        ?prio pmo:extendedFamilyPriority ?fxprio .
        ?pat pmo:priorityReference ?fxprio .
  FILTER (?x = pat:EP01301662B1-20100526) }
```

```
# Triadic patent families
SELECT ?pat
WHERE {
  ?x pmo:transitivePriorityReference ?pat ;
     pmo:authority pmo:US, pmo:EP, pmo:JP .
  FILTER (?x = pat:EP01301662B1-20100526) }
```

Modeling patent family definitions that are manually defined by human experts (cf. [183]) would be possible by adding a new family relation to PMO-F and manual semantic annotation. This type of patent family information has to be considered as implicit and external metadata since it only can be established by considering external sources.

As described in section 2.1.1 the WIPO ST.36 standard allows listing for each patent its corresponding patent family members. But it is important to note that this only allows taking a snapshot of the current state. As soon as new patent family members come up, this definition is outdated. Therefore a semantic modeling of patent families as described above has a clear benefit.

3.7 Classification Module (PMO-C)

The PMO classification module (PMO-C) provides the terminology for representing general hierarchical classification schemes and for linking patent documents with entries of these schemes. In order to be extensible and to support the construction of custom schemes, PMO-C only defines higher level concepts and relations that are to be specialized by concrete profiles. An example of such a profile for the International Patent Classification (IPC) is given bellow. Currently, the IPC is the most important classification scheme used by all major patent offices for the classification of their patents. An overview of the IPC is given in section 2.1.2. The possibility for defining customized classification schemes is an important aspect, in particular for highly specialized companies that need a more detailed classification scheme then provided by the IPC but restricted on a few technical fields only.

The core class of PMO-C is *pmo:ClassificationEntry*. It allows the definition of hierarchically organized entries. Each entry except the root entry has one parent and can have an arbitrary collection of child entries. The *pmo:childIndex* property is for specifying the order of the children. The *pmo:hierarchyLevel* property defines the depth of an entry in the classification hierarchy. The *pmo:ClassificationEntryRole* class defines the functional role of an entry (see IPC profile bellow for specialized entry roles). Patent documents are linked with classification entries using two object properties: *pmo:classifiedAs* and *pmo:indirectlyClassifiedAs*. While the first states that a patent document has been classified exactly with the associated classification entry, the latter indicates an indirect correspondence, for example for linking all ancestors along the classification hierarchy up to the root. For the definition of labels, references and comments the standard RDFS properties *rdfs:label*, *rdfs:seeAlso*, and *rdfs:comment* are used.

IPC Profile

The PMO-C IPC profile uses a separate namespace *http://ipc.patexpert.org/* with prefix *ipc*. Each IPC entry is modeled as instance of *ipc:ClassificationEntry*. Its role is

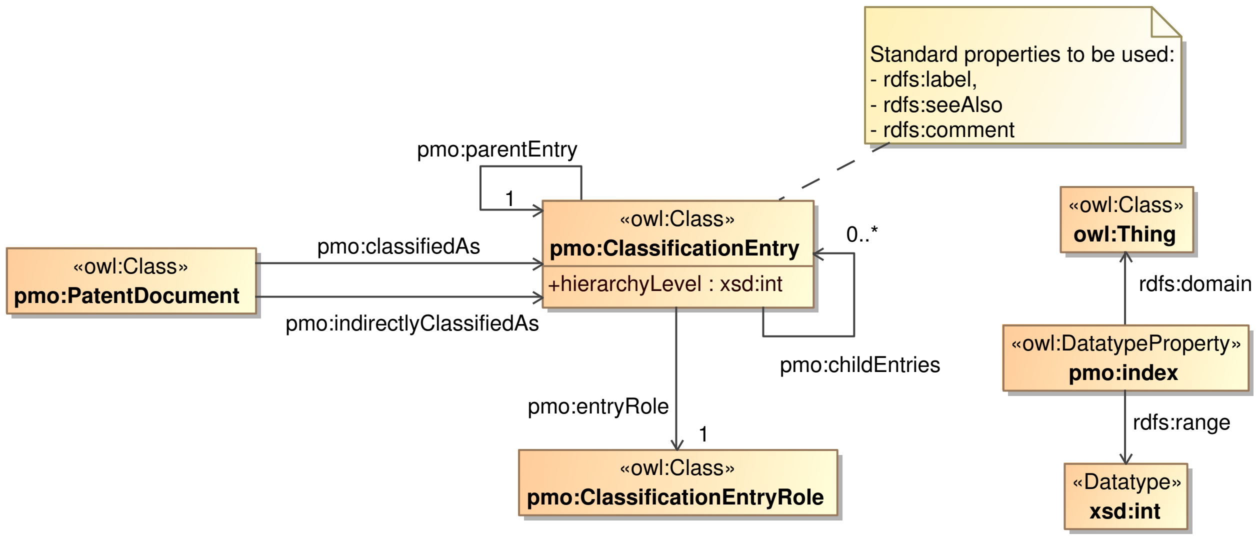


Figure 3.6: PMO-C vocabulary

defined by associating a specific *ipc:ClassificationRole* instance, for example (*ipc:Section* for sections, *ipc:Class* for classes, or *ipc:SubClass* for subclasses.

As mentioned before, IPC titles may be composed of several parts and also may contain references. Because of this, IPC titles are explicitly modeled as collections of *ipc:TitlePart* instances. Each such title part may reference one or more *ipc:Classification-Entry* instances using the *ipc:Reference* association class. The latter provides a container for adding additional properties, such as the context or the IPC level of the reference. With this model detailed analyses of the IPC structure are much easier than parsing references from plain text titles. For applications that do not need this granularity titles can also be provided as plain text using the standard *rdfs:label* property. It is important to note that the *pmo:index* property is used to define the order of classification entries, title parts and references rather than using RDF collections. The reason is to simplify queries on the IPC representation.

The mapping of IPC codes to URLs is straightforward by using zero digits as padding and removing the slash character, for example ‚H01S 1/00’ becomes ‚H01S0001000000’. Guidance headers and notes are represented as blank nodes.

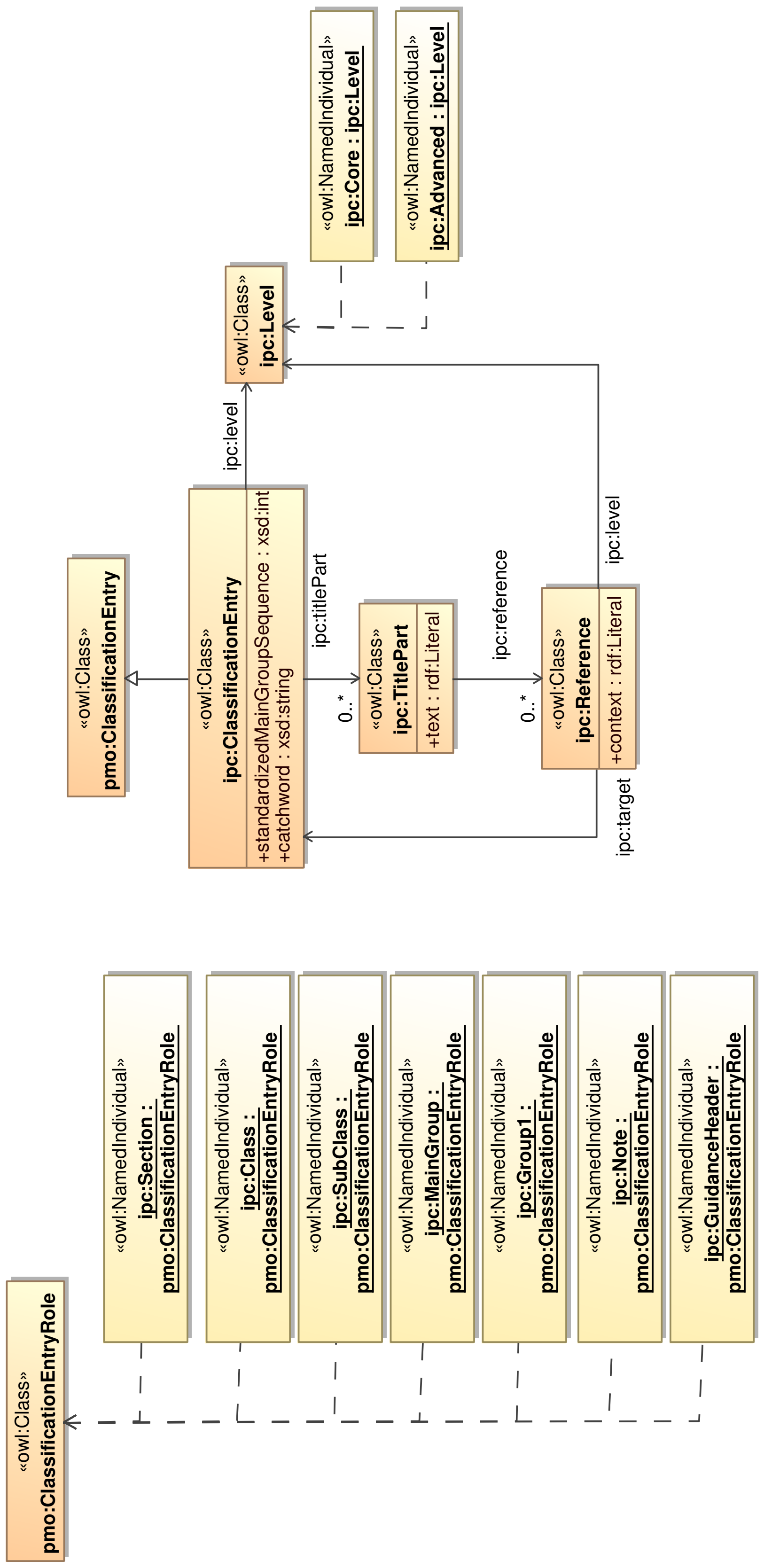


Figure 3.7: PMO-C vocabulary for IPC profiles

Example: The following code shows a fragment of the IPC representation:

```
@prefix ipc:  <http://ipc.patexpert.org/> .
@prefix pmo:  <http://pmo.patexpert.org/> .
@prefix xsd:  <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

ipc:H01S a ipc:ClassificationEntry ;
        rdfs:label "H01S" ;
        pmo:entryRole ipc:SubClass ;
        pmo:hierarchyLevel "3"^^xsd:int ;
        pmo:index "14"^^xsd:int ;
        pmo:parentEntry ipc:H01 ;
        ipc:level ipc:Core ;
        ipc:titlePart [
                ipc:titleText "DEVICES USING STIMULATED EMISSION"^^en .
        ] ;
        pmo:childEntry [ a ipc:ClassificationEntry ;
                pmo:entryRole ipc:Note ;
                pmo:index "1"^^xsd:int ;
                ipc:titlePart [ a ipc:TitlePart ;
                        pmo:index "1"^^xsd:int ;
                        ipc:text "This subclass covers devices for the generation..."^^en .
                ] ;
        ] ;
        pmo:childEntry [ a ipc:ClassificationEntry ;
                pmo:entryRole ipc:GuidanceHeader ;
                pmo:index "2"^^xsd:int ;
                ipc:titlePart [ a ipc:TitlePart ;
                        pmo:index "1"^^xsd:int ;
                        ipc:text "MASERS"^^en ;
                        ipc:reference [ ipc:target ipc:H01S0001000000 . ]
                ] ;
                # ...
        ] ;
        pmo:childEntry ipc:H01S0001000000 ;
        pmo:childEntry ipc:H01S0003000000 ;
        pmo:childEntry ipc:H01S0004000000 ;
        pmo:childEntry ipc:H01S0005000000 .
```

Currently, the PMO-C IPC profile does not consider (i) revision concordance lists which give information on how subject matter has been transferred between different places in the IPC as a result of its revisions, (ii) chemical formulas or graphic illustrations which have been introduced for illustrating classification entries or explaining them in more detail, and (iii) a catchword index which provides further assistance for finding relevant classifications.

3.8 Conclusions

In this chapter an ontology-based approach for semantic representation of patent metadata has been described. The proposed *Patent Metadata Ontology (PMO)* provides a detailed terminology for all major patent metadata aspects. PMO is based on Semantic Web standards, in particular on OWL as ontology language. Since patent metadata are an integral part of almost every patent search task (see section 2.1.3), it is important to integrate them into an overall semantic representation framework in order to allow the formulation of metadata restrictions. PMO fills this gap in the PATExpert ontology framework.

Furthermore, PMO provides a homogeneous semantic view on patent metadata and can serve as «lingua franca» for integrating heterogeneous patent metadata sources. It is assumed that for each data source an adapter is provided which translates the input data into the representation that has been described in this chapter.

Additionally, PMO considers the standardization efforts of the World Intellectual Property Organization (WIPO) by providing a mapping to the terms used in the WIPO ST.32 and ST.36 standards (see section 2.1.1) and thus promotes practicability and understandability for common use.

The alignment with upper level ontologies, such as SUMO, is not part of the PMO core vocabulary. One reason is that corresponding upper level class concepts may have a large number of generalized class concepts. For example, the class concept «sumo:Patent» is transitively inherits from ten more general class concepts. When «pmo:Patent» inherits from «sumo:Patent» many additional triples will be introduced, when inferred statements are materialized. Thus, the used approach is to let applications decide, whether they need alignment with upper level concepts or not. As best practice it is recommended to add upper level classes only, if they are required for reasoning and to omit them otherwise.

Due to the large number of available patents, scalability has been an important requirement. Therefore, the OWL RDFS/Plus subset (see section 2.2.3) has been used for defining the ontology. It is important to mention that alignment with other ontologies can change this language type and complexity class. For example, alignment with an OWL full ontology will cause PMO to be also of type OWL full.

PMO has been divided into four ontology modules: *Patent Identification Module (PMO-I)*, *Bibliographic Data Module (PMO-B)*, *Patent Family Module (PMO-F)*, and *Patent Classification Module (PMO-C)*. Each module defines a subset of the terminology and covers a specific patent metadata aspect. PMO can be extended using the OWL standard mechanisms: either by adding new modules or by adding new concepts to existing modules. It is notable that the added concepts can be in any namespace, since the use of multiple schemes and versions is one of the key features of OWL and RDF.

For evaluation purposes, PMO has been populated with instances based on a set of 8,513 patent documents from IPC class B23. The patents have been downloaded from the *Open Patent Services* [21] and transformed into RDF using the PMO terminology. A detailed description of the transformation process and statistical evaluation can be found in the diploma thesis of Stäbler [230].

A more detailed evaluation with criteria suggested in [248, 117] has not been done as part of this work and is subject of future research. Furthermore, PMO currently only considers basic rules, as described in section 3.4.1. Investigating rules and rule languages, e.g. for deriving the legal state of a patent, would be another interesting topic for future research.

CHAPTER 4

Partial Encryption of Semantic Annotations

Giving information a well-defined meaning is on one hand the basis for intelligent applications in an emerging Semantic Web, but on the other hand can have profound consequences when considering privacy, security, and intellectual property rights issues. In the Semantic Web vision agents automatically gather and merge semantically annotated data, infer new data and re-use the data in different contexts [70]. However seemingly harmless pieces of data could reveal a lot of information when combined with others. In the Semantic Web there will also be the need of integrating data which is sensitive in some contexts.

Therefore, methods for specifying *who* is allowed to use *which* data are important in the next step towards the Semantic Web. There are two approaches to achieve this. The first is to specify access rights, to control the data access and to secure the communication channel when the data is transferred. The second attempt is to use cryptographic methods to protect the sensitive data itself.

4.1 Related Work

There has been a considerable amount of work about access control for the Web (cf. [64, 278]). However, all these approaches need trustworthy infrastructures for specifying and controlling the data access. If sensitive data is stored in (potentially) insecure environments, such as public web-spaces, shared desktop systems, mobile devices, etc. the only way to do this is to locally encrypt the data before uploading or storing it. The ability to merge distributed data and to re-use the data have been important design aspects for the Semantic Web. From that perspective, partial encryption – where only the sensitive data are encrypted while all other data remain publicly readable – is desirable.

A common practice for encrypting sensitive data in an RDF graph is to cut the data from the original graph, store the data in a separate file, encrypt the file and finally link the encrypted file to the original graph. This approach has some shortcomings: (1) the original RDF graph is separated into different physical resources, (2) the encrypted files are not RDF compliant and therefore could not be consistently processed by common RDF frameworks, (3) the linking has to be done manually, and (4) no rules are given for re-integrating the data into the RDF graph after decryption. Another practice for encrypting RDF graphs is to serialize them in XML and to use XML Encryption [251] and XML Signature [263] based security frameworks. One problem with this approach, is the structural difference between the tree-based XML Information Set data model [258] and the graph-based RDF Abstract Syntax data model [257]. Another problem is that this approach only allows to handle XML serializations of RDF graphs.

To address these problems, we propose a method for *partial RDF encryption (PRE)* which allows for fine-grained encryption of arbitrary fragments of an RDF graph without creating additional resources. Both encrypted data and plaintext data are represented in a single RDF compliant model together with the metadata describing the encryption parameters. PRE uses the XML Encryption and XML Signature standards to represent the encryption metadata. This work is partly based on the diploma thesis of Zhou [299].

The rest of this chapter is organized as follows. In the next section gives an overview of the partial encryption process for RDF graphs. The subsequent sections look at important realization aspects: encryption and decryption of RDF fragments (section 4.3), description of graph transformations necessary to keep encrypted graphs RDF compliant (section 4.4), a graph pattern based method for dynamic selection fragments to be encrypted and a notion of encryption policies (section 4.5). The last section summarizes the partial RDF encryption approach and gives an outlook to future work.

4.2 Partial RDF Encryption

A *directed labeled graph (DLG)* encodes two different types of information: structural information and label information. Encrypting structural information means to hide the topology of the graph, whereas encrypting the label information means to hide individual node and arc values. With regard to RDF graphs, label information is encoded by the URI-references and literals of subjects, predicates and objects. Structural information is encoded in terms of triples. It should be noted that blank nodes only provide structural information but no label information.

RDF graphs can be interpreted as restricted DLGs having the following properties: (1) structural and label information of nodes are both encoded in terms of URI-references and literals – changing a node label also changes the structure of the graph; (2) node labels can be distributed over several triples. Thus, changing a node label can cause several triples of an RDF graph to be changed; (3) all nodes are connected by at least on arc – there are no isolated nodes. Thus, the encryption of a triple, can cause the encryption of the connected subject and object nodes if they are only connected by that triple; (4) RDF makes the constraint, that subject and predicate labels have to be URIs. Encrypted labels are not words of the URI language. Therefore, encrypted labels have to be represented as objects which can have arbitrary literal values. As a consequence, graph transformations have to be performed in order to keep an encrypted graph RDF-compliant.

The following three encryption types for RDF graphs can be distinguished: (1) encryption of subjects and objects (equivalent to the encryption of node labels) (2) encryption of predicates (equivalent to the encryption of arc labels) (3) encryption of triples (equivalent to the encryption of nodes, arcs and subgraphs). An arc is represented by a single triple, a node by a set of triples having the node label either as subject or object, and a subgraph can be any subset of a triple set).

Partial RDF Encryption (PRE) is a transaction which is composed of the six steps showed in Fig. 4.1. We will briefly describe each step.

1. **Fragment Selection:** The first step is the selection of the RDF fragments to be encrypted. RDF fragments can either be subjects, predicates, objects, or triples. The selected fragments are called *encryption fragments* and the remaining fragments are called *plaintext fragments*. Selection can be done, for example, by explicitly enumerating the encryption fragments (static selection), by specifying selection patterns which check specific properties (dynamic selection), by random selection, etc. This step is described in more detail in section 4.5.

2. **Encryption:** In this step, each encryption fragment is serialized and encrypted. The result of this step is a data structure containing both, encrypted data and

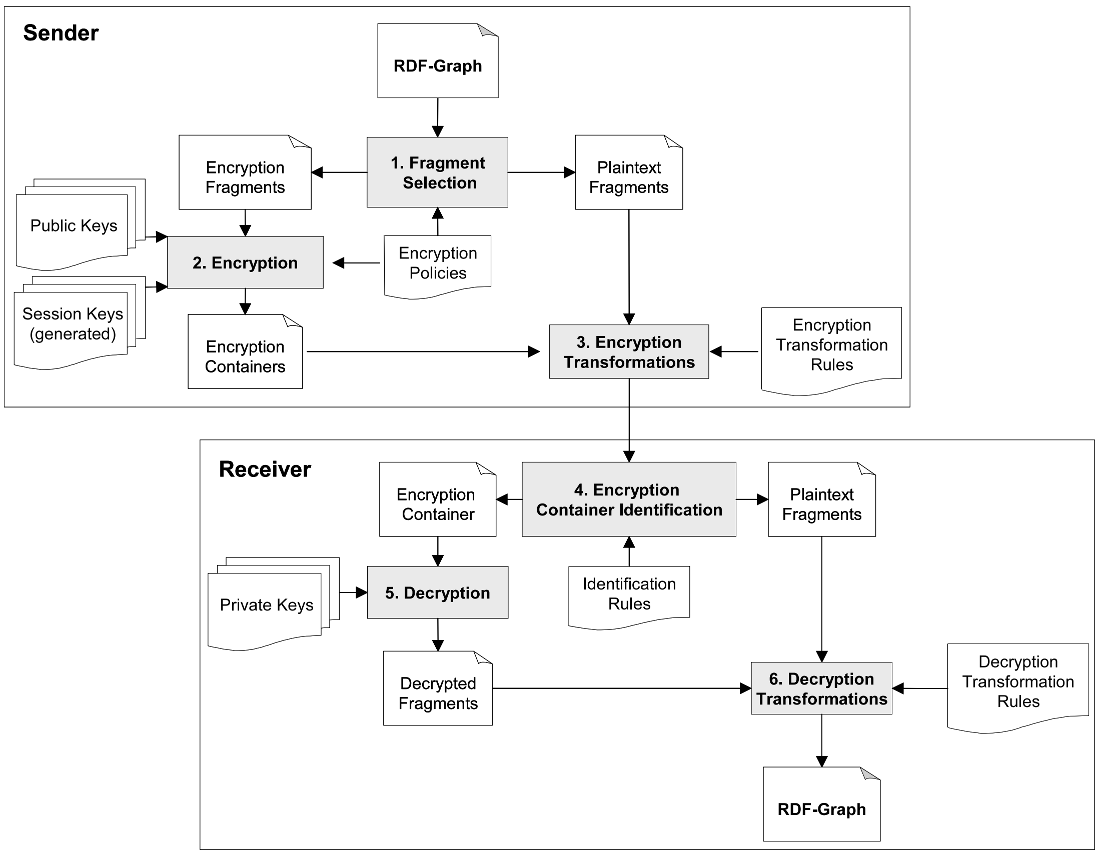


Figure 4.1: Partial encryption process

encryption metadata. We will call this structure an *Encryption Container (EC)*. An encryption container can be serialized and represented as literal value. This step is described in more detail in section 4.3.

3. **Encryption transformations:** All encryption fragments are replaced by their corresponding encryption containers. The result is a single self-describing RDF-compliant graph containing three different kinds of components: (1) encrypted data, (2) encryption metadata and (3) plaintext fragments. In order to fulfill RDF well-formedness constraints – in particular the constraint that literals are only allowed as the object of a triple – graph transformations have to be performed. This step is described in more detail in section 4.4.

4. **Encryption container identification:** Encryption containers and encryption metadata are identified and extracted. This can be done by using an RDF query language.

5. **Decryption:** In this step, the encryption containers are decrypted according to the parameters specified in the encryption metadata. If a receiver does not have an appropriate decryption key, the decryption fails.

6. **Decryption transformations:** The last step is the re-construction of the RDF graph by replacing the encryption containers with the corresponding decrypted values. Graph transformations have to be performed which are inverse to the encryption transformations in step three. If a recipient has the keys to decrypt all encryption containers, then the re-constructed RDF graph is identical to original RDF graph. In the case that keys are missing, there will be remaining encryption containers in the RDF graph.

4.3 Encryption of RDF Fragments

The normal session-key scheme (see section 2.3) can be extended to be able to encrypt a set of messages for a set of recipients. Let $M = \{m_1, \dots, m_m\}$ be a non-empty set of messages to be encrypted, $P = \{pub_1, \dots, pub_n\}$ be a non-empty set of public keys, and $P_i \subseteq P$ be a non-empty set of public keys representing the recipients of message $m_i \in M$. For each message m_i a new session key k_i is generated. m_i is encrypted using the symmetric function f parameterized by k_i. Then k_i is encrypted $|P_i|$-times using the asymmetric functions g parameterized by $pub_i \in P_i$ (Fig. 4.2). The encryption of M takes $|M|$ symmetric and $\sum_{i=1}^{|M|} |P_i| \leq |M| \cdot |P|$ asymmetric encryption function calls.

For each message, the extended session-key scheme creates a set of key ciphers $c_{k_1}, \dots, c_{k_n}$ of which at the most one can be correctly decrypted using a given private key. A naive approach would be to decrypt sequentially each key cipher and to check the integrity of the decrypted values. Providing additional information about the public keys used for encryption (such as finger prints, certificate serial number, etc.) helps to identify the corresponding private key in advance. Thus, key information is an important class of encryption metadata.

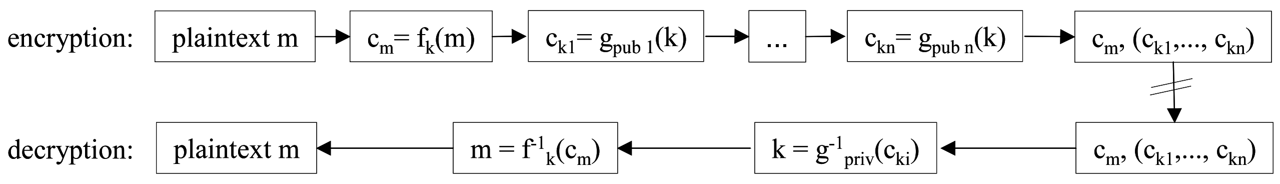


Figure 4.2: Extended session-key scheme

4.3.1 Digests

An important idea in PRE is using hash values for merging RDF graphs, similar to the inverse functional property *mbox_sha1sum* defined in the FOAF vocabulary [48]. *mbox_sha1sum* contains the digest of an email to prevent publishing the email but to allow for merging based on the email. Partially encrypted fragments of an RDF graph can be used for merging, if they (1) are object fragments, are inverse functional, and provide a direct hash value and (2) are subject fragments and provide a direct hash value.

There are cases in which it is not secure to use direct hash values, for example if the range of a property only contains few values. When using a direct hash for a 4-digit bank account PIN, it takes less than 1000 tests to know the correct PIN by comparing the hash values. In this case a randomization of the value before computing the digest is necessary. Randomized hash values provide a higher security. They still can be used for testing the data integrity but cannot be used for merging. So it is a trade-off between security and data integration. For the representation of randomized values, we use a simple XML-based method. The original fragment serialization is embedded as the content of the one and only *FragmentValue* element. It is a child of *RandomizedValue* which contains randomly generated bytes. The structure is described by the following XML Schema:

```
<xs:schema targetNamespace="http://rdfenc.berlios.de/pre-random\#"
           xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name='RandomizedValue'/>
    <xs:complexType mixed='true'>
      <xs:choice>
        <xs:element name='FragmentValue' type='xs:string'/>
      </xs:choice>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

The original fragment serialization can be retrieved using the XPath expression:

```
//*[namespace-uri()='http://rdfenc.berlios.de/pre-random#' and

local-name()='FragmentValue']/text()
```

4.3.2 Encryption Metadata

To allow for a abstract definition of the encryption process, encryption metadata has to be specified, such as the encryption algorithms and their parameters, the computed hash values, key information for public key identification, canonicalization methods, transformation to be performed, etc. The encryption metadata is stored together with the ciphers in a single data structure – the *Encryption Container (EC)*. There are different approaches to integrate encryption containers into RDF graphs. We take the approach of serializing the encryption containers into XML and including the serializations as XML literals.

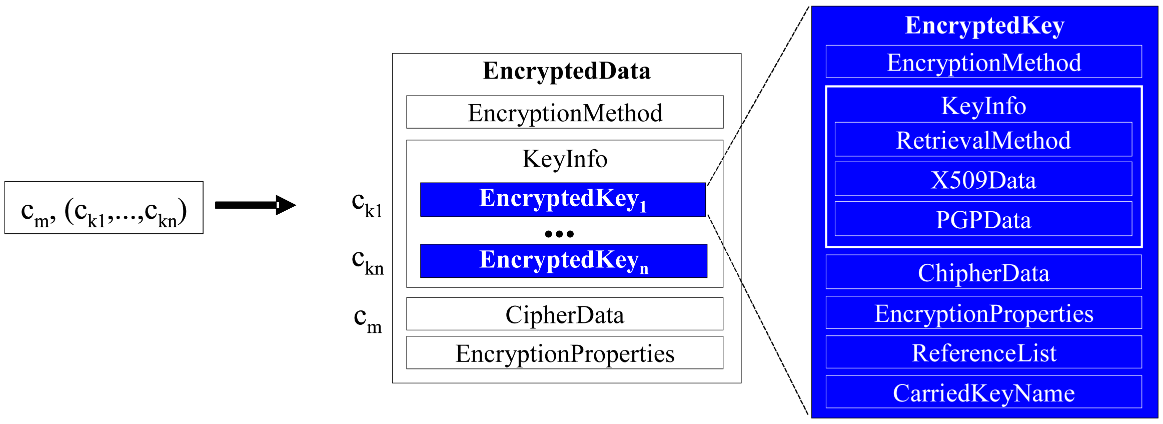


Figure 4.3: Overall encryption container structure

The general EC structure is shown in Fig.4.3. The key ciphers $c_{k_1}, \ldots, c_{k_n}$ are each stored in an *EncryptedKey* slot and the message cipher c_m is stored in an *EncryptedData* slot. Both, *EncryptedKey* and *Encrypted Data* have a similar structure. The XML-Encryption recommendation [251] provides a detailed description about the structure. The *EncryptionMethod* slot specifies the used encryption algorithm as a unique URI. The *KeyInfo* slot provides information about the key used for encrypting the cipher. When using the extended session-key scheme, the *KeyInfo* slot inside *EncryptedData* contains a sequence of *EncryptedKey* slots, whereas the *KeyInfo* slot inside *EncryptedKey* contains information about the public key, for example a certificate or a certificate reference. The

CipherData slot stores the concrete cipher value computed by the encryption function as Base64 encoded string. The *EncryptionProperties* slot contains additional information such as the digest value, the digest algorithm, data type information, the language used for serializing the data, etc.

Example 1: Alice has annotated the resource http://www.sample.de/alice.htm in RDF. To access the resource, a username and password is needed. Alice wants to store the access data together with other annotations in the same RDF graph, so that only Bob and Chris can read the access data while all other annotations are publicly readable. Alice has the X.509 certificates of Bob and Chris and wants to encrypt the following RDF triples using AES 128-bit:

```
<http://www.sample.de/alice.htm> <http://sample.de/schema#username> "alice" .
<http://www.sample.de/alice.htm> <http://sample.de/schema#password> "secret" .
```

First, the triples are serialized using an RDF language (N-Triples in this example). Second, the SHA1 digest is computed (`/84Cdz6BdYd6kY9zSa6sT1IjLoo=`). Then, the data is AES block encrypted in CBC mode with a generated 128-bit session key k. The result looks like `37++haErMYLidG...` Then, the k is RSA encrypted twice using the public keys contained in the X.509 certificates of Bob and Chris. The result looks like `rrOC4FYSNogKsi...` (for Bob). Finally, the ciphers, the digest, the certificate, and the algorithm names and parameters are combined in an encryption container. An XML-Encryption/Signature conforming serialization looks like:

```
<xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc\#"
                    xmlns:ds="http://www.w3.org/2000/09/xmldsig\#">
  <xenc:EncryptionMethod Algorithm="\&xenc;\#aes128-cbc"/>
  <ds:KeyInfo>
    \color{blue}<xenc:EncryptedKey>
      <xenc:EncryptionMethod Algorithm="\&xenc;\#rsa-1_5"/>
      <ds:KeyInfo>
        <ds:X509Data>
          <ds:X509Certificate>MIICQjCCAasCBE...</ds:X509Certificate>
        </ds:X509Data>
      </ds:KeyInfo>
      <xenc:CipherData>
        <xenc:CipherValue>rrOC4FYSNogKsi...</xenc:CipherValue>
      </xenc:CipherData>
    </xenc:EncryptedKey>
    <xenc:EncryptedKey>encrypted key of Chris...</xenc:EncryptedKey>\color{black}
  </ds:KeyInfo>
  <xenc:CipherData>
```

```
    <xenc:CipherValue>37++haErMYLidG . . . </xenc:CipherValue>
  </xenc:CipherData>
  <xenc:EncryptionProperties>
    <xenc:EncryptionProperty>
      <ds:DigestMethod Algorithm="\&ds;\#sha1"/>
      <ds:DigestValue>/84Cdz6BdYd6kY9zSa6sT1IjLoo=</ds:DigestValue>
    </xenc:EncryptionProperty>
  </xenc:EncryptionProperties>
</xenc:EncryptedData>
```

4.4 Transformations

Since in RDF only the objects can represent literal values, encrypted subjects and predicates cannot directly be replaced by their corresponding encryption container serializations. Instead, graph transformations have to be performed. An overview of the transformations for integrating the encrypted content is given in Fig. 4.4 (literals containing the encryption container serialization are marked with a 'lock' icon). We will briefly describe each transformation.

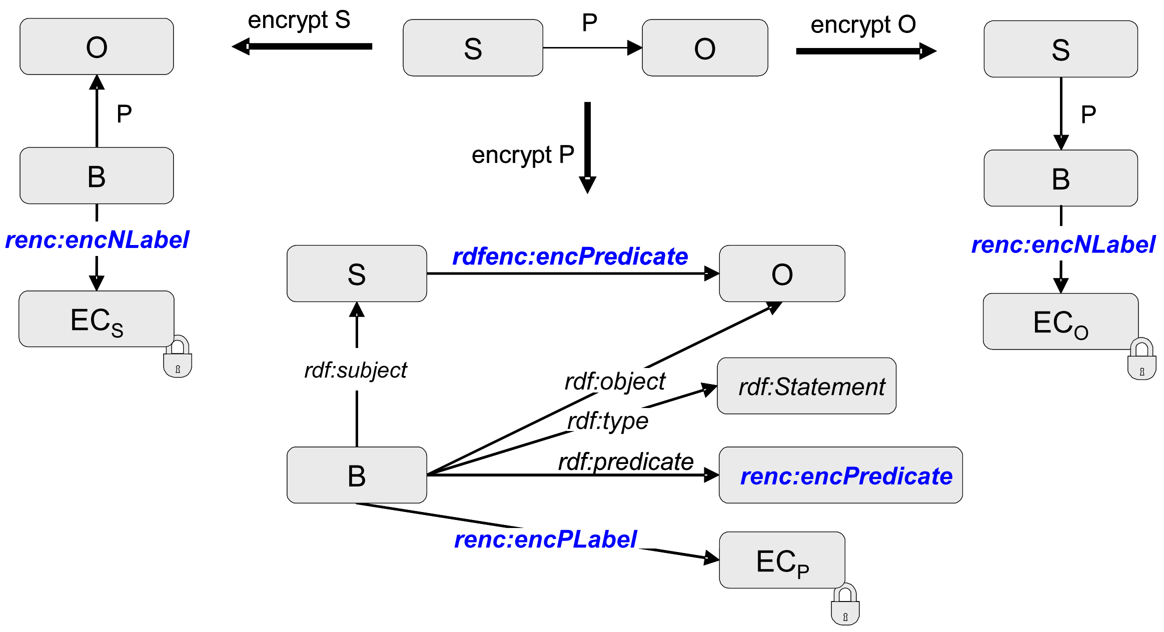


Figure 4.4: Subject, object and predicate transformations

1. *Subject transformation*: In order to encrypt a subject S, a new triple $\langle B$, *renc:encNLabel*, $EC_S\rangle$ is added to the graph. EC_S contains the XML serialization of the

encryption container of S. All references to S are replaced by references to B. Therefore all triples containing S either as object or subject have to be changed.

2. *Object transformation*: Objects could directly be replaced by their encryption container serializations. But this would also change the datatype into *rdf:XMLLiteral*. Therefore, a blank node is introduced which replaces the original object node.

   A new triple $\langle B,\ renc{:}encNLabel,\ EC_O\rangle$ is added to the graph. EC_O contains the XML serialization of the encryption container of O including the original datatype information. All references to O have to be replaced by references to B.

3. *Predicate transformation*: Since in RDF only URI references are allowed as predicates, blank nodes cannot be used for bridging between arcs and their encrypted label data. Instead a RDF reification [253] based approach is used. The transformation is carried out in three steps. First, the predicate P of the original triple t is replaced by the URI reference *renc:encPredicate*. Second, a new reification quad is added for identifying t. Finally, a new property $renc{:}encPLabel = EC_P$ is added to the reification quad stating that the real predicate of t is encrypted in EC_P.

4. *Triple set transformation*: The encryption of a non-empty triple set $T_{enc} = \{t_i, \ldots, t_{i+m}\}$ takes the following steps. First, T_{enc} is serialized into a string s using an RDF serialization language. Second, an encryption container EC_T is constructed containing the encrypted string s together with the encryption metadata. Third, a new triple $\langle B,\ renc{:}encTriples,\ EC_T\rangle$ is added to the graph. Finally, all triples in T_{enc} are removed from the graph (see fig. 4.5).

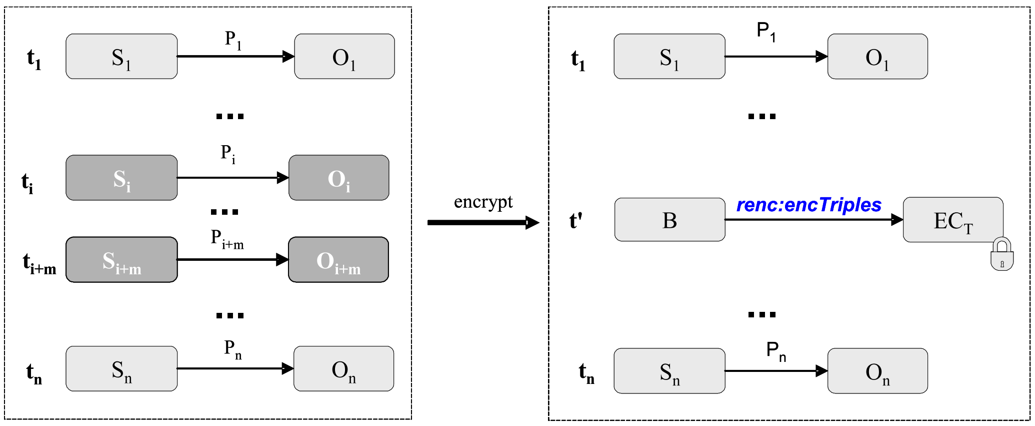


Figure 4.5: Triple set transformation

Handling of Blank Nodes

The described transformations can be directly applied to RDF graphs that do not contain blank nodes (ground graphs). As noted earlier, a blank node identifier is not regarded as node label and thus cannot be encrypted. However blank nodes may be contained in triple sets that are to be encrypted. Blank node identifiers have to be unique in one RDF graph. They are not required to be globally unique and may be changed to some internal representation by RDF frameworks. In order to be able to encrypt triples containing blank nodes, additional information is needed to uniquely identify the blank nodes after decryption, since their identifiers might have changed.

Therefore, a Universally Unique Identifier (UUID) [46] is generated for each blank node contained in a triple to be encrypted. The UUID is assigned to the blank node as URI value of the *renc:assignedURI* property. The blank nodes of the triples to be encrypted are then replaced by the generated URIs. During decryption, the generated URIs are used for identifying the original blank nodes. Blank nodes can occur as the subject of a triple, as the object of a triple or both. Fig. 4.6 gives an overview and shows the corresponding transformations.

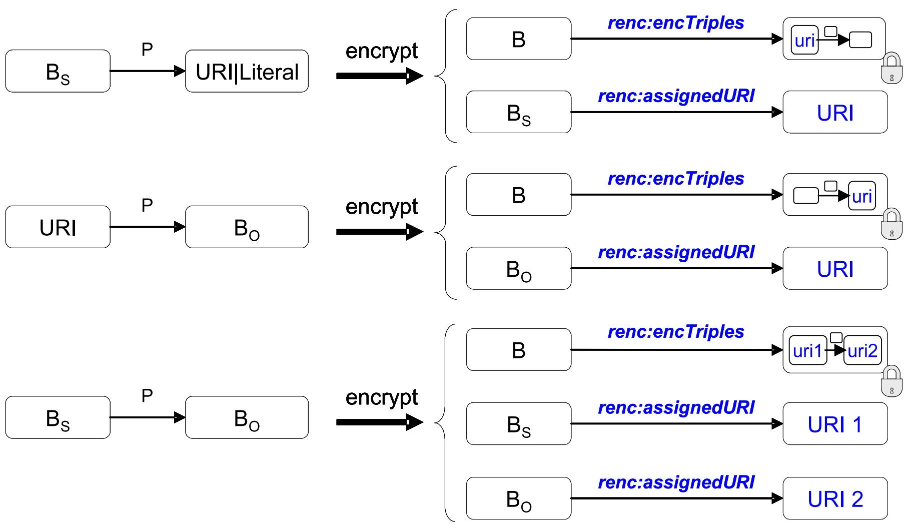


Figure 4.6: Graph transformations for blank nodes

Example 2: Alice wants to encrypt the *foaf:knows* relation between her and Bob expressed by triple t_{enc}. Since persons have no adequate URI representation, blank nodes are used for bundling properties about the person which are the email addresses in this example. Fig. 4.7 shows the result of the encryption. The triple t_{enc} is removed. Three new triples are added: two triples for identifying the blank nodes (t_1 and t_2) and one triple containing the encrypted data (t_3). The blank node identifiers for B_1 and B_2 are replaced by the generated UUIDs (uri_1 and uri_2) before the encryption. During decryption the t_3 is decrypted, parsed, and removed. Let T_{dec} denote the decrypted triples. In a second step, the objects of all triples having an renc:assignedURI predicate are tested against the subject and object URIs of the triples in T_{dec}. If a correspondence is detected (the object of t_1 with uri_1 and the object of t_2 with uri_2), the URI references are replaced by the corresponding blank nodes and the identification triples (t_1, t_2) are removed.

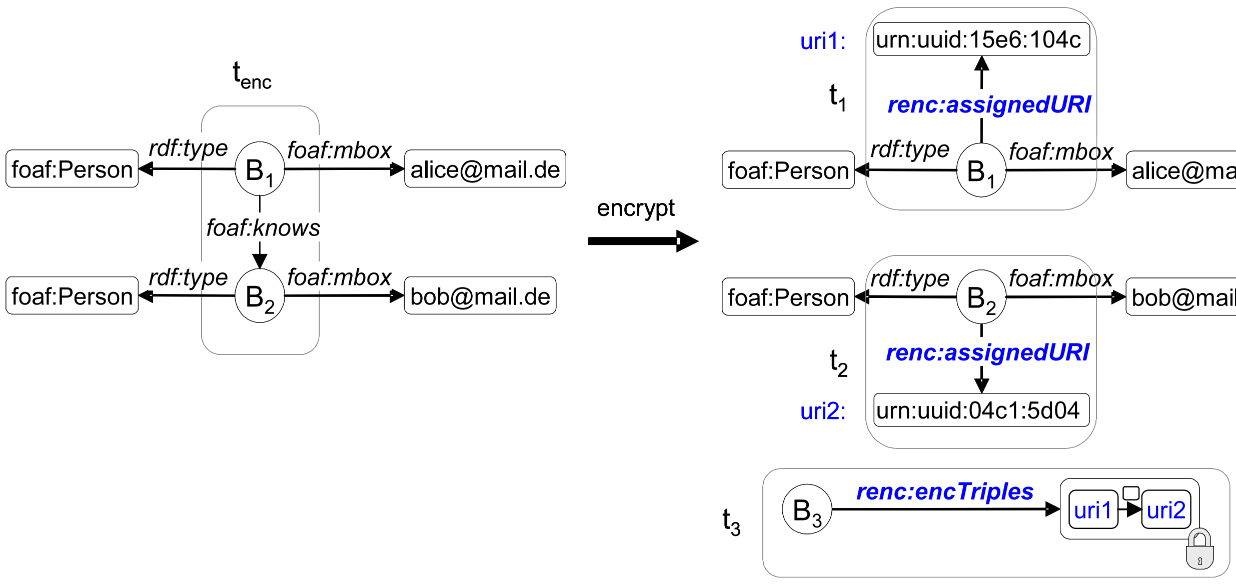


Figure 4.7: Blank node example

4.5 Encryption Policies

Encryption policies for RDF graphs define *which* fragments to encrypt and *how* to encrypt them. The *PRE Policy Language (PRE-PL)* uses a graph pattern based approach that allows for dynamic selection of encryption fragments. PRE-PL uses the RDQL [45] query language that comes with the Jena framework [91, 43]. The result of a query can be interpreted as a set of fragments which are instances of the same 'category' defined by the search pattern. Each category is encrypted in the same way (the same keys, algorithms, etc.). RDQL mainly defines a list of triple patterns which are mapped to concrete triples in an RDF graph. A triple pattern generally has the form

TriplePattern ::= '(' (Var | URI) (Var | URI) (Var | Const) ')'

where *Var* are variables, *URI* are URI references and *Const* are URI references or (typed) literals. The result of a query is a set of bindings, in which the variables are bound to concrete RDF items (subjects, predicates or objects).

RDQL has been adapted in a way that it returns a set of triples bound to each triple pattern instead of returning variable bindings. Based on the ordered triple pattern sequence, the encryption fragments are identified by using the markers s, p, o, or t. The marker s (p, o) will cause the encryption of the subjects (predicates, objects) of the bound triple set. The marker t will cause the encryption of each triple in the set. This mechanism allows the encryption of fragments which are not bound to variables, e.g. named values. Additionally, it has to be specified how to encrypt the selected fragments, i.e. which encryption method, keys, parameters, etc. to use. In the following we give a short example.

Example 3: The rule *„encrypt the email addresses of all persons"* using Triple-DES as block cipher algorithm and the RSA keys provided in the certificates of Bob and Alice can be formulated in PRE-PL as follows:

```
<pre:PREPolicy xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
               xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
               xmlns:pre="http://rdfenc.berlios.de/pre-pl#">
  <ds:KeyInfo>
    <ds:X509Data id="alice">...</ds:X509Data>
    <ds:X509Data id="bob">...</ds:X509Data>
  </ds:KeyInfo>

  <pre:DefaultEncryptionScheme>
    <pre:Symmetric><xenc:EncryptionMethod Algorithm="xenc:tripledes-cbc"/></pre:Symmetric>
    <pre:Asymmetric><xenc:EncryptionMethod Algorithm="xenc:rsa-1\_5"/></pre:Asymmetric>
    <pre:Digest type="pre:directDigest"><ds:DigestMethod Algorithm="ds:sha1"/></pre:Digest>
    <pre:RDFLanguage name="pre:N-Triples"/>
    <pre:DefaultKeys><pre:KeyRef id="alice"/></pre:DefaultKeys>
  </pre:DefaultEncryptionScheme>

  <pre:GraphPattern xmlns:foaf="http://xmlns.com/foaf/0.1/"
                    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns\#">
    <pre:TriplePattern subj="?x" pred="rdf:type" obj="foaf:Person"/>
    <pre:TriplePattern subj="?x" pred="foaf:mbox" obj="?y">
      <pre:Encryption target="o"><KeyRef id="bob"/></pre:Encryption>
    </pre:TriplePattern>
  </pre:GraphPattern>
</pre:PREPolicy>
```

Each *PREPolicy* has a *KeyInfo* section for key definition. Each child element provides key material which is referenced in the *GraphPattern* sections. Note, that the external keys can be referenced using the XML-Signature reference mechanism. Each PRE policy also defines one *DefaultEncryptionScheme* section which defines the default encryption parameters: the symmetric and asymmetric algorithms, the digest algorithm and additional randomization, the RDF serialization language for triples and the default keys for each fragment. Additional encryption schemes can be defined which can be referenced in *Encryption* elements. Each *GraphPattern* section has a list of triple patterns and optional constraints which are mapped to an RDQL query. For each *TriplePattern* it can be defined how to encrypt the bound fragments. In the above example, the object of the second triple pattern (the email address) is encrypted using the default encryption scheme and the additional key with the ID 'bob'.

4.6 Conclusions

A method to partially encrypt RDF graphs has been presented. It differs from other approaches in that the result is a single self-describing RDF-compliant graph containing both, encrypted data and plaintext data. The method allows for fine-grained encryption of subjects, objects, predicates and subgraphs of RDF graphs. Encrypted fragments are included as XML literals which are represented using the XML-Encryption and XML-Signature recommendations. Graph transformations have been described that are necessary to keep the encrypted RDF graph well-formed. The proposed method is adoptable for different algorithms and processing rules by using encryption metadata. We have motivated the usage of randomized digests for high sensitive data (such as credit card number, passwords, etc.) and direct digests for low sensitive data (such as email, phone number, etc.) in order to allow a trade-off between security and application integration.

We also have introduced the idea of encryption policies for RDF. For this purpose, the *PRE Policy Language* has been proposed which uses RDQL queries for dynamic selection of fragments to be encrypted. A prototypical implementation (PRE4J [27]) is available under GNU Lesser General Public License for the Jena Framework [43].

As with all partial encryption methods, encrypted data has a certain context which can be used for guessing the corresponding plaintext data. Semantic Web applications also typically make use of ontologies. An ontology formulates a strict conceptual scheme about a domain containing the relevant classes, instances, properties, data types, cardinalities, etc. This information can be used for attacks or even inferring encrypted content. Property definitions for example can dramatically reduce the search space for guessing the plaintexts and can be used for systematically checking the hash value provided in the encryption container. Concerning the confidentiality of encrypted data, it is also crucial to know if the data to be encrypted is inferable. This topic has not been evaluated in detail, yet. First investigations of this topic have been described in [132, 200].

CHAPTER 5

Visualization of Resource-Level Metadata

Due to the huge amount existing patent documents, there is an increasing demand for computer aided patent analysis. Since patents mainly comprise natural language content supplemented by figures, diagrams, chemical formulas, gene sequences, etc. their analysis still requires human experts for getting the best results. Interactive patent visualization therefore can support human experts in deriving new insights from large patent collections by producing visual representations of abstract patent data and their analyses. Interaction techniques can help adapting these visual representations to the needs of the users.

This chapter describes new visualization techniques for exploration and coarse analysis of patent metadata, for example as the result of a patent query. The focus is on graphs and treemaps for visualizing multivariant data. The techniques described in this chapter extend existing approaches by new features and adapt them for the patent domain. Section 5.2 describes a dynamic visual mapping based on the SPARQL query language and a graph visualization ontology. Section 5.3 shows an application of this mapping for patent networks. Section 5.4 introduces a different layout algorithm for temporal paten metadata. The result is an interactive matrix. Section 5.5 describes treemaps for the presentation of classificatory patent information. It introduces an efficient 3D treemap layout that combines a classical 2D layout for the generation of textures with a 3D layout for the generation of the treemap topology. With stacked 3D treemaps it is further possible to show another attribute for each classification, similar to stacked bar charts. Additionally, hierarchical 3D edge bundles are used for visualizing relationships between classifications, for example for co-classification analysis.

5.1 Related Work

In the recent years there have been various approaches in visualizing patent information. One can distinguish between visualization approaches at content-level and resource-level. For content-level visualizations dimension reduction methods like Multidimensional Scaling [174], Factor Analysis [63], or Self-Organizing Maps [171] are applied to the textual content of patents. Examples of software systems in this area are SPIRE/ThemeScape [290] or VxInsight [79]. An overview of visualizations based on shallow semantic analysis of general scientific documents can be found in [82].

Common resource-level visualizations use lists, tables and charting techniques, such as line charts, bar charts, or scatter plots. In particular, for time-oriented patent metadata, such as application, filing, or publication date, charts with one time axis are used [53].

While chart-based visualizations are suitable for displaying low dimensional data (typically one to three dimensions), they are less suitable for presenting multidimensional data, i.e. data with more than one independent variables (cf. [291]). The latter can be presented as a combination of different 'simple' visualizations as Coordinated and Multiple Views (CMV) [208]) or within a 'complex' visualization like Parallel Coordinates [153, 297, 144] or Table Lens [202, 237, 160].

Multidimensional patent metadata which basically contain nominal data further can be visualized as *graphs* for focusing on relations between the data items. In the patent domain such items include patent documents, legal entities, patent families, classifications, etc. A general overview of graph layout techniques can be found in [107, 163]. Relations can be based on computed similarity measures, co-classifications, co-citations, term co-occurrence, or some other shared bibliographic data (cf. Jaffe and Trajtenberg [157]). Visual clustering based on similarity measures has been described by Boyack et al. [78]. Co-citation networks have been investigated by Chen [96].

Trees or hierarchies are special cases of graphs, in which nodes only have one ingoing relation. *Treemaps* [161, 222, 224] allow the presentation of large hierarchies by dividing the display area into a nested sequence of polygons (usually rectangles) whose areas correspond to an attribute of the data set. Several layout algorithms have been proposed for treemaps, for example slice-and-dice [161], cluster treemaps [245] or squarified treemaps [84]. Vliegen and Wijk [246] have generalized the construction of treemaps in order to fit into arbitrary polygons. Balzer and Deussen [61] proposed a combination of treemaps with Voronoi diagrams, which they call Voronoi treemaps. Further optimization regarding human cognition can be achieved when using cushion filling [245] and shading techniques [154]. The use of treemaps for visualizing the hierarchic organization of patent collections among the United States Patent Office (USPTO) classification has first been described by Kutz [176]. 3D treemap-like approaches have been proposed by Wettel and Lanza [279] (Code City) and by Balzer and Deussen [60] (Hierarchical Net) for the visualization of software systems.

5.2 SPARQL-based Visual Mapping

This section proposes a filtering and visual mapping approach for constructing graph based on the *Graph Visualization Ontology (GVO)* and dynamic SPARQL [261] queries. GVO is a simple general purpose ontology for describing nodes and edges as well as their visual variables (e.g. color, label, size, etc.). The SPARQL query selects relevant information from an underlying semantic representation (see chapter 3) and maps this information to graph nodes and edges. The visual mapping is directly encoded in SPARQL queries. This is different from existing approaches such as Fresnel [76].

The aim is to provide a flexible interface for doing ad-hoc analyses of patent collections. The nodes and relations of graph represent entities of the patent domain, such as patent documents, legal entities, patent families, classifications, etc. The rest of this section describes GVO (section 5.2.1) and two SPARQL mappings: one for general graphs based on SPARQL construct statements (section 5.2.2) and one for hierarchies based on SPARQL select statements (section 5.2.3).

5.2.1 Graph Visualization Ontology

The Graph Visualization Ontology (GVO) is a lightweight ontology for setting up a graph description in RDF. It uses the namespace *http:gvo.patexpert.org* prefixed as *gvo*. The most important classes are *gvo:GraphNode*, the class of nodes, and *gvo:GraphEdge*, the class of edges. Each resource of type *gvo:GraphNode* is interpreted to be a graph node. Each resource of type *gvo:GraphEdge* is interpreted to be a graph edge defined with *gvo:source* referencing the source node and *gvo:target* referencing the target node. The reason for modeling edges as *owl:Class* rather then *owl:ObjectProperty* is because it allows adding properties to edges without the using the reification vocabulary (cf. [255, ch. 5.3]). Labels, for example, can be associated using the *gvo:label* property. Other properties for styling nodes and edges are derived from Cascading Style Sheets (CSS) [267]. An overview of the vocabulary is shown in figure 5.1.

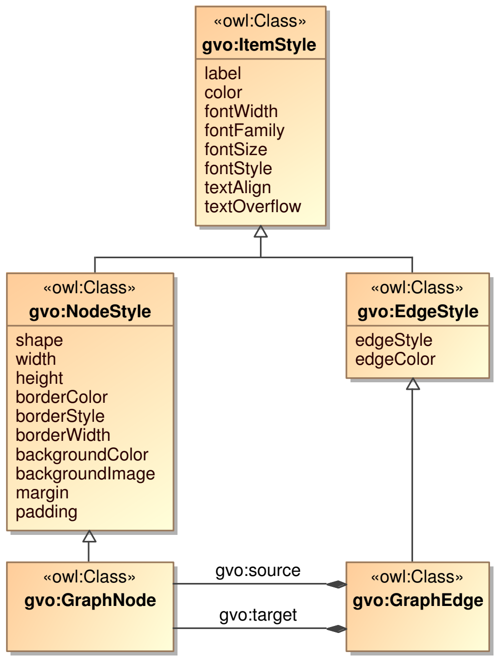


Figure 5.1: GVO vocabulary

A GVO instance is visualized by a corresponding client which 'knows' about the semantics of the classes and properties. An example for such a client is shown in the next section. An important design decision was to inherited style properties rather than using delegation in order to simplify the SPARQL statements. The properties defined in GVO for styling are not intended to be complete. They include just a basic collection of CSS level 1 attributes suitable for the client. New styling properties, e.g. for shading, 3D layout, etc. can easily be added to the ontology by extending the corresponding classes.

5.2.2 Graph Mapping with SPARQL Construct

This section discusses the visual mapping for general graphs based on SPARQL construct statements. The following SPARQL construct query gives an example how a patent information model can be mapped to a graph specification. The result of this query is shown in figure 5.2.

The *where* clause selects all applicants with "UNIV MIAMI" as name and binds them to the variable *?applicant*. It binds all associated patent document numbers to the variable *?pat*, all associated inventors and applicants to the variables *?inventor* and *?applicant* respectively. Based on these variable bindings the GVO instance is created in the *construct* clause. Graph nodes are created for patents, inventors and applicants and connected by edges. As an example for styling, colors are defined using the *gvo:backgroundColor* property.

```
CONSTRUCT {
  ?pat rdf:type gvo:GraphNode .
  ?pat gvo:label ?documentNumber .
  ?pat gvo:backgroundColor "#FFFF00" .
  ?inventor rdf:type gvo:GraphNode .
  ?inventor gvo:label ?inventorName .
  ?inventor gvo:backgroundColor "#FFFFFF" .
  ?applicant rdf:type gvo:GraphNode .
  ?applicant gvo:label ?applicantName .
  ?inventor gvo:backgroundColor "#FF0000" .
  _:inventorEdge rdf:type gvo:GraphEdge .
  _:inventorEdge gvo:source ?pat .
  _:inventorEdge gvo:target ?inventor .
  _:applicantEdge rdf:type gvo:GraphEdge .
  _:applicantEdge gvo:source ?pat .
  _:applicantEdge gvo:target ?applicant .
}
WHERE {
  ?applicant pmo:name ?applicantName .
  FILTER regex(?applicantName, "UNIV MIAMI")
  ?pat rdf:type pmo:PatentDocument .
  ?pat pmo:docummentNumber ?documentNumber .
  ?pat pmo:applicant ?applicant .
  ?pat pmo:inventor ?inventor .
  ?inventor pmo:name ?inventorName .
}
```

In figure 5.2 yellow nodes represent patent documents, red nodes applicants and white and gray nodes inventors (gray nodes are highlighted because they have been manually selected). One can notice that there are ambiguities regarding the inventor and applicant names ('Ryan James', 'Ryan J W', 'Ryan James W', etc.). These ambiguities are difficult to resolve automatically. By using graph visualizations that place inventor and applicants near to their defining patent documents it would easily be possible to manually resolve such ambiguities on the fly by inserting corresponding *owl:sameAs* assertions into the knowledge base. If later a user selects one of the nodes that represent for example 'Ryan James' all equivalent nodes could also be highlighted.

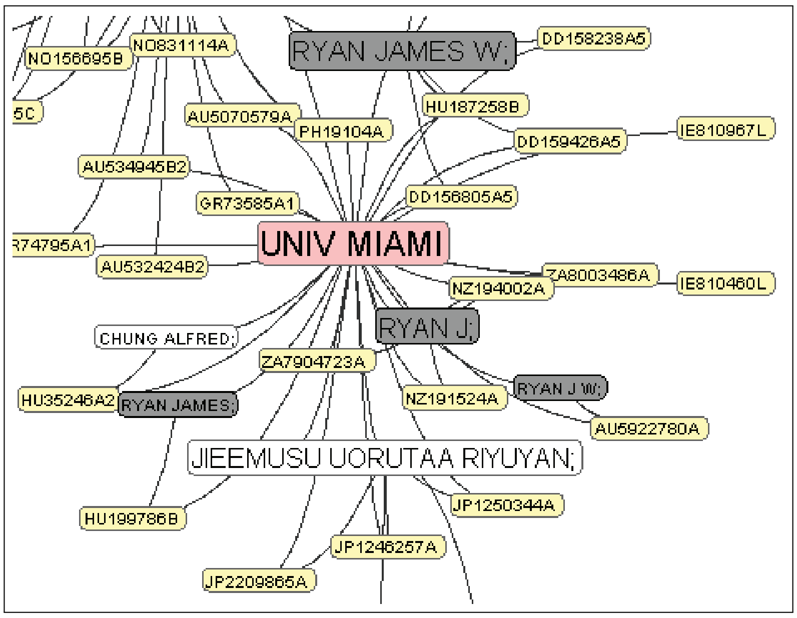
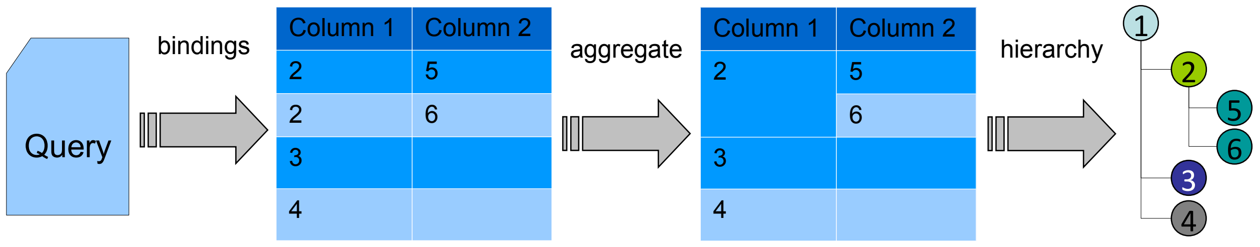


Figure 5.2: Relations between patents, applicants and inventors

This approach allows for a flexible visual mapping of RDF triple sets in general and Patent Information Models in particular onto graph visualizations. The advantage of this approach is that the patent analyst has full control over how the graph is constructed because the presentation logic is combined with the selection logic and appropriate filters can be applied. A possible shortcoming is that an analyst has to know about the Patent Metadata Ontology, about the Graph Visualization Ontology and about the SPARQL query language for defining the mapping. Providing appropriate templates for SPARQL based graph mappings therefore would be a good balance between complexity and flexibility. A simpler mapping for hierarchical data is described in the next section.

5.2.3 Hierarchy Mapping with SPARQL Select

This section proposes a simple mapping of hierarchical data onto tree structures. Similar to the previous section a SPARQL query is used for selecting relevant information from an underlying RDF triple set in general and a Patent Information Model in particular. The idea here is to implicitly map the order of the variables in the SPARQL SELECT statement onto a tree structure. The results bound to the first variable are mapped to the first tree level. The results bound to the second variable are mapped to the second tree level, and so on. In an intermediate step, the bindings are aggregated by column and value before they are mapped to a tree structure. The schema is depicted in figure 5.3.



Column 1	Column 2
2	5
2	6
3	
4	

Column 1	Column 2
2	5
	6
3	
4	

Figure 5.3: Steps of the tree structure generation, root node (1) is automatically added

Example

The following SPARQL query defines four variables *?catchword*, *?year*, *?applicant*, *?pat* that are to be bound to a graph pattern (cf. [261, ch. 5]). It defines a search for IPC entries with hierarchy level 2 (classes level) that are associated with a given catchword. Since patent document instances are connected with these IPC entries, the pattern also binds all associated patent documents, their year of publication and their applicants.

The order of the SELECT clause is catchword → year → applicant → patent document number. This order implicitly determines the grouping order that is mapped to a tree structure. A fragment of the generated tree visualized as node-link diagram is shown in figure 5.4.

```
SELECT ?catchword ?year ?applicant ?pat
WHERE {
  ?ipc ipc:hierarchyLevel "2" .
  ?ipc ipc:catchword ?catchword .
  ?pat pat:classification ?ipc .
  ?pat pat:applicant ?applicant .
  ?pat pat:documentDate ?year .
}
```

Algorithm

Input is a table with the rows $r_1, \ldots, r_n$. Each row is a list of name-value-pairs (n,v). The names are the column names defined by the variables in the SPARQL SELECT statement. The number and order of these pairs is determined by the number and order of SELECT variables. The first step of the algorithm is an aggregation of query bindings by column and value.

Algorithm 5.1 Aggregation of Query Bindings

```
1.  procedure makeHierarchy(bindings[], names[])
2.      $map \leftarrow$ new empty map
3.      for each $b_i$ in bindings[]
4.          processBinding(map, 0, $b_i$, names)
5.      end for
6.  return $map$

7.  procedure processBinding(map, level, binding, names[])
8.      $name \leftarrow$ names[level]);
9.      $value \leftarrow$ binding.get(name);
10.     if $level = |names| - 2$
11.         $leafSet \leftarrow$ map.get(value)
12.         if not exists leafSet
13.             $leafSet \leftarrow$ new empty set
14.             $map$.put($leafSet$)
15.         end if
16.         leafSet.add(binding.get(names[level+1]))
17.     else
18.         $nextMap \leftarrow$ map.get(value)
19.         if not exists nextMap
20.             $nextMap \leftarrow$ new empty map
21.             $map$.put($nextMap$)
22.         end if
23.         processBinding(nextMap, level + 1, result, names);
24.     end if
```

This creates an aggregation structure in which simple values represent leaf nodes and maps represent inner nodes. The keys of a map at level i represent nodes at level i. The values of a map at level i represent nodes at level $i + 1$.

1. level: map_1<Object, set<Object> >

2. level: map_1<Object, map_2<Object, set<Object> > >

3. level: map_1<Object, map_2<Object, map_3<Object, set<Object> > > >

n. level: map_1<Object, ... map_{n-2}<Object, map_{n-1}<Object, set<Object> > > >...>

Finally, the aggregation structure is transformed into a tree structure appropriate for visualization which contains *gvo:GraphNode* and *gvo:GraphEdge* instances as described in section 5.2.

Algorithm 5.2 Transformation into a Tree

```
1.  procedure doConvert(map, depth, level, tree, parent, names)
2.      $keys$[] ← map.getKeys /* can be sorted */
3.      for each $key$ in keys[]
4.          $childNode$ ← tree.makeChildEdge(parent) /* optionally label with names[level] */
5.          $childNode$ add relevant properties of key
6.          if $level < depth - 2$ /* inner nodes */
7.              $nextMap$ ← map.get(key)
8.              doConvert(nextMap, depth, level + 1, tree, childNode, names);
9.          else /* leaf nodes */
10.             $leafSet$ ← map.get(key)
11.             for each $leafNode$ in leafSet
12.                 $childNode$ ← tree.makeChildEdge(parent)
13.                 $childNode$ transfer all relevant properties of leaf
14.             end for
15.         end if
16.     end for
```

Discussion

The proposed method allows a simple visual mapping of RDF triple sets onto a tree structures. In contrast to the dynamic graph mapping described in section 5.2.2 this

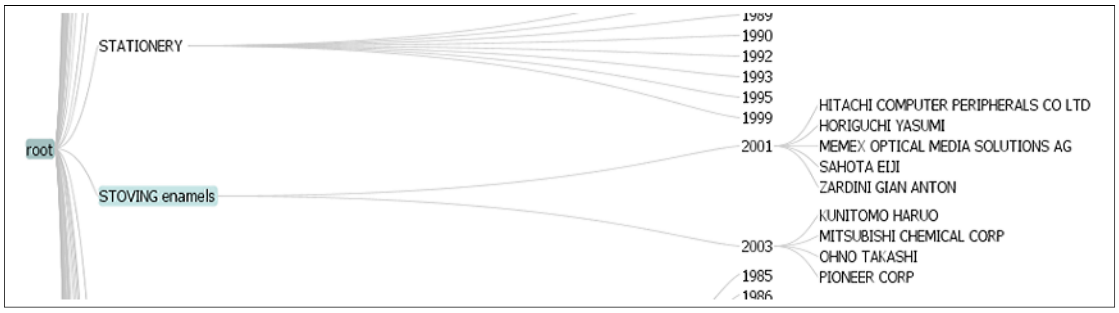


Figure 5.4: Fragment of a generated hierarchy visualized as node-link diagram

approach implicitly generates tree nodes and edges and currently does not support the modification of visual attributes. They have to be set up programmatically for each column. But this limitation on the other hand simplifies the usage and allows the analyst to create fast results even for complex tasks. The mapping is implicitly defined by the order of the SELECT variables.

Very helpful for generating statistics are additional SPARQL extensions, for example ARQ plug-ins [2] that provide aggregate functions such as COUNT, GROUP-BY, AVG or SUM. A future extension could be a combination with faceted browsing, for example as described by Heim and Ziegler [143], for showing the used attributes and their value and for using them as customizable filters. Also a dynamic configuration of visual attributes for each mapped column is planned for the future.

5.3 Visualization of Patent Networks

Graph Drawing is an important area within Information Visualization. A good graph layout can effectively convey the key features of complex structures to a wide range of users [95]. Di Battista et al. [107] give some common aesthetic criteria for what makes a good layout, for example uniform edge length, minimized edge crossings, minimized node overlapping or symmetric distribution of nodes. Another import criterion is the time necessary to layout a graph. Since graph drawing algorithms in general have the problem of scaling up to tens or hundreds of thousands of nodes, the focus in this chapter mid-size patent networks with less than 1000 nodes.

Entities of the patent information model, such as patent documents, inventors, applicants, classifications, etc. (see chapter 3) can be represented as nodes of a *patent net-*

work. Patent networks therefore usually are heterogeneous networks with several node types. Relations between entities, for example priority, co-citation, classification or other bibliographic relations, can be represented as edges. Additionally, nodes and edges can have attributes, for example time-based or statistical information, which can be bound to visual variables such as size, color, shape, etc. in order to visualize their specific characteristics.

In combination with dynamic filters executed locally and hit highlighting, graph based visualizations provide a good overview of a patent information model, which is retrieved from a backend as the result of an initial patent query. For the visualization of patent networks various graph layout algorithms can be applied as described below.

As an example, figure 5.5 shows a patent network in which the yellow nodes represent published patent documents and the brown nodes priority applications. If a priority node is connected with a patent node by an edge, then this represents a priority relation.

The layout algorithm used in figure 5.5 places the nodes based on a simulation of interacting forces. A customizable force simulator is used to drive the layout. It allows for setting a variety of forces. By default nodes repel each other, edges act as springs, and drag forces similar to fluid resistance are applied. The algorithm requires running multiple iterations. The current state of the graph is shown as an animation.

The implementation uses a Barnes-Hut [62] based hierarchical space decomposition method provided by the Prefuse framework [142] with a time complexity of $O(E+N\ log\ N)$ where N is the number of nodes and E the number of edges. The default values for masses, springs, gravity, iteration cycle, etc. can be overridden for each node type to optimize the layout for specific needs. But the addition of custom force calculations may increase the algorithm's time. A further improvement could be a migration to the hierarchical $O(N)$ force calculation algorithm proposed by Dehnen [104].

In contrast to other force-directed algorithms, such as Fruchterman and Reingold [113] or Kamada and Kawai [162], the final state of the algorithm is not determined by minimal energy of the force system, but is stopped manually by the user if the layout is considered as sufficient.

Since the Barnes-Hut algorithm does not consider different node sizes (e.g. due to different labels), there are possible overlappings in the calculated graph. In order to reduce overlapping a version of the *Online Dynamic Natural Spring Length* (ODNSL) algorithm [181] is applied. This algorithm has a time complexity of $O(N^2)$ and is executed

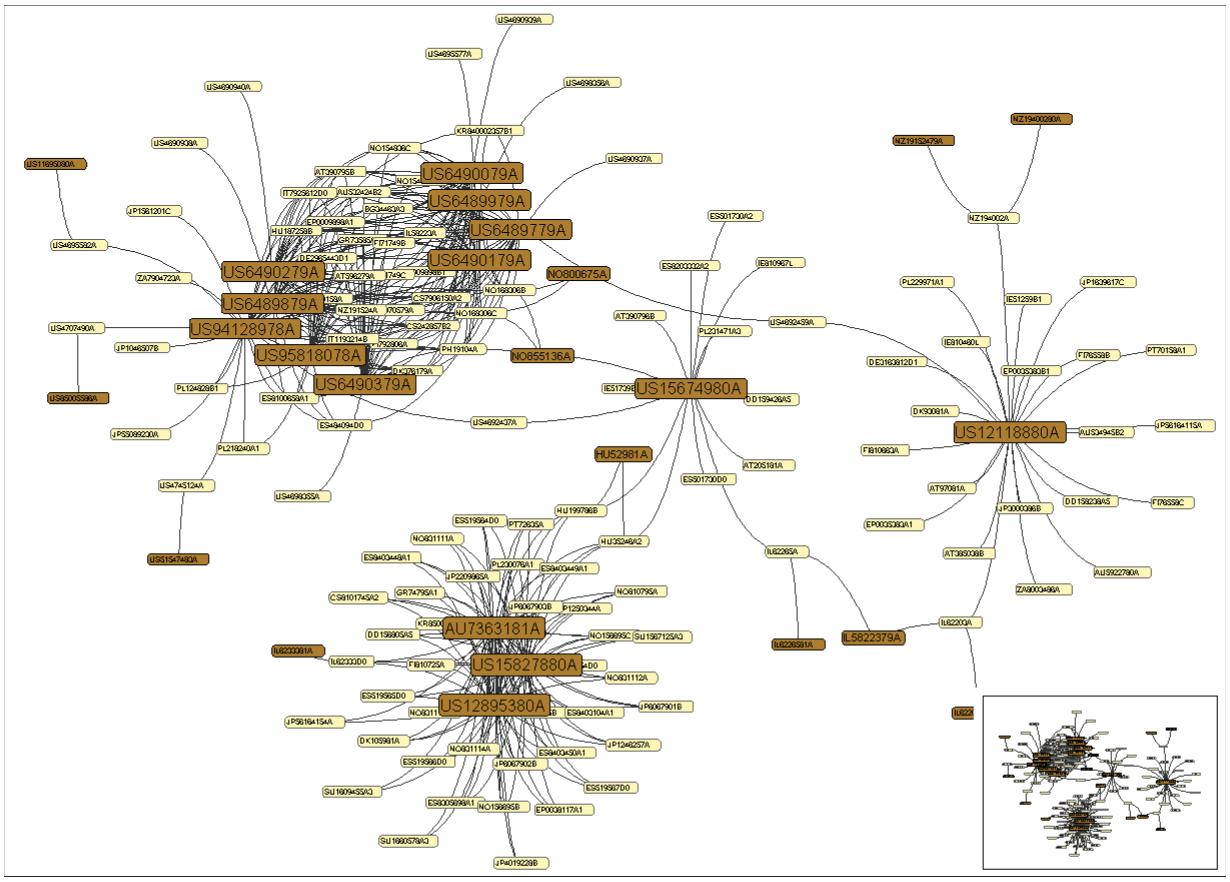


Figure 5.5: Family clusters in a patent network

after the final layout has been calculated using the Barnes-Hut algorithm.

The layout of very large graphs with thousands of nodes using these techniques can take up to several minutes and therefore is not suitable for interactive scenarios. Although force-directed layouts algorithms in general produce a good layout according to the mentioned aesthetic criteria, it is difficult to identify specific nodes in a large graph, since their position is different after each layout [107]. To find specific patent nodes in large graphs an additional full-text search index is maintained providing fast search and also filter and highlighting capabilities.

In order to further improve the readability a Focus+Context technique has been introduced. Selected nodes are enlarged and put into front of the graph. Potential overlapping of nodes is removed using ODNSL while keeping the relative position between nodes. Using an animated transition triggered upon node selection further supports the mental model. Unselected nodes are put into the background by applying a blur filter.

As an example, figure 5.6 shows a selected node with its direct neighbors. All other nodes are in the background. This technique allows browsing in larger graphs. Animations support in maintaining the context. The highlighted nodes are also enlarged in order to make their labels more readable. For comparison, the unselected graph is depicted in figure 5.7.

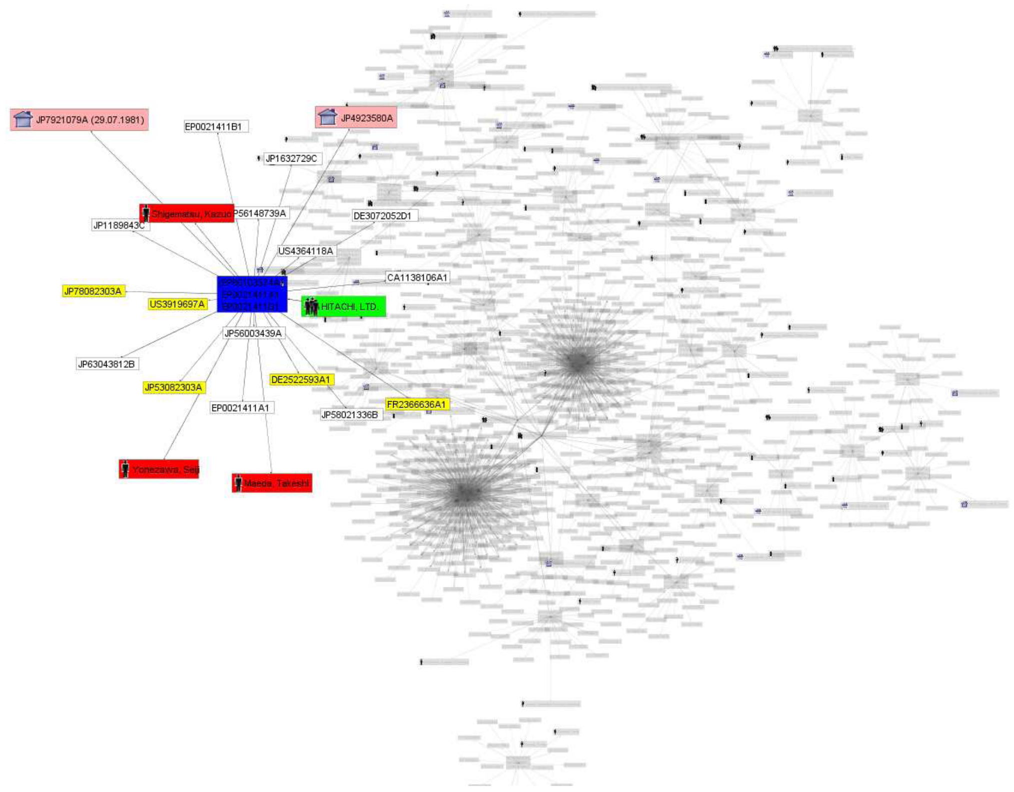


Figure 5.6: Patent network with one node selected and adjacent nodes zoomed

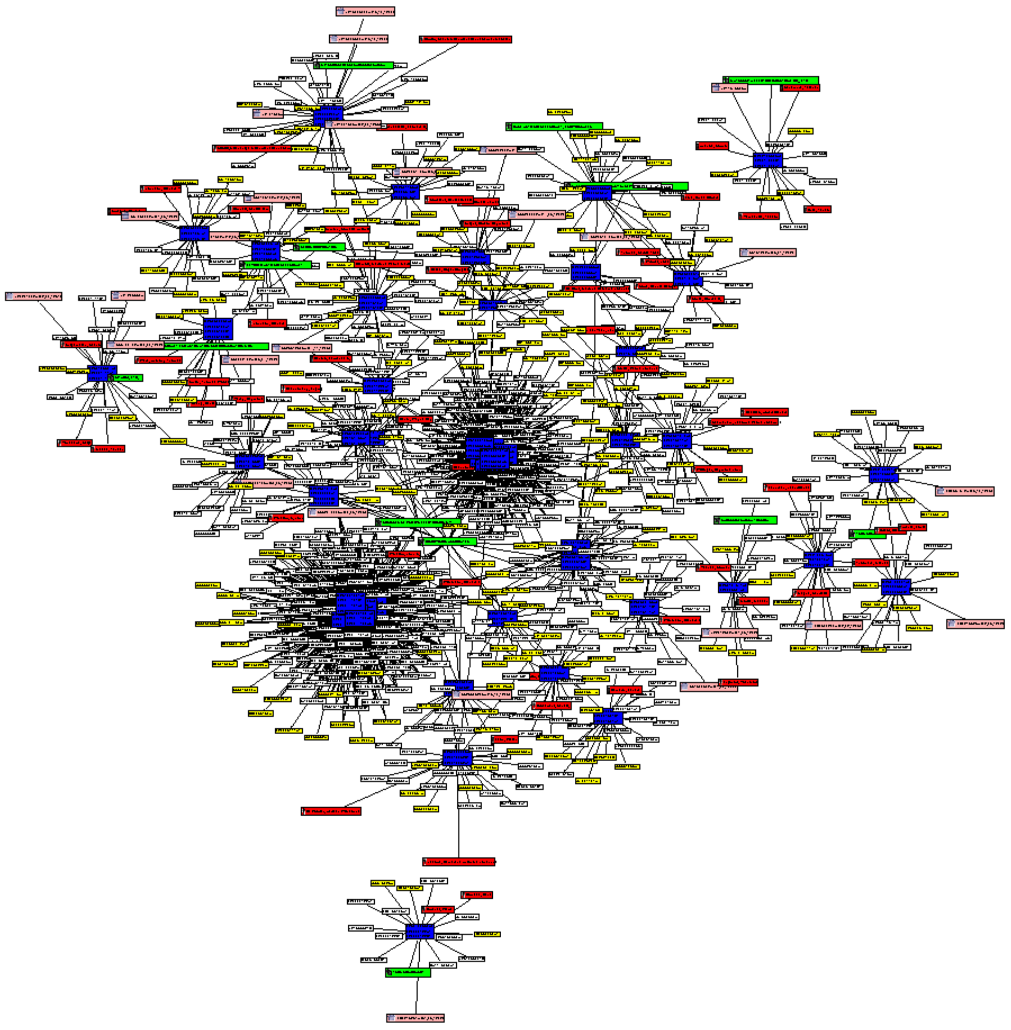
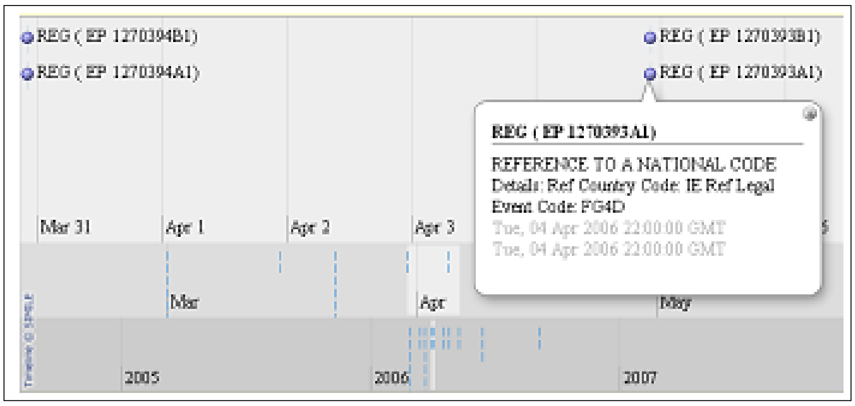


Figure 5.7: Patent network with no selection

5.4 Visualization of Time-Dependent Patent Metadata

The primary time-dependent patent metadata are the filing, publication and priority dates of patent documents. Other time-dependent patent metadata are for example the date of legal status events, the adding dates of new family members or derived time information such as the time that a granted patent is in force. Time information can be combined with other patent information, for example to show the development of a patent portfolio of a company over time [176]. Therefore, time-dependent information is an import aspect during a patent analysis.

Because time defines a strict order it can be mapped to ordinal or numeric axes. A good overview of general visualization methods for time-dependent data can be found in [184]. A common and natural visualization method for temporal aspects is a timeline visualization. An example of a timeline visualization of legal status events is shown in figure 5.8. For this prototype the SIMILE Timeline widget [29] has been used to show the dates of legal events of a given patent set. Each event can be expanded to look at more details. The widget shows a stacked timeline in different resolutions: years at the bottom, months in the middle and day at the top.



REG (EP 1270393A1)

REFERENCE TO A NATIONAL CODE
Details: Ref Country Code: IE Ref Legal Event Code: FG4D
Tue, 04 Apr 2006 22:00:00 GMT
Tue, 04 Apr 2006 22:00:00 GMT

Figure 5.8: Timeline of the legal events of a patent family

Often time-dependent patent metadata has to be combined with other metadata aspects such as filing country, applicant, classification, etc. Therefore the timeline visualization is extended to a matrix visualization in which each data point can be connected with others. Based on the visualization framework for graphs and hierarchies as described in the previous section a matrix can be constructed by exchanging the layout algorithm for positioning of nodes.

The rest of this section describes an interactive matrix visualization for time-dependent patent metadata. All matrix nodes represent objects that can be ordered according to their time and a second category, usually patent documents with different roles, such as publications, applications, citations, or family members. The visualization additionally shows bibliographic relations between patent documents, when the user moves the mouse over a node. The arcs can be generalized as domain and range restricted additional dimensions (domain and range have to be of a node role).

Figure 5.9 shows an example of a patent matrix visualization. The horizontal position (x-axis) is calculated based on the publication date of patent publications (shown as magenta nodes) and the filing date of patent applications (shown as blue nodes).

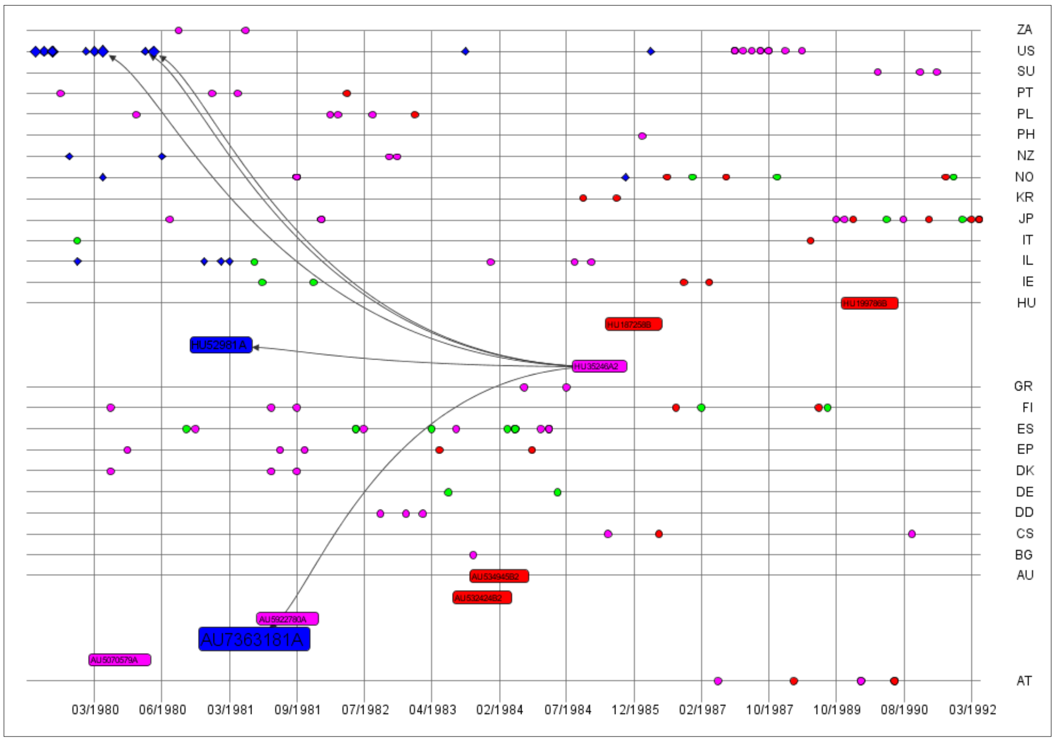


Figure 5.9: Interactive matrix visualization with explicit linking

Cited and citing patent documents are shown in the visualization as red and green nodes. The vertical position (y-axis) reflects the country or organization where the patents or applications have been filed. Each horizontal line shows the documents filed in the same country. The lines can be interactively expanded in order to present more details for that country, which is shown for the countries codes *HU* and *AU*. When a country is expanded, more space is available and the publication/application number is shown. Because this requires additional space, the nodes are stacked vertically. When hovering a node the priority and citation relations are visualized as arcs pointing from a patent to its priority, cited or citing documents. The node size reflects the number of ingoing and outgoing arcs of a node. This allows highlighting patent documents that are often referenced.

5.5 Visualization of Classificatory Information

Patent classifications organize patents into predefined technology categories. Patent classification is based on the outcomes of investigations of the patent offices during examination. Due to the diversity of today's technological fields, a patent may be classified with several classes. Technology classes could therefore be an appropriate unit of co-classification analysis to show linkages between technical fields.

In particular the International Patent Classification (IPC) (see section 2.1.2) became an important source for investigating the state of the art in given fields of technology and the assessment of technological development in these fields. The IPC can also be used as taxonomy for comparing patent portfolios of different applicants. The aim therefore is to improve analysis of large patent collections and portfolios by visualizing their classificatory distributions.

This section describes treemap-based visualization techniques for the IPC. There are various options for presenting the classificatory distribution of a patent collection among the IPC. Figure 5.10 shows a treemap in which the amount of matching patents is indicated in the labels of the IPC categories. Additionally the outline of matching IPC categories is colored red. The label itself shows the IPC code part that corresponds to the hierarchy level, e.g. A for sections, 01 for classes, etc. If not enough space is available, the labels are omitted (see subclasses of section B figure 5.10). Sometimes it is useful to filter out IPC categories that do not have associated patents, in order to get a visualization that provides more details. A filtered IPC treemap is shown in figure 5.11.

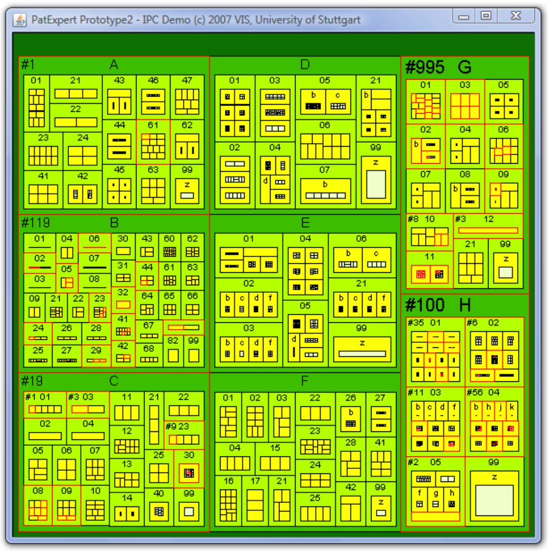

PatExpert Prototype2 - IPC Demo (c) 2007 VIS, University of Stuttgart
#1 A
D
#995 G
#119 B
E
#100 H
#19 C
F

Figure 5.10: Classificatory distribution of a patent collection shown as treemap

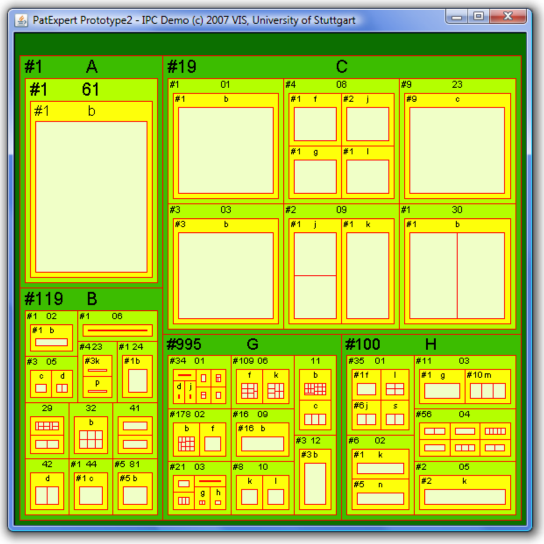

PatExpert Prototype2 - IPC Demo (c) 2007 VIS, University of Stuttgart
#1 A
#1 61
#1 b
#19 C
#119 B
#995 G
#100 H

Figure 5.11: Treemap with unused categories filtered and zoomed into A & C

To give the user more control of the visualization, an important issue is structural zooming. Structural zooming is a Focus+Context technique that adds further details on the structural level (i.e., deeper levels of the IPC) when more display space is available. Figure 5.12 shows an example for structural zooming. The subclass G11b has been zoomed in. As a result, the next level (level of subgroups in blue) is presented to the user. In the example, the zoom factor for G11b is 4.0 and at the same time all siblings are reduced by factor 0.5 to allow more space for details while still keeping the current context.

5.5.1 Comparison of Patent Portfolios

One use case is the comparison of different patent portfolios, for example of the portfolios of organizations or countries. The comparison can take the complete IPC into consideration or can be restricted to specific IPC categories. It is further useful to support a restriction to certain IPC levels, e.g. to the IPC main group level. The latter aspect requires a propagation of classification relations along the IPC inheritance structure. For this purpose the properties *pmo:indirectlyClassifiedAs* and *pmo:directlyClassifiedAs* (see section 3.7) are used.

For example, if a patent is directly classified as G11B7/08 and G11B7/09, then it is also indirectly classified as G11B7/00, G11B, G11 and G. It is important to note that if the

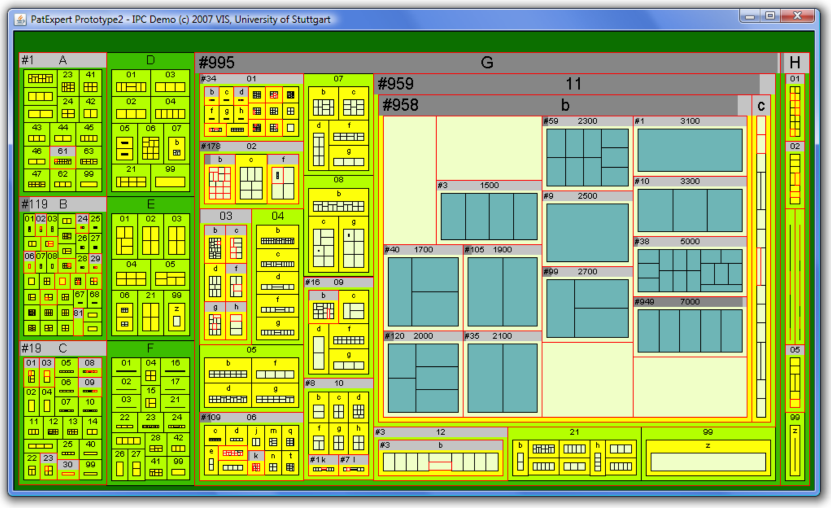

PatExpert Prototype2 - IPC Demo (c) 2007 VIS, University of Stuttgart
#1 A
D
#995
G
H
#959
11
#958
b
c
#119 B
E
#19 C
F

Figure 5.12: Structural zooming in an IPC treemap visualization

direct classifications are in the same IPC subtree (as in this example, where the groups G11B7/08 and G11B7/09 have the main group G11B7/00 as common parent category) the indirect classification relations are generated only once. In practice the patents are classified beginning at IPC main group level. Top-level categories, such as IPC sections, classes and subclasses usually are not used for direct classifications.

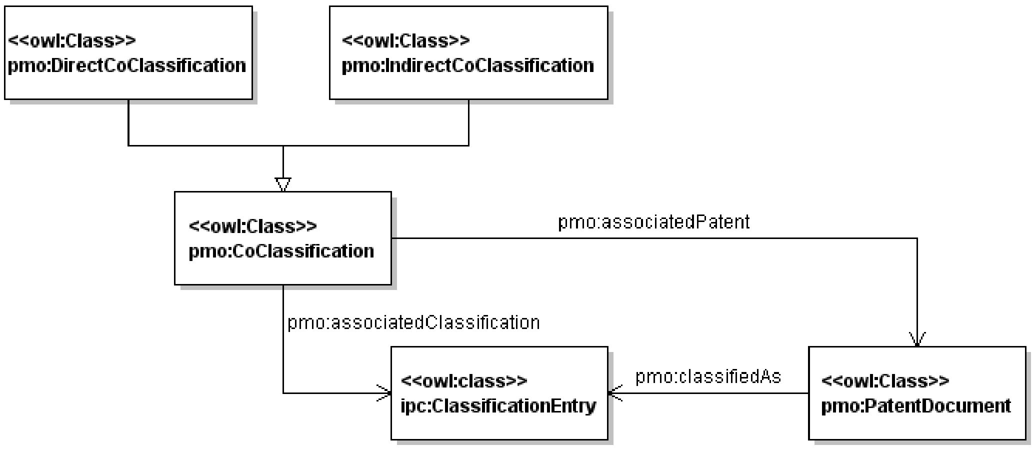
For the comparison of different patent portfolios multiple views can be generated (one view for one portfolio) or multiple portfolios can be integrated into one view. An example for latter approach is the *stacked 3D treemap* technique depicted in figure 5.18.

5.5.2 Co-Classification

A patent can be classified into multiple IPC categories, each specifying a technological field. A patent that is classified in multiple technological fields therefore indicates some relationship between these fields. The aim of a co-classification analysis is to use this information, for example for identifying new trends or for estimating the maturity grade of technological fields.

Because it is possible that a patent can be co-classified in a higher-level category and a descendant category at the same time, it is important to differentiate between different types of classifications. In figure 5.13 an extension of the PMO Classification Module (see section 3.7) is described that allows a flexible modeling of co-classification relations. It introduces a new *pmo:CoClassification* relation between a pair of *pmo:Classification-*

Entry individuals. The meaning of this binary relation between two classification entries c_1 and c_2 is that there is at least one patent that is classified as c_1 as well as c_2. The relation is also attributed with the set of *pmo:PatentDocuments* that include this co-classification relation.



<<owl:Class>> pmo:DirectCoClassification
<<owl:Class>> pmo:IndirectCoClassification
<<owl:Class>> pmo:CoClassification
pmo:associatedPatent
pmo:associatedClassification
<<owl:class>> ipc:ClassificationEntry
pmo:classifiedAs
<<owl:Class>> pmo:PatentDocument

Figure 5.13: Extension of the PMO classification module for co-classification analysis

Figure 5.14 shows a simple co-classification example, where the patent EP0095852A1 is classified as G11B7/08, G11B7/085 and G11B7/09.

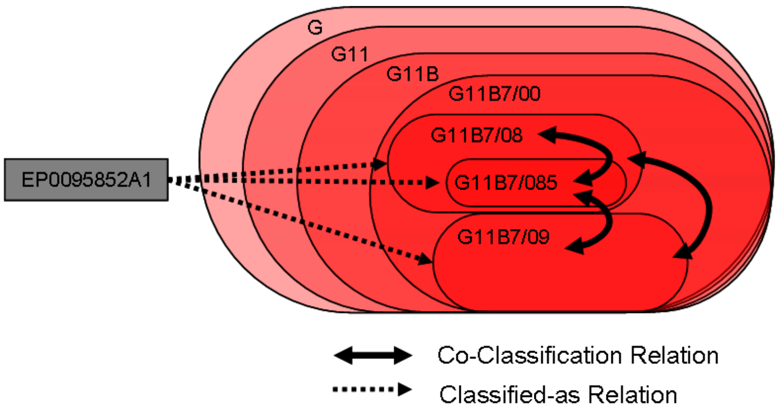


Figure 5.14: Example for patent co-classification

The co-classifications of a given patent set are generated with the following algorithm. The time complexity of this algorithm is linear with the size of the patent set and quadratic with the number of classifications of the patents. The *isDirect* parameter determines whether to generate *pmo:DirectCoClassification* or *pmo:IndirectCoClassification* relations. The foundation of this algorithm is the pairing of the classifications of a patent. If a patent for example has the classifications $\{c_1, c_2, c_3\}$ then the pairing returns the set of symmetric pairs $\{(c_1, c_2), (c_1, c_3), (c_2, c_3)\}$.

Algorithm 5.3 Generation of Co-Classification Relations

```
1.  procedure generateCoClassifications(patentSet, isDirect) returns CoClassifications
2.    $ccSet \leftarrow$ empty set of CoClassifiactions
3.    for each $patent$ in $patentSet$
4.      if $isDirect$
5.        $classificationSet \leftarrow$ patent.directClassifications
6.      else
7.        $classificationSet \leftarrow$ patent.indirectClassifications
8.      end if
9.      /* pairing of classifications of the patent */
10.     $pairSet \leftarrow$ makePairs(classificationSet)
11.     for each $pair$ in $pairSet$
12.       $cc \leftarrow$ lookup CoClassification with given pair in ccSet
13.       if not found $cc$
14.         /* create new CoClassification and add it to ccSet */
15.         if $isDirect$
16.           $cc \leftarrow$ create new DirectCoClassification
17.         else
18.           $cc \leftarrow$ create new IndirectCoClassification
19.         end if
20.         $cc \leftarrow$ add classifications in $pair$ as associatedClassification
21.         $ccSet \leftarrow$ add $cc$
22.       end if
23.       $cc \leftarrow$ add $patent$ as associatedPatent
24.     end for
25.   end for
26.   return $ccSet$
```

RDF representation of the above example (indirect co-classifications are omitted).

```
@prefix ipc:  <http://ipc.patexpert.org/> .
@prefix pmo:  <http://pmo.patexpert.org/> .
@prefix pat:  <http://pat.patexpert.org/> .

ipc:G            a ipc:ClassificationEntry .
ipc:G11          a ipc:ClassificationEntry ; pmo:parentEntry ipc:G .
ipc:G11B         a ipc:ClassificationEntry ; pmo:parentEntry ipc:G11 .
ipc:G11B7/00     a ipc:ClassificationEntry ; pmo:parentEntry ipc:G11B .
ipc:G11B7/08     a ipc:ClassificationEntry ; pmo:parentEntry ipc:G11B7/00 .
ipc:G11B7/085    a ipc:ClassificationEntry ; pmo:parentEntry ipc:G11B7/08 .
ipc:G11B7/09     a ipc:ClassificationEntry ; pmo:parentEntry ipc:G11B7/00 .

pat:EP0095852A1 pmo:directlyClassifiedAs ipc:G11B7/08 .
pat:EP0095852A1 pmo:directlyClassifiedAs ipc:G11B7/085 .
pat:EP0095852A1 pmo:directlyClassifiedAs ipc:G11B7/09 .

pat:EP0095852A1 pmo:indirectlyClassifiedAs ipc:G11B7/00 .
pat:EP0095852A1 pmo:indirectlyClassifiedAs ipc:G11B .
pat:EP0095852A1 pmo:indirectlyClassifiedAs ipc:G11 .
pat:EP0095852A1 pmo:indirectlyClassifiedAs ipc:G .

_:cc1 a pmo:DirectCoClassification ;
        pmo:associatedClassification ipc:G11B7/08 ;
        pmo:associatedClassification ipc:G11B7/085 ;
        pmo:associatedPatent pat:EP0095852A1 .

_:cc2 a pmo:DirectCoClassification ;
        pmo:associatedClassification ipc:G11B7/08 ;
        pmo:associatedClassification ipc:G11B7/09 ;
        pmo:associatedPatent pat:EP0095852A1 .

_:cc3 a pmo:DirectCoClassification ;
        pmo:associatedClassification ipc:G11B7/085 ;
        pmo:associatedClassification ipc:G11B7/09 ;
        pmo:associatedPatent pat:EP0095852A1 .
```

5.5.3 3D Treemap Visualization Method

The overall benefit of 3D over 2D is an ongoing discussion in information visualization (cf. [95, 225]). The 3D treemap approach proposed in this section is motivated by the ability of integrating additional visual information into the presentation of hierarchical structures. Other examples are the integration of temporal information into maps [241], emphasizing linkages between items in one visualization [60] or between related data shown in different visualizations [101]. Further examples of 3D visualizations approaches can be found in [238].

Following Ware [277, p. 8], who proposes „use 3D objects to represent data entities [...] emphasize 2D layout", the classificatory distribution patent portfolios is visualized in an interactive 3D treemap, in which the third dimension represents the number of patents associated with a category. A key feature of the proposed 3D treemap visualization technique is the synchronization of a 2D treemap layout with a 3D treemap layout in order to use the 2D treemap for generating the textures for the 3D items. The benefit is a better rendering performance because the large number of leaf level items are not rendered as 3D items but are textures instead.

3D Treemap with 2D Textures

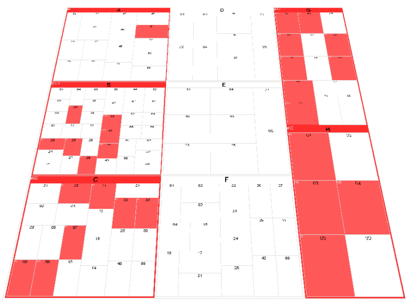


Figure 5.15: 2D treemap texture

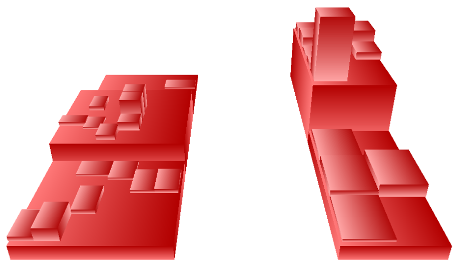


Figure 5.16: 3D treemap shape

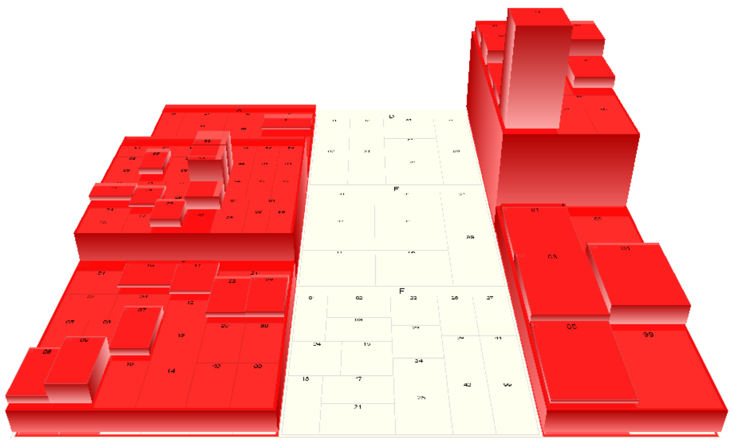


Figure 5.17: 3D treemap shape with 2D texture

The 3D treemap visualization is based on the ordered treemap layout algorithm by Shneiderman and Wattenberg [224], which creates a sorted 2D layout for the IPC hierarchy. The number of patents associated with each category is then mapped to the 3rd dimension forming 'category cubes'. To allow a more efficient rendering, categories that do not have associated patents are represented by a 2D texture. The figures 5.15-5.17 show the composition for the 3D treemap for the IPC hierarchy up to the 2nd level. Each leaf category has a uniform area of 1.

5.5.4 Stacked 3D Treemaps

The set of *pmo:PatentDocument* instances associated with a pmo:ClassificationEntry can be divided into a collection of subsets by grouping the patent documents using the paths provided by the underlying Patent Information Model. Subsets can be grouped for example by the same applicant, inventor, filing year, etc. With this model it is possible to compare patent portfolios of different applicants or to look at the temporal development of a technical field represented by its IPC category.

Multiple patent subsets associated with an IPC category can be visualized by stacking them on each other like in a stacked bar chart. The height of each stack is determined by the number of patents in the corresponding patent set. An example of stacked category cubes is shown in figure 5.18. The patents classified with an IPC category are divided into two sets: one containing the patents of an applicant A (shown in red) the other of applicant B (shown in blue).

3D Edge Bundling

Co-classification relations can be visualized as 3D edges connecting two category cubes in the 3D treemap. An example is shown in figure 5.20 which uses cubic curves for rendering 3D edges.

To better highlight the structure of co-classification relations they can be bundled by taking the IPC hierarchy into consideration. An example of bundled co-classification is shown in figure 5.19. The example uses the Hierarchic Edge Bundling Algorithm of Holten [148] extended for 3D.

The edges are rendered as GL_LINE_STRIP simulating 3D BSplines. The curve points are computed using the free Curve API library [4]. The fundamental extension for

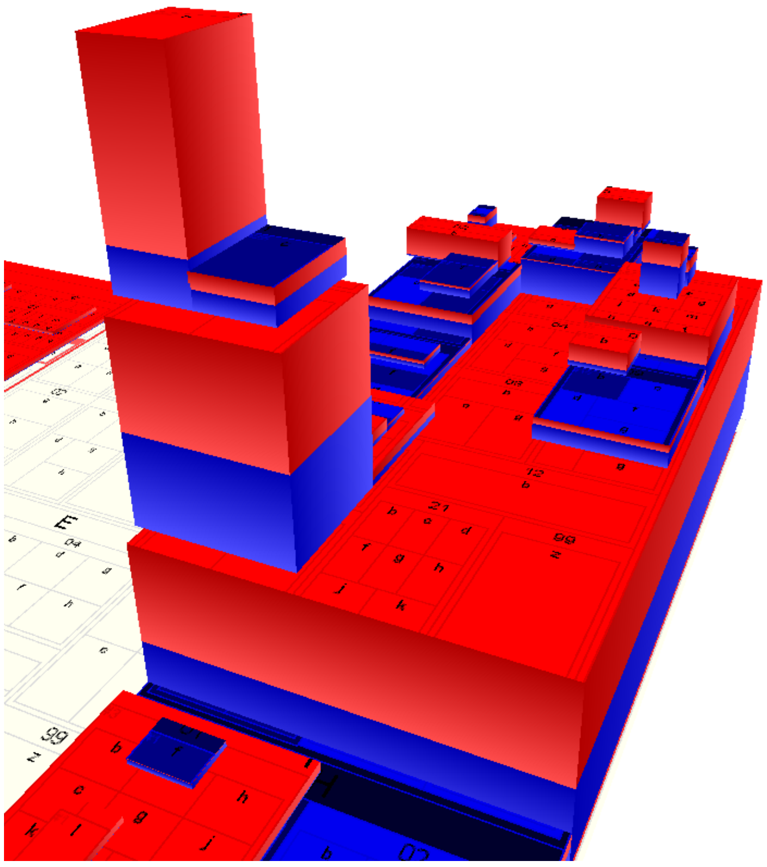


Figure 5.18: Stacked 3D treemap for classification analysis of the patents of two applicants, one is shown in red and the other in blue)

3D compared to Holten's algorithm is the computation of the z-coordinate for each point. This computation is done by linear approximation of the distance between the z-coordinate of the source and target nodes. This implementation has the feature that the resulting bundles go through the category cubes. Therefore they have to be rendered transparently.

In figure 5.19 the co-classification relations of a single patent set are shown. The right upper category has been selected. The selected edge bundles are shown in red, the others in blue. For comparison, figure 5.20 shows the same sample without edge bundling and figure 5.21 depicts the 2D bundling version.

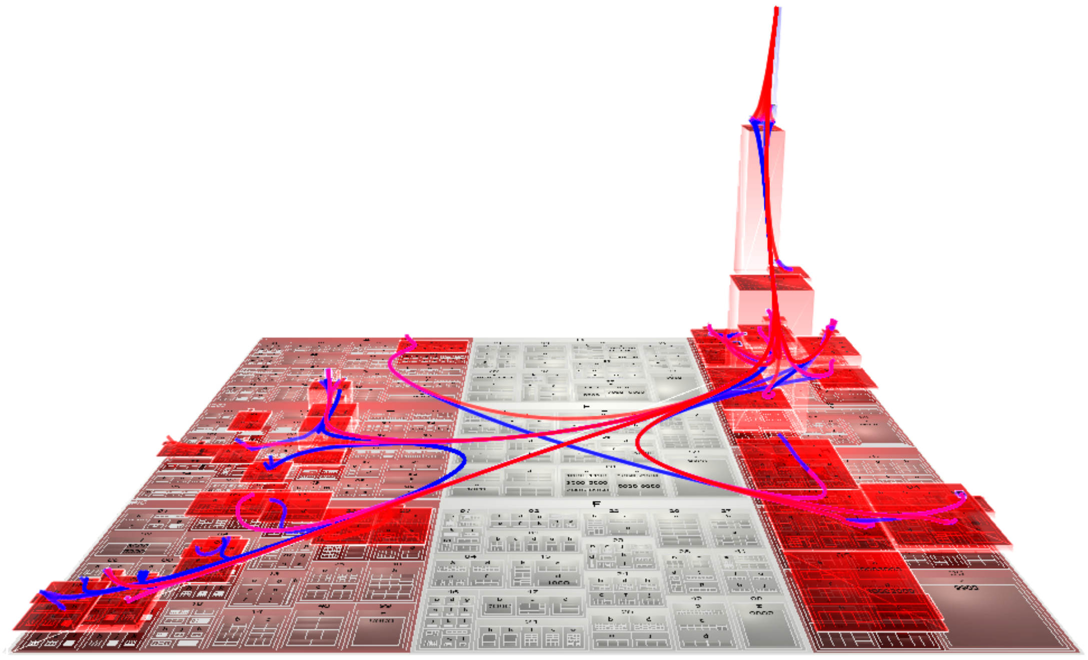


Figure 5.19: Co-classification relations shown as 3D edge bundles

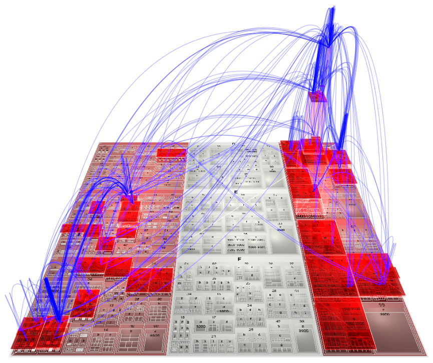


Figure 5.20: Co-classification relations without edge bundles

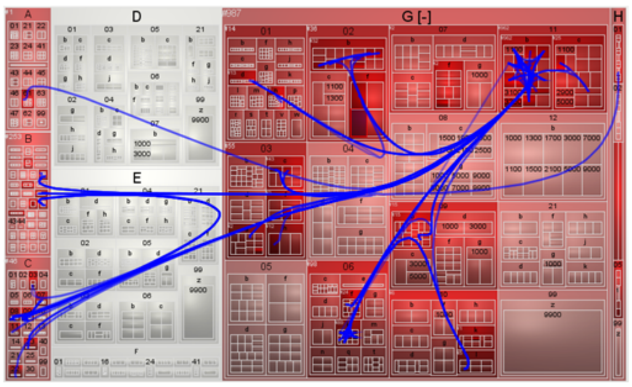


Figure 5.21: Co-classification relations shown as 2D edge bundles

Implementation Issues

For the implementation of the 3D treemap prototype the *Prefuse* framework [142, 141] has been extended with the described 3D treemap layout, 3D edge bundling algorithm, and new OpenGL rendering components using the *Java Binding for the OpenGL API*

(JOGL) [15]. This work is partly based on the diploma thesis of Schröck [216]. Figure 5.22 shows the architecture of the Prefuse extension for 3D.

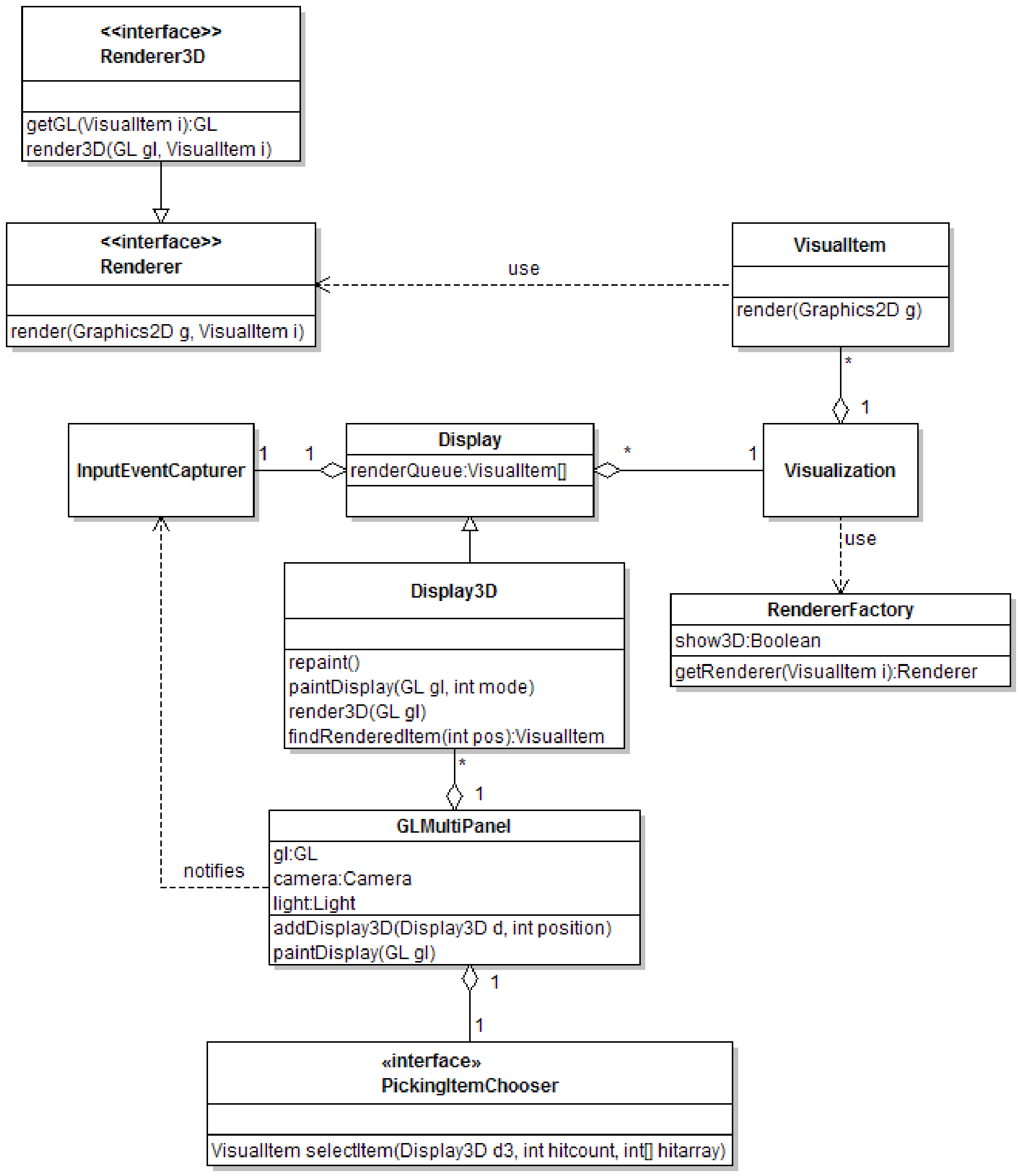


Figure 5.22: Extension of the *Prefuse* framework for 3D and OpenGL

This architecture addresses three main issues. The first is to exchange the 2D graphics context with a 3D context. The second is to exchange all renderer components that use a 2D context with renderer components that use an OpenGL context. And the third is to re-use existing Prefuse controller implementations for handling the user input.

1. The first issue is solved by introducing the *GLMultiPanel* and *Display3D* classes.

The *GLMultiPanel* provides access to the OpenGL API. It also manages multiple *Display3D* instances. A *Display3D* is a subclass of the *Display* class with the purpose of transparently exchanging the 2D graphics API with the OpenGL 3D graphics API.

2. The second issue is solved by customizing the *RendererFactory* so that it can return a *Renderer* for 2D and 3D.

3. To solve the third issue the *GLMultiPanel* functions as a proxy for all mouse and keyboard events. If an event is received, it uses the picking capabilities of OpenGL to determine the corresponding *VisualItem*. It then forwards the event to the *InputEventCapturer* as in the 2D case. Therefore, the 3D extension is completely transparent for the controller instances.

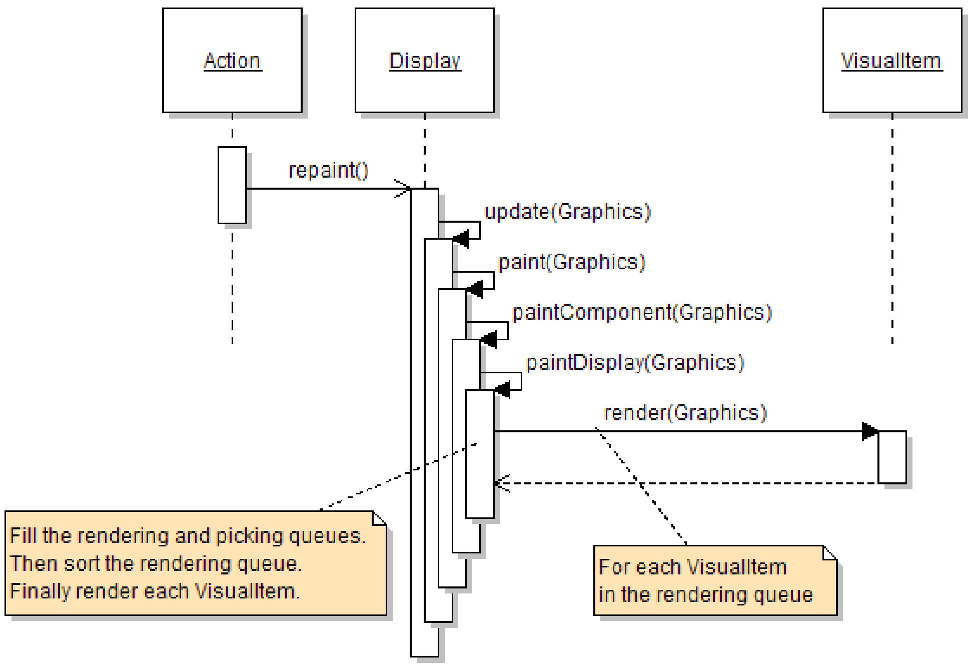

Action
Display
VisualItem
repaint()
update(Graphics)
paint(Graphics)
paintComponent(Graphics)
paintDisplay(Graphics)
render(Graphics)
Fill the rendering and picking queues. Then sort the rendering queue. Finally render each VisualItem.
For each VisualItem in the rendering queue

Figure 5.23: Normal 2D rendering sequence

Figure 5.23 shows how the normal 2D rendering is performed. After the last *Action* has changed the visual structure, an asynchronous paint request is triggered by calling the *repaint()* method. This causes the event dispatching thread to invoke *update()*, *paint()*, *paintComponent()* and *paintDisplay()* in this order on the *Display* instance. The latter fills the rendering and picking queues, sorts the rendering queue and finally renders each VisualItem by calling its *render()* method.

The changed 3D rendering sequence is shown in figure 5.24. As in the 2D rendering sequence an asynchronous paint request is triggered by calling the *repaint()* method of *Display3D*. In contrast to the 2D sequence this triggers a *repaint()* on the *GLMultiPanel* instance. The OpenGL event dispatching thread in turn provides a valid 3D context, which is forwarded to the *paintDisplay()* method of each Display3D associated with the *GLMultiPanel*. The *render3D()* method is responsible for updating the rendering queue and for sorting of the visual items based on their z-index and transparency. All non-transparent items are rendered first.

The *render()* method is invoked the same way as in for 2D with the only difference that no 2D context is provided. Instead, the 3D context is read from the corresponding *Display3D* container (figure 5.25). Finally, the rendering is done by reading required information from the *VisualItem* and by performing calls on the OpenGL API. It has been a design decision not to extend the *VisualItem* class, because of backward compatibility reasons and the possibility of updating to new Prefuse versions. If this is not required, the *VisualItem* class could be extended by overloading the *render()* method to take a 3D context as parameter.

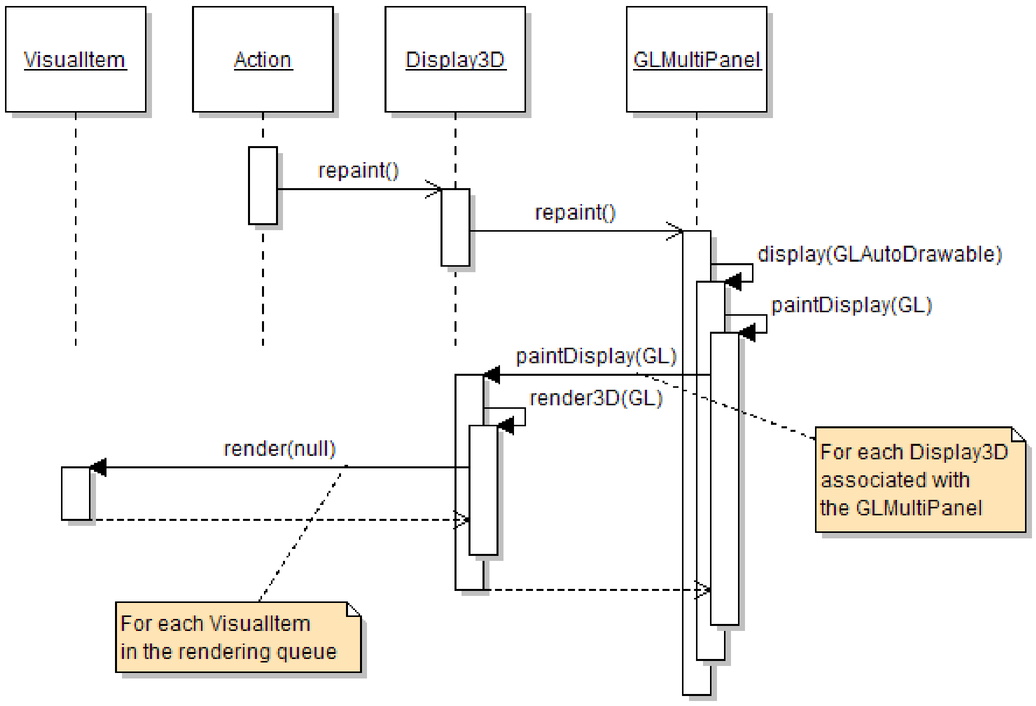


Figure 5.24: 3D rendering sequence

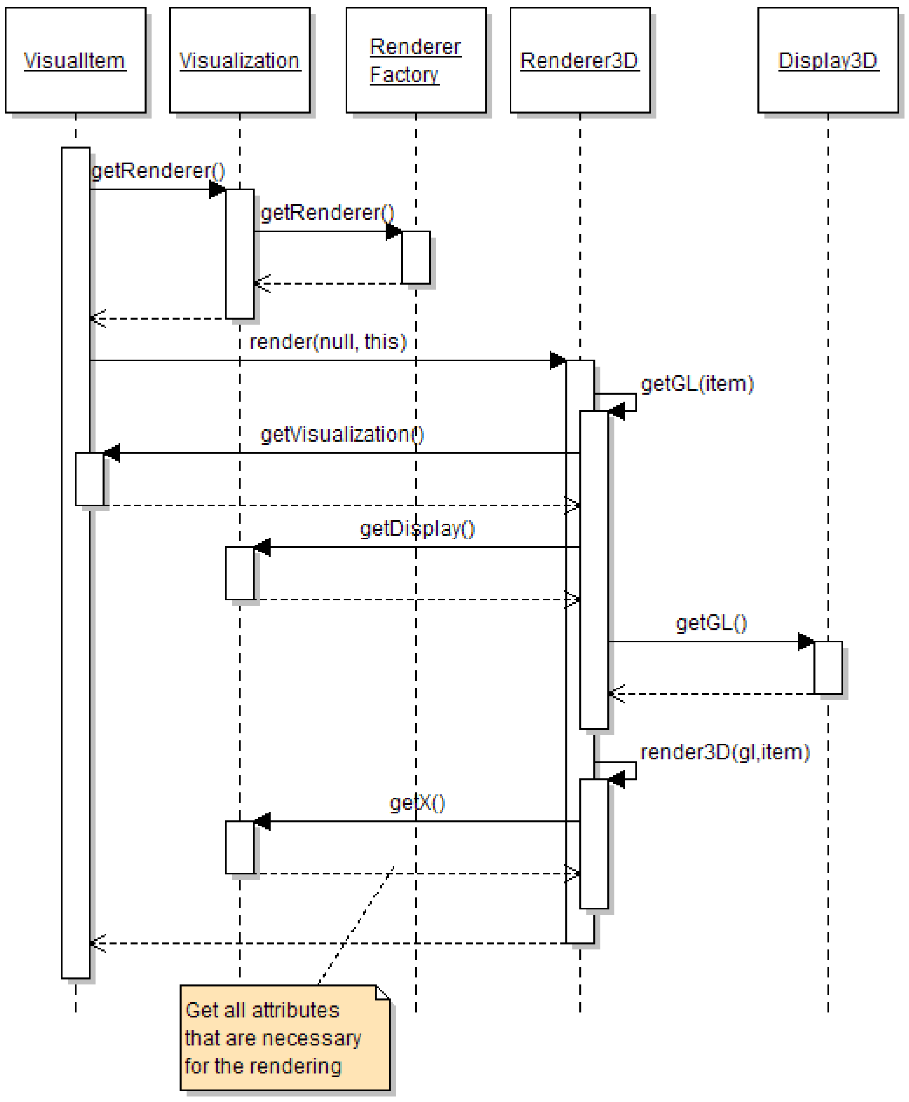


Figure 5.25: 3D rendering sequence (continued)

Discussion

The proposed 3D treemap visualization technique uses a 2D layout algorithm to generate textures and to compute the x and y-coordinates of the 3D items. The z-coordinate and the height are determined by the amount of patents associated with an IPC category. In order to limit the visualization's height, absolute numbers have to be scaled. The data model allows to distinguish between direct and indirect classifications. If a depth limit is set in the visualization, for example only to show IPC categories up to the main-group level, then it is useful to also display indirect classifications because otherwise information will be lost.

The 3D treemap visualization technique also allows for example the comparison of patent portfolios of different applicants by utilizing *'stacked category cubes'*. Stacked cubes can further be used for analyzing other aspects, such as the temporal development of categories, by utilizing the available paths in the Patent Information Model Graph. Each stacked cube is shown in a different color or with a different texture. The readability of this visualization is limited if there is a large number of stacks.

For supporting co-classification analysis of patent sets, the visualization shows co-classification relations as 3D edge bundles. The 3D edge bundling is based on the hierarchic edge bundling technique proposed by Holten [148], but has been extended for 3D. Currently the z-coordinate of curve points is computed by a linear approximation of the distance between the z-coordinate of the source and target nodes. For future work Holten's edge bundles could be combined with the routing of entity relations in the Hierarchical Net proposed by Balzer and Deussen [60]. Also for further analyses of co-classifications the edge bundles could be inspected in more details, for example as proposed by Panagiotidis et al. [189] in their *Edge Analyzer* system.

5.6 Conclusions

In this chapter several techniques for visualizing resource-level patent metadata have been described. The focus has been on the visualization of bibliographic relations between patents, temporal, geographic and classificatory distributions. The aim of the proposed visualizations is to support users in getting an overview of the structure of patent collections by providing a visual user interface for the contained patent metadata.

Bibliographic data and their relations are visualized as graph. The node types are customizable and can be, for example, applicants, inventors, patents, classifications, etc. Temporal distributions are show in timeline or interactive matrix visualizations. Additionally, to a normal scatter plot, the interactive matrix also shows relations between the items as edges. Therefore, an interactive matrix more behaves like a graph then a scatter plot. Finally, treemap based visualizations for presenting classificatory distributions of patent sets and their co-classification relations have been described for 2D and 3D. Technically, it has been shown how the 2D Prefuse visualization framework can be extended for 3D. First investigation on this topic have been conducted by Schröck within his diploma thesis [216].

A central new feature is a dynamic visual mapping that is defined by SPARQL queries on an underlying Patent Metadata Ontology instance. Two different visual mapping approaches have been described in section 5.2. The first uses SPARQL CONSTRUCT statements in order to map fragments of the *Patent Metadata Ontology* onto a *Graph Visualization Ontology*. The second approach supports a simple and fast visual mapping by interpreting the variables of SPARQL SELECT statements for the generation of hierarchies. The main advantages of such SPARQL based visual mappings are the flexibility given to the user and the seamless integration of visualizations with semantic networks in RDF. A shortcoming is that the user has to have detailed knowledge about SPARQL and the used ontologies.

The visualizations are intended to be combined in a multiple view environment, such as the *PatWiki* described in chapter 8 or the *PatViz Desktop* described in chapter 9.3.1. In such environments the visualizations can also be used for defining filters in order to interactively change the overall display of the underlying patent collection. Further, other user interfaces such as charts, tables, previews, etc. can be incorporated.

So far there has been no detailed user study that compares the proposed visualizations with other user interfaces but a promising preliminary evaluation of the PatViz Desktop including the visualizations can be found in Koch et al. [170]. This article also describes how the visualizations can interact with each other in order to be efficiently used for complex patent analysis scenarios.

Beside the mentioned visualization approaches of resource-level patent metadata the integration with content-level visualization is essential. The latter will be described in the next chapter.

CHAPTER 6

Visualization of Content-Level Metadata

Content-level annotations semantically describe content parts of a resource, such as paragraphs, sentences, multi-term expressions or even single characters. Content-level annotations typically are instances of domain concepts linked with natural language expressions or relations between such instances, for example motion or part-whole relation as described by Lenzi and Pianta [178]. Content-level annotations therefore bridge semantic abstractions with concrete lexical forms.

Content-level annotations may be connected with other concepts. They form a semantic network we call *content annotation graph*. Nodes that are directly linked with content parts are called *annotation nodes*. They are associated with exactly one content part in one document. All other nodes are called *additional nodes*. Figure 6.1 gives an overview.

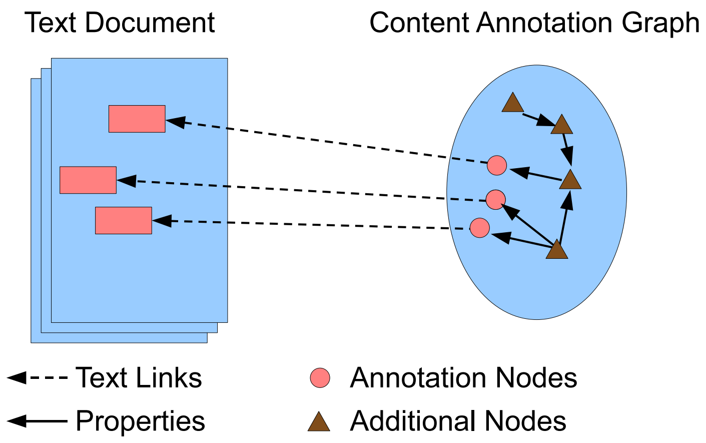


Figure 6.1: Content annotation graph

The basic idea proposed in the chapter is to visually align text documents with their content annotation graphs. The approach is to provide a layered visualization that shows the text document at one layer and its associated content annotation graph at another layer. The text layout is used for the layout of annotation nodes. The position and size of the annotation nodes is determined by the bounding boxes of the corresponding text expressions. The edges are routed so that the text is permanently readable. Additional nodes are included on demand to show further information.

The next section describes the state of the art in this area. Afterwards, it follows a discussion of how to associate content-level annotation with text documents (section 6.2). Further, section 6.3 describes a suitable data structure for the visualizing content-level annotations. A new visualization technique for content-level annotations is proposed in section 6.4. Finally, section 6.5 discusses a distortion technique for reducing the distance between annotation nodes in the document.

6.1 Related Work

There have been various approaches for visualizing graph structures [146, 54, 228]. Prominent examples of applying these algorithms to RDF graphs are Welkin [35] or IsaViz [195]. For interacting with large graphs various Focus+Context techniques have been proposed [114, 67, 159]. Beside these graph-based approaches the Fresnel RDF display vocabulary [196] provides a simple rule based mapping from RDF graphs to a presentation layer such as HTML and CSS. Fresnel for example is used in the Lena [17] and Longwell [18] RDF browsers which provide faceted explorative browsing of RDF data. RDFa [269] is a standard for embedding RDF statements in XHTML documents. Although it is originally intended as a different encoding for RDF, it could be used for highlighting the marked up elements by appropriate interpretation and manipulation of the DOM structure of RDFa documents. Examples for processing documents that contain RDFa are Google's Rich Snippets [10] and RDFa Bookmarklets [28].

Qiu and Elsayed [201] proposed a technique for supporting text-based reading by non-linear navigation. Additionally to the original text they visualize a hierarchical semantic structure as a tree. Semantic concepts can be associated with sentences of the text. If a sentence is highlighted, then the associated concepts are shown in a third tree visualization that shows the current subset of the hierarchical semantic structure. However, this approach does not support the visualization of content annotation graphs because it is restricted to hierarchies. It further does not visually integrate a content annotation graph into the original document.

6.2 Association of Content-Level Annotations

Generally there are two different methods of associating content-level annotations with patent resources. One technique is *embedding* the annotations into the document and the other is *linking* the annotations using a path expression language. As an example the following code fragment shows a fragment of claim 7 of patent EP93111773B1 encoded in ST.36.

```
...
<claim num="claim-7">
   <claim-text>A toner pack according to any one of claims 1 to 6, characterized in
      that the toner pack (30) further comprises a flange (32) ... </claim-text>
</claim>
...
```

As result of a semantic analysis the annotation a *sumo:hasPart* relation has been identified between *'toner pack'* and *'flange'*. It follows a comparison between the embedding and linking method.

6.2.1 Embedding

A common technique for embedding semantic annotations into XML documents is RDFa [269]. RDFa defines a collection of attributes and a set of processing rules how these attributes are to be interpreted. RDFa parsers have to implement these rules in order to extract the contained RDF statements. An online RDFa parser is for example available at [20]. The RDF extraction process has to consider the RDFa attributes and their location within the XML document structure. An important feature of RDFa is the usage of selected text parts in XHTML for the definition of RDF literals in order to link between the human- and the machine-readable representations.

The following code fragment gives an example of using RDFa for annotating ST.36 encoded patent documents. Two additional *span* elements are introduced that contain the RDFa attributes. The *about* attribute specifies the subject of a statement. In the example there are two blank nodes (_:tonerPack and _:flange) as subjects. The *property* attribute defines the predicate between the current subject and the current element content as literal. The *rel* attribute in contrast defines the predicate between the current subject and an RDF resource as object given by the *resource* attribute.

```
<ep-patent-document id="EP93111773B1"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:sumo="http://www.patexpert.org/sumo#" >
  <claim id="claim-7">
     <claim-text>A <span about="_:tonerPack" property="rdfs:label" resource="_:flange" rel="
         sumo:hasPart">toner pack</span> according to any one of claims 1 to 6, characterized in
        that the toner pack (30) further comprises a <span about="_:flange" property="rdfs:label">
            flange</span> (32) ... </claim-text>
  </claim>...
</ep-patent-document>
```

Based on this RDFa code the following RDF statements can be extracted:

```
_:tonerPack rdfs:label "toner pack" .
_:tonerPack sumo:hasPart _:flange .
_:flange rdfs:label "flange" .
```

RDFa gives publishers the possibility to enrich their content with semantic annotations so that the content can be easier interpreted by machines. The shortcomings of this approach are that (i) the original documents have to be changed, (ii) existing XML Schema definitions [259] are potentially violated, because new attributes are introduced, and (iii) the link between the human- and the machine-readable representations is lost in the extracted RDF triples.

Schmedding [214] proposed a first solution to the latter problem by integrating RDFa into XML parsers and extending the Document Object Model (DOM) so that DOM elements are associated to those RDF statements that have been defined in their scope. A different approach is described in the next section.

6.2.2 Linking

Linking using *Uniform Resource Identifiers (URI)* [69] allows the association of separately stored semantic annotations with web resources. URIs are also the basic concept in RDF for making statements about resources. An advantage of linking is that original documents do not have to be changed. This is in particular important for resources that are not accessible for writing, such as published patent documents. Also existing XML Schema definitions cannot be violated when semantic annotations are linked. A shortcoming of linking is that linked annotations may become invalid without any notification when the content structure is changed. This is not the case for published patent documents, because they are immutable.

A simple way for referencing specific content parts is using URI fragment identifiers. Unfortunately there is no defined semantics for general URI fragment identifiers. The *XML Pointer Language (XPointer)* [252] fills this gap. XPointer provides different schemes that define how fragment identifiers are to be interpreted. It further allows the definition of custom schemes, for example for using the *XML Path Language (XPath)* [250]. This offers a great flexibility for addressing the internal structures of XML documents. For linking RDF annotations a custom *token()* scheme is defined as an abbreviation for XPath expressions as described below. This approach enables the usage of URI references as the subject and object of RDF triples in contrast to blank nodes as in the embedding approach described before. But it requires that unique identifiers are defined for addressed each XML element, which is the case for our target content format ST.36.

The following RDF snippet gives an example for external content-level annotations of a patent document stored in the ST.36 format using XPointer in combination with XPath. The fragment identifiers refer to the textual content of a *claim* element with an *id* attribute with the value 'claim-7'. The subject references the multi-word expression 'tone pack' (character 3 to 13), whereas the object references the term flange (character 115 to 121). Referring to the normalized text content of the claim element has the advantage that optional markup for highlighting and formatting is ignored.

```
pat:EP93111773B1#xpath(substring(normalize-space(string(//claim(@id='claim-7'))),3,13))
  sumo:hasPart
    pat:EP93111773B1#xpath(substring(normalize-space(string(//claim(@id='claim-7'))),115,121)) ;
  rdfs:label
    "tone pack" .
pat:EP93111773B1#xpath(substring(normalize-space(string(//claim(@id='claim-7'))),115,121))
  rdfs:label
    "flange" .
```

In order to abbreviate the URI fragment identifiers a new customized *token()* scheme is introduced as XPointer scheme. It is used for referencing expressions for example in the claim and description section.

The *token()* scheme has three arguments:

1. Element identifier
2. Start position of the token in the content element's text
3. End position of the token

Using this scheme the above fragment identifiers can be abbreviated as

```
pat:EP93111773B1#token('claim7',3,13) a pmo:Annotation ;
  sumo:hasPart pat:EP93111773B1#token('claim-7',115,121) ;
  pmo:token "tone pack" .
pat:EP93111773B1#token('claim-7',115,121) a pmo:Annotation ;
  pmo:token "flange" .
```

6.3 Data Model

This section describes an appropriate data model for visualizing content annotation graphs together with its associated text documents. There are two major aspects. One is the realization of the linking between RDF graphs and XML documents and one is the visual abstraction. Figure 6.2 gives an overview of the data model.

Linking between RDF and XML

An *RDFGraph* instance is a collection *RDFTriple* instances. Each triple has a *RDFResource* as subject, a *RDFProperty* as predicate and a *RDFNode* as object component. *RDFLiteral* and *RDFUriResource* are subclasses of *RDFNode*. *Annotation* is a specialized *RDFUriResource* which interprets the fragment part of the URI as XPointer. It references a fragment of the text content of a specific *DOMElement*. Annotations bridge RDF graphs and DOM documents using XPointers as described in the previous section. The resource is resolved by using a *UriResolver* instance, for example for resolving patent documents via HTTP.

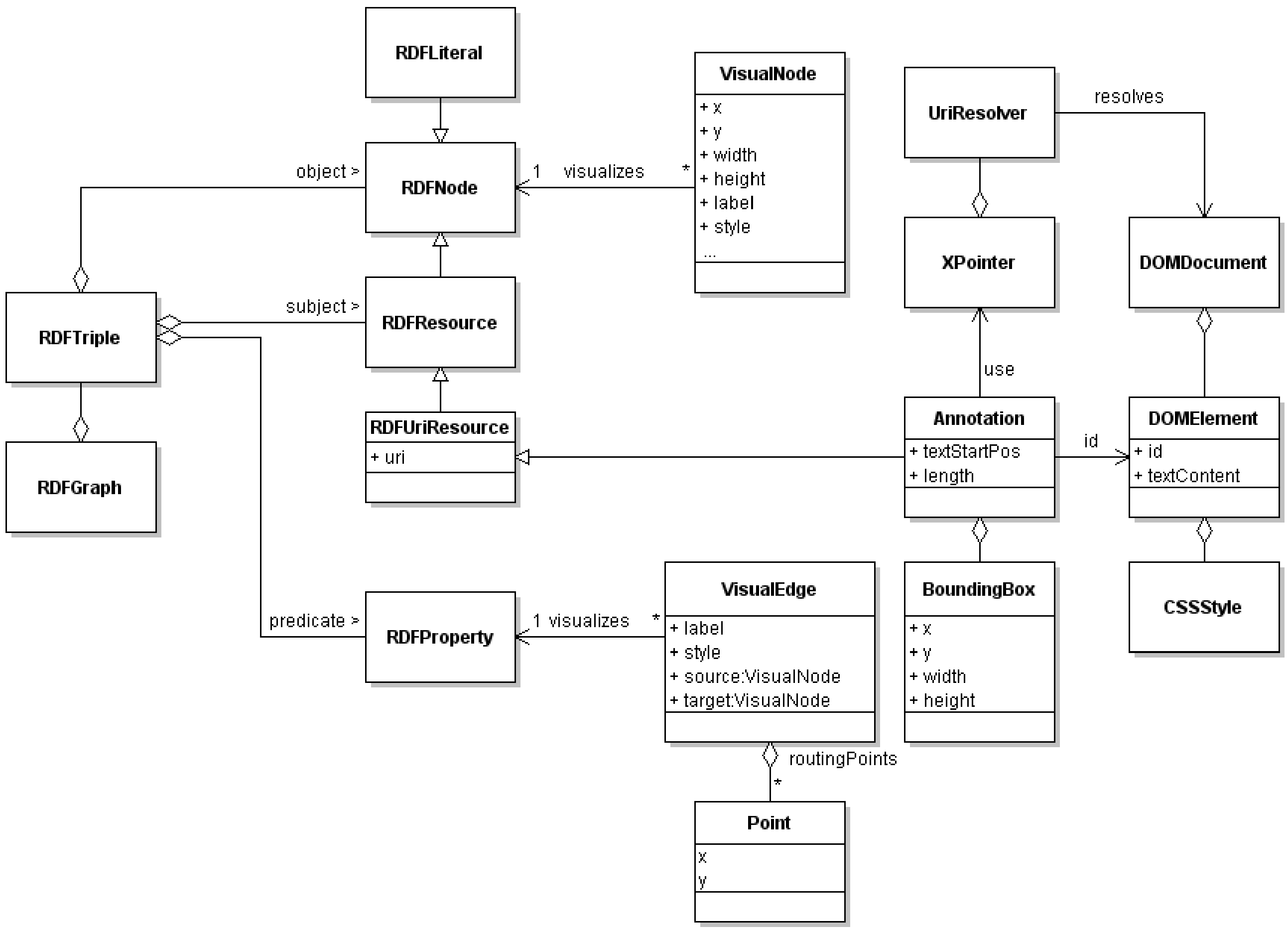


Figure 6.2: Data model for visualizing semantic annotations

Visual Abstraction

The purpose of *VisualNode* instances are to graphically present content annotation nodes and additional nodes. *VisualEdge* instances are created for the presentation of relations between the nodes. Edges can be routed as explained in the next section. The result of the routing is stored in a routing points array for being interpreted by an edge renderer. A *BoundingBox* is computed for each text part referenced by an annotation. The bounding box defines the position and size in screen coordinates. It usually depends on the style definitions of the element (font-style, font-size, etc.). Information about the bounding box is retrieved from the component that is responsible for the document layout.

It is important to mention that one *RDFNode* can be visualized by multiple *VisualNodes*. Also one *RDFProperty* can be visualized by multiple *VisualEdges*. This is to give concrete visualizations more flexibility in the presentation, for example to redundantly show additional nodes or to show a content annotation node in several parts if there are line breaks between the associates text parts.

Algorithm

1. Based on the input graph the annotations are extracted by looking up instances with the type *pmo:Annotation*. The result is a list of *Annotation* objects initialized with an URI that points to the annotated text fragment.
2. For each annotation object (1) the XPointer is resolved, (2) the bounding box is computed based on the text position of the referenced token and (3) a *VisualNode* instance is allocated, linked with the annotation and initialized with the annotation's bounding box.
3. For each relation between two annotations (determined by a SPARQL query) a *VisualEdge* instance is allocated.
4. For each non-annotation *RDFNode* associated with an annotation a *VisualNode* instance is allocated. The result is a list of context nodes.
5. For each relation starting with or ending at a context node a *VisualEdge* instance is created. The result is a list of context edges.

The creation of nodes and edges is defined by SPARQL queries. By adding additional constraints to the queries the creation process can be filtered. Also the presentation style for nodes and edges can be controlled by SPARQL queries. For example instances of type *pulo:Material* could be shown in green while instances of type *pulo:Function* could be shown in blue. The next section describes the visualization technique in more detail.

6.4 Visualization Technique

The proposed technique for visualizing content-level annotations combines text layout with a graph-based layout. This is reached by showing different aligned layers on top of each other. At the first layer the content annotation graph is shown. The second layer on top of the first shows the text document with a transparent background. To have the text on top of the graph ensures that the text is permanently readable. For the text layer a standard HTML engine is used. The implementation uses the HTML 3.2 compliant *JEditorPane* shipped with the Java2 standard edition. For the graph layer the Prefuse visualization framework [142] is used.

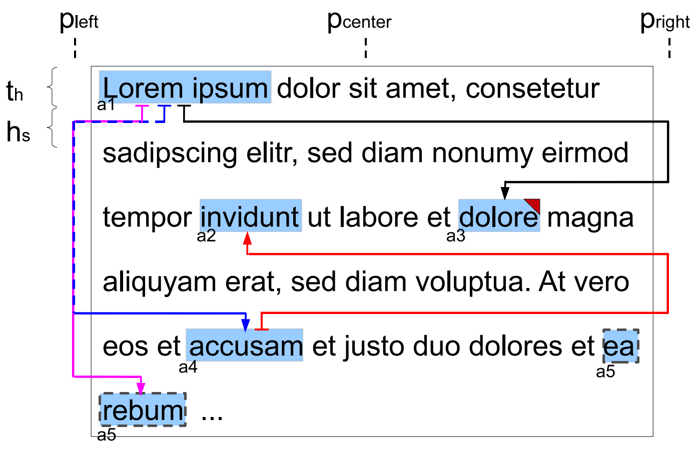

pleft
pcenter
pright
th
hs
Lorem ipsum dolor sit amet, consetetur
a1
sadipscing elitr, sed diam nonumy eirmod
tempor invidunt ut labore et dolore magna
a2
a3
aliquyam erat, sed diam voluptua. At vero
eos et accusam et justo duo dolores et ea
a4
a5
rebum ...
a5

Figure 6.3: Routing of edges between content annotations

Figure 6.3 shows an example of content-level annotations for a sample text. The annotation nodes $a_1, \ldots, a_5$ are visualized as rectangles. The position and size is determined by the position and size of the associated text fragment. For the case that an annotation references a multi-word expression, the first and the last character of the expression define the bounding box. If the expression spans multiple lines, i.e. when there is a line break in between, multiple annotations nodes are shown. It is indicated that both belong together by using a dashed stroke as shown for annotation a_5 (ea rebum).

Domain specific aspects, such as material used, process type, connection type, etc. can be encoded by using different background colors. For a customizable mapping of ontology classes to style information the Fresnel vocabulary could be used. Alternatively, SPARQL queries could be enriched with style information as proposed in section 5.2.

The availability of additional information, for example comments that have been added manually, can be indicated by additional glyphs on top of an annotation node. Annotation a_3 (dolore) gives an example. It shows a red triangle glyph as indicator as known from different office applications. For computing the layout of arcs and additional nodes different techniques can be applied.

6.4.1 Arc Routing

Relations between annotated text parts are visualized as arcs. Direct lines are a simple realization of arcs. The problem with this approach is that arcs overlap with the text. The text may be more difficult to read. One approach to solve this problem is the representation of arcs as list of routed line segments rendered between the text lines. An example of edge routing is shown in figure 6.3.

The input to the edge routing algorithm is a list of VisualEdge instances each containing two VisualNode objects, one source and one target node[1]. As output the algorithm computes an array of points interpreted as line strips. The algorithm has a time complexity of $O(n)$ with n the number of relations.

The points can be computed locally for each relation based on the position and the size of its source and target nodes. There are two basic cases depending on whether the source and target nodes are adjacent or not. Nodes are adjacent if they are shown in the same text row or in two text rows the directly follow each other. In the case that nodes are not adjacent the vertical segment is routed either on the left side or on the right side with position x_{left} or x_{right}. To check whether if the segment is to be routed left or right, the x position of the target node is checked against the center of the page (x_{center}) The edge routing algorithm described below uses the following functions:

$$up_1(node) = \begin{pmatrix} node.x + node.width/2 \\ node.y \end{pmatrix}$$

$$up_2(node) = \begin{pmatrix} node.x + node.width/2 \\ node.y - hs/2 \end{pmatrix}$$

$$down_1(node) = \begin{pmatrix} node.x + node.width/2 \\ node.y + node.height \end{pmatrix}$$

$$down_2(node) = \begin{pmatrix} node.x + node.width/2 \\ node.y + node.height + hs/2 \end{pmatrix}$$

[1] We will write r.s for the source and r.t for the target node of a relation r.

$$left_{down}(node) = \begin{pmatrix} x_{left} \\ node.y + node.height + hs/2 \end{pmatrix}$$

$$left_{up}(node) = \begin{pmatrix} x_{left} \\ node.y - hs/2 \end{pmatrix}$$

$$right_{down}(node) = \begin{pmatrix} x_{right} \\ node.y + node.height + hs/2 \end{pmatrix}$$

$$right_{up}(node) = \begin{pmatrix} x_{right} \\ node.y - hs/2 \end{pmatrix}$$

Algorithm

1. For each relation r:

2. Is r.t is adjacent to r.s?

2.1. yes: define $points(r)$ as:

$$points(r) = \begin{cases} r.t.y < r.s.y: & [up_1(r.s), up_2(r.s), down_1(r.t), down_2(r.t)] \\ r.t.y > r.s.y: & [down_1(r.s), down_2(r.s), up_1(r.t), up_2(r.t)] \\ r.t.y = r.s.y: & [up_1(r.s), up_2(r.s), up_1(r.t), up_2(r.t)] \end{cases}$$

2.2. no: check if $r.t.x \leq x_{center}$

2.2.1 yes: define $points(r)$ as:

$$points(r) = \begin{cases} r.t.y < r.s.y: & [up_1(r.s), up_2(r.s), left_{up}(r.s), left_{down}(r.t), down_1(r.t), down_2(r.t)] \\ r.t.y > r.s.y: & [down_1(r.s), down_2(r.s), left_{down}(r.s), left_{up}(r.t), up_1(r.t), up_2(r.t)] \end{cases}$$

2.2.2 no: define $points(r)$ as:

$$points(r) = \begin{cases} r.t.y < r.s.y: & [up_1(r.s), up_2(r.s), right_{up}(r.s), right_{down}(r.t), down_1(r.t), down_2(r.t)] \\ r.t.y > r.s.y: & [down_1(r.s), down_2(r.s), right_{down}(r.s), right_{up}(r.t), up_1(r.t), up_2(r.t)] \end{cases}$$

3. set $r.points = points(r)$

This algorithm has the following features:

- The text is always readable. There are no intersections of text and arcs.

- Edge bundles can be visualized easily
- Multiple edges between the same two nodes are routed along the same path, therefore it can be difficult to identify the origin and/or the target.
- Line spacing has to be large enough, it is assumed that the $hs <= ht$

6.4.2 Additional Nodes

There are different approaches to visualize additional information associated with annotation nodes, each with its pros and cons. One approach is to show this information in a popup window similar to a tool tip. Textual information could be appropriately presented. But it would be difficult to visualize structural information as arcs between additional and annotation nodes.

A second approach is to present additional information as a graph, either in a third layer on top of the text or at a different display region that does not overlap with the text. The node layout can be computed for example using a force-directed or radial algorithm with annotation nodes having a fixed position. Structural information can directly be visualized as arcs. Disadvantages are interchanging node positions in case of force-directed layout and a segmented presentation of property name-value pairs with property names shown as arc labels and property values as nodes.

6.4.3 Discussion

If there is a large number of annotation or additional nodes – the worst case would be that each text part is annotated and linked with other text parts – the proposed visualization technique may negatively influence the reading process. A solution here is to filter relevant annotations based on facets or based on a semantic search in advance.

If there are multiple arcs between two annotation nodes or if there are multiple outgoing or incoming edges it may be difficult to identify the source and target nodes. One solution can be edge bundling [148] or using an edge analyzer as described in Panagiotidis et al. [189]. Also interaction techniques can be applied for emphasizing the ingoing and de-emphasizing the outgoing arcs, when moving the mouse pointer over it. A related problem is the labeling of edges and rendering of arrow icons. If there are multiple ingoing arcs, the labels may overlap and may be difficult to read. Another problem occurs, if annotations are distributed among a long text so that they are not in same view port. A solution to this problem is described in the next section in terms of a distortion technique.

6.5 Annotation Driven Focus+Context

This section describes an interaction technique, called *semantic shrink*, for showing annotation nodes close to each other that are originally far away from each other in the text. The semantic shrink technique has basically two application scenarios. First, it can be applied to a pair of annotation nodes connected by a semantic relation, when the user selects that relation. Second, it can be applied for highlighting the hits of a semantic search. The underlying hypothesis for the second scenario is that the hits are of special interest to the user. Therefore the attention of the user is drawn to these hits by emphasizing them and de-emphasizing text parts that are away from the hits. This makes it possible to read the context around the hits and to show more hits in the current view port with less scrolling. It also improves the perception of semantic relations visualized as routed arcs as described in the previous section, since their structure can better be shown in the available view port.

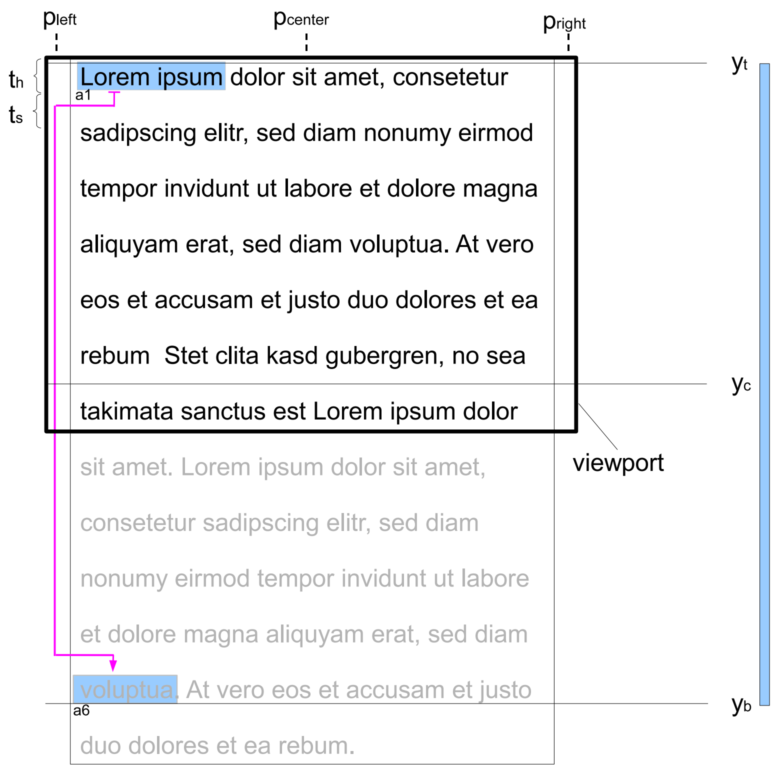
The basic idea of semantic shrink is to reduce the vertical distance by applying a distortion function. In contrast to general fisheye views [114, 236] that magnify areas of greater interest, semantic shrink is used to demagnify areas of lower interest. The distortion function will cause text lines between annotation nodes to be compressed depending on the distance to the annotations.

As shown in figure 6.4 a pair of annotation nodes define the top (y_t) and bottom (y_b) of the area to be reduced. This area is called shrink area. The center y_c is defined as the arithmetic mean ($y_c = y_t + \frac{y_b - y_t}{2}$).

Each line within the shrink area is applied to a positioning function P_{shrink} and a scaling function S_{shrink}. The positioning function modifies the vertical position of a text line. The scaling function modifies the text height and width. Both functions take two parameters, x and a. The first parameter x is the relative vertical position of a text line within the shrink area. x is normalized to the interval $[0, 1]$ ($x = 0$ for the first line, $x = 1$ for the last). The second parameter a denotes the ratio of the current height of the shrink area and its original uncompressed height with $0 < a \leq 1$.

Distortion Functions

$P_{shrink}(x, a)$ denotes the relative vertical position of the distorted line of text. For an uncompressed shrink area ($a = 1$) the distortion function is required to return $f_1(x, 1) = x$. For a highly compressed shrink area (a close to 0) the distortion function must retain $f_2(0, a) = 0$ and $f_2(1, a) = 1$, but have similar values for x around 0.5 to place middle lines very close together. This allows keeping above average line heights for lines at the beginning and end of the shrink area. The distortion function P_{shrink} is defined by linearly interpolating f_1 for $a = 1$ and f_2 for $a = 0$:



p_{left}
p_{center}
p_{right}
t_h
t_s
a1
a6
y_t
y_c
y_b
viewport
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum.

Figure 6.4: Semantic shrink

$$P_{shrink}(x,a) = af_1(x) + (1-a)f_2(x) \tag{6.1}$$

An appropriate function for f_1 is the identity function.

$$f_1(x) = x \tag{6.2}$$

An appropriate function for f_2 is given in equation 6.3. Figure 6.5 (left) shows the function plots for different degrees. This finally leads to the positioning function defined in equation 6.4.

$$f_2(x) = 2^{n-1}(x - \frac{1}{2})^n + \frac{1}{2} \quad \text{with odd } n \geq 3 \tag{6.3}$$

$$P_{shrink}(x,a) = ax + (1-a) \cdot 2^{n-1}(x - \frac{1}{2})^n + \frac{1}{2} \quad \text{with } 0 \leq x \leq 1\ ,\ 0 < a \leq 1 \text{ and } n \geq 3 \tag{6.4}$$

Good results can be obtained for $n = 5$:

$$P_{shrink}(x,a) = ax + (1-a)16(x - \frac{1}{2})^5 + \frac{1}{2} \tag{6.5}$$

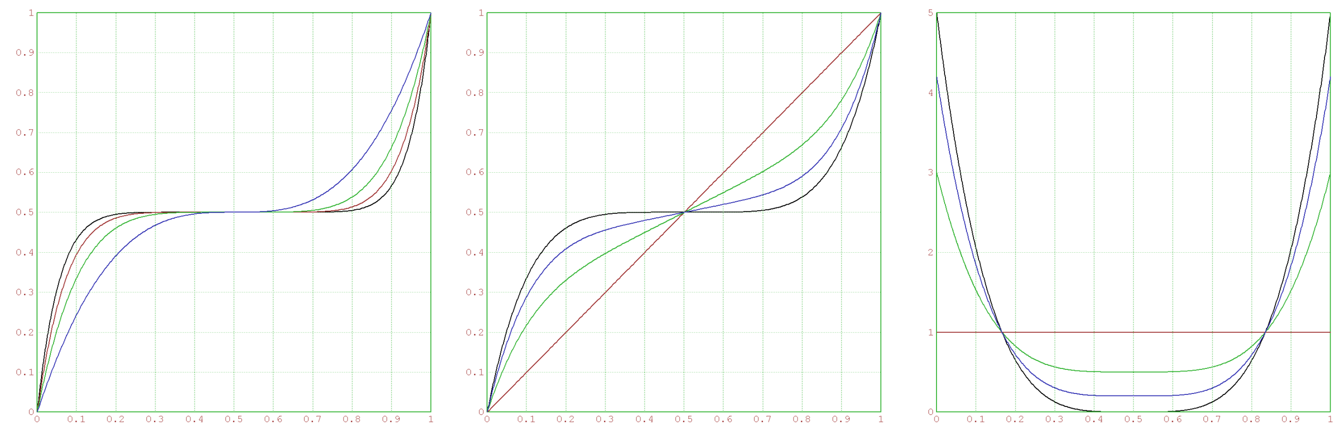


Figure 6.5: Left: Plot of formula 6.3 with n=9 (black), n=7 (red), n=5 (green) and n=3 (blue); Center: $P_{shrink}(x, a)$; Right: $S_{shrink}(x, a)$ both for $n = 5$ and $a = 0.0001$, $a = 0.2$, $a = 0.5$ and $a = 1$

In order to scale the height of the individual lines in accordance with the overall distortion, the height is multiplied with a scaling factor returned by the scaling function S_{shrink} which is defined as linear interpolation of the first derivative of f_1 and f_2:

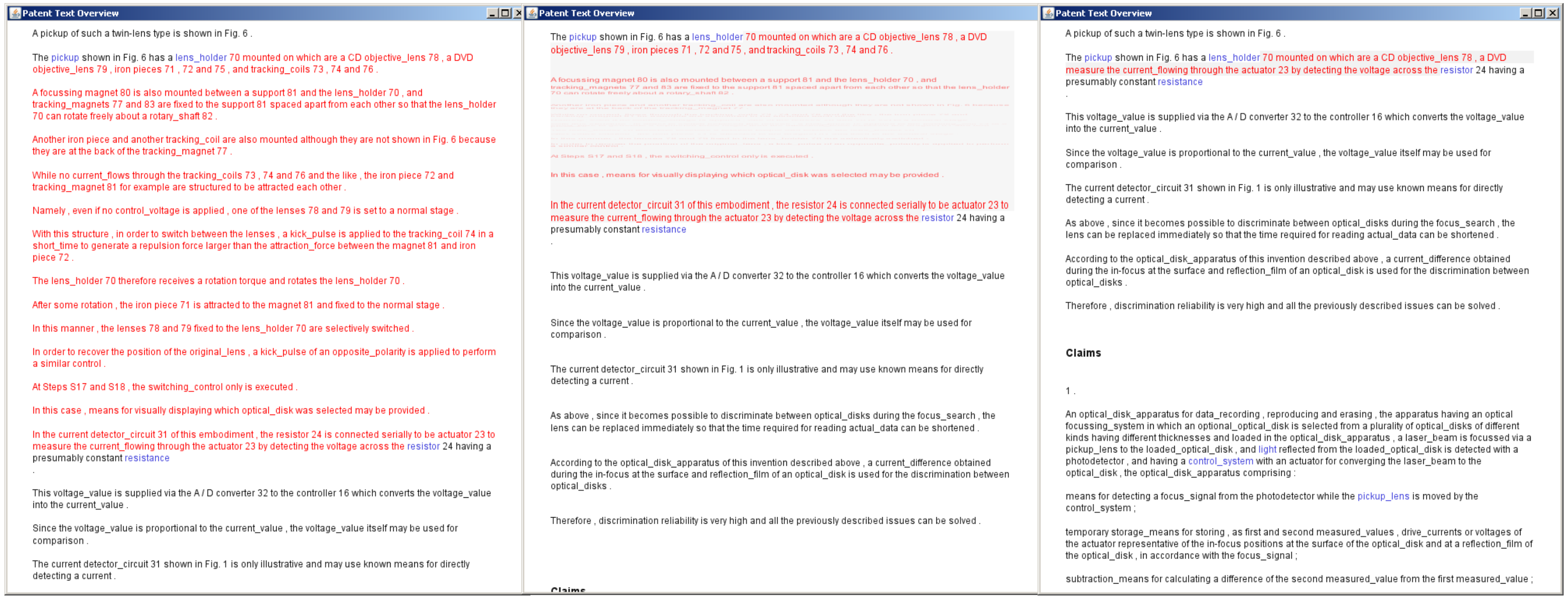
$$S_{shrink}(x, a) = af_1'(x) + (1 - a)f_2'(x) \tag{6.6}$$

For $n = 5$ the scaling function is:

$$S_{shrink}(x, a) = a + (1 - a) \cdot 80 \cdot (x - \frac{1}{2})^4 \tag{6.7}$$

Iteration of a in the interval (0,1] allows to animate the shrink process. The function plots of $P_{shrink}(x, a)$ and $S_{shrink}(x, a)$ for different values of a are shown in figure 6.5 (center and right). As an example figure 6.6 shows the animated shrinking of a text area (red) between two annotations (blue).

Figure 6.7 shows the distortion of a sample text with the corresponding scale factor (0.0 red, 1.0 green and 2.0 blue). If text lines get too small to be read a threshold can be set to skip these lines. The following lines can be moved up, as shown in figure 6.8.



A pickup of such a twin-lens type is shown in Fig. 6 .

The pickup shown in Fig. 6 has a lens_holder 70 mounted on which are a CD objective_lens 78 , a DVD objective_lens 79 , iron pieces 71 , 72 and 75 , and tracking_coils 73 , 74 and 76 .

A focussing magnet 80 is also mounted between a support 81 and the lens_holder 70 , and tracking_magnets 77 and 83 are fixed to the support 81 spaced apart from each other so that the lens_holder 70 can rotate freely about a rotary_shaft 82 .

Another iron piece and another tracking_coil are also mounted although they are not shown in Fig. 6 because they are at the back of the tracking_magnet 77 .

While no current_flows through the tracking_coils 73 , 74 and 76 and the like , the iron piece 72 and tracking_magnet 81 for example are structured to be attracted each other .

Namely , even if no control_voltage is applied , one of the lenses 78 and 79 is set to a normal stage .

With this structure , in order to switch between the lenses , a kick_pulse is applied to the tracking_coil 74 in a short_time to generate a repulsion force larger than the attraction_force between the magnet 81 and iron piece 72 .

The lens_holder 70 therefore receives a rotation torque and rotates the lens_holder 70 .

After some rotation , the iron piece 71 is attracted to the magnet 81 and fixed to the normal stage .

In this manner , the lenses 78 and 79 fixed to the lens_holder 70 are selectively switched .

In order to recover the position of the original_lens , a kick_pulse of an opposite_polarity is applied to perform a similar control .

At Steps S17 and S18 , the switching_control only is executed .

In this case , means for visually displaying which optical_disk was selected may be provided .

In the current detector_circuit 31 of this embodiment , the resistor 24 is connected serially to be actuator 23 to measure the current_flowing through the actuator 23 by detecting the voltage across the resistor 24 having a presumably constant resistance

.

This voltage_value is supplied via the A / D converter 32 to the controller 16 which converts the voltage_value into the current_value .

Since the voltage_value is proportional to the current_value , the voltage_value itself may be used for comparison .

The current detector_circuit 31 shown in Fig. 1 is only illustrative and may use known means for directly detecting a current .

Patent Text Overview

The pickup shown in Fig. 6 has a lens_holder 70 mounted on which are a CD objective_lens 78 , a DVD objective_lens 79 , iron pieces 71 , 72 and 75 , and tracking_coils 73 , 74 and 76 .

A focussing magnet 80 is also mounted between a support 81 and the lens_holder 70 , and tracking_magnets 77 and 83 are fixed to the support 81 spaced apart from each other so that the lens_holder 70 can rotate freely about a rotary_shaft 82 .

[illegible]

At Steps S17 and S18 , the switching_control only is executed .

In this case , means for visually displaying which optical_disk was selected may be provided .

In the current detector_circuit 31 of this embodiment , the resistor 24 is connected serially to be actuator 23 to measure the current_flowing through the actuator 23 by detecting the voltage across the resistor 24 having a presumably constant resistance

.

This voltage_value is supplied via the A / D converter 32 to the controller 16 which converts the voltage_value into the current_value .

Since the voltage_value is proportional to the current_value , the voltage_value itself may be used for comparison .

The current detector_circuit 31 shown in Fig. 1 is only illustrative and may use known means for directly detecting a current .

As above , since it becomes possible to discriminate between optical_disks during the focus_search , the lens can be replaced immediately so that the time required for reading actual_data can be shortened .

According to the optical_disk_apparatus of this invention described above , a current_difference obtained during the in-focus at the surface and reflection_film of an optical_disk is used for the discrimination between optical_disks .

Therefore , discrimination reliability is very high and all the previously described issues can be solved .

Claims

Patent Text Overview

A pickup of such a twin-lens type is shown in Fig. 6 .

The pickup shown in Fig. 6 has a lens_holder 70 mounted on which are a CD objective_lens 78 , a DVD measure the current_flowing through the actuator 23 by detecting the voltage across the resistor 24 having a presumably constant resistance

.

This voltage_value is supplied via the A / D converter 32 to the controller 16 which converts the voltage_value into the current_value .

Since the voltage_value is proportional to the current_value , the voltage_value itself may be used for comparison .

The current detector_circuit 31 shown in Fig. 1 is only illustrative and may use known means for directly detecting a current .

As above , since it becomes possible to discriminate between optical_disks during the focus_search , the lens can be replaced immediately so that the time required for reading actual_data can be shortened .

According to the optical_disk_apparatus of this invention described above , a current_difference obtained during the in-focus at the surface and reflection_film of an optical_disk is used for the discrimination between optical_disks .

Therefore , discrimination reliability is very high and all the previously described issues can be solved .

Claims

1 .

An optical_disk_apparatus for data_recording , reproducing and erasing , the apparatus having an optical focussing_system in which an optional_optical_disk is selected from a plurality of optical_disks of different kinds having different thicknesses and loaded in the optical_disk_apparatus , a laser_beam is focussed via a pickup_lens to the loaded_optical_disk , and light reflected from the loaded_optical_disk is detected with a photodetector , and having a control_system with an actuator for converging the laser_beam to the optical_disk , the optical_disk_apparatus comprising :

means for detecting a focus_signal from the photodetector while the pickup_lens is moved by the control_system ;

temporary storage_means for storing , as first and second measured_values , drive_currents or voltages of the actuator representative of the in-focus positions at the surface of the optical_disk and at a reflection_film of the optical_disk , in accordance with the focus_signal ;

subtraction_means for calculating a difference of the second measured_value from the first measured_value ;

Figure 6.6: Animated shrinking (left: no, center: partial, right: full reduction)

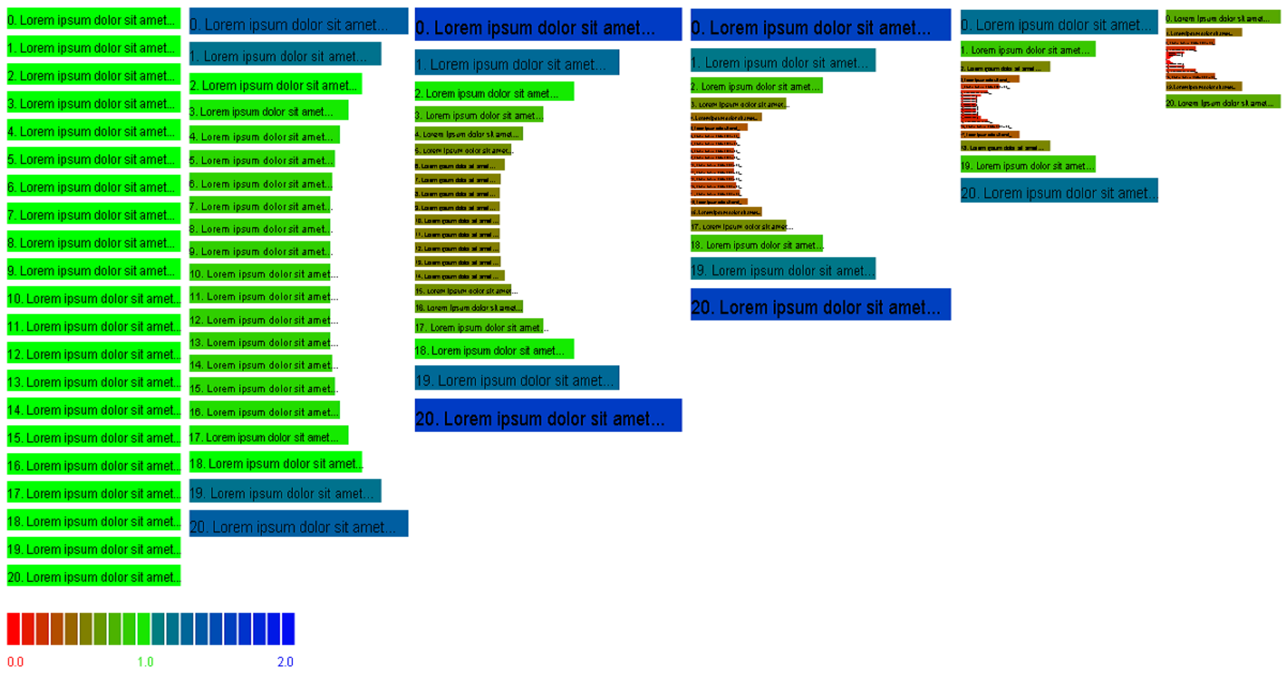


Figure 6.7: Distortion of a sample text with a=1.0, a=0.9, a=0.5, a=0.3 and a=0.15 (left to right).

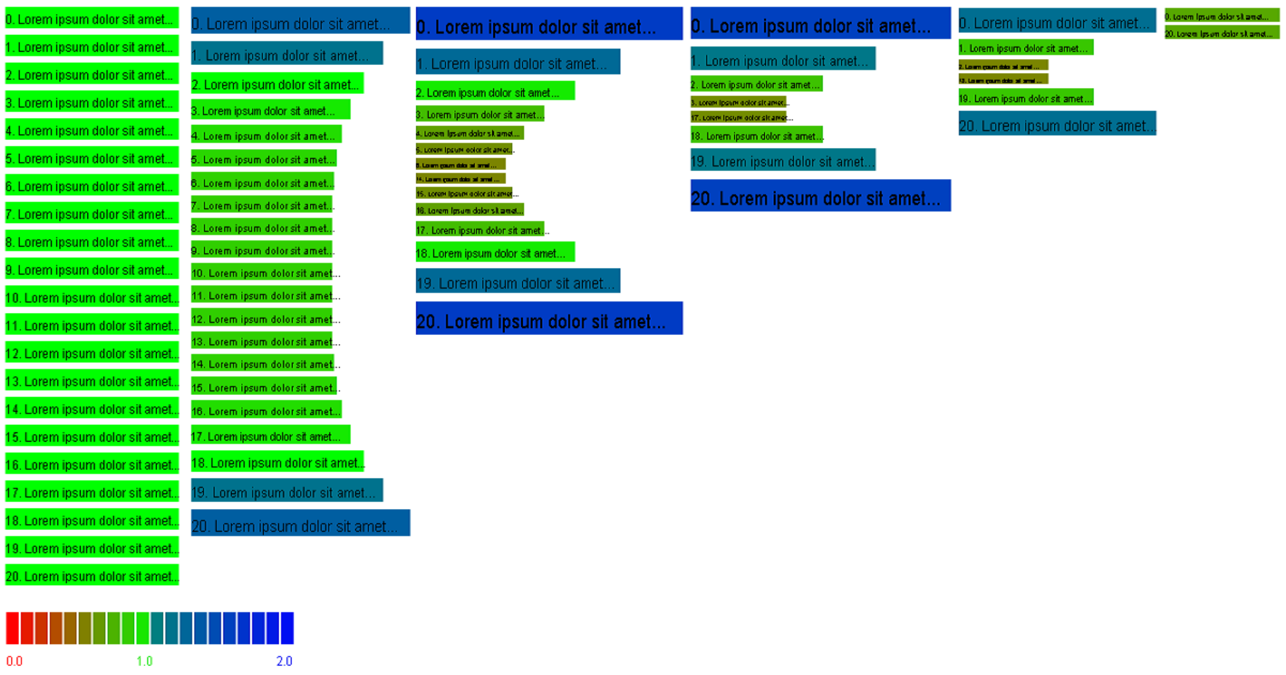


Figure 6.8: Distortion of a sample text as in figure 6.7 but with threshold set to 0.5.

6.6 Application: Search Result Highlighting

The presentation of semantic search results is more difficult than the presentation of traditional textual search results. For the latter the matching text parts usually are highlighted together within a certain textual context. In contrast, semantic search results[2] are a collection of graph nodes directly or indirectly connected with annotation nodes. Each annotation node in the result defines a *result context*.

In order to explain the results to the user the result contexts have to be visualized appropriately. The approach that has been used is the following:

1. All annotation nodes contained in a result context are highlighted. All annotation nodes not contained in a result context are ignored.
2. All relations between highlighted annotation nodes are visualized as routed arcs.
3. All additional nodes that are directly associated with an annotation node are shown as tool tip when the user hovers an annotation node. Additional nodes that are indirectly associated with an annotation node are ignored.

Example

A user wants to highlight the text fragments that describe actuators having optical lenses as part. The user could formulate the following query (namespaces are omitted):

```
SELECT ?actuator ?part
WHERE {
        ?actuator rdf:type auto:actuator ;
                  sumo:hasPart ?part .
        ?part rdf:type ordo:lens .
}
```

Let us assume that first result context contains the following RDF triple, in which two annotation nodes are connected by a part-of relation.

```
pat:EP0805439A1#token('claim-1',13,8) sumo:hasPart pat:EP0805439A1#token('claim-1',16,4) .
```

Based on these annotation nodes the directly associated additional nodes are selected based on the following implicit and dynamically generated query.

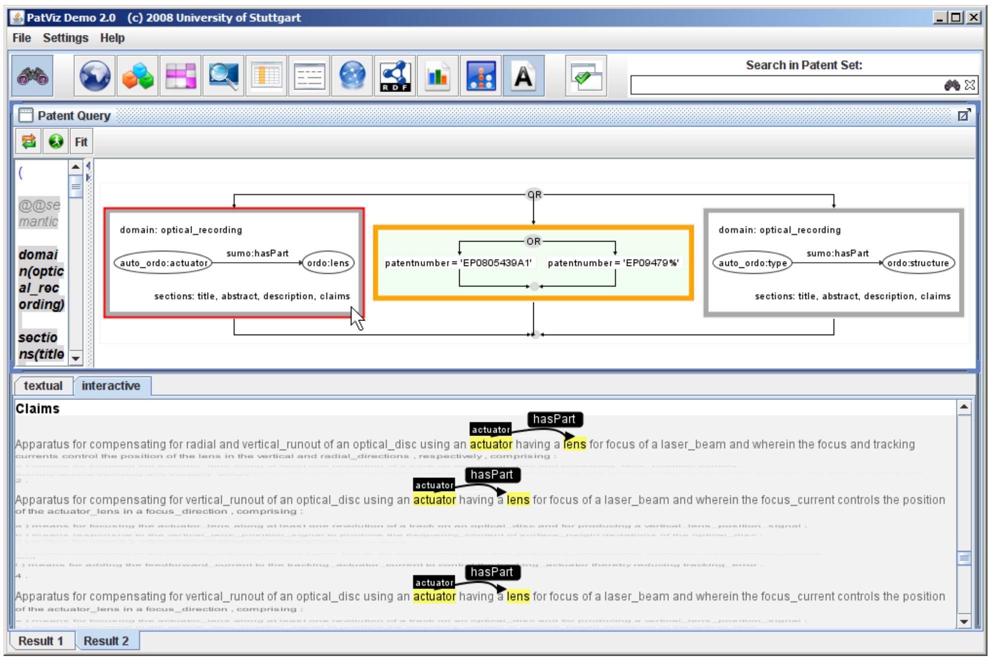
[2]We define a semantic search to be a SPARQL query on the Content Annotation Graph.

```
SELECT ?annotation ?anyProp ?additional
WHERE  {
       ?annotation ?anyProp ?additional .
       FILTER (
            ?annotation = <http://pat.patexpert.org/EP0805439A1#token('claim-1',13,8)> ||
            ?annotation = <http://pat.patexpert.org/EP0805439A1#token('claim-1',16,4)>
       )
}
```

In the example the type information is mapped to additional nodes.

```
pat:EP0805439A1#token('claim-1',13,8) rdf:type auto:actuator .
pat:EP0805439A1#token('claim-1',16,4) rdf:type ordo:lens .
```

Figure 6.9 shows an example of a semantic search visualization. The user has visually formulated a semantic search using the PatViz query visualization front end [129, 170]. The result contexts of the semantic search are highlighted in the window below. Annotation nodes are highlighted with a yellow background. Relations between annotation nodes are visualized as curves. Additional nodes as visualized as tool tips. To increase the number of result contexts shown in the current view port the text fragments in between have been collapsed as described in section 6.5.


PatViz Demo 2.0 (c) 2008 University of Stuttgart
File Settings Help
Search in Patent Set:
Patent Query
Fit
domain: optical_recording
sumo:hasPart
auto_ordo:actuator
ordo:lens
sections: title, abstract, description, claims
OR
patentnumber = 'EP0805439A1'
patentnumber = 'EP09479%'
auto_ordo:type
ordo:structure
textual
interactive
actuator
hasPart
Result 1
Result 2

Claims

Apparatus for compensating for radial and vertical_runout of an optical_disc using an actuator having a lens for focus of a laser_beam and wherein the focus and tracking currents control the position of the lens in the vertical and radial_directions , respectively , comprising :

Apparatus for compensating for vertical_runout of an optical_disc using an actuator having a lens for focus of a laser_beam and wherein the focus_current controls the position of the actuator_lens in a focus_direction , comprising :

Apparatus for compensating for vertical_runout of an optical_disc using an actuator having a lens for focus of a laser_beam and wherein the focus_current controls the position of the actuator_lens in a focus_direction , comprising :

Figure 6.9: Highlighting of semantic search results

6.7 Conclusions

In this chapter different approaches for associating content-level annotations with textual fragments contained in XML documents have been described. For patent documents encoded in ST.36 the linking approach with custom *token* schema XPointers is considered to be the most appropriate one.

Based on this design decision a data model and a new technique for visualizing content level annotations has been proposed. The central abstraction of this model is the *Annotation* class which integrates RDF graphs with XML documents. The layout information of *Annotation* instances is derived from the layout of their associated text fragments and propagated to the visualization system. The proposed visualization technique combines text layout and graph layout in two layers that are aligned to each other. Each *Annotation* instance is associated with a *VisualNode* referred to as annotation node. The highlighting of annotated text fragments is determined by the way the annotation nodes are rendered. For the visualization of relations between annotation nodes a new edge routing algorithm has been described. The central idea of the algorithm is to route edges so that they do not overlap with the text.

Finally an annotation driven Focus+Context technique has been proposed for reducing the vertical space between two annotations by applying a distortion function. In contrast to general fisheye views that magnify areas of greater interest, the semantic shrink technique demagnifies areas of lower interest. The distortion function will cause text lines between annotation nodes to be compressed depending on the distance to the annotations. One application of this technique is to increase the number of hits of a semantic search shown in the view port in order to provide a better overview to the user.

A general application is the presentation of semantic search results. The presentation of semantic search results usually is more difficult than the presentation of traditional textual search results. For the latter the matching text parts usually are highlighted together within a certain textual context. In contrast, semantic search results are a collection of graph nodes directly or indirectly connected with annotation nodes. A first implementation is provided within the PatViz prototype (section 9.3.1).

CHAPTER 7

Semantic Lenses

In semi-automatic analytical processes, such as patent analyses, users need to connect information from different sources in order to perceive relationships between entities of interest. However, when using different retrieval tools and methods, users often have to switch between various user interfaces.

With *implicit queries* it is possible to automatically take advantage of the user's context for getting explanatory or additional information. We refer to this kind of information as *context information*. Users can be visually informed about the existence of such information without having to switch the context. On demand the available context information can be seamlessly integrated and visualized. Thus, the concept of implicit queries is a useful approach when working with semi-automatic analytical processes.

Although there might be an overhead in executing queries whose results are not used later, we expect a great benefit from the implicit query concept because of the following reasons: It can be checked very quickly *if* there is context information for certain entities available at all. Users are able to detect the existence of multiple meanings for one and the same entity, which is important especially when working in interdisciplinary fields. Further, the users might be motivated to inspect context information, when their availability is visualized.

In our approach we assume that the user context is given by a web page within a certain set of knowledge domains. For each domain we assume the existence of knowledge bases or services that provide additional information about the concepts and instances of a domain. The key phrases of a web page act as parameters of implicit queries. They are matched with the information contained in the underlying knowledge bases or retrieved from underlying services. The availability of additional information is indicated by highlighting the matched phrases in the web page.

When users interactively select a highlighted phrase, the context information is presented visually by embedding it into a special panel that we call a *semantic lens*. This representation allows displaying the information integrated into the current web page without changing the layout. The context information is presented as compound document containing graphical and textual representations that provide background or detail information about the entity.

The next section describes related work. Section 7.2 presents applications and scenarios in which the proposed methods are used. Then the overall system architecture is described in more detail in section 7.3. Interaction techniques and the results of a first user evaluation are presented in the sections 7.4 and 7.5. In section 7.6 the semantic lens approach and possible limitations are discussed.

7.1 Related Work

The representation of complex data structures is one of the key problems in information visualization research. Special graph layouts help to improve the understanding of the data [293, 177, 165]. Visual graph analysis techniques support the users in identifying the overall structure of a graph and some of its key features [220, 292].

The term *semantic lens* is based on the concept of *magic lenses* where a panel can be moved like a magnifying glass over the objects on a screen in order to provide a special view on the objects. In contrast to magic lenses, the visualizations presented in semantic lenses are based on the information retrieved from knowledge stores. The goal of semantic lenses is to provide quick access paths to related context information that is seamlessly integrated into web pages and thus to support the users in better understanding the content.

The concept of using lenses in user interfaces has been published already for different fields of application. Magic lenses were published by Bier et al. and Stone et al. [73, 232]. Their work is based on earlier research that was done in more specialized techniques like fisheye views by Furnas [114] and by Perlin and Fox [193]. Lenses have been integrated into web browsers to interactively change the rendering of web pages by Phelps and Wilensky [194]. Furthermore, Janecek et al. describe a technique called Semantic Fisheye Views that can integrate rich semantic models into the search process, such as WordNet [158, 159].

Link augmentation is a hypermedia technique in adaptive hypertext systems. Bailey et al. combine link augmentation with a model of the user's spatial context to achieve cross-domain adaptive navigational support [58]. A method called fluid links that provide additional information at a link source to support readers in choosing among links and understanding the structure of a hypertext [295].

In particular, when describing knowledge domain concepts and instances using RDF as data model (see section 2.2), the visual structure of the resulting RDF graphs may not be intuitive to understand. Great challenges are the visual presentation of and the navigation in RDF graphs. Different tools have been developed to support the visualization of RDF graphs, such as *IsaViz* [13, 195], *Welkin* [35], or *GViz* [112, ch. 9.3], to name only the most prominent.

The semantic lens approach differs from previous RDF visualization tools in the following aspects:

- Only relevant parts of a knowledge base are visualized using the key terms of a web page as search criteria.
- Semantic lenses provide both, graph-based visualizations to outline the overall structure of the data and textual presentations for additional or detail information.
- This approach allows a seamless integration into web pages by using Web Integration Compound Documents (WICD) [272] that embed *Scalable Vector Graphics (SVG)* [268] and other markups in XHTML documents.
- Semantic lenses can also be used to integrate non-RDF content and allows e.g. for previews of web pages, images, or PDF documents.

7.2 Applications

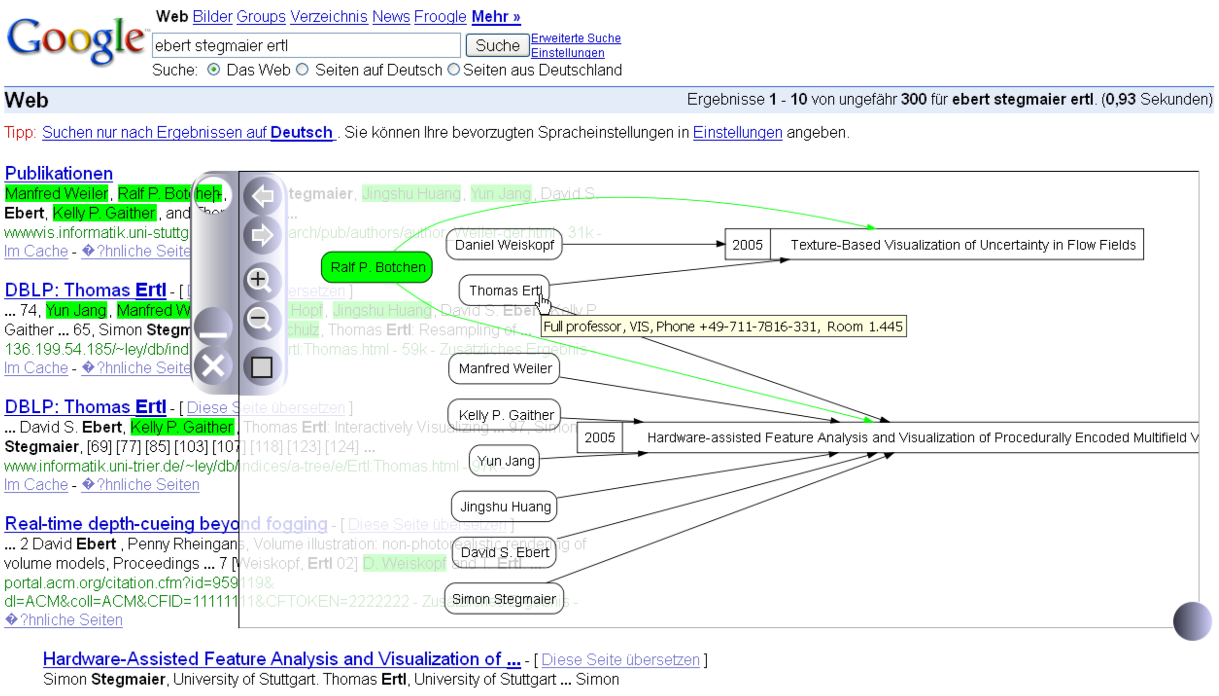
In this section three prototypical web based applications for different knowledge stores and services are described that take advantage of the augmentation with implicit queries and of the semantic lenses.

The first application uses a bibliographic database that has been crawled from the web. It highlights the author names in a web page, and presents the related publications and co-authors as a node-link graph, when clicking on the author's name. The second application uses the Open Patent Service. It highlights the patent numbers and presents legal event information about patent families. The third application highlights the names of Wikipedia article titles and presents the full text article in a semantic lens.

7.2.1 Bibliographic Lenses

The bibliographic augmentation works on gathered BibTeX data that are crawled from web sites. The data is mapped to an RDF representation and added to a knowledge store. Furthermore, existing databases that consist of relations between authors, publication name, publication location, publication year, etc. may be used within this scenario.

Using the bibliographic data the names of authors can be highlighted in web pages by presenting the availability of data set entries. When clicking on a name, the related publications and the co-authors of this person are presented in the semantic lens as a node-link diagram.



Web Bilder Groups Verzeichnis News Froogle **Mehr »**

ebert stegmaier ertl | Suche | Erweiterte Suche | Einstellungen

Suche: [✓ Das Web] Seiten auf Deutsch Seiten aus Deutschland

Web Ergebnisse **1** - **10** von ungefähr **300** für **ebert stegmaier ertl**. (**0,93** Sekunden)

Tipp: Suchen nur nach Ergebnissen auf **Deutsch** . Sie können Ihre bevorzugten Spracheinstellungen in Einstellungen angeben.

Publikationen
Manfred Weiler, Ralf P. Botchen, … tegmaier, Jingshu Huang, Yun Jang, David S.
Ebert, Kelly P. Gaither, and …
wwwvis.informatik.uni-stuttg … arch/pub/authors/author_Weiler-ger.html - 31k -
Im Cache - �?hnliche Seite

DBLP: Thomas **Ertl** - [… ersetzen]
... 74, Yun Jang, Manfred W … Hopf, Jingshu Huang, David S. **Ebert**, Kelly P.
Gaither ... 65, Simon **Stegm** … chulz, Thomas **Ertl**: Resampling of ...
136.199.54.185/~ley/db/ind … rtl:Thomas.html - 59k - Zusätzliches Ergebnis
Im Cache - �?hnliche Seite

DBLP: Thomas **Ertl** - [Diese Seite übersetzen]
... David S. **Ebert**, Kelly P. Gaither, Thomas **Ertl**: Interactively Visualizing ... 97, Simon
Stegmaier, [69] [77] [85] [103] [107] [118] [123] [124] ...
www.informatik.uni-trier.de/~ley/db/indices/a-tree/e/Ertl:Thomas.html - 97k -
Im Cache - �?hnliche Seiten

Real-time depth-cueing beyond fogging - [Diese Seite übersetzen]
... 2 David **Ebert** , Penny Rheingans, Volume illustration: non-photorealistic rendering of
volume models, Proceedings ... 7 [Weiskopf, **Ertl** 02] D. Weiskopf and T. **Ertl** ...
portal.acm.org/citation.cfm?id=959119&
dl=ACM&coll=ACM&CFID=11111111&CFTOKEN=2222222 - Zusätzliches Ergebnis -
�?hnliche Seiten

Hardware-Assisted Feature Analysis and Visualization of ... - [Diese Seite übersetzen]
Simon **Stegmaier**, University of Stuttgart. Thomas **Ertl**, University of Stuttgart ... Simon

Ralf P. Botchen
Daniel Weiskopf
Thomas Ertl
Full professor, VIS, Phone +49-711-7816-331, Room 1.445
Manfred Weiler
Kelly P. Gaither
Yun Jang
Jingshu Huang
David S. Ebert
Simon Stegmaier
2005
Texture-Based Visualization of Uncertainty in Flow Fields
2005
Hardware-assisted Feature Analysis and Visualization of Procedurally Encoded Multifield V

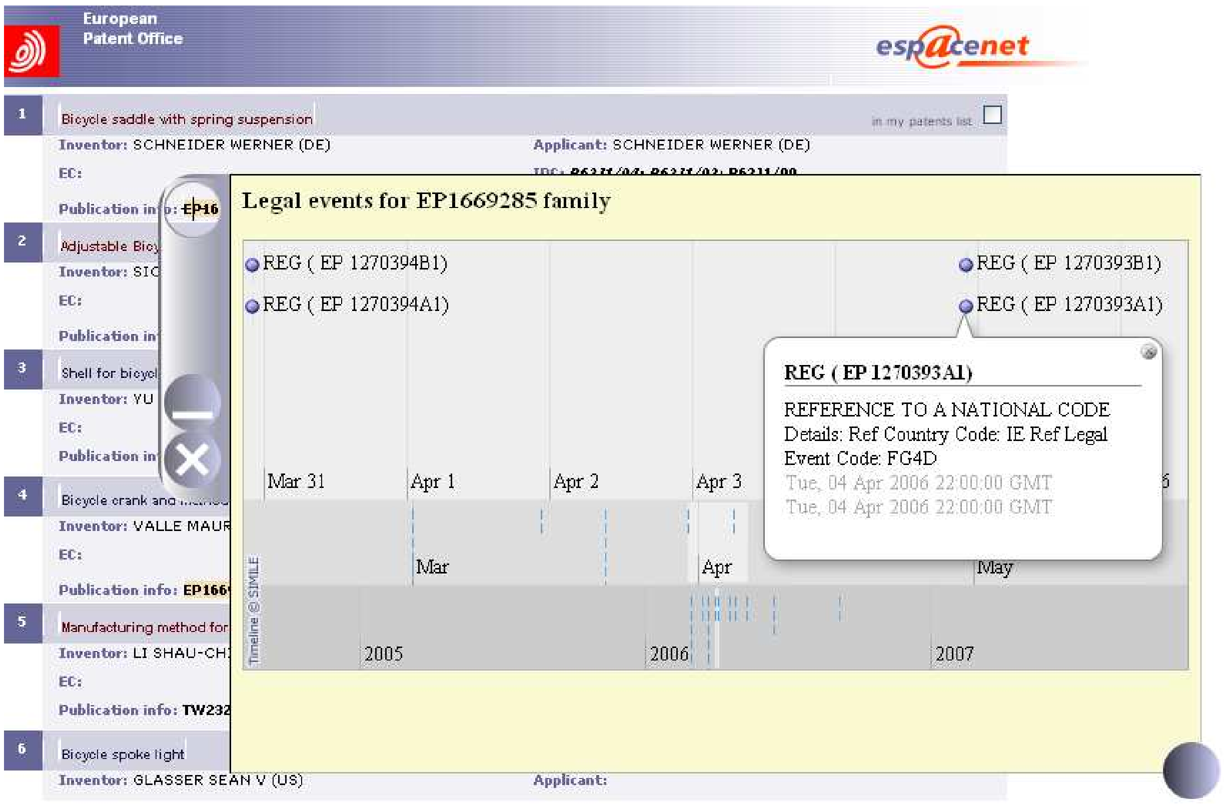
Figure 7.1: Semantic lens showing the publication relations of the selected person

As an example figure 7.1 shows the context information embedded in a semantic lens for an author. In the example, the lens is integrated into a search result web page from Google. To determine the authors all phrases on the web page are matched against the instances that occur in the knowledge base as authors of a published paper.

7.2.2 Patent Lenses

Patent lenses provide a structured view on patent metadata such as patent families, citations, applicant/inventor networks, or legal event information. When looking at web pages containing patent numbers (e.g. when doing a patent inquiry), this application highlights potential patent publication numbers by applying regular expressions.

When selecting such a number, the Open Patent Services [21] are called and the legal events as well as the family information is retrieved. The result is converted into an RDF representation based on the Patent Metadata Ontology (see chapter 3) which serves as input for the context information visualization component.



1 Bicycle saddle with spring suspension

Inventor: SCHNEIDER WERNER (DE) Applicant: SCHNEIDER WERNER (DE)

EC:

Publication info: EP16

2 Adjustable Bicy

Inventor: SIC

EC:

Publication in

3 Shell for bicycl

Inventor: YU

EC:

Publication in

4 Bicycle crank and

Inventor: VALLE MAUR

EC:

Publication info: EP166

5 Manufacturing method for

Inventor: LI SHAU-CH

EC:

Publication info: TW232

6 Bicycle spoke light

Inventor: GLASSER SEAN V (US) Applicant:

Legal events for EP1669285 family

REG (EP 1270394B1)

REG (EP 1270394A1)

REG (EP 1270393B1)

REG (EP 1270393A1)

REG (EP 1270393A1)

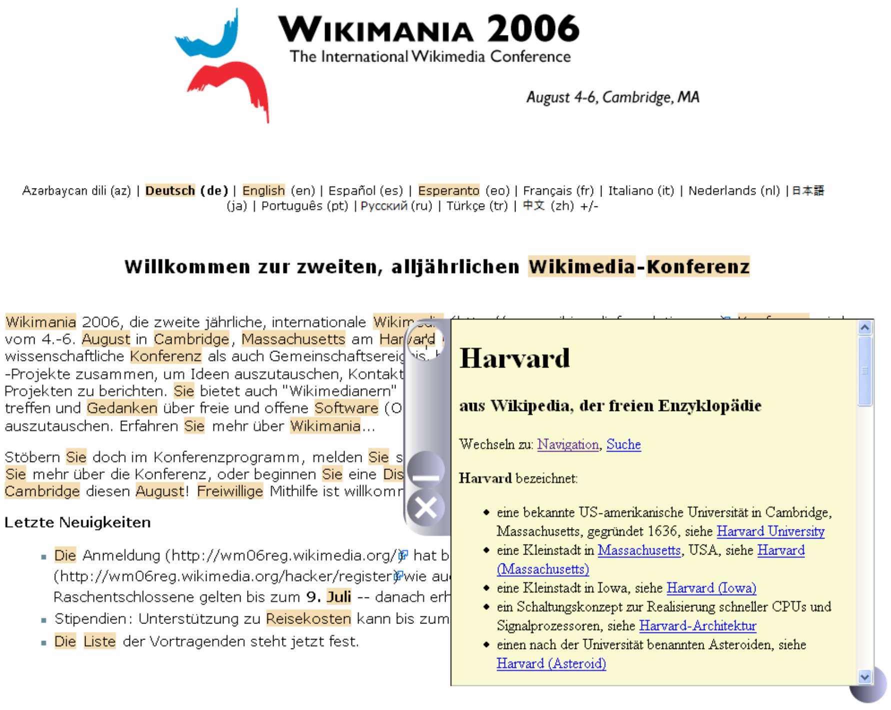
REFERENCE TO A NATIONAL CODE
Details: Ref Country Code: IE Ref Legal Event Code: FG4D
Tue, 04 Apr 2006 22:00:00 GMT
Tue, 04 Apr 2006 22:00:00 GMT

Figure 7.2: Semantic lens showing the legal events of a patent family in a timeline

Using the timeline widget [29] provided by MIT's SIMILE (Semantic Interoperability of Metadata and Information in unLike Environments) project, the visualization component renders the data as a timeline showing the temporal sequence of the legal events. When the user clicks on an event, its details are presented in an info box (figure 7.2). This kind of presentation allows a quick overview of the legal status of a patent and its associated family members directly within the web page without forcing the user to perform a context switch.

7.2.3 Wikipedia Lenses

Wikipedia has become one of the most important information sources when searching for definitions and background information. But using the Wikipedia search form forces the users to leave their current context. In order to improve the integration of Wikipedia into the user's context, Wikipedia lenses allow the presentation of related Wikipedia articles directly within the current web page (cf. Rotard et al. [210]).


WIKIMANIA 2006
The International Wikimedia Conference
August 4-6, Cambridge, MA

Azərbaycan dili (az) | **Deutsch (de)** | English (en) | Español (es) | Esperanto (eo) | Français (fr) | Italiano (it) | Nederlands (nl) | 日本語 (ja) | Português (pt) | Русский (ru) | Türkçe (tr) | 中文 (zh) +/-

Willkommen zur zweiten, alljährlichen Wikimedia-Konferenz

Wikimania 2006, die zweite jährliche, internationale Wikim
vom 4.-6. August in Cambridge, Massachusetts am Harvard
wissenschaftliche Konferenz als auch Gemeinschaftsereig
-Projekte zusammen, um Ideen auszutauschen, Kontakt
Projekten zu berichten. Sie bietet auch "Wikimedianern"
treffen und Gedanken über freie und offene Software (O
auszutauschen. Erfahren Sie mehr über Wikimania...

Stöbern Sie doch im Konferenzprogramm, melden Sie s
Sie mehr über die Konferenz, oder beginnen Sie eine Dis
Cambridge diesen August! Freiwillige Mithilfe ist willkomm

Letzte Neuigkeiten

- Die Anmeldung (http://wm06reg.wikimedia.org/) hat b
(http://wm06reg.wikimedia.org/hacker/register) wie au
Raschentschlossene gelten bis zum **9. Juli** -- danach erh
- Stipendien: Unterstützung zu Reisekosten kann bis zum
- Die Liste der Vortragenden steht jetzt fest.

Harvard

aus Wikipedia, der freien Enzyklopädie

Wechseln zu: Navigation, Suche

Harvard bezeichnet:

- eine bekannte US-amerikanische Universität in Cambridge, Massachusetts, gegründet 1636, siehe Harvard University
- eine Kleinstadt in Massachusetts, USA, siehe Harvard (Massachusetts)
- eine Kleinstadt in Iowa, siehe Harvard (Iowa)
- ein Schaltungskonzept zur Realisierung schneller CPUs und Signalprozessoren, siehe Harvard-Architektur
- einen nach der Universität benannten Asteroiden, siehe Harvard (Asteroid)

Figure 7.3: Semantic lens showing the full text article of a highlighted term

The original web page is augmented with links to Wikipedia articles by matching the phrases of the web page against the Wikipedia titles in the knowledge store. The results of the inverse query are highlighted in the web page and linked to the article on the Wikipedia server (see figure 7.3). Optional filters may be defined to select the article categories to include, e.g. only articles about biology and medicine. For each article the categories have to be stored in the RDF knowledge store to enable this feature. This information is extracted from the Wikipedia XML database dumps. Furthermore, the main categories for each article have to be determined.

The users can click on a highlighted phrase to present the latest Wikipedia article in a semantic lens. To optimize the available screen space in the lens only the body of the article is presented not including the navigation bars on the top and on the left.

7.3 System Architecture

In this section we describe the general system design in more detail. The system is implemented as a web application that can be used with standard browsers[1]. It has three major compo-

[1]Currently the system is optimized for the Firefox browser, since Firefox supports compound document by inclusion for XHTML and SVG.

nents: The first component is the *augmentor service* which preprocesses a web page, checks the availability of context information by implicit queries, and augments the original web page with highlighting and linking information. The second is the *context information visualization service* which generates a visual presentation of the available context information. The third component, the *semantic lens engine*, handles the user interactions and the visual presentations on the client side. An overview of the components and their invocation sequence is shown in figure 7.4.

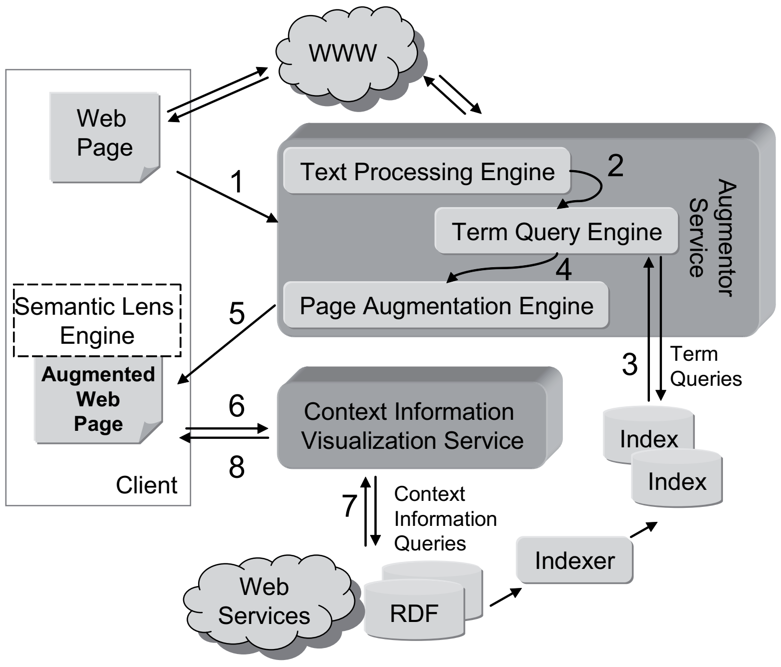


Figure 7.4: Major components of a semantic lens system

The services are called via HTTP. The augmentor service can be invoked with the following simple JavaScript statement that can be added to the browser's bookmark list:

```
javascript:location.href='http://SERVERNAME/Augmentor
?url='+encodeURIComponent(location.href)
```

When users visit a web page and request context information about the phrases on the page, the augmentor service is invoked with the URL of the page as argument (figure 7.4, step 1).

The augmentor service consists of three sub-components: The text processing engine downloads a web page, extracts the textual content from the page, determines the language by using characteristic stop words and trigrams, and analyzes and lemmatizes the content using the statistical TreeTagger [215]. Heuristic filtering, e.g. nouns, can be applied to reduce the amount of phrases[2] for lookup. After this step, the remaining phrases are used to generate queries on the index.

7.3.1 Implicit Queries

Key phrases of a web page are used to automatically generate queries on underlying indexed knowledge bases. Multiple queries (several hundreds for each phrase) have to be matched with millions of small sized terms, e.g. Wikipedia article titles. This differs from classical information retrieval where only one query is matched with a document set containing millions of medium to large sized documents.

The implicit query scenario is outlined in figure 7.5 and can be described more formally as follows: The goal is to match a text having a word sequence $w_1, ..., w_m$ of length $m > 0$ with a set of index terms $\{t_1, ..., t_n\}$, where each term t_i has a word sequence $w_{i_1}, ..., w_{i_x}$ of length $i_x > 0$. Typical values are $m < 2.500$, $i_x < 5$, and $n > 1.000.000$. The result of the matching procedure is a list of matching phrases and their positions in the text (e.g., the table in figure 7.6).

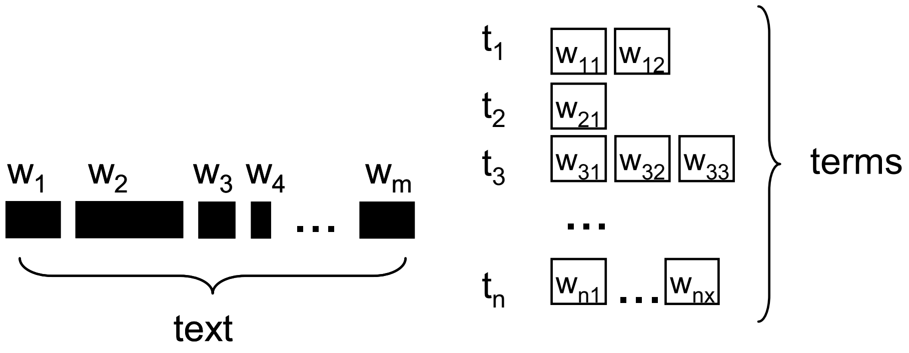
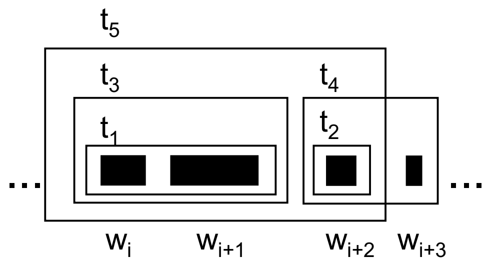


Figure 7.5: Implicit term query scenario

Overlapping matches are possible, so that one word of a phrase can be contained in different index terms. In figure 7.6, for example, the word w_{i+2} at position $i + 2$ is part of the terms t_2, t_4, and t_5. A phrase can have different meanings, and thus different context information associated to it. Different meanings are characterized by having multiple terms associated with one phrase, e.g. the phrase (w_i, w_{i+1}) has two meanings, one defined by term t_1 and one defined by term t_3.

[2]In this context, we define that a phrase is a sequence of words in a natural language contained in the content of a web page. We define a term to be a word sequence in an index.

Overlapping and multiple matches have to be taken into consideration, when phrases are visually highlighted and links to the associated context information are generated. We group overlapping matches into the longest continuous match and provide for each group a menu, in which the user can select an appropriate meaning.



term	start pos	end pos
t1	i	i+1
t2	i+2	i+2
t3	i	i+1
t4	i+2	i+3
t5	i	i+2

Figure 7.6: Example for overlapping matches

In the current implementation, we only use a simple algorithm for matching phrases with index terms. For both, phrases and index terms, lemmatized word versions are used (verbs in infinitive form, nouns in nominative singular form, etc.). First the queries are constructed using single words, then using two-word phrases, then using three-word phrases and so on. The described base line algorithm has a cost of $m * l * x$, where m is the amount of words, l is the maximum phrase length to lookup and x is the effort for executing one query. In typical web pages $m < 500$ and $l < 5$, so that x is the limiting factor.

7.3.2 Web Page Augmentation

The last step within the service is augmenting the original web page with additional HTML tags and JavaScript code, so that each matched phrase is highlighted and initialized to trigger the creation of one or more corresponding semantic lenses. The initialization includes the generation of linking information that either triggers a context information query in one or more knowledge stores and a subsequent visualization procedure, or links to a resource having further information about the phrase, for example a link to an encyclopedia.

An augmented web page is an enriched version of the original web page and also contains JavaScript code to handle the user interaction on the client side that we call the *semantic lens engine*. An augmented page is returned by the augmentor service as response to the client request (figure 7.4, step 5).

7.3.3 Visualization of Context Information

When the user clicks on a highlighted phrase, the associated context information is generated and presented to the user. The service is called with three parameters: (1) the phrase for which context information is created, (2) the knowledge stores, where the information has to be looked up, and (3) the presentation description specifying how the context information has to be collected and presented.

Since the result of context information queries is an RDF graph, we implemented the visualization service based on customizable SPARQL queries [261] which supports a more generic way of defining the presentation of context information and sharing this presentation knowledge between semantic lens applications.

The task of presenting RDF graphs in a human-readable way mainly consists in specifying *what* information contained in an RDF graph should be presented and *how* this information should be presented. Fresnel's two basic concepts are lenses and formats. Lenses define which properties of one or more RDF resources to display and the order of presentation. Formats determine how to render the resources, their properties and values.

Context information can be presented graphically, for example as node-link diagrams, or textually, for example using XHTML. In our approach, we combine the two paradigms in order to generate compound documents containing graphical elements in SVG inside general XHTML documents. The context information visualization service returns a compound document as response to the user's request (figure 7.4, step 8).

On the client side, the compound document is integrated in an overlay, which we call a *semantic lens*. The *semantic lens engine* and the available interaction techniques are described in more detail in section 7.4.

7.3.4 Customization

The framework can be customized for different applications by performing two steps: First the augmentation has to be adapted. This can be done in various ways. When working on RDF knowledge bases, one way would be to specify queries that identify the instances to be highlighted. The simplest interface to the augmentation service is a list of index terms. This has been done for the Wikipedia example.

The second step involves the specification of SPARQL queries to define what context information to present and further a visual mapping that specifies how to present it. For the rendering a set of serializers is provided, e.g. a renderer for generating node-link diagrams as shown in figure 7.1, or a renderer for generating timelines as shown in figure 7.2.

7.4 Interaction Techniques

The interaction techniques are one of the most important aspects for a new tool to be widely accepted. In this section we describe the interactive handling of the semantic lens. Figure 7.7 shows the functions of the handle on the semantic lens. The right part of the handle is visible when the content in the context information is presented as graph.

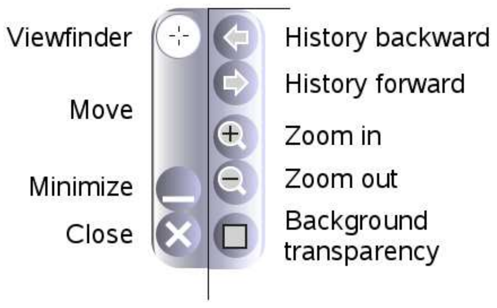


Figure 7.7: Handle of a semantic lens

By clicking on a highlighted key term the lens is launched with the context information near the corresponding key term. The lens can be minimized and closed by pressing the corresponding button on the handle. The user is not restricted to have just one semantic lens. Our system can handle multiple lenses that can be integrated into a web page and can overlap each other. When the user scrolls the web page the lens remains fixed in that position related to the context. On double clicking the lens handle the context information is presented in full screen. At the top of the handle there is a viewfinder. To show the semantic context information of a key term the viewfinder can be used by moving it over the key term. By using the viewfinder key terms can be selected that are inside a hyperlink. Therefore any open semantic lens can be used. Additionally, a new lens can be opened with a button in the toolbar of the browser.

A graph displayed as context information can be larger than the entire display area of the semantic lens. Therefore zooming functionality is necessary, which can be found in the two buttons on the handle of the lens. The viewport can also be moved by dragging the background of the graph. While the user moves the viewport of the graph the mouse cursor changes its symbol as feedback for the user. Since the graph is rendered as SVG the text in the context information graph is accessible and can be searched. The button on the handle with the square icon toggles the transparency of the background of the lens in three steps between full, semitransparent, and opaque.

Each semantic lens builds up a history of the context information when the content is a graph. This history can be navigated forward and backward with the corresponding buttons on the right side of the handle. Each node of the graph can contain a hyperlink to the corresponding context information graph of its key term. This enables the users to explore the context information by browsing the graph structure and by navigating in the history.

Data from different knowledge stores can be combined in one graph of the context information. A tool tip can be assigned to each node that presents additional information (e.g. contact information, full titles, hierarchy level, etc.) that comes from another knowledge store (figure 7.1).

7.5 Evaluation

The usefulness and the usability of the semantic lens was tested in a short evaluation with twelve employees and students at our university. Before we started each test we introduced the subjects into the 'thinking-aloud' method. The duration of each test was ten minutes. Before the test the subject were given a short tutorial on how to augment the web page by using the buttons in the toolbar of the browser and how to handle the lens with the viewfinder. After the free exploration phase the subjects were asked to fill out a questionnaire, in which they had to rate the usefulness as well as the ease of use in several questions. Additionally the subjects were asked to propose some applications that would benefit from the semantic lens concept.

An evaluation led to some first results. All subjects noticed the highlighted key terms and intuitively clicked on them. Then the first semantic lens appeared and showed the context information to the highlighted key term. All users started to explore the functionalities of the lens and most of them pointed out that the handling is easy to use. Some of them had stated problems in using the viewfinder. The navigation within the context information was found to be easy and intuitive.

The questions about the usefulness of the semantic lens where rated positive without exception. In particular, the implicit queries where pointed out to be very useful. Especially the key term selection by using the viewfinder was new but easy to handle for the users. The navigation inside a graph of the context information was not discovered by all subjects.

7.6 Discussion

In the following sections we discuss pros and cons of our approach, in particular the system architecture and limitations.

The construction of semantic lenses in the way described in section 7.3 has obvious advantages:

- Semantic lenses provide a flexible mechanism for integrating knowledge sources into the context of web pages.
- The lenses are displayed adjacent to the phrases in the web page but can be moved to any position on the page. This also allows them to be used as magic lenses for different kinds of content, like context information, web page previews, PDF documents, images, etc.
- With our approach it is possible to have multiple independent semantic lenses integrated into one page, which enables the comparison of context information from different data sources.
- The integration of semantic lenses does not disturb the layout of the web page and in case of a semitransparent presentation does not occlude the web page and saves precious screen space.
- Our approach does not depend on having special browser plugins that have to be developed separately for each browser.

But the approach also has the following shortcomings:

- Client functionality is programmed in terms of JavaScript and additional HTML tags. When working with different browsers, existing inconsistencies have to be handled by different program code.
- There are certain limitations, when augmenting web pages using HTTPS. Also, HTTP sessions can cause problems since the server is not aware of the session context. Frames and existing JavaScript code on the web page can also cause problems.
- Current SVG viewers still do not support hardware accelerated rendering. Complex graphs will be visualized slowly.

Due to the ambiguity of the natural languages, it is possible that a phrase has a different context on the web page than in the knowledge base. In the current stage the mapping between phrases on a web page and instances in a knowledge store is purely syntactic and only morphological features are considered.

Furthermore, there are still problems for matching names concerning semantic and syntactic ambiguities. Semantic ambiguities exist for example, when different persons have the same name (cf. [273]). Examples for syntactic ambiguities are abbreviations (e.g. Jeffrey D. Ullman, J. D. Ullman, J. Ullman, or Jeff Ullman), nicknames (e.g. Michael vs. Mike), permutations (e.g. Liu Bin vs. Bin Liu), different transcriptions (e.g. Andrei vs. Andrej vs. Andrey), accents (e.g. Muller vs. Müller vs. Mueller), ligatures (e.g. Weiß vs. Weiss), case (e.g. Al-AAli vs. Al-Aali), or hyphens (e.g. Hans-Peter vs. Hans Peter) (cf. [180]). In general, when working with data of one domain one can expect fewer semantic and syntactic ambiguities.

When looking at scalability issues, we found, that the limiting factor is the time needed to execute queries on the underlying RDF stores. This applies for the retrieval of index terms and also to the retrieval of context information. Unfortunately, it is difficult to make a general statement about the performance of the system, since it heavily depends on the application as well as on the structure of the knowledge bases or services.

7.7 Conclusions

For applications in the field of visual analytics we have proposed a new method for visually integrating semantic resources into standard web pages. Our method is based on implicit queries to configurable knowledge stores by using the phrases of the current web page. The results are highlighted in the web page. We have described a special panel that we call a semantic lens that can be integrated directly into the current web page. In this lens the context information related to the selected key phrase is visualized interactively. To select the highlighted phrases a viewfinder on the semantic lens can be used. Furthermore, web page previews of linked documents can be displayed in the lens by using the viewfinder.

We have discussed our proposal and pointed out that our user interface has advantages for the user in the handling and the presentation. Furthermore, there are disadvantages in some technical aspects especially the performance on large knowledge bases. We have presented the results of a first evaluation. Although the semantic lens was new to all test persons it turned out to be very useful. The user test results are very promising and show us that some work has to be done in the graph layout and the presentation of the compound documents.

In the evaluation we have asked the subjects to propose some applications where our concept could be used in and their answers were multifaceted. Since viewers for *Rich Site Syndication (RSS)* can be easily integrated into web browsers, our concept also allows to augment RSS feeds. Even e-mails that are presented with unified messaging systems can be combined with our system. Furthermore, messages of Usenet discussion groups that are embedded into web pages could be processed.

CHAPTER 8

Sharing of Patent Information

In this chapter a visualization enhanced semantic wiki system is described as user interface for patent information. The proposed system is characterized by the following aspects: First, to use a semantic model for the representation of patent information and user annotations. Second, to use semantic wiki technology for adding semantic annotations to the original patent data within a community process. Third, an integration of interactive visualizations for classificatory, temporal, geographic, and other patent information aspects.

The approach is motivated by using the functionality of modern wiki systems to directly implement major user requirements: the possibility to annotate content, to manage different versions (how do the query results change over time), and to send notifications when content has been updated. The proposed system combines user interactions (search and annotation) with automatic actions (adding new patents to the system, repeating searches on a weekly base, notifying users, etc.).

In traditional wikis content pages are created, edited and annotated by users. In our system content pages are automatically generated based on the patent search activities of users. Additionally to its textual representation, semantic concepts and metadata are either linked to the pages or embedded as described in section 6.2. Generated pages cannot be edited directly, because they contain immutable information published by a patent office. However, commenting and semantic annotation of pages is supported, e.g. discussions of patent content, patent reviews, patent rating, linking to prior art, etc.

8.1 Related Work

Commercial databases, such as Derwant World Patents Index, STN, or PatBase, provide 'added values' by integrating data from different patent offices and by providing abstracts and translations of new patents. Users have to pay if they want to access the data or use the tools. Besides the commercial data providers there are also non-commercial activities such as PatentLens [23], Freepatentsonline [9], or WikiPatents [36]. Patent Lens and freepatentsonline are free patent portals that provide a web-based user interface for searching and reading patents. Both approaches still lack the tight integration of users within a community process for commenting and reviewing patents. WikiPatents on the other hand is a wiki system that contributes to the US patent system by reviewing issued patents and pending patent applications. WikiPatents is related to our approach, but does not use a semantic model - neither for representing patent information nor for representing user annotations. WikiPatents also does not provide advanced visualizations for analyzing large patent sets.

Semantic wikis, such as Semantic MediaWiki [247, 240], are going to become an important component in the Semantic Web for adding machine-interpretable content. Semantic MediaWiki provides an extension to the wiki syntax and an enhanced article view to present the interpreted semantic data. It enhances the Wikipedia category concept and introduces typed links, attributes and semantic templates. Semantically annotated wiki pages can be exported in RDF.

Further, there are various visualization techniques that have been applied to patent information. An overview has been given in chapter 5. Since well designed visualizations provide for aggregated overviews of large patent collections, we believe that it is important to also add visualizations to a patent wiki system.

8.2 Architecture

This section describes the proposed approach for combining patent search, patent visualization and patent annotation in a semantic wiki system. Figure 8.1 gives an overview of the components and the data flow between them. There are three ways to add content to the wiki. First, a user executes a query (step 1) and accepts the results (step 6). Second, the user adds semantic annotations (step 7). Third, the update manager adds newly published patent data to the wiki system in an asynchronous process. In the following we describe a typical search scenario.

The first step is the formulation of a query. A patent index is looked up in order to retrieve the patent numbers of the resulting patent documents (step 2). In PATExpert this step involves four different search engines: a full-text search engine, a semantic search engine for searching in the knowledge base, a metadata search engine and an image search engine. The result of step 2 is a ranked list of patent document numbers.

In step 3 the patent contents are looked up in the cache. In PATExpert the cache is realized as a relational database, which uses the data from the Open Patent Services (OPS) [21] and the European Publication Server [7]. In step 3 also the knowledge base is queried for relevant semantic annotations. The result of step 3 is a patent model. A patent model can be serialized in RDF according to the patent ontologies described in section 3.

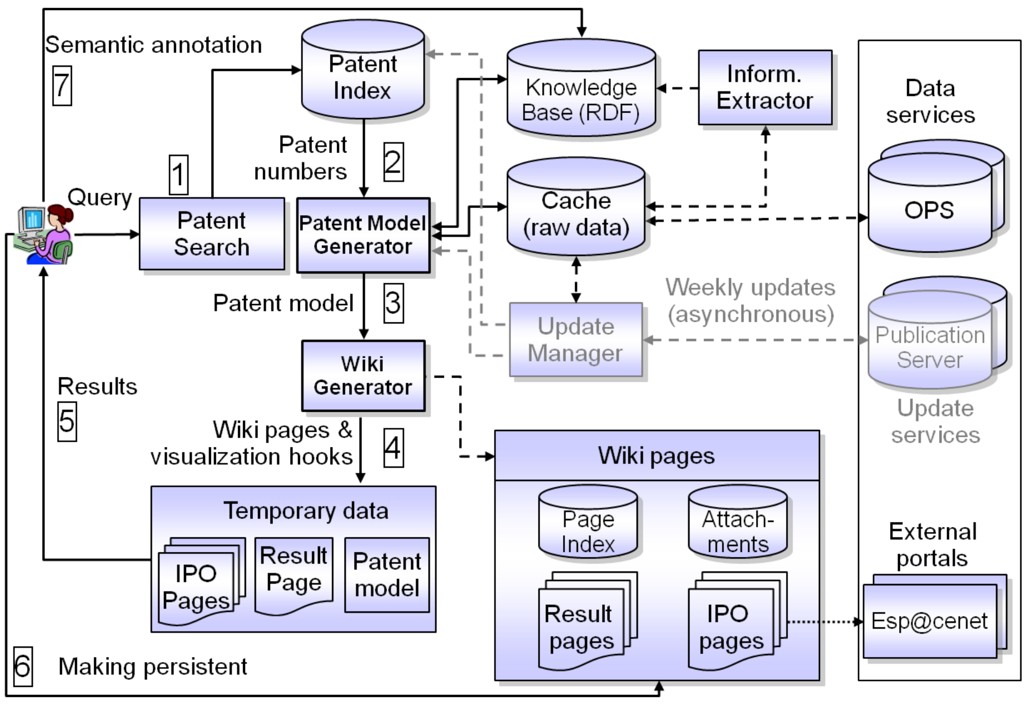


Figure 8.1: Architecture of the wiki system

In step 4 the wiki pages are generated based on the patent model. This generation is customized by scripts. A summarization page is created that contains the resulting patents, the patent model as attachment and visualization hooks. Visualization hooks are wiki plug-ins that take a patent model as input and create a visualization as output.

Visualizations are currently implemented in two ways: as Java applets on the client-side or as images dynamically generated on the server-side. The following code gives an example for a client-side applet visualization plug-ins for the JSPWiki [16] system.

```
[{patviz code='patviz.prefusext.applet.PrefuseXTApplet'
  archive='Applets/patviz.jar' script='Scripts/ipcInit.js'
  model=' patents.gz' width='100\%' height='600'}]
```

The parameters *code*, *archive width* and *height* are the applet parameters. The applet in this example is initialized with a script (*script* parameter) and a patent model (*model* parameter). All model references are made relative to the current wiki page. Models are attached to page.

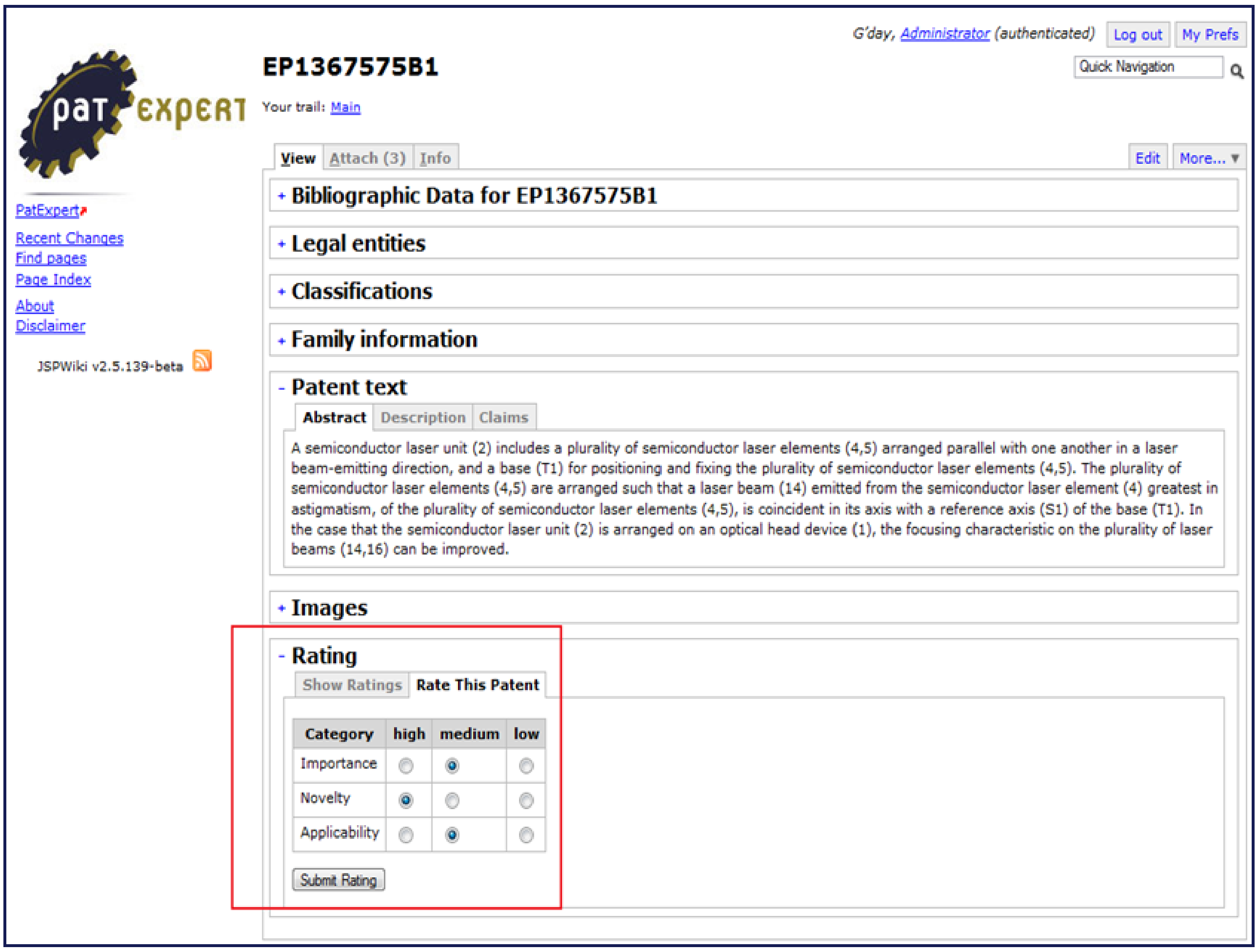
In the next step (5) the user can view the results and examine the details. The user can either discard the results and reformulate the query, or add the generated contents to the wiki system (step 6). Wiki pages can be semantically annotated, ranked and reviewed by other users. Adding new pages to the system may also trigger an update of existing pages (e.g. new patent pages could be added to inventor, applicant or category pages). Updates of existing pages trigger RSS notifications for users that have registered, for example to get notified when new patents of specific applicants are added.

User annotations are stored in a global knowledge base and registered users can add annotations. The interface for annotating patent pages currently is realized by customizable form plug-ins. Figure 8.2 shows a simple patent rating plug-in and its definition.

```
[{FormOpen form='rateForm'}] ...
| Importance | [{FormInput type='radio' name='imp' value='high'}]
| [{FormInput type='radio' name='imp' value='medium'}]
| [{FormInput type='radio' name='imp' value='low'}] ... [{FormClose}]
[{FormOutput form='rateForm' handler='RatingRDFPlugin' populate='handler'}]
```

The snippet defines a form and attaches it with a *RatingRDFPlugin*. Each form plug-in defines a form handler, which is responsible for adding appropriate statements to the knowledgebase. All user annotations are stored as RDF named graphs [90] with the user's unique login as graph name. This is also considered necessary to prevent spam added to the wiki.

As visualization framework a variant of PatViz (see chapter 5) is used. The aim was to provide multiple interactive visualizations for various patent information aspects. The visualizations developed so far support the analysis of temporal, geographic and classificatory information as well as term distributions of patent sets.



EP1367575B1

Your trail: Main

+ Bibliographic Data for EP1367575B1

+ Legal entities

+ Classifications

+ Family information

- Patent text

Abstract | Description | Claims

A semiconductor laser unit (2) includes a plurality of semiconductor laser elements (4,5) arranged parallel with one another in a laser beam-emitting direction, and a base (T1) for positioning and fixing the plurality of semiconductor laser elements (4,5). The plurality of semiconductor laser elements (4,5) are arranged such that a laser beam (14) emitted from the semiconductor laser element (4) greatest in astigmatism, of the plurality of semiconductor laser elements (4,5), is coincident in its axis with a reference axis (S1) of the base (T1). In the case that the semiconductor laser unit (2) is arranged on an optical head device (1), the focusing characteristic on the plurality of laser beams (14,16) can be improved.

+ Images

- Rating

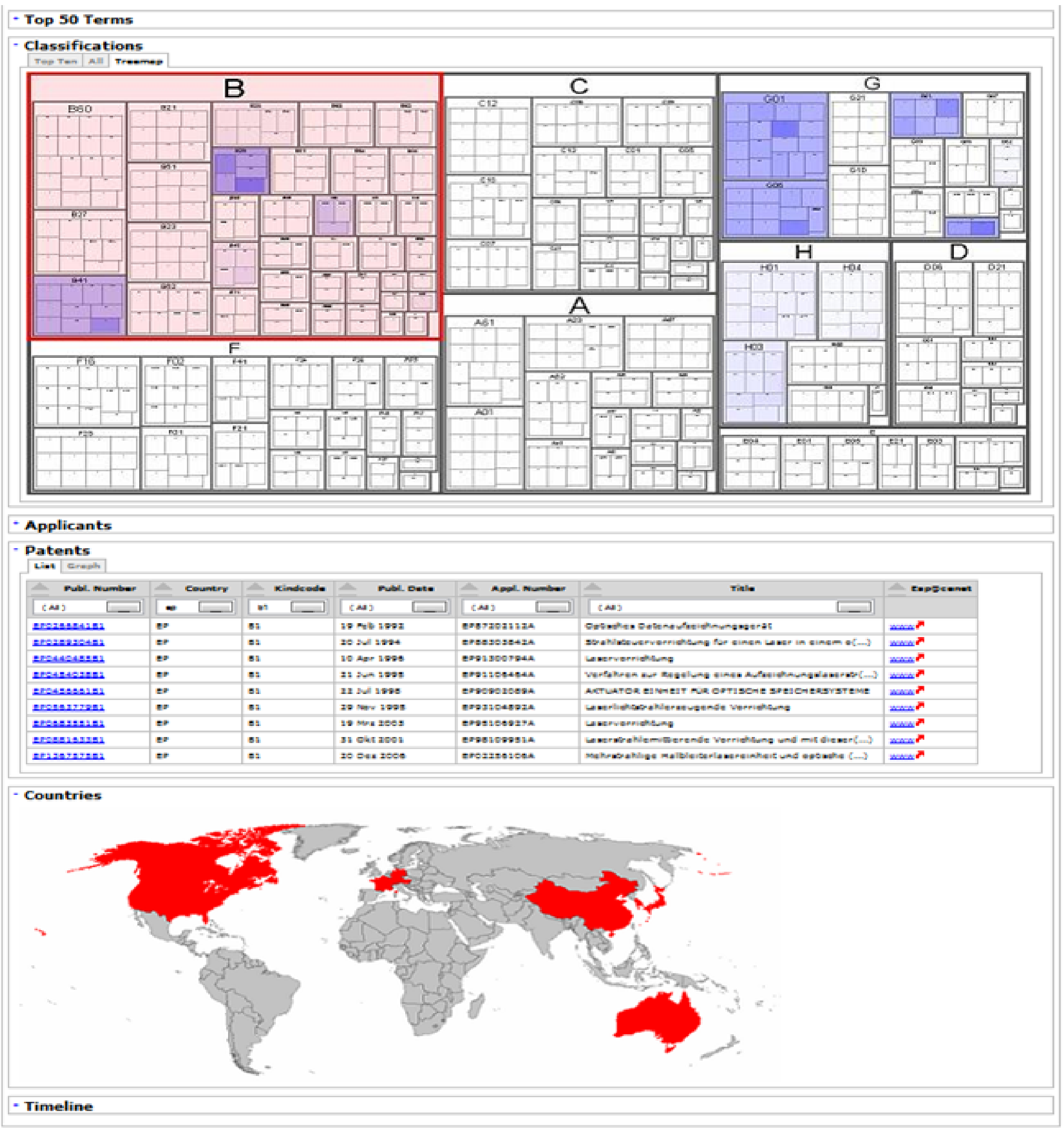
Show Ratings | Rate This Patent

Category	high	medium	low
Importance	○	●	○
Novelty	●	○	○
Applicability	○	●	○

Figure 8.2: Simple rating plug-in

8.3 Example

Input for the visualizations are the patent information models (see chapter 3) attached to wiki pages. Patent information models are RDF graphs encapsulated as Java object model. For the mapping between RDF and Java the Jastor framework [14] has been used. One important design goal was to minimize the loading time for the data transferred to the client as well as the time needed for parsing. We therefore use efficiently compressed binary object streams that are directly deserialized into a Java object model on client-side without any parsing. We are aware that this is specific to the patent domain, since here we have the simple case of immutable data (patent data are read-only once they have been published).



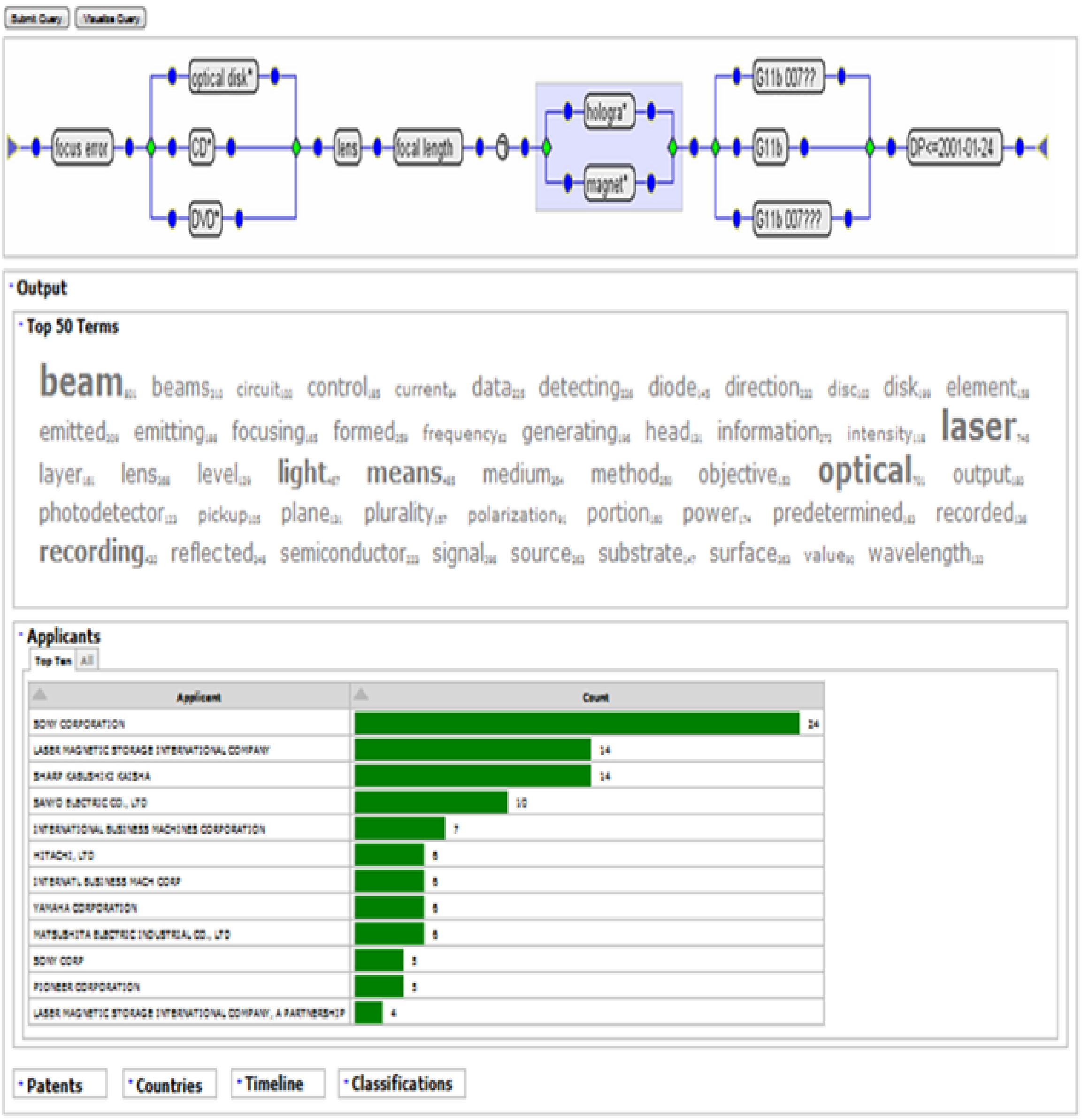
- Top 50 Terms
- Classifications
- Applicants
- Patents

Publ. Number	Country	Kindcode	Publ. Date	Appl. Number	Title	Esp@cenet
(All)	ep	B1	(All)	(All)	(All)	
EP0288884B1	EP	B1	19 Feb 1992	EP87202112A	Optisches Datenaufzeichnungsgerät	www
EP0289304B1	EP	B1	20 Jul 1994	EP88203842A	Strahlsteuervorrichtung für einen Laser in einem o(...)	www
EP0440485B1	EP	B1	10 Apr 1996	EP91300794A	Laservorrichtung	www
EP0484022B1	EP	B1	21 Jun 1995	EP91106464A	Verfahren zur Regelung eines Aufzeichnungslaserstr(...)	www
EP0425661B1	EP	B1	22 Jul 1998	EP90902089A	AKTUATOR EINHEIT FÜR OPTISCHE SPEICHERSYSTEME	www
EP0583779B1	EP	B1	29 Nov 1995	EP93104892A	Laserlichtstrahlerzeugende Vorrichtung	www
EP0588355B1	EP	B1	19 Mrz 2003	EP95106927A	Laservorrichtung	www
EP0881633B1	EP	B1	31 Okt 2001	EP98109951A	Laserstrahlemittierende Vorrichtung und mit dieser(...)	www
EP1287787B1	EP	B1	20 Dez 2006	EP02256106A	Mehrstrahlige Halbleiterlasereinheit und optische (...)	www

- Countries
- Timeline

Figure 8.3: Embedded treemap, table and world map visualizations

Figures 8.3 and 8.4 show the visualizations generated from the attached patent information model embedded within a result page. The upper graphic in figure 8.3 shows the distribution of the patent set according to the International Patent Classification (IPC). The amount of classified patents of a certain IPC category is indicated by color. The user can zoom in and expand sub-categories. The center graphic shows the patent set as search and filterable table using a provided wiki format. In the lower visualization the geographical distribution is presented as a colored word map, where the colors indicate the number of patents in a certain country.



beam beams circuit control current data detecting diode direction disc disk element emitted emitting focusing formed frequency generating head information intensity laser layer lens level light means medium method objective optical output photodetector pickup plane plurality polarization portion power predetermined recorded recording reflected semiconductor signal source substrate surface value wavelength

Applicant	Count
SONY CORPORATION	24
LASER MAGNETIC STORAGE INTERNATIONAL COMPANY	14
SHARP KABUSHIKI KAISHA	14
SANYO ELECTRIC CO., LTD	10
INTERNATIONAL BUSINESS MACHINES CORPORATION	7
HITACHI, LTD	6
INTERNATL BUSINESS MACH CORP	6
YAMAHA CORPORATION	6
MATSUSHITA ELECTRIC INDUSTRIAL CO., LTD	6
SONY CORP	5
PIONEER CORPORATION	5
LASER MAGNETIC STORAGE INTERNATIONAL COMPANY, A PARTNERSHIP	4

Figure 8.4: Embedded query, tag cloud and chart visualizations

The upper part in figure 8.4 shows a version of the PatViz query visualization developed by Yang in his diploma thesis [294]. The center figure shows a tag cloud of containing the most relevant terms in the current patent information model. The lower chart shows the number of patent documents filed by each applicant.

8.4 Discussion

We believe that integrating visualizations into the patent wiki allows for additional analysis of patent sets. Since the focus was to provide client-side interaction and rich visualizations we used and extended an existing visualization framework (see chapter 5) and integrated the visualizations into the wiki as Java applets. In contrast to the PatViz prototype the applets are standalone and do not know each other. As a consequence there is no brushing and linking functionality at the moment.

For the basic interactions within a wiki page on the client-side, for example for hovering, selecting items, expanding and collapsing panels, the Mootools [19] JavaScript framework is used because it is already integrated in the JSPWiki System. The visualization framework itself is managed as a special wiki page, which has the framework code and the configuration scripts attached. As a simple extension the script pages could be associated to different user roles in order to provide some further level of personalization. For fully personalization the scripts could be associated with the user home page. The latter is not foreseen at the moment.

Another important aspect is privacy. We are aware of the fact that searching for patent information is generally considered as sensitive, because it could give competitors hints about the directions in which a company could go concerning the current and future research. We want to give two possible solutions to this issue. First, wiki user management functionality can be used to restrict the access to certain pages considered as sensitive. This means to trust the provider who runs the wiki service. A second solution is to encrypt the sensitive parts of the information so that only the owner or a restricted group can access this information. The aspect of partial encryption of RDF graphs has been discussed in chapter 4.

Currently semantic annotations can be made via special plug-ins, such as the previously described ranking plug-ins. An extension to the wiki markup language for using RDFa [269] in conjunction with arbitrary ontologies is planned for the future.

A current limitation is the need of a dedicated patent index. The used patent index contains about 100.000 patent references that have been crawled using OPS as test patent corpus developed in the PATExpert project. The patent domains investigated in PATExpert are restricted to 'optical recording' and 'machine tools'. This limitation will change in the future because OPS already provides search functionality in its current version 2.6.2. Full-text and metadata search is then possible on the complete patent database of the European Patent Office. Additional semantic search is possible on the patents added to the wiki. By using a semantic representation formalism for patent information it is expected that other patent services can be better integrated and merged in the future.

8.5 Conclusions

In this chapter a new approach has been described for searching, visualizing and annotating patent information by using a semantic wiki. The JSPWiki system has been extended in order to integrate interactive visualizations as Java applets, e.g. for showing the classificatory, geographical, and temporal distribution of patent sets.

The approach differs from typical wiki usage scenarios in the sense that it combines automatic content generation based on patent search activities of the users with user driven semantic annotation of patent information. Currently the aspects of patent rating, linking with prior art, reviews, and general comments and discussions are considered. The content generation is based on a semantic patent information model, which is described in terms of a patent ontology. The aim of the proposed system was to use and extend a modern wiki for semantic annotation, to integrate patent search functions, and to combine it with rich visualizations in order to support the knowledge-intensive tasks of patent search and understanding.

Another aspect that has been realized was the usage of wiki pages as distributed storage for patent information models. Depending on the page type, either a result set representation or a patent document representation is attached to a page. One advantage of this approach is that no explicit repository has to be maintained, but with the limitation that there is no central access point for semantic search.

CHAPTER 9

Architecture for Visual Patent Analysis

In this chapter an architecture for visual patent analysis is described. This architecture integrates the semantic representation model described in chapter 3, the partial encryption method for semantic models described in chapter 4, and the visualization techniques described in chapter 5 and 6.

In the second part three prototypes developed as part of this thesis are reviewed from an architectural perspective. It is discussed how the different technologies used in the PatLens (section 7.2.2), PatWiki (chapter 8), and PatViz (section 9.3.1) prototypes play together for realizing the patent analysis use cases introduced in section 2.1.3.

9.1 Patent Analysis Process

Koch et al. [170] described a patent analysis loop which consists of the three stages: patent retrieval, result set analysis, and detail analysis. This loop can be extended by the ability of adding semantic annotations in each phase. We call this extension *abstract patent analysis process*. It consists of the activities and transitions shown in figure 9.1.

1. The first phase is the formulation of a new patent query or the reformulation of an existing one. Queries can be annotated with additional information, e.g. concerning the purpose of the query, chosen or discarded alternatives, findings, or bibliographic metadata. Standard vocabularies, for example Adobe's Extensible Metadata Platform [42] or DCMI Metadata Terms [47] can be used for this purpose.

2. The second phase is a coarse analysis of a result set. Results of patent analyses can be added as annotations to the result set representation. For this purpose standard or specialized vocabularies can be used.

3. The third phase is a detailed analysis of individual patent documents. Findings, comments, ratings, etc. can be documented as annotations to the semantic representation entities described in chapter 3.

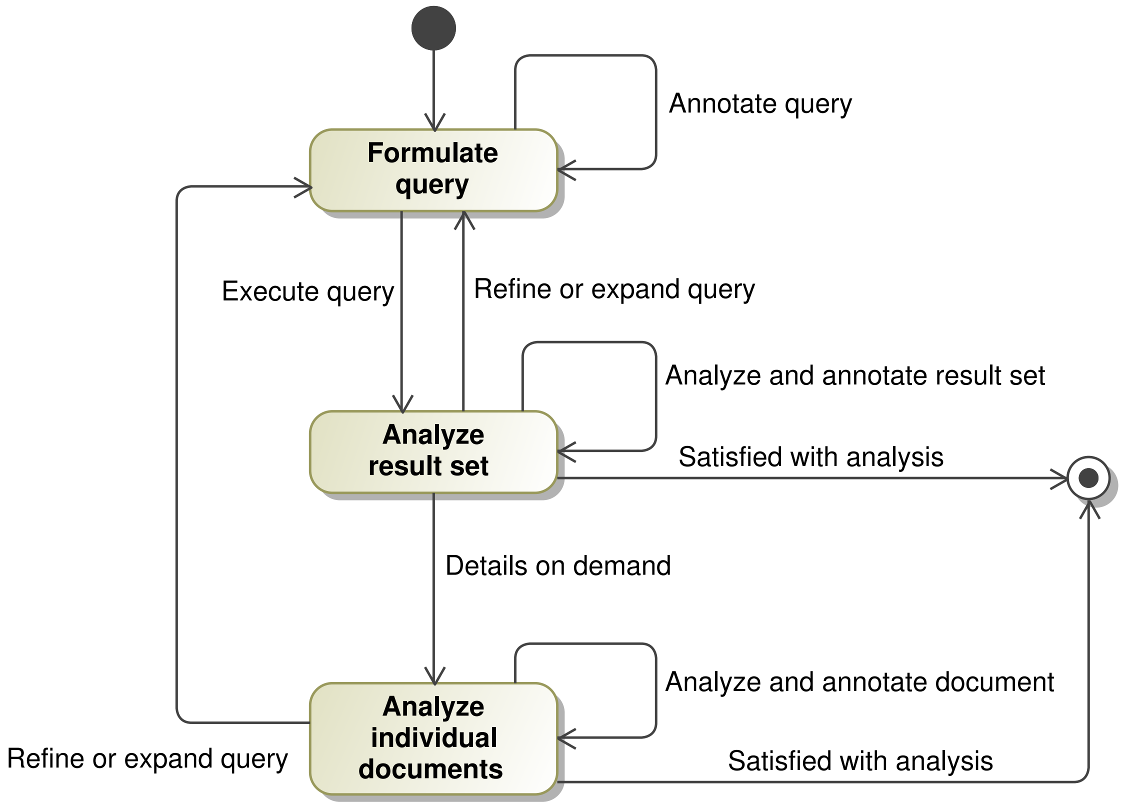


Figure 9.1: Abstract patent analysis process

The process is performed iteratively until the users are satisfied with the result. For each step different tools for analysis, annotation and visualization can help the users in performing their tasks. This abstract patent analysis process has been used as template for the proposed architecture and prototypes described in the following two sections.

9.2 Architecture for Visual Patent Analysis

Visual patent analysis is carried out by analysts using a patent analysis system. The analysts interact with the system by using a visual user interface that provides highly interactive visualizations for patent information and associated analysis results. The basic requirements for visual patent analysis are the retrieval, analysis, representation, secure persisting and interactive visual presentation of patent information and associated analysis results. Analyses augment an initial model with new information that ideally enables experts to gain new insights. It can be distinguished between automatic, semi-automatic and manual analyses. The later are also referred to as annotations. Visual patent analysis also has to deliberate the sharing of patent analyses with other users or systems. Since the results of patent analyses are in general regarded as sensitive information, security issues also have to be considered.

9.2.1 Components and Interfaces

The component diagram in figure 9.2 shows the major parts of the system, their dependencies and the interfaces each component provides or requires.

- *Patent information model (PIM)*: A model that contains all relevant information necessary to carry out a visual patent analysis. It provides the data structures for representing patent information and associated analysis results. An RDF based approach for building such a model has been described in chapter 3.

- *Visual user interface*: The central component which is responsible for the visual presentation of *patent information models* to the user and for handling user interaction. It also allows to compose and edit *query models* and *analysis models*. Invokes the *orchestrator* component in order to perform a patent analysis.

- *Analysis model*: A model that holds the parameters required by *patent analysis services*. Parameters can be simple key-value pairs or complex graph structures represented in RDF.

- *Query model*: A model for representing patent queries of a patent analysis. One query may change over time during the analysis process. Therefore the model

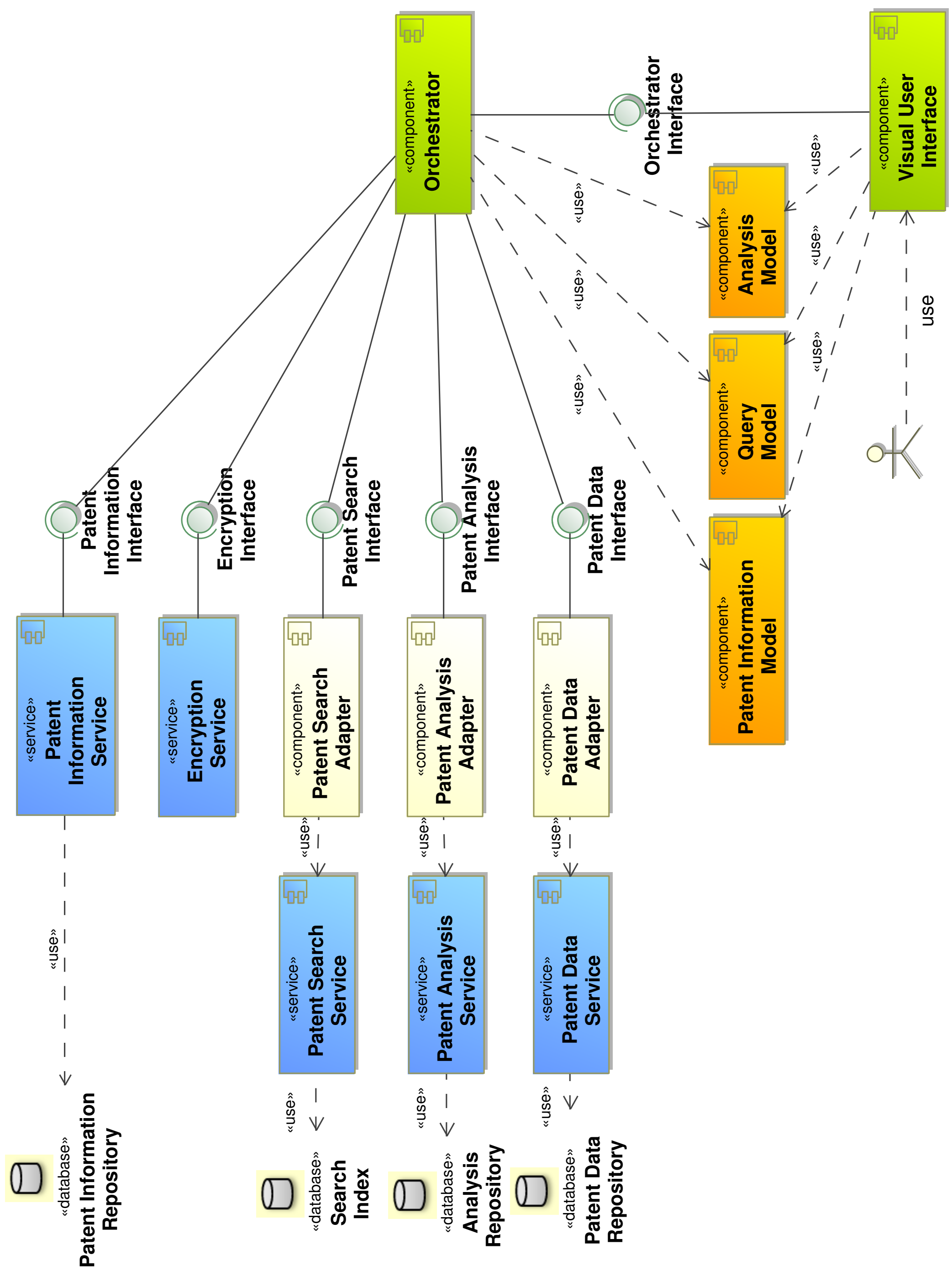

«component» Orchestrator
Orchestrator Interface
«component» Visual User Interface
«use»
«use»
«use»
«use»
«use»
«use»
use
«component» Analysis Model
«component» Query Model
«component» Patent Information Model
Patent Information Interface
Encryption Interface
Patent Search Interface
Patent Analysis Interface
Patent Data Interface
«service» Patent Information Service
«service» Encryption Service
«component» Patent Search Adapter
«component» Patent Analysis Adapter
«component» Patent Data Adapter
«use»
«use»
«use»
«use»
«service» Patent Search Service
«service» Patent Analysis Service
«service» Patent Data Service
«use»
«use»
«use»
«database» Patent Information Repository
«database» Search Index
«database» Analysis Repository
«database» Patent Data Repository

Figure 9.2: Overview of the components involved in visual patent analysis

should be able to manage different versions of one query. A query could be annotated, for example tagged or commented by an analyst. A query model may contain different queries in different query languages for different search services.

- *Orchestrator*: The central component that implements the business logic. It is responsible for invoking the services and adapters, aggregating the data and building up a PIM. Since services may change in the future, it is important to minimize the service dependencies. Therefore adapters are defined for abstracting data formats and service invocation details.

- *Patent search service*: A service that takes a patent query as input, looks up a search index and returns a list of patent document identifiers that match the query. There can be different such search services, for example for metadata search, full-text search, semantic search, document similarity search or image search. In general, each search service manages its own index. The *patent search adapter* and *patent search interface* provide an abstraction of how the search services are called and the results are mapped.

- *Patent information service*: A service that looks up a repository for available patent information. For RDF based PIMs the repository is a SPARQL endpoint. It returns the relevant fragments of previously added PIMs. The service is invoked using the *patent information interface*.

- *Patent data service*: A service that takes a list of patent document identifiers and other additional service options as input, looks up a repository and returns the result data that match the query. Again, there can be various such data services, for example for the retrieval of bibliographic data, patent families, patent contents, images, etc. The *patent data adapter* and *patent data interface* abstract the communication with concrete services. A patent data adapter also is responsible for transforming the result data into a structure and format appropriate for the PIM.

- *Patent analysis service*: An analysis service can be anything from a simple term frequency calculation up to a complex dimension reduction or ontology learning. Very time intensive analyses could be performed in advance in order to allow answering the request within an acceptable time limit. Usual online analysis services perform statistical processing or apply business rules on the semantic model. A patent analysis service takes patent information and analysis parameters as input, analyses the information and gives back the results of the analysis. Typically, there are multiple analysis services involved, for example for summarizing patent contents, for extracting features, for ontology learning, for computation of document similarities, etc. In general, two types of analysis services can be distinguished: services that perform online or offline analyses. An online service needs all the information required for the analysis, whereas an offline service already has performed the analysis and has stored the results in an analysis index. In practice offline and online characteristics are combined. Each analysis service has an associated *patent analysis adapter* defining a corresponding *patent analysis interface*.

- *Encryption service*: A service for encrypting and decrypting sensitive fragments within a PIM. It takes a model, appropriate credentials and an encryption specification as input. This specification defines which fragments to encrypt. The decryption does not require an encryption specification. Details about the encryption method can be found in chapter 4.

9.2.2 Interaction between Components

A typical service invocation sequence is depicted in figure 9.3. The sequence shows the service calls needed for an initial patent analysis. The sequence is described below.

An analyst composes a query using a visual user interface. The visual user interface updates the query model. The user has finished the query composition and starts the analysis. The orchestrator is called with a query and analysis model (1). The orchestrator invokes one or more patent search services via a patent search adapter (2). The response of the search service is a ranked list of patent document identifiers that fulfill the specified constraints (3). In case multiple search services have been invoked the results are merged. Based the list of patent document identifiers the patent information service is called (4). It returns the information available for each patent identifier in terms of patent information model fragments (5). Missing patent data are requested from corresponding patent data services, e.g. the Open Patent Services [21] (6). The structure and format of the result data depend on the used data service. The responses are transformed by a patent data adapter into the formalism used by the PIM (7). Patent analysis services are called with the merged patent information model and analysis model (8).

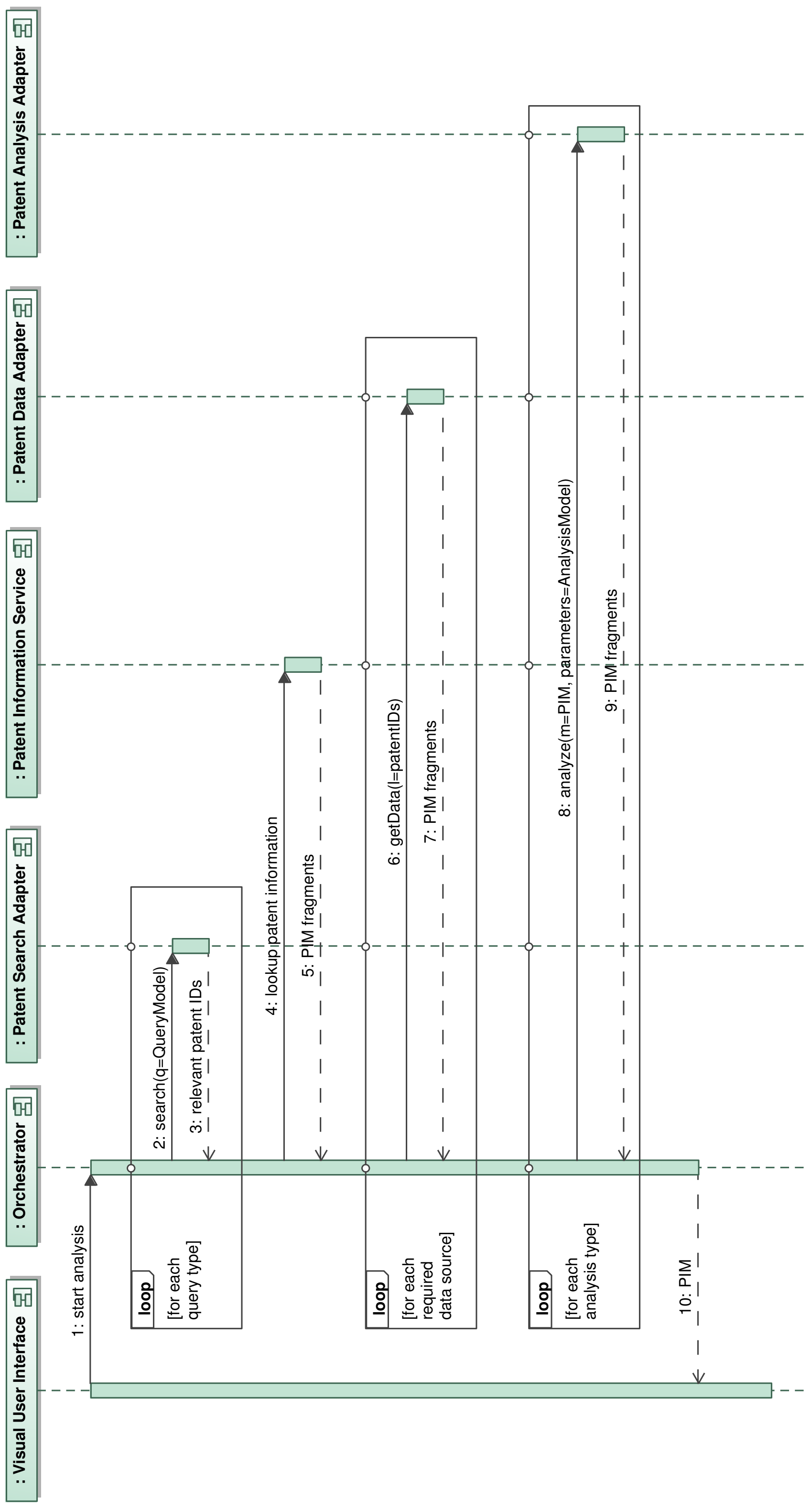


Figure 9.3: Calling sequence in a typical patent analysis process

After an analysis has finished the results are returned and transformed by a corresponding patent analysis adapter into the formalism used by the PIM (9). The PIM is updated and the data are merged with the existing model. The visual user interface is notified that the retrieval of the PIM has been successful (10).

As it can be seen in this sequence diagram the response time of the system heavily depends on the used search, retrieval and analysis services. Therefore parallelization, pre-fetching and caching of requests are essential for optimizing the performance of the overall system. It is also important to note that this architecture explicitly considers multiple patent search, patent data and patent analysis services.

The central idea is the division of the overall retrieval task into two parts: first, the retrieval of unique identifiers and second, the retrieval of the data associated with the identifiers. The merging of identifiers can be specified in terms of set operations. The merging of associated data is handled based on the native merging capability of RDF. Section 9.3.1 will give an example for the logical composition of higher level queries that integrate different retrieval services. Another important aspect is the usage of a well-defined terminology as service contract instead of fixed software interfaces.

After this initial sequence the query can be iteratively modified either by narrowing the request to remove noise from the results, or by widening the search because the user wants to find additional relevant patents. In both cases, the insight that generates a user's desire for refinement is normally perceived through the presentation of the previous results. Therefore an efficient feedback loop is important to transfer insights from result set exploration to the query formulation. This aspect will later be discussed in section 9.3.1. The next section shows a white box view of the visual user interface architecture.

9.2.3 Visual User Interface

In user interface design typically data (model), presentation (view) and interaction (controller) concerns are separated, also known as *model view controller (MVC)* paradigm [172, 115]. Information visualization interfaces have further requirements concerning the presentation of data and user interaction and therefore should use proven design patterns, such as the patterns described in [141]. Figure 9.4 gives an overview of the visual user interface architecture. The patterns used to realize this architecture are described below.

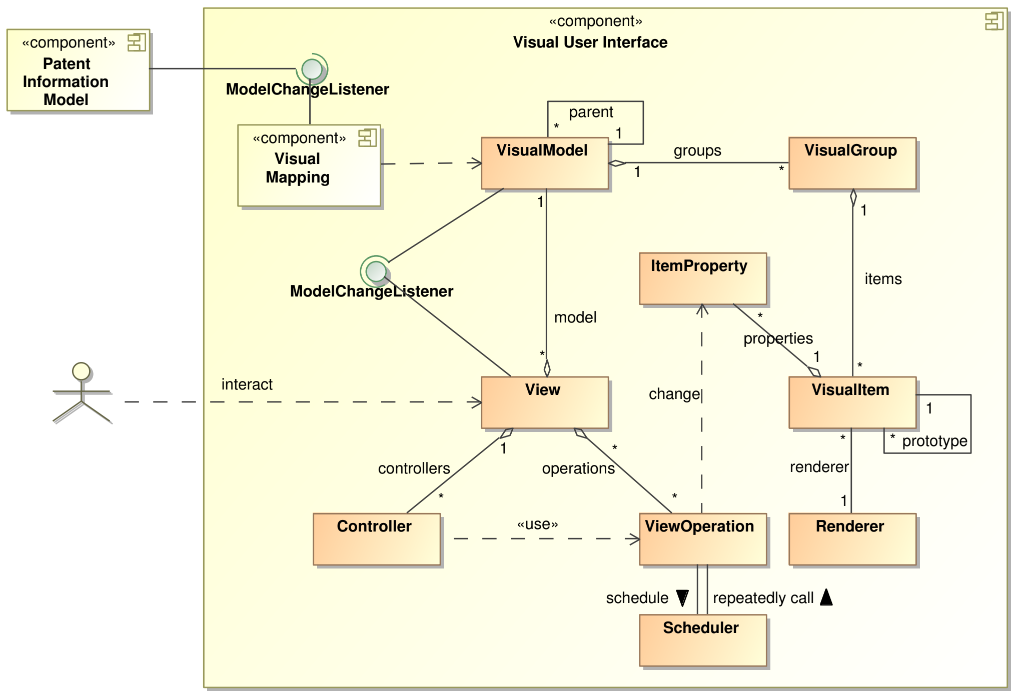


Figure 9.4: Visual user interface components

Visual Mapping

Visual mapping is the process of selecting relevant data in an underlying data model (*PatentInformationModel*) and mapping it to structures appropriate for visualization (*VisualModel*). This mapping process is handled by a *VisualMapping* component.

The realization of such a mapping has to take the targeted visual design into account. For example, a mapping for a graph visualization may differ from a mapping for a scatter plot visualization, since a graph needs relations to be explicitly set. A visual mapping can be specific providing all the necessary visual attributes (position, size, shape, color, etc.) or it can be general delegating the specification of visual attributes later processes. The visual mapping therefore also relates to *VisualOperations* which will be discussed later.

Visual Model

A *VisualModel* contains all items (*VisualItem*) that are to be visualized in one or more *View*. In order to support different layers or different item types, a group concept (*VisualGroup*) has been introduced. Each *VisualItem* is further described by collection of properties (*ItemProperty*).

In general it can be distinguished between data properties, visual properties and state properties. Examples for data properties are strings, numbers, or complex objects. Complex objects can be composed recursively by other complex objects. This allows to build up tree or graph structures. Examples for visual properties are color, shape, size, position, font, margins, etc. In order to separate the styling from the visualization and interaction design, it has been a good practice to move static visual properties to external styles sheets. The state of an item depends on previous user interactions, for example whether an item has been selected, hovered, disabled, etc. and is also modeled as *ItemProperty*.

It is also useful to provide the concept of dynamic properties which take other properties as input and produce an output value whenever the input parameters change. Dynamic properties can be defined using *Expression* pattern (cf. [141, 2.6]). The properties of an item are interpreted by a *Renderer*, which finally draws the item. Whenever a property is changed or when items are added to or removed from a group, the model is notified. For bulk updates the notification is optimized so that it is only triggered once.

Multiple Views

Multiple views support the investigation of a single conceptual entity by two or more distinct views. The views can differ in the data they show or in the visual presentation of the data. Multiple views are often used, when there is a diversity of attributes or levels of abstraction in order to partition complex data into manageable chunks. Multiple views are also used to bring out correlations or disparities between the data [59].

For the purpose of visualizing one model in multiple views, the concept of prototypical inheritance is applied for looking up the properties of an item[1]. When a property is read that is not directly contained in an item, the item's prototype is looked up recursively. If a property is set, it is directly set on the item without traversing the prototype chain. This means that values can be overridden without changing the value of the prototype.

Whenever a *VisualModel* is associated with a *View*, it is recursively cloned by cloning its *VisualGroup* and *VisualItem* instances (figure 9.5). The prototype of each cloned *VisualItem* is linked with the original item. View operations can be performed on all item properties of the parent model and can override the properties on the cloned items without modifying the parent model.

[1]*Prefuse* implements prototypical inheritance by using the *Cascaded Data Tables* pattern [141, 2.3]

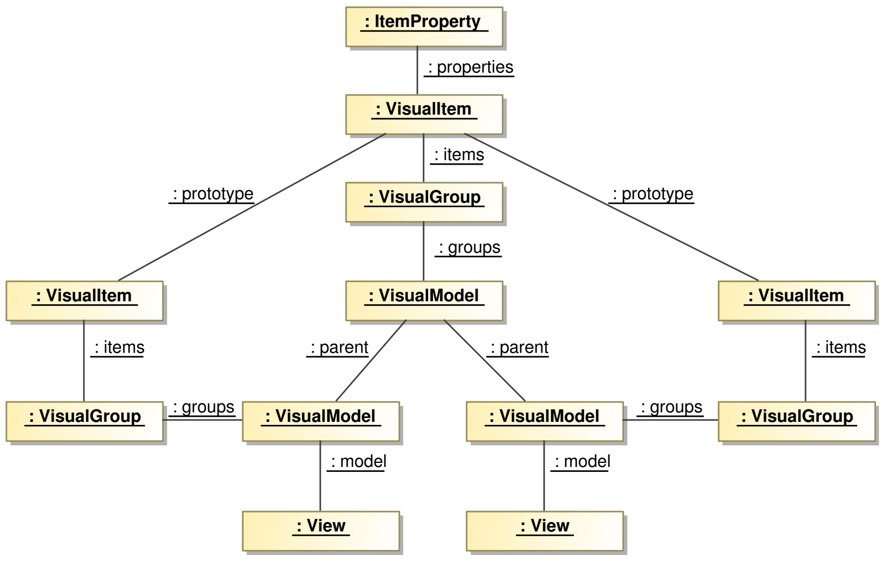


Figure 9.5: Multiple views accessing the same parent model

Decomposition of visual processing

To improve the development of flexible, configurable, testable and reusable visualization components the visual data processing is decomposed into a series of *ViewOperations*. The main task of each such operation is to change item properties. Common operations are for example are filtering and layout. A filter operation usually sets the visibility state and whether an item should be included in the layout.

A layout operation typically computes the position and size of the items. For example the layout of hierarchical structures could be tree, treemap or sunburst layout. In some cases there are dependencies between layout and renderer components. By providing additional item properties and making the renderers more general, these dependencies can be minimized. *ViewOperations* can be chained, for example to first filter the items and then to re-layout the remaining items.

Dynamic visualizations often require time-sensitive recurring *ViewOperations*. A common example is changing a visual property at regular time intervals for animation. A *ViewOperation* registers itself at a *Scheduler*, which then invokes the operation repeatedly over given duration and at specified intervals. If the interval is non-linear, different pacing behavior can be implemented [141, 2.7].

Rendering

A *Renderer* interprets the properties of an item and transforms them into a set of low-level graphic API calls. By exchanging the rendering and view implementation, different target platforms can be supported, for example Java 2D, OpenGL, or HTML5 canvas. Whenever the item properties change, the *View* needs to be rendered again. Elaborated visualization frameworks provide complex optimizations to only update the regions that have changed in order to make the rendering as efficient as possible. This is especially important for animations. The rendering mechanism used in the *Prefuse* framework is described in [142].

Interaction

The handling of user input is essential for each visual user interface. Mouse and keyboard input is handled by one or more *Controller* instance each being associated with one view. A controller can change the view, for example applying affine transformations, or it can change item properties of the model. In most cases a controller calls a *ViewOperation* that encapsulates the manipulation of item properties. By composition of view operations, sophisticated interaction techniques such as brushing and linking, focus+context or dynamic filtering can be realized (see section 2.4.2). A controller can also change the data model by submitting a new query. Changing the data model is, for example, necessary for realizing details-on-demand or semantic-zooming interactions.

9.3 Review of Prototypes

This section gives an overview of the PatViz, PatWiki, and PatLens prototypes developed as part of this thesis. PatViz is partly based on the theses of Koch [168] and Bosch [77] and has been jointly developed with Koch and Bosch in the PATExpert project. The PatWiki prototype uses a query visualization that is based on the thesis of Yang [294]. The PatLens prototype has been jointly developed with Rotard as part of our Semantic Lens approach [211]. The focus of the author has been on implicit queries, the visualization modules and the RDF based repositories.

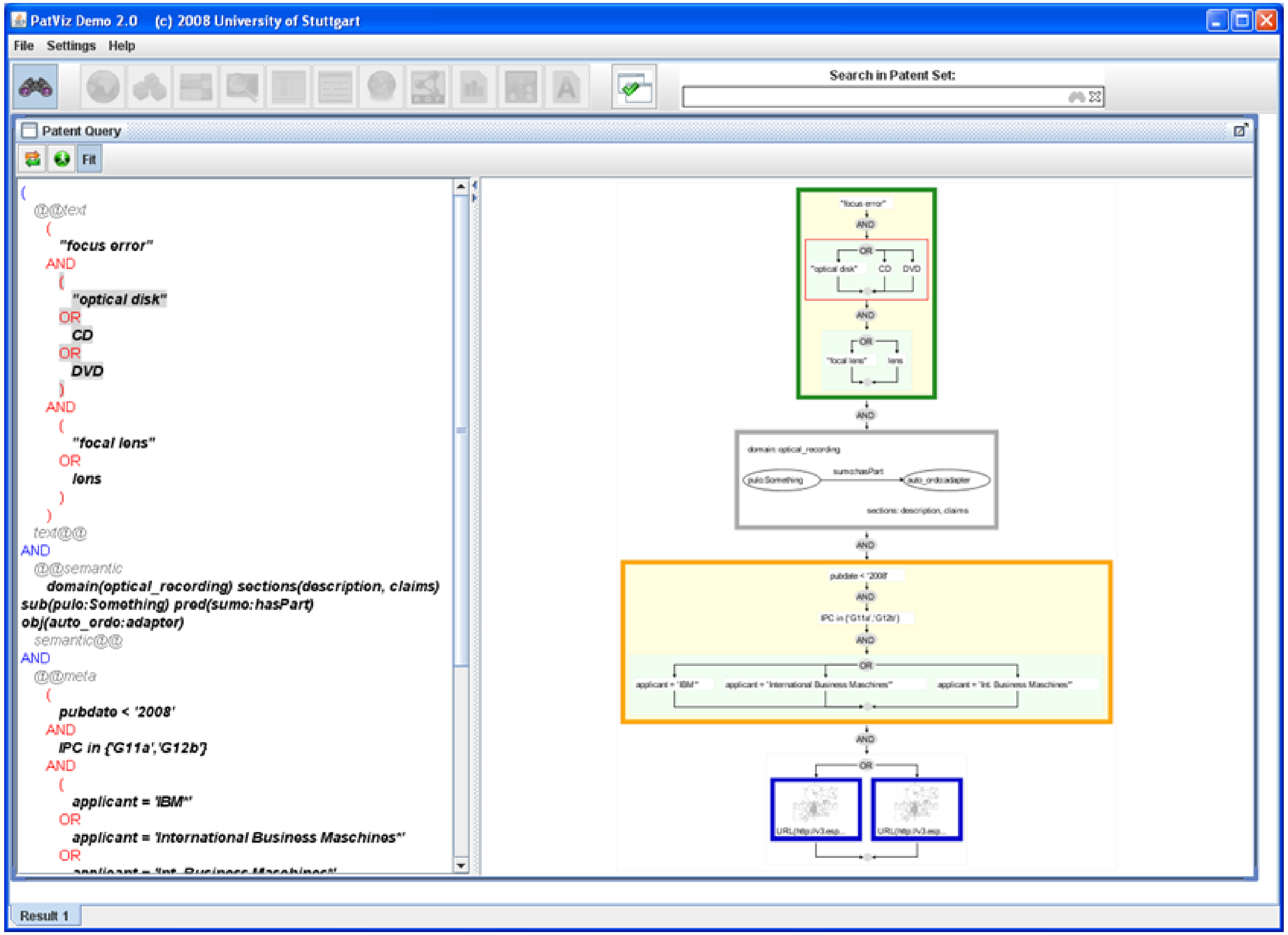
9.3.1 PatViz

The PatViz prototype was built as an interactive visual client for the different patent retrieval and analysis services developed in the PATExpert project. The focus of the PatViz prototype is on unified visualization of different query languages and on the tight integration of result visualizations with query refinement [128, 170].

In PATExpert there are four different query engines for full-text, metadata, image similarity and semantic search. Each search engine comes with its own query language. The results of each engine are combined by a Boolean integration language. The actual merging of the results is performed by a *merger* service, so that the orchestrator only has to communicate with a single patent search service. The PATExpert search facility is described in more detail in [100]. For illustration, figure 9.6 shows a query that contains four query fragments for full-text, metadata, semantic and image search.

All four fragments are combined with the logical AND-operator. The textual presentation is shown on the left side and the graphical equivalent on the right. Both can be edited simultaneously. The queries as well as the corresponding query results are visualized by multiple views in the PatViz desktop. By offering multiple views, each for a specific aspect, users get a faster overview of the current PIM.

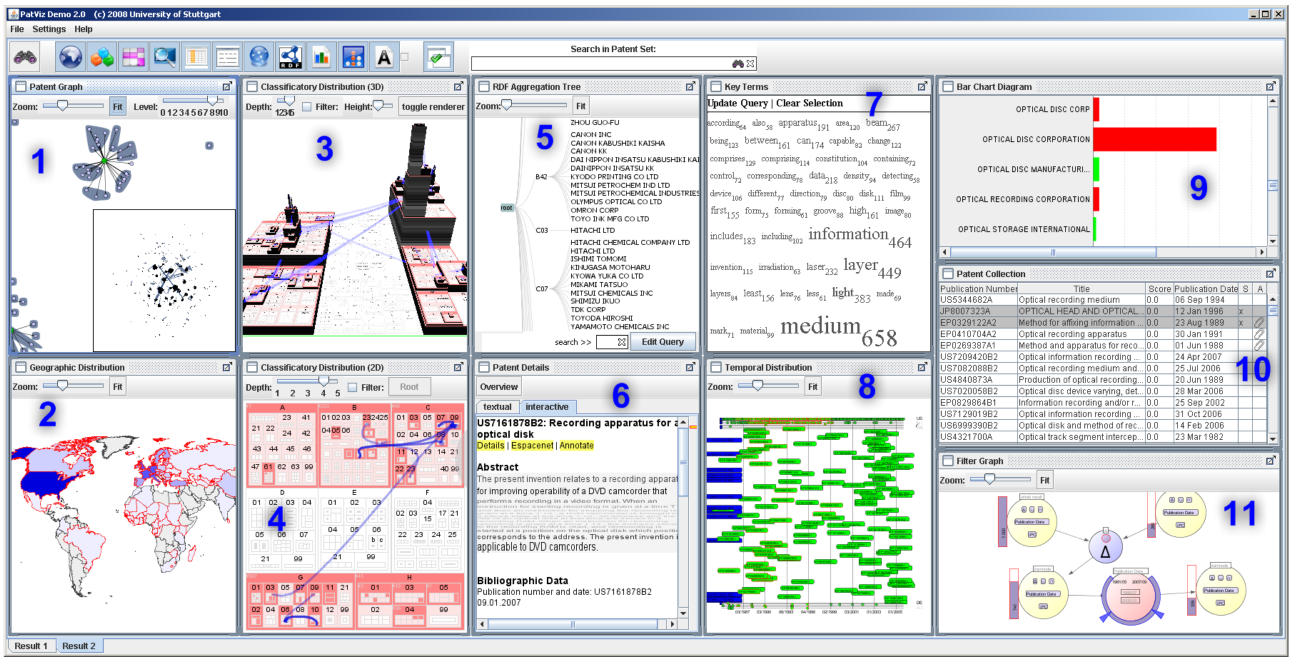
Users can locally analyze this model by exploring different filter settings. An obvious idea therefore was to use the filters for refining or expanding the query and thus to interactively map the user's insights from the analysis of the current result set to its underlying query. The main benefit of this approach is that it enables users to start with an initial query and then iteratively refine their query directly from within the visualizations. A user could for example select countries in a geographic map visualization, categories on an IPC distribution treemap or terms on a tag cloud. These filters can be used to update the query, so that the user does not have to remember the query syntax. The logic for mapping the filter settings of the visualizations onto a corresponding query is implemented by the *Query Model* component. When the query is submitted to the system, a new PIM is generated and visualized in PatViz desktop. PatViz provides visualizations for result set and single patents using the visualization techniques described in chapter 5 and 6.



```
(
  @@text
   (
     "focus error"
   AND
     (
      "optical disk"
     OR
      CD
     OR
      DVD
     )
   AND
     (
      "focal lens"
     OR
      lens
     )
   )
  text@@
AND
  @@semantic
   domain(optical_recording) sections(description, claims)
sub(pulo:Something) pred(sumo:hasPart)
obj(auto_ordo:adaptor)
  semantic@@
AND
  @@meta
   (
     pubdate < '2008'
   AND
     IPC in {'G11a','G12b'}
   AND
     (
      applicant = 'IBM*'
     OR
      applicant = 'International Business Maschines*'
     OR
      applicant = 'Int. Business Maschines*'
```

Figure 9.6: Example of a combined query

Figure 9.7 show an example of the PatViz desktop with the following visualizations: (1) A configurable graph view that shows various connections between entities of the current PIM, for example between patent documents, applicants, inventors and IPC classes; (2) A world map for the distribution of the patent documents over the filing countries; (3 and 4) Treemap visualizations for the distribution of the patent documents over the IPC classification schema; (5) A dynamic tree visualization that is constructed based on a typed in query expression; (6) A viewer for patent documents that uses the overlay technique described in chapter 6 for showing available analysis results; (7) A tag cloud containing the most frequent terms used in the current patent document collection; (8) An extended timeline visualization that also includes the geographic distribution and priority relations between patent documents; (9) A chart for counting patent documents contained in the current patent collection according to a customizable metadata field; (10) A table view containing the most important patent metadata number, title or applicant; (11) A graph based selection management tool to store, combine and adjust selections [77].



US7161878B2: Recording apparatus for a optical disk
Details | Espacenet | Annotate

Abstract
The present invention relates to a recording apparat for improving operability of a DVD camcorder that performs recording in a video format. When an [illegible] corresponds to the address. The present invention i applicable to DVD camcorders.

Bibliographic Data
Publication number and date: US7161878B2
09.01.2007

Publication Number	Title	Score	Publication Date	S	A
US5344682A	Optical recording medium	0.0	06 Sep 1994		
JP8007323A	OPTICAL HEAD AND OPTICAL...	0.0	12 Jan 1996	x	
EP0329122A2	Method for affixing information ...	0.0	23 Aug 1989	x	📎
EP0410704A2	Optical recording apparatus	0.0	30 Jan 1991		📎
EP0269387A1	Method and apparatus for reco...	0.0	01 Jun 1988		📎
US7209420B2	Optical information recording ...	0.0	24 Apr 2007		
US7082088B2	Optical recording medium and...	0.0	25 Jul 2006		
US4840873A	Production of optical recording...	0.0	20 Jun 1989		
US7020058B2	Optical disc device varying, det...	0.0	28 Mar 2006		
EP0829864B1	Information recording and/or r...	0.0	25 Sep 2002		
US7129019B2	Optical information recording ...	0.0	31 Oct 2006		
US6999390B2	Optical disk and method of rec...	0.0	14 Feb 2006		
US4321700A	Optical track segment intercep...	0.0	23 Mar 1982		

Figure 9.7: Views in the PatViz desktop

9.3.2 PatWiki

The PatWiki prototype described in chapter 8 is an alternative visual front-end for the PATExpert services. The focus of this prototype is on secure sharing and manual annotation of PIMs. The central idea is to use a wiki for this purpose. The used JSPWiki [16] provides the capabilities to comment on pages, manage attachments in different versions, and send notifications when a page has been commented or changed.

In traditional wikis content pages are created, edited and annotated by users. PatWiki generates pages automatically based on a patent search activity of a user. Each search is edited on a special search page, which contains a plug-in for visually composing the query. When the query is submitted to the system, two types of pages are generated: pages for presenting search results and pages for presenting individual patent documents. Figure 8.4 showed a generated page that contains the query and the corresponding search results. Generated pages cannot be edited directly. However, text comments can be added to a page, for example for discussions on patent content, reviews, linking to prior art, etc.

Additionally, the system can be extended for adding semantic annotations by using plug-ins. The annotations are stored as RDF attachments to the page. As an example for semantic annotation, a plug-in has been developed that allows to rate the importance, novelty and applicability of a patent (figure 8.2).

The PatWiki components incorporated in the general architecture are shown in figure 9.8. The differences in comparison to the PatViz prototype are in the realization of the visual user interface and the orchestrator. The user interface is realized as a collection of Java applets (one applet for each type visualization), wiki markup (tables and term cloud) and JavaScript components. Its configuration and setup has been described in section 8.

The normal flow is as follows. The user interactively composes a query on a special *SearchPage*. For this purpose a query visualization is provided as applet, which updates a *QueryModel* and triggers the *Orchestrator*, when the user submits the query. The *Orchestrator* invokes the Patent Search and Patent Data services (no analysis is performed with this prototype) as described in the sequence of figure 9.3. The result of these service invocations is a *Patent Information Model (PIM)*. This model serves as the foundation for generating the wiki pages. The generation is done by a *Wiki Page Generator*, which cre-

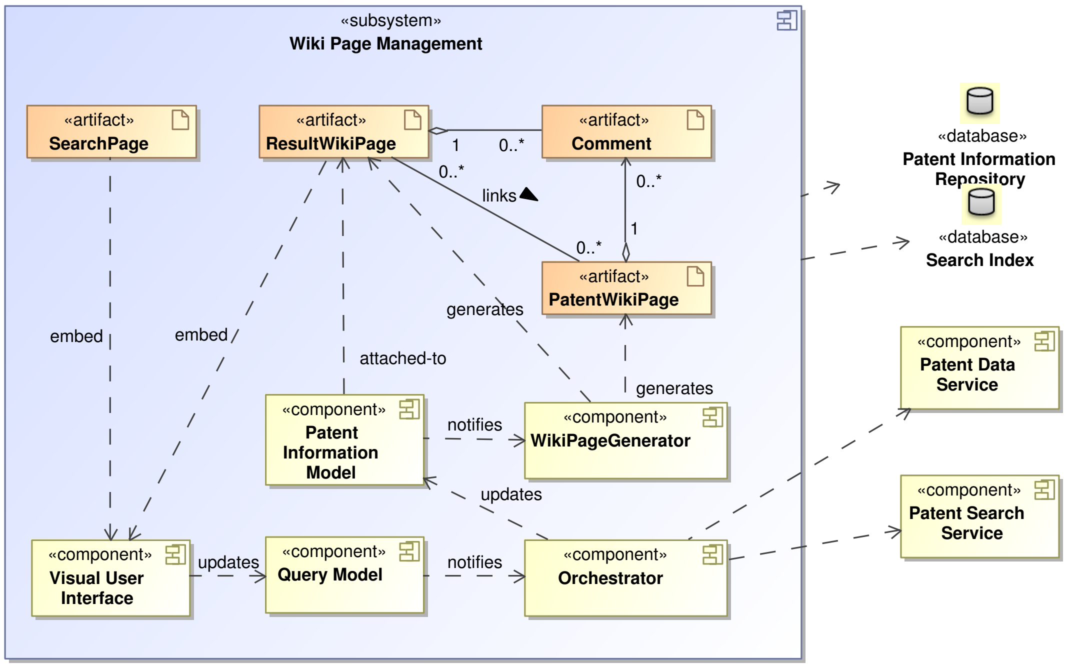

«subsystem»
Wiki Page Management
«artifact» SearchPage
«artifact» ResultWikiPage
«artifact» Comment
«artifact» PatentWikiPage
1
0..*
0..*
links
0..*
1
0..*
embed
embed
attached-to
generates
generates
«component» Patent Information Model
notifies
«component» WikiPageGenerator
updates
«component» Visual User Interface
updates
«component» Query Model
notifies
«component» Orchestrator
«database» Patent Information Repository
«database» Search Index
«component» Patent Data Service
«component» Patent Search Service

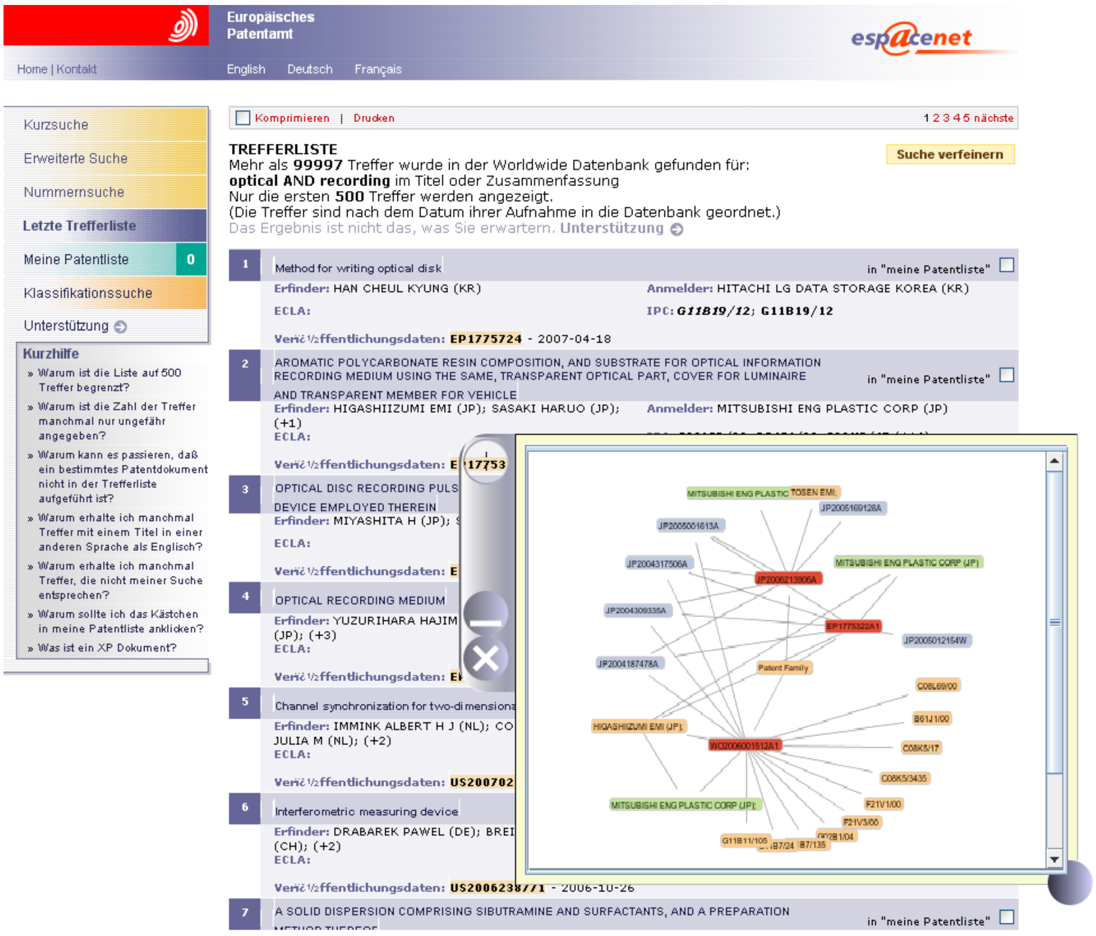
Figure 9.8: Architecture of the PatWiki prototype

ates one *Results Wiki Page* for presenting the patent collection and one *Patent Wiki Page* for each patent document, if it does not already exist. The *Patent Information Model* and the *Query Model* are attached to the *Results Wiki Page*. If a *Patent Wiki Page* already exists for a patent, for example due to a prior search, the page content is replaced with the new content if it has changed. This ensures that a *Patent Wiki Page* always reflects the current state. In contrast to the original architecture the *Visual User Interface* is not notified, when the *Patent Information Model* is changed. This is due to the fact that Java applets have a different lifecycle. As soon as the user changes or reloads a wiki page, the applets retrieve the attached model during their initialization cycle. A server push mechanism is currently not implemented, but can technically be realized for example by long-lived bidirectional TCP connections [191].

All wiki pages can be commented using the native commenting capabilities of the wiki system. Additionally semantic annotations can be made if special plug-ins are provided. The annotations can be encrypted if necessary. They are stored as attachments to the page. All activities are stored by the system so that they can be searched. All queries are stored in the *Query Repository*, all generated pages in the *Search Index* and all *Patent Information Models* in the *Patent Information Repository*.

9.3.3 PatLens

The focus of the *PatLens* prototype described in section 7.2.2 is to seamlessly integrate patent information into the current analysis context for any web application. In contrast to the aforementioned implementations, the query is constructed implicitly using the analysis context contained in the user's current web page. Figure 9.9 gives another example for a PatLens applied to the esp@cenet website. The PatLens shows the members of the patent family associated with the patent number currently selected by the lens.



Europäisches Patentamt

Home | Kontakt   English   Deutsch   Français

esp@cenet

Kurzsuche
Erweiterte Suche
Nummernsuche
Letzte Trefferliste
Meine Patentliste 0
Klassifikationssuche
Unterstützung

Kurzhilfe
» Warum ist die Liste auf 500 Treffer begrenzt?
» Warum ist die Zahl der Treffer manchmal nur ungefähr angegeben?
» Warum kann es passieren, daß ein bestimmtes Patentdokument nicht in der Trefferliste aufgeführt ist?
» Warum erhalte ich manchmal Treffer mit einem Titel in einer anderen Sprache als Englisch?
» Warum erhalte ich manchmal Treffer, die nicht meiner Suche entsprechen?
» Warum sollte ich das Kästchen in meine Patentliste anklicken?
» Was ist ein XP Dokument?

Komprimieren | Drucken   1 2 3 4 5 nächste

Suche verfeinern

TREFFERLISTE
Mehr als **99997** Treffer wurde in der Worldwide Datenbank gefunden für:
optical AND recording im Titel oder Zusammenfassung
Nur die ersten **500** Treffer werden angezeigt.
(Die Treffer sind nach dem Datum ihrer Aufnahme in die Datenbank geordnet.)
Das Ergebnis ist nicht das, was Sie erwartern. **Unterstützung**

1 Method for writing optical disk — in "meine Patentliste"
Erfinder: HAN CHEUL KYUNG (KR) **Anmelder:** HITACHI LG DATA STORAGE KOREA (KR)
ECLA: **IPC:** ***G11B19/12***; **G11B19/12**
Verï¿½ffentlichungsdaten: **EP1775724** - 2007-04-18

2 AROMATIC POLYCARBONATE RESIN COMPOSITION, AND SUBSTRATE FOR OPTICAL INFORMATION RECORDING MEDIUM USING THE SAME, TRANSPARENT OPTICAL PART, COVER FOR LUMINAIRE AND TRANSPARENT MEMBER FOR VEHICLE — in "meine Patentliste"
Erfinder: HIGASHIIZUMI EMI (JP); SASAKI HARUO (JP); (+1) **Anmelder:** MITSUBISHI ENG PLASTIC CORP (JP)
ECLA:
Verï¿½ffentlichungsdaten: **E 17753**

3 OPTICAL DISC RECORDING PULS DEVICE EMPLOYED THEREIN
Erfinder: MIYASHITA H (JP);
ECLA:
Verï¿½ffentlichungsdaten: **E**

4 OPTICAL RECORDING MEDIUM
Erfinder: YUZURIHARA HAJIM (JP); (+3)
ECLA:
Verï¿½ffentlichungsdaten: **EP**

5 Channel synchronization for two-dimensiona
Erfinder: IMMINK ALBERT H J (NL); CO JULIA M (NL); (+2)
ECLA:
Verï¿½ffentlichungsdaten: **US200702**

6 Interferometric measuring device
Erfinder: DRABAREK PAWEL (DE); BREI (CH); (+2)
ECLA:
Verï¿½ffentlichungsdaten: **US2006238771** - 2006-10-26

7 A SOLID DISPERSION COMPRISING SIBUTRAMINE AND SURFACTANTS, AND A PREPARATION — in "meine Patentliste"

Figure 9.9: Lens showing patent family information

The main architectural differences of this prototype are in the wiring of the components. Figure 9.10 gives an overview of the design. Similar to the PatWiki prototype, the user interface also is realized as Java applet, HTML and JavaScript loaded from a backend on demand.

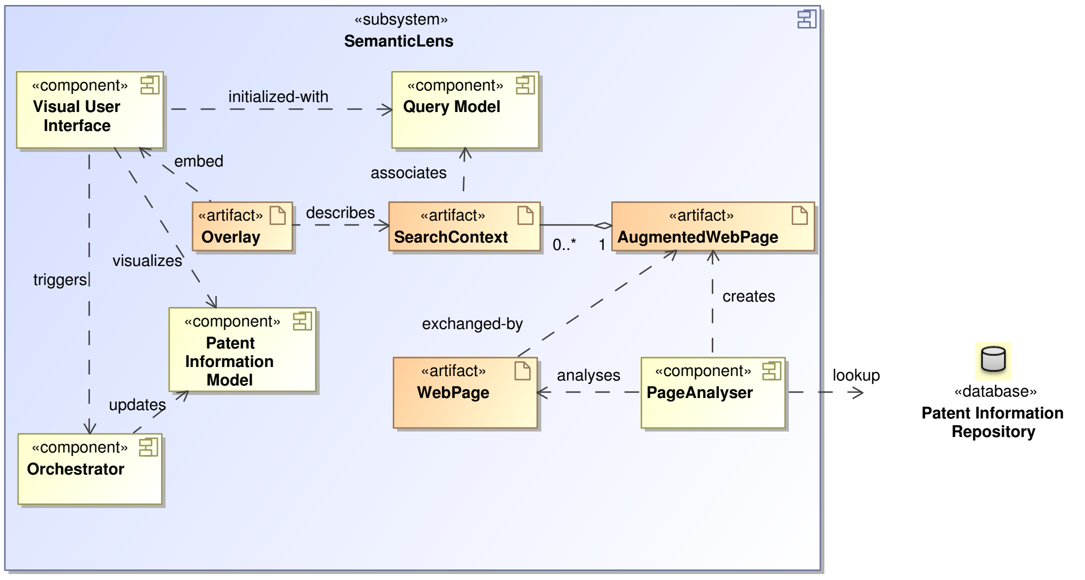


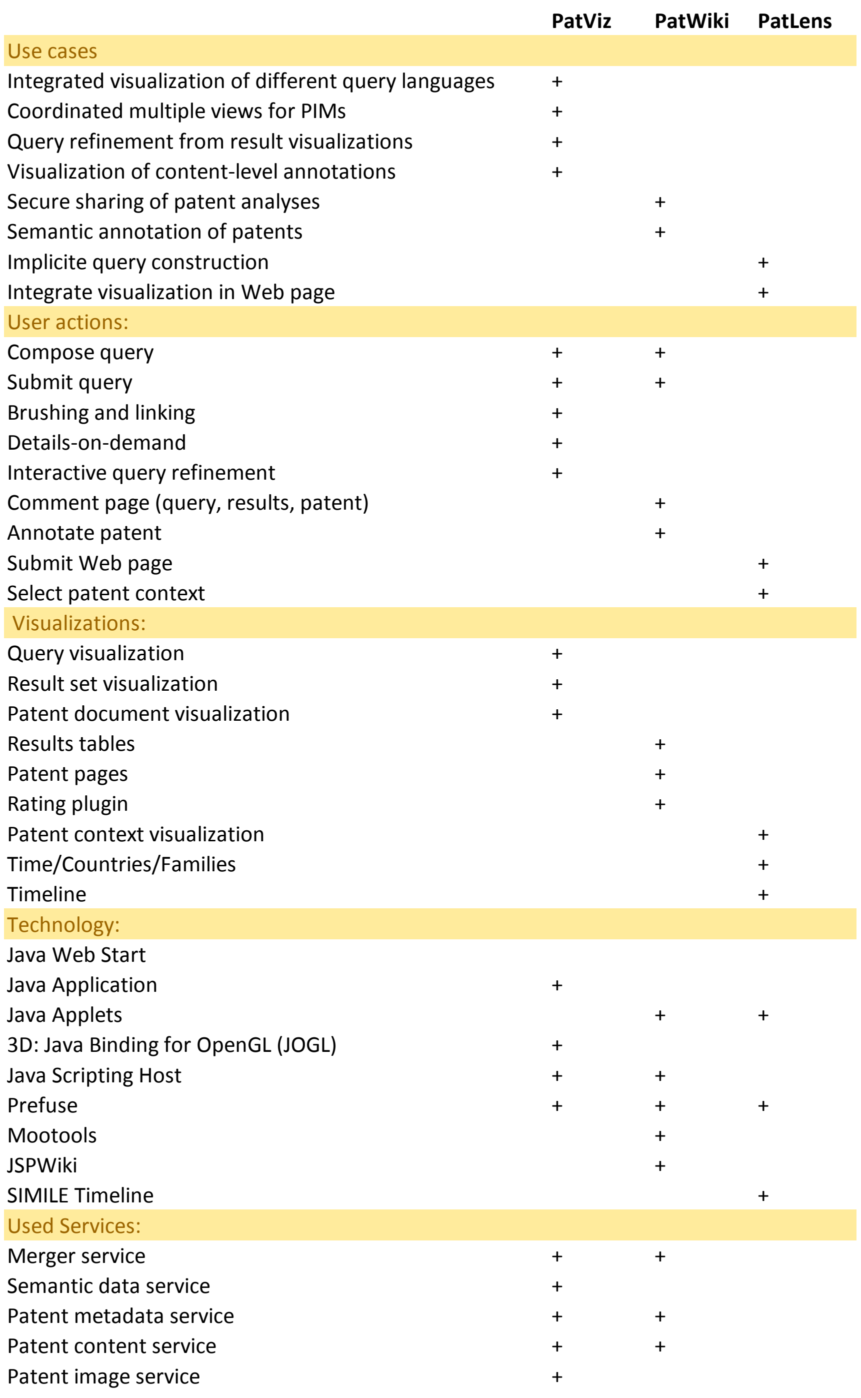
Figure 9.10: Architecture of the PatLens prototype

The normal flow is as follows. The user browses a Web site and wants to check if there is additional information available for the current context. For this purpose the user sends the current Web page to a *Page Analyzer* by using an active bookmark. The analyzer compares the page content with available resources stored in a *Patent Information Repository* . For example it looks for inventor and applicant names, keywords, or searches for number schemes such as IPC codes or patent document identifiers. Based on this page analysis the original *Web Page* is transformed into an *Augmented Web Page* containing the found search contexts. A *Search Context* is a special query for all information associated with one resource as the context, for example all information associated with one patent document or all information associated with one applicant. A *Search Context* therefore is a simplified *Query Model*.

When the user selects a highlighted area on the augmented page indicating a context for which the system provides additional information, then a lens overlay is created. The overlay is parameterized so that the *Visual User Interface* is loaded as applet. In the initialization of the applet the *Search Context* is transferred to the server. This triggers the *Orchestrator* which creates at *Patent Information Model*. The model is send back to the applet as response of the initial request. Finally the model gets visualized in the applet according to section 9.2.3.

9.3.4 Overview

The following table gives an overview of the prototypes.



	PatViz	PatWiki	PatLens
Use cases			
Integrated visualization of different query languages	+		
Coordinated multiple views for PIMs	+		
Query refinement from result visualizations	+		
Visualization of content-level annotations	+		
Secure sharing of patent analyses		+	
Semantic annotation of patents		+	
Implicite query construction			+
Integrate visualization in Web page			+
User actions:			
Compose query	+	+	
Submit query	+	+	
Brushing and linking	+		
Details-on-demand	+		
Interactive query refinement	+		
Comment page (query, results, patent)		+	
Annotate patent		+	
Submit Web page			+
Select patent context			+
Visualizations:			
Query visualization	+		
Result set visualization	+		
Patent document visualization	+		
Results tables		+	
Patent pages		+	
Rating plugin		+	
Patent context visualization			+
Time/Countries/Families			+
Timeline			+
Technology:			
Java Web Start			
Java Application	+		
Java Applets		+	+
3D: Java Binding for OpenGL (JOGL)	+		
Java Scripting Host	+	+	
Prefuse	+	+	+
Mootools		+	
JSPWiki		+	
SIMILE Timeline			+
Used Services:			
Merger service	+	+	
Semantic data service	+		
Patent metadata service	+	+	
Patent content service	+	+	
Patent image service	+		

Figure 9.11: Feature overview of the prototypes

9.4 Conclusions

In this chapter a general architecture for visual patent analysis has been proposed. The four main objectives for this architecture have been: (i) using a semantic representation as underlying data model, (ii) incorporating manual and automatic patent analyses, (iii) providing interactive visualizations of patent information and patent analysis results, and (iv) providing a security mechanism for semantic annotations in open Web environments. The architecture has been implemented by three prototypes, each focusing on different use cases as summarized in the previous section.

The first objective has been realized by using Semantic Web standards (section 2.2) for semantic representation of patent information. The fact that merging of different schemes is a key capability of RDF, has led to a blackboard oriented architecture design with the patent information model as central data structure. Patent data coming from different services are merged in this model. In addition, this model has been designed to integrate the results of (automatic) patent analysis services or (manual) annotations. Concerning visual user interfaces, using a model that is based on the open world assumption has turned out to be flexible and useful, since additional data structures could transparently be added without changing existing visual mappings.

When looking at the quality characteristics described in section 2.6.2, the general emphasis has been on: (i) Modifiability: the system behavior can be changed by adding new services and changing the wiring in the orchestrator; (ii) Interoperability: the system can exchange data with other systems using services and adapters; (iii) Usability: the system is encompassed to various users characteristics by providing multiple graphical views on patent data and analysis results; (iv) Confidentiality: the system protects the user's data against threats by providing an encryption service for semantic annotations.

One basic assumption has been that patent analysis is conducted by an expert user who builds up his or her private analysis model within the context of one analysis. This model can be stored (PatViz) or can be shared (PatWiki). Furthermore, models can be merged in one or multiple repositories.

In this context, aspects of provenance (where do the assertions come from?), trust (can I trust these assertions?) and security (only a certain set of addressees are allowed to see my assertions.) play an important role. A possible realization of provenance is using named graphs (section 2.2.1). The trust aspect typically is carried out by signatures, whereas security aspects can be realized with the partial RDF encryption method proposed in chapter 4. Additionally, for realizing trust and security aspects it is important to set up a public key infrastructure.

Furthermore, the scalability of a system that implements the proposed architecture heavily correlates with the scalability of the used patent data and analysis services. Since the orchestration is isolated per user, the system can be scaled by adding parallel processes to be able to increase the number of parallel users.

Concerning scalability of the visual user interface one can say that most visualizations have an upper data limit, and if this limit is exceeded the usability of the visualization is reduced. For example, as mentioned in section 5.3, the upper limit of the proposed graph approach is somewhere at thousand nodes, whereas the upper limit of statistical charts can be significantly higher.

CHAPTER 10

Results and Outlook

Patents are exclusive rights granted by a sovereign state in exchange for the public disclosure of an invention. This makes patents a valuable source for technological knowledge. But patent analysis is a complex, time and knowledge intensive process. This thesis has investigated the question of how patent analysis can be supported by techniques and methods from the field of visual analytics. The results of this thesis are a general architecture for visual patent analysis, an ontology for patent metadata, new visualization techniques for patent metadata, and a new partial encryption based method for secure sharing of semantic models.

1. A general architecture for visual patent analysis has been proposed which is based on a semantic representation model. The model functions as a blackboard and can be accessed from analysis and presentation components, enriched with analysis results, published and reused. Analysis and presentation components are loosely coupled and exchangeable based a shared terminology defined by ontologies.

   A fundamental assumption for the proposed architecture is that each patent analysis is conducted by an expert user who builds up his or her private model within the context of a single analysis for a defined purpose. This model can then be stored, shared, and merged in one or more repositories. Based on this architecture the PatViz, PatLens and PatWiki prototypes (section 9.3) have been implemented each focusing on a different research aspect as described in the section before. It could be shown that the proposed architecture is flexible and extensible to be used in different scenarios and to be used with different technologies.

2. For semantic representation of patent metadata an ontology-based approach has been described. The *Patent Metadata Ontology (PMO)* provides a detailed terminology for all major patent metadata aspects. PMO is based on Semantic Web standards, in particular on OWL as ontology language. Because patent metadata are an integral part of almost every patent search task (see section 2.1.3), it is important to integrate them into an overall semantic representation framework in order to allow the formulation of metadata restrictions.

   PMO fills this gap in the PATExpert ontology framework. It further provides a mapping to the terms used in the WIPO ST.32 and ST.36 standards (section 2.1.1) and thus promotes practicability and understandability for common use. For efficiency reasons, PMO uses the OWL RDFS/Plus subset (section 2.2.3) as ontology language which reduces the amount of inferred assertions but on the other hand also reduces reasoning capabilities. PMO has been divided into four ontology modules and is easily extensible.

3. New visualization techniques have been described for resource level and content level patent metadata. Concerning the visualization of resource level patent metadata, the focus has been on bibliographic relations between patents, temporal, geographic and classificatory distributions. A central new feature is a SPARQL based dynamic visual mapping. Two different visual mapping approaches have been described in section 5.2. The first uses SPARQL construct statements in order to map PMO fragments onto a *Graph Visualization Ontology*. The second approach supports a simple and fast visual mapping by interpreting the variables of SPARQL select statements for the generation of hierarchies.

   Concerning the visualization of content level patent metadata, a graph overlay based technique for visualizing dependencies between text parts based on semantic relations has been proposed. Semantic relations are expressed using a linking approach with custom *token* schema for XPointers which also provides the capability to annotate contents to which no write access is available, such as published patent information.

4. A new method for partially encrypting RDF graphs has been developed for secure sharing of semantic models to a known group of recipients. The main idea of this method is to only encrypt sensitive information while keeping all non-sensitive

information publicly readable (partial encryption). This method allows the usage of open infrastructures for the exchange of sensitive semantic models.

It differs from other approaches in that the result is a single self describing RDF compliant graph containing both, encrypted data and plaintext data. The method allows for fine grained encryption of RDF subjects, objects, predicates and subgraphs. Encrypted fragments are included as XML literals which are represented using the XML-Encryption and XML-Signature recommendations. Graph transformations have been described that are necessary to keep the encrypted RDF graph well-formed. The proposed method is adoptable for different algorithms and processing rules by using encryption metadata.

This method can also be the foundation for new business models, for example by encrypting the results of analyses in order to sell them to paying customers that receive an appropriate decryption key while non-paying customers are only able to see the public information.

Evaluation

To show the suitability of the visual interface a first 'think-aloud' evaluation with patent specialists has been carried out by Koch et al. [170]. A problem in this context were the limitations in PATExpert to the domains 'optical recording' and 'machine tools' on the one hand and the limited subset of patents for each of those domains on the other. This made it difficult to find a representative number of external patent specialists for evaluating the visual interfaces with realistic queries during their daily work. Therefore patent specialists from the PATExpert consortium were asked to take part in the evaluation. Because of this, the validity of the evaluation obviously is limited. But it never the less gives some first expert feedback.

In general, all users agreed that the visual interface is a valuable means for creating and editing complex queries for different search engines. Further, the synergetic effects of using different views of the same set in parallel were appreciated by the users. The users also commented positively on the variability and power of the system resulting from the degrees of freedom in moving back and forth in the stages of the analysis process and between different perspectives within one stage of the process. An important benefit identified by the expert users was the support for iterative refinement of queries. Brushing and linking was extensively used after the system had become more familiar.

But further evaluations in this context are necessary, including ontology evaluations and usability tests. The results of this thesis can be seen as a step towards visual patent analysis. Directions for future research include improving scalability at various levels and support for collaborative patent analysis. Both aspects are briefly discussed below.

Improving Scalability

An important issue for all visual analytics applications is *visual scalability*, i.e. „the capability of visualization representations and visualization tools to display massive data sets effectively" (Thomas and Cook [239, p. 91]). Although displays get larger from year to year flow of information is increasing faster. Therefore, information reduction techniques, including Latent Semantic Indexing, Multidimensional Scaling or Self-Organizing Maps, can improve the visualization of massive document collections by focusing on the information subset that is relevant in user context. An ongoing research project in the area is the DFG Priority Program 1335 Scalable Visual Analytics [30] with special focus on high-dimensional visual analytics, visual geo-temporal analytics, and visual document analytics.

Another important aspect for visual patent analysis is the aspect of *information scalability*, i.e. „the capability to extract relevant information from massive data streams" (ibid. 91). Since patent information is more document oriented rather than stream oriented, information scalability in this context has to deal with extracting relevant semantic information from large document collections. For patent metadata which are also contained or associated with patent documents, rules could be provided for making implicit knowledge explicit. As an example, the current legal status of a patent could be determined by rules on the document and legal events published by patent offices.

Related to information scalability is *analytic scalability*, i.e. „the capability of the mathematical algorithms to efficiently accommodate large data sets" (ibid. 91). This aspect has not been considered in this thesis, since the focus was not on concrete analysis techniques. By defining analytical reasoning as a service, algorithms have been abstracted and are exchangeable. This exchangeability is also an important feature regarding the *software scalability*, i.e. „the capability of the software to accommodate data sets of varying sizes" (ibid. 91). In this context, also techniques that use graphics processing units (GPUs) rather than central processing units (CPUs) may play an important role for speeding up visualization and analysis by parallelizing computations.

Collaborative Patent Analysis

The focus within this thesis was on patent analysis conducted by a single user building up a private model within the context of a single analysis for a defined purpose. Since patent analysis often is carried out by a group of experts, further research in the area of collaborative analysis is necessary. New hardware, including large displays (e.g., tiled LCD walls or projector arrays) and input devices (e.g., Microsoft's motion sensing device Kinect), will have influences on the future developments in this area. But also new tablet hardware may play an important role for collaborative patent analysis, for example when put together to one virtual screen in an online meeting. An example of using PatViz on a large display is shown in figure 10.1. Experts could discuss their findings face to face. Challenges in this regard are usable interaction techniques, for example for interacting with large visualizations or for documenting the findings.

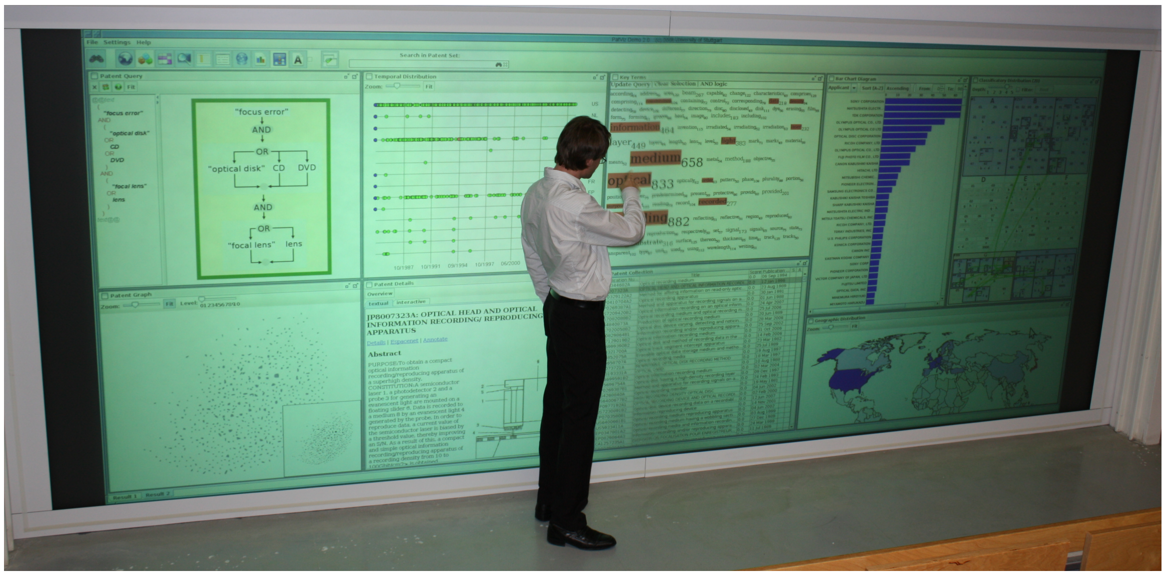


Figure 10.1: Patent analysis on large displays

Additionally, provenance and trust aspects are important issues in remote collaboration scenarios, in which the users cannot communicate face to face. Provenance information is necessary in order to validate where assertions come from and if they are trusted or not. With evolving analysis techniques also security and privacy issue become more and more important or as stated by Thomas and Cook: „The most innovative analytical tools can be ineffective or even harmful if implemented and deployed without regard to security and privacy considerations" (Thomas and Cook [239, p. 157]).

Bibliography

[1] AllegroGraph. Webpage, last visit December 10, 2012.
http://www.franz.com/agraph/allegrograph/.

[2] ARQ. Webpage, last visit December 10, 2012.
http://jena.sourceforge.net/ARQ/documentation.html.

[3] Continuing Patent Applications - What Are They & When Are They Appropriate.
http://www.invention-protection.com/ip/publications/docs/Continuing_Patent_Applications.html.

[4] Curves API (CAPI) Library. Webpage, last visit December 10, 2012.
http://sourceforge.net/projects/curves/.

[5] ECLA - European Classification. Webpage, last visit December 10, 2012.
http://v3.espacenet.com/eclasrch.

[6] Esp@cenet. Webpage, last visit: December 10, 2012.
http://ep.espacenet.com.

[7] European Publication Server. Webpage, last visit: 30.3.2011.
https://publications.european-patent-office.org/.

[8] FI/F-Term Search. Webpage, last visit December 10, 2012. http://www4.ipdl.inpit.go.jp/Tokujitu/tjftermena.ipdl?N0000=114.

[9] Freepatentsonline. Webpage, last visit: 30.3.2011.
http://www.freepatentsonline.com/.

[10] Google Rich Snippets. Webpage, last visit December 10, 2012.
http://www.google.com/support/webmasters/bin/topic.py?topic=21997.

[11] INPADOC Legal Status Database. Webpage, last visit December 10, 2012.
http://www.epo.org/searching/subscription/raw/product-14-11.html.

[12] International Patent Classification (Version 2011) Guide. http://www.wipo.int/export/sites/www/classifications/ipc/en/guide/guide_ipc.pdf.

[13] IsaViz. Webpage, last visit: 30.3.2011.
http://www.w3.org/2001/11/IsaViz/.

[14] Jastor Project Homepage. Webpage, last visit: 30.3.2011.
http://jastor.sourceforge.net/.

[15] Java bindings for OpenGL (JOGL). Webpage, last visit December 10, 2012.
http://java.net/projects/jogl/.

[16] JSPWiki. Webpage, last visit: 30.3.2011.
http://www.jspwiki.org/.

[17] LENA - RDF/Linked Data Browser. Webpage, last visit December 10, 2012.
http://code.google.com/p/lena/.

[18] Longwell Browser. Webpage, last visit December 10, 2012.
http://simile.mit.edu/wiki/Longwell.

[19] Mootools. Webpage, last visit: 30.3.2011.
http://mootools.net/.

[20] Online RDFa Parser. Webpage, last visit December 10, 2012.
http://rdf-in-html.appspot.com/.

[21] Open Patent Services (OPS). Webpage, last visit: 30.3.2011.
http://www.epo.org/searching/free/ops_de.html.

[22] OWLIM. Webpage, last visit December 10, 2012.
http://www.ontotext.com/owlim.

[23] Patent Lens. Webpage, last visit: 30.3.2011.
http://www.patentlens.net.

[24] PATExpert Project. Webpage, last visit December 10, 2012.
http://www.patexpert.org.

[25] Peer To Patent. Webpage, last visit December 10, 2012.
http://www.peertopatent.org/.

[26] Provisional Application for Patent brochure. Webpage, last visit December 10, 2012.
http://www.uspto.gov/patents/resources/types/provapp.jsp.

[27] RDF Encryption Project Homepage. Webpage, last visit December 10, 2012.
http://sourceforge.net/projects/pre4j/.

[28] RDFa Bookmarklets. Webpage, last visit December 10, 2012.
http://www.w3.org/2006/07/SWD/RDFa/impl/js/.

[29] SIMILE Timeline Widged. Webpage, last visit: December 10, 2012.
http://simile.mit.edu/timeline.

[30] SPP - Scalable Visual Analytics. Webpage, last visit December 10, 2012.
http://www.visualanalytics.de/.

[31] Unified Modeling Language (UML). Webpage, last visit December 10, 2012.
http://www.uml.org/.

[32] United States Patent Classification (USPC).
http://www.uspto.gov/go/classification/.

[33] Virtuoso RDF Data Store. Webpage, last visit December 10, 2012.
http://virtuoso.openlinksw.com/rdf-quad-store/.

[34] W3C Semantic Web Standards. Webpage, last visit December 10, 2012.
http://www.w3.org/standards/semanticweb/.

[35] Welkin RDF Browser. Webpage, last visit: 30.3.2011.
http://simile.mit.edu/welkin/.

[36] WikiPatents. Webpage, last visit:30.3.2011.
http://www.wikipatents.com/.

[37] WIPO Standards. Webpage, last visit: December 10, 2012.
http://www.wipo.int/standards/en/part_03_standards.html.

[38] The MD5 Message-Digest Algorithm. RFC 1321, April 1992.
http://www.faqs.org/rfcs/rfc1321.html.

[39] Data Encryption Standard (DES). FIPS Publication 46-3, October 1999. http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf.

[40] Advanced Encryption Standard (AES). FIPS Publication 197, November 2001. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[41] Secure Hash Standard. FIPS Publication 180-2, April 2002. http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf.

[42] Extensible Metadata Platform (XMP). Webpage, last visit: 30.3.2011, January 2004. http://www.adobe.com/devnet/xmp.html.

[43] Jena Project. Webpage, last visit December 10, 2012, 2004. http://jena.sourceforge.net/.

[44] OWL Web Ontology Language Guide, February 2004.

[45] RDQL - A Query Language for RDF. W3C Member Submission, January 2004. http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/.

[46] A Universally Unique IDentifier (UUID) URN Namespace. RFC 4122, July 2005. http://www.ietf.org/rfc/rfc4122.txt.

[47] DCMI Metadata Terms, October 2010. http://dublincore.org/documents/dcmi-terms/.

[48] FOAF Vocabulary Specification 0.98. Webpage, last visit December 10, 2012, August 2010. http://xmlns.com/foaf/spec/.

[49] EPO Worldwide Patent Statistical Database (PATSTAT), 2011. http://www.epo.org/searching/subscription/raw/product-14-24_de.html.

[50] J. Abello, J. Korn, and I. Finocchi. Graph Sketches. In *IEEE Symposium on Information Visualization 2001*, INFOVIS '01, pages 67–72. IEEE Computer Society, 2001.

[51] J. Abello and F. van Ham. Matrix Zoom: A Visual Interface to Semi-External Graphs. In *IEEE Symposium on Information Visualization*, INFOVIS '04, pages 183–190. IEEE Computer Society, 2004.

[52] C. Ahlberg, C. Williamson, and B. Shneiderman. Dynamic Queries for Information Exploration: An Implementation and Evaluation. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, 1992.

[53] W. Aigner, S. Miksch, H. Schumann, and C. Tominski. *Visualization of Time-Oriented Data*. Human-Computer Interaction Series. Springer, 2011.

[54] R. Albertoni, A. Bertone, and M. D. Martino. Semantic Web and Information Visualization. In *Proceedings of the 1st Italian Workshop on Semantic Web Applications and Perspectives*, 2004.

[55] D. Allemang and J. Hendler. *Semantic Web for the Working Ontologist – Effective Modeling in RDFS and OWL*. Morgan Kaufman, 2008.

[56] R. Amar, J. Eagan, and J. T. Stasko. Low-Level Components of Analytic Activity in Information Visualization. In *IEEE Symposium on Information Visualization*, InfoVis '05, pages 111–117. IEEE Computer Society, 2005.

[57] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.

[58] C. Bailey, S. R. El-Beltagy, and W. Hall. Link Augmentation: A Context-Based Approach to Support Adaptive Hypermedia. In *Proceedings of the OHS-7/SC-3/AH-3*, pages 239–251, 2001.

[59] M. Q. W. Baldonado, A. Woodruff, and A. Kuchinsky. Guidelines for Using Multiple Views in Information Visualization. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, pages 110–119. ACM, 2000.

[60] M. Balzer and O. Deussen. Hierarchy based 3D Visualization of Large Software Structures. In *Poster at IEEE Visualization (Vis)*, 2004.

[61] M. Balzer and O. Deussen. Voronoi Treemaps. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis)*, 2005.

[62] J. Barnes and P. Hut. A Hierarchical O(NlogN) Force-Calculation Algorithm. *Nature*, 324(6096):446–449, 1986.

[63] A. Basilevsky. *Statistical Factor Analysis and Related Methods: Theory and Applications*. John Wiley and Sons, 1994.

[64] L. Bauer, M. Schneider, and E. Felten. A General and Flexible Access-Control System for the Web. In *Proceedings of the 11th USENIX Security Symposium*, San Francisco, CA, Aug 2002.

[65] K. Beard, H. Deese, and N. R. Pettigrew. A Framework for Visualization and Exploration of Events. *Information Visualization*, 7(2):133–151, 2008.

[66] R. A. Becker and W. S. Cleveland. Brushing Scatterplots. *Technometrics*, 29(2):127–142, May 1987.

[67] B. Bederson. Fisheye Menus. In *Proceedings of ACM Conference on User Interface Software and Technology (UIST 2000)*, pages 217–225. ACM Press, 2000.

[68] K. Benzineb and J. Guyot. Automated Patent Classification. In M. Lupu, K. Mayer, J. Tait, and A. J. Trippe, editors, *Current Challenges in Patent Information Retrieval*, volume 29 of *THE INFORMATION RETRIEVAL SERIES*, pages 239 – 263. Springer, 2011.

[69] T. Berners-Lee, R. Fielding, and L. Masinter. RFC 3986 – Uniform Resource Identifiers (URI): Generic Syntax. Request for Comments, Januar 2005.
http://www.ietf.org/rfc/rfc3986.txt.

[70] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, May 2001.

[71] J. Bertin. *Semiology of graphics: Diagrams, Networks, Maps*. University of Wisconsin Press, 1967/1983. Übersetzung von William J. Berg.

[72] S. Bettray. Patent Maps – Landkarten der Innovation. *Wissensmanagement*, 5:50–51, 2008.

[73] E. Bier, M. Stone, K. Pier, W. Buxton, and T. DeRose. Toolglass and magic lenses: the see-through interface. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 73–80, 1993.

[74] M. Bishop. *Introduction to Computer Security*. Pearson Education, 2005.

[75] C. Bizer, T. Heath, and T. Berners-Lee. Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems*, 5(3):1–22, 2009.

[76] C. Bizer, R. Lee, and E. Pietriga. Fresnel - A Browser-Independent Presentation Vocabulary for RDF. In *Proceedings of the End User Semantic Web Interaction Workshop at the ISWC 2005*, 2005.

[77] H. Bosch. Interaktive graphbasierte Filterung von Ergebnismengen. Master's thesis, Universität Stuttgart, Institut für Visualisierung und Interaktive Systeme, 2007.

[78] K. Boyack, B. Wylie, G. Davidson, and D. Johnson. Analysis of Patent Databases using VxInsight. In *Workshop on New Paradigms in Information Visualization and Manipulation*, 2000.

[79] K. W. Boyack, B. N. Wylie, and G. S. Davidson. Domain Visualization Using VxInsight for Science and Technology Management. *Journal of the American Society for Information Science and Technology*, 53(9):764–774, Aug. 2002.

[80] J. Breuker and R. Hoekstra. Epistemology and ontology in core ontologies: FOLaw and LRI-Core, two core ontologies for law. In *Proceedings of EKAW Workshop on Core ontologies*, pages 15–27, 2004.

[81] D. Brickley and L. Miller. FOAF Vocabulary Specification 0.98 (Marco Polo Edition), August 2010. `http://xmlns.com/foaf/spec/`.

[82] K. Börner, C. Chen, and K. Boyack. Visualizing Knowledge Domains. In B. Cronin, editor, *Annual Review of Information Science & Technology*, volume 37, chapter 5. Information Today Inc., 2003.

[83] S. Brockmans, R. Volz, A. Eberhart, and P. Löffler. Visual Modeling of OWL DL Ontologies using UML. In *The Semantic Web – ISWC 2004*, pages 198–213. Springer, 2004.

[84] M. Bruls, K. Huizing, and J. van Wijk. Squarified Treemaps. In *Proc. of Joint Eurographics and IEEE TCVG Symp. on Visualization*, pages 33–42. IEEE Press, 2000.

[85] J. Buchmann. *Einführung in die Kryptographie*. Springer Verlag, 3rd edition, 1999.

[86] S. Card and J. Mackinlay. The structure of the information visualization design space. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 92–99, Oct. 1997.

[87] S. Card, J. Mackinlay, and B. Schneiderman, editors. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufman, 1999.

[88] J. V. Carlis and J. A. Konstan. Interactive Visualization of Serial Periodic Data. In *Proceedings of the 11th Annual ACM Symposium on User Interface Software and Technology*, UIST '98, pages 29–38. ACM, 1998.

[89] M. S. T. Carpendale, D. J. Cowperthwaite, and F. D. Fracchia. Extending Distortion Viewing from 2D to 3D. *IEEE Comput. Graph. Appl.*, 17(4):42–51, 1997.

[90] J. Carroll, C. Bizer, P. Hayes, and P. Stickler. Named Graphs, Provenance and Trust. Technical report, HP Laboratories Bristol, 2004.

[91] J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson. Jena: Implementing the Semantic Web Recommendations. Technical report, HP Labs Semantic Web Research, 2003.

[92] J. J. Carroll, C. Bizer, P. Hayes, and P. Stickler. Named Graphs, Provenance and Trust. In *Proceedings of the 14th International Conference on World Wide Web*, WWW '05, pages 613–622. ACM, 2005.

[93] G. Cascini and M. Zini. Measuring patent similarity by comparing inventions functional trees. In G. Cascini, editor, *Proceedings of the Conference on Computer-Aided Innovation (CAI)*, volume 277 of *IFIP International Federation for Information Processing*, pages 31–42. Springer Boston, 2008.

[94] M. Chau. Visualizing Web Search Results Using Glyphs: Design and Evaluation of a Flower Metaphor. *ACM Trans. Manage. Inf. Syst.*, 2(1):2:1–2:27, 2011.

[95] C. Chen. *Information Visualization Beyond the Horizon*. Springer, 2nd edition, 2006.

[96] C. Chen and R. Paul. Visualizing a Knowledge Domain's Intellectual Structure. *Computer*, 34(3):65–71, 2001.

[97] E. H.-H. Chi and J. Riedl. An Operator Interaction Framework for Visualization Systems. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 63–70, Oct 1998.

[98] M. C. Chuah and S. G. Eick. Information Rich Glyphs for Software Management Data. *IEEE Comput. Graph. Appl.*, 18(4):24–29, 1998.

[99] A. Cockburn, A. Karlson, and B. B. Bederson. A Review of Overview+Detail, Zooming, and Focus+Context Interfaces. *ACM Comput. Surv.*, 41(1):2:1–2:31, 2009.

[100] J. Codina, E. Pianta, S. Vrochidis, and S. Papadopoulos. Integration of Semantic, Metadata and Image Search Engines with a Text Search Engine for Patent Retrieval. In S. Bloehdorn, M. Grobelnik, P. Mika, and D. T. Tran, editors, *SemSearch*, CEUR Workshop Proceedings, 2008.

[101] C. Collins and S. Carpendale. VisLink: Revealing Relationships Amongst Visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1192–1199, 2007.

[102] H. Cunningham, V. Tablan, I. Roberts, M. A. Greenwood, and N. Aswani. Information Extraction and Semantic Annotation for Multi-Paradigm Information Management. In M. Lupu, K. Mayer, J. Tait, and A. J. Trippe, editors, *Current Challenges in Patent Information Retrieval*, volume 29 of *THE INFORMATION RETRIEVAL SERIES*, pages 307 – 329. Springer, 2011.

[103] R. Cyganiak and A. Jentzsch. Linking Open Data Cloud Diagram. Webpage, last visit December 10, 2012. http://lod-cloud.net/.

[104] W. Dehnen. A Hierarchical O(N) Force Calculation Algorithm. *Journal of Computational Physics*, 179:27–42, 2002.

[105] J. Delgado, I. Gallego, S. Llorente, and R. García. IPROnto: An Ontology for Digital Rights Management. In *Proceedings of the 16th Conference on Legal Knowledge and Information Systems (JURIX'03)*, 2003.

[106] J. Delgado, I. Gallego, S. Llorente, and R. García. Regulatory ontologies: An Intellectual Property Rights approach. In *Workshop on Regulatory Ontologies and the Modeling of Complaint Regulations (WORM CoRe 2003)*, 2003.

[107] G. Di Battista, P. Eades, R. Tamassia, and I. Tollis. *Graph drawing: algorithms for the visualization of graphs*. Prentice Hall, 1999.

[108] L. Ding and T. Finin. Characterizing the Semantic Web on the Web. In *Proceedings of the 5th International Semantic Web Conference*, November 2006.

[109] M. Endrei, J. Ang, A. Arsanjani, S. Chua, P. Comte, P. Krogdahl, D. M. Luo, and T. Newling. *Patterns: Service-Oriented Architecture and Web Services*. IBM Corp., 2004.

[110] C. Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

[111] R. T. Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, UNIVERSITY OF CALIFORNIA, 2000.

[112] F. Frasincar, A. Telea, and G. Houben. Adapting graph visualization techniques for the visualization of RDF data. In V. Geroimenko and C. Chen, editors, *Visualizing the Semantic Web*. Springer, 2nd edition, 2006.

[113] T. Fruchterman and E. Reingold. Graph Drawing by Force-Directed Placement. *Software-Practice and Experience*, 21(11):1129–1164, November 1991.

[114] G. W. Furnas. Generalized fisheye views. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 16–23, New York, NY, USA, 1986. ACM Press.

[115] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.

[116] A. Gangemi. Ontology Design Patterns for Semantic Web Content. In Y. G. et al., editor, *ISWC 2005*, pages 262–276. Springer, 2005.

[117] A. Gangemi, C. Catenacci, M. Ciaramita, and J. Lehmann. Ontology Evaluation and Validation: An Integrated Formal Model for the Quality Diagnostic Task. Technical report, Laboratory for Applied Ontology, ISTC-CNR, Roma/Trento (Italy), 2005. http://www.loa.istc.cnr.it/Files/OntoEval4OntoDev_Final.pdf.

[118] A. Gangemi, N. Guarino, c. Masolo, A. Oltramari, and L. Schneider. Sweetening Ontologies with DOLCE. In *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management*, pages 166–181. Springer-Verlag, 2002.

[119] Gapminder Foundation. Trendanalyzer. Website, last visit December 10, 2012, 2010. http://www.gapminder.org/world/.

[120] D. Garlan and M. Shaw. An introduction to Software Architecture. In *Advances in Software Engineering and Knowledge Engineering*, pages 1–39. Publishing Company, 1993.

[121] N. Ghoula, K. Khelif, and R. Dieng-Kuntz. Supporting Patent Mining by using Ontology-based Semantic Annotations. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, 2007.

[122] M. Giereth. On Partial Encryption of RDF-Graphs. In *The Semantic Web - ISWC 2005*, volume 3729 of *Lecture Notes in Computer Science*, pages 308–322. Springer, 2005.

[123] M. Giereth. Partial RDF Encryption as a Method for Addresse Oriented Publishing in the Semantic Web (Poster). In *Poster and Demo Proceedings of the 2nd European Semantic Web Conference*, Heraklion, Greece, June 2005.

[124] M. Giereth. PRE4J - A Partial RDF Encryption API for Jena. In *Proceedings of the 1st Jena User Conference*, Bristol, UK, May 2006.

[125] M. Giereth, H. Bosch, and T. Ertl. A 3D Treemap Approach for Analyzing the Classificatory Distribution in Patent Portfolios (Poster). In *Posters of the IEEE Symposium on Visual Analytics Science and Technology (VAST 2008)*, 2008.

[126] M. Giereth and T. Ertl. Design Patterns for Rapid Visualization Prototyping. In *Proceedings of the 12th International Conference on Information Visualisation (IV 2008)*, pages 569–574. IEEE Computer Society, 2008.

[127] M. Giereth and T. Ertl. Visualization Enhanced Semantic Wikis for Patent Information. In *Proceedings of the 12th International Conference on Information Visualisation (IV 2008)*, pages 185–190. IEEE Computer Society, 2008.

[128] M. Giereth, S. Koch, H. Bosch, and T. Ertl. Visual Patent Retrieval. In E. Schweighofer, A. Geist, G. Heindl, and C. Szücs, editors, *Komplexitätsgrenzen der Rechtsinformatik, Tagungsband des 11. int. Rechtsinformatik Symposiums (IRIS) 2008*. Boorberg, 2008.

[129] M. Giereth, S. Koch, and T. Ertl. D6.2 Advanced Visualization and Navigation Techniques. Technical report, PATExpert Consortium, 2007.

[130] M. Giereth, S. Koch, Y. Kompatsiaris, SymeonPapadopoulos, E. Pianta, L. Serafini, and L. Wanner. A Modular Framework for Ontology-based Representation of Patent Information. In A. Lodder and L. Mommers, editors, *Legal Knowledge and Information Systems - JURIX 2007: The Twentieth Annual Conference*, volume 165 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2007.

[131] M. Giereth, S. Koch, M. Rotard, and T. Ertl. Web Based Visual Exploration of Patent Information. In *Proceedings of the 11th International Conference on Information Visualisation (IV 2007)*, pages 150–155. IEEE Computer Society, 2007.

[132] M. Giereth and Y. Qian. Testing the Inferability of RDF-Data for Secure Partial Encryption (Poster). In *Poster and Demonstration Proceedings of the 4nd International Semantic Web Conference*, Galway, Ireland, November 2005.

[133] M. Giereth, A. Stäbler, S. Brügmann, M. Rotard, and T. Ertl. Application of Semantic Technologies for Representing Patent Metadata. In *Informatik 2006 - Informatik für Menschen, Band 2, Beiträge der 36. Jahrestagung der Gesellschaft für Informatik e.V. (GI)*, pages 297–304, 2006.

[134] M. Giereth, M. Wörner, H. Bosch, P. Baier, and T. Ertl. Utilization of Semantic Annotations in Interactive User Interfaces for Large Documents. In *INFORMATIK 2008, Beherrschbare Systeme - Dank Informatik, Band 2, Beiträge der 38. Jahrestagung der Gesellschaft für Informatik e.V. (GI)*, pages 706–711. Springer, 2008.

[135] A. Goeb and K. Lochmann. A Software Quality Model for SOA. In *Proceedings of the 8th International Workshop on Software Quality*, WoSQ '11, pages 18–25. ACM, 2011.

[136] B. C. Grau, I. Horrocks, B. Motik, B. Parsia, P. F. Patel-Schneider, and U. Sattler. OWL 2: The next step for OWL. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(4):309–322, 2008.

[137] T. R. Gruber. A Translation Approach to Portable Ontologies. *Knowledge Acquisition*, 5(2):199–220, 1993.

[138] R. Haber and D. McNabb. *Visualization in Scientific Computing*, chapter Visualization Idioms: A Conceptual Model for Scientific Visualization Systems, pages 74–93. IEEE Computer Society Press, 1990.

[139] R. L. Harris. *Information Graphics: A Comprehensive Illustrated Reference*. Oxford University Press, Inc., 1999.

[140] T. Heath and C. Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Synthesis Lectures on the Semantic Web: Theory and Technology. Morgan & Claypool, 2011.

[141] J. Heer and M. Agrawala. Software Design Patterns for Information Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):853–860, 2006.

[142] J. Heer, S. K. Card, and J. A. Landay. Prefuse: A Toolkit for Interactive Information Visualization. In *Proceedings of the CHI*, 2005.

[143] P. Heim and J. Ziegler. Faceted Visual Exploration of Semantic Data. In *Proceedings of the 2nd Workshop on Semantic Models for Adaptive Interactive Systems (SEMAIS)*, 2009.

[144] J. Heinrich and D. Weiskopf. Continuous Parallel Coordinates. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1531–1538, 2009.

[145] N. Henry and J.-D. Fekete. MatrixExplorer: A Dual-Representation System to Explore Social Networks. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):677–684, 2006.

[146] I. Herman, G. Melancon, and M. S. Marshall. Graph Visualization and Navigation in Information Visualization: A Survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24–43, Jan 2000.

[147] P. Hitzler, M. Krötzsch, and S. Rudolph. *Foundations of Semantic Web Technologies*. Chapmann and Hall, 2010.

[148] D. Holten. Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):741–748, 2006.

[149] S. Hong. The Magic of Patent Information. Webpage, last visit December 10, 2012, 2006.

[150] D. Hunt, L. Nguyen, and M. Rodgers, editors. *Patent Searching: Tools & Techniques*. John Wiley & Sons, 2007.

[151] E. L. Hutchins, J. D. Hollan, and D. A. Norman. Direct Manipulation Interfaces. *Human-Computer Interaction*, 1(4):311–338, 1985.

[152] K. Indukuri, A. Ambekar, and A. Sureka. Similarity Analysis of Patent Claims Using Natural Language Processing Techniques. In *Proceedings of the Conference on Computational Intelligence and Multimedia Applications*, pages 169 – 175, 2007.

[153] A. Inselberg and B. Dimsdale. Parallel Coordinates: A Tool for Visualizing Multi-Dimensional Geometry. In *Proceedings of the 1st Conference on Visualization '90*, pages 361–378. IEEE Computer Society Press, 1990.

[154] P. Irani, D. Slonowsky, and P. Shajahan. Human perception of structure in shaded space-filling. *Information Visualization*, 5(1):47–61, 2006.

[155] ISO/IEC 25010:2011. Systems and Software Engineering – Systems and Software Quality Requirements and Evaluation – System and Software Quality Models, 2011. http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=35733.

[156] ISO/IEC/IEEE 42010:2011. Systems and Software Engineering – Architecture Description., 2011.

[157] A. Jaffe and M. Trajtenberg. *Patents, Citations and Innovocations*. MIT, 2002.

[158] P. Janecek and P. Pu. Opportunistic Search with Semantic Fisheye Views. In *Proceedings of the 5th International Conference on Web Information Systems Engineering (WISE2004)*, pages 668–680, 2004.

[159] P. Janecek, V. Schickel-Zuber, and P. Pu. Concept Expansion Using semantic Fisheye Views. In *Proceedings of the ICADL 2005*, pages 273–282, 2005.

[160] M. John, C. Tominski, and H. Schumann. Visual and Analytical Extensions for the Table Lens. In *Proceedings of Visualization and Data Analysis (VDA)*, 2008.

[161] B. Johnson and B. Shneiderman. Tree-Maps: a space-filling approach to the visualization of hierarchical information structures. In *Proceedings of the 2nd IEEE Conference on Visualization '91*, 1991.

[162] T. Kamada and S. Kawai. An Algorithm for Drawing General Undirected Graphs. *Information Processing Letters*, 31(1):7–15, 1989.

[163] M. Kaufmann and D. Wagner, editors. *Drawing Graphs: Methods and Models*. Springer, 2001.

[164] D. A. Keim. Designing Pixel-Oriented Visualization Techniques: Theory and Applications. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):59–78, 2000.

[165] D. A. Keim. Information Visualization and Visual Data Mining. *IEEE Transactions on Visualization and Computer Graphics*, 7(1):100–107, 2002.

[166] D. A. Keim, G. Andrienko, J. d. Fekete, C. Gorg, J. Kohlhammer, and G. Melancon. Visual Analytics: Definition, Process, and Challenges. Technical report, Dagstuhl Group Report, 2007.

[167] D. A. Keim, F. Mansmann, A. Stoffel, and H. Ziegler. Visual Analytics. In *Encyclopedia of Database Systems*. Springer, 2009.

[168] S. Koch. Intelligente Visualisierung von Patentdokumenten. Diplomarbeit, Universität Stuttgart, Institut für Visualisierung und Interaktive Systeme, 2006.

[169] S. Koch, H. Bosch, M. Giereth, and T. Ertl. Iterative Integration of Visual Insights during Patent Search and Analysis. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (VAST 2009)*, 2009.

[170] S. Koch, H. Bosch, M. Giereth, and T. Ertl. Iterative Integration of Visual Insights during Scalable Patent Search and Analysis. *IEEE Transactions on Visualization and Computer Graphics*, 17(5):557–569, May 2011.

[171] T. Kohonen. *Self-Organizing Maps*. Springer Verlag, 1995.

[172] G. Krasner and S. Pope. A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System. *Journal of Object-Oriented Programming*, 1(3):26–49, 1988.

[173] M. Kreuseler, N. Lopez, and H. Schumann. A Scalable Framework for Information Visualization. In *Proceedings of the IEEE Symposium on Information Vizualization 2000*, INFOVIS '00, pages 27–34. IEEE Computer Society, 2000.

[174] J. B. Kruskal. Multidimensional Scaling and Other Methods for Discovering Structure. In K. Enslein, A. Ralston, and H. Wilf, editors, *Statistical Methods for Digital Computers*. John Wiley and Sons, 1977.

[175] J. B. Kruskal and J. M. Landwehr. Icicle Plot: Better Displays for Hierarchical Clustering. *The American Statistician*, 37(2):162–168, 1983.

[176] D. O. Kutz. Examining the Evolution and Distribution of Patent Classifications. In *Proceedings of the 8th International Conference on Information Visualisation (Iv'04)*. IEEE Computer Society, 2004.

[177] B. Lee, C. S. Parr, C. Plaisant, B. B. Bederson, V. D. Veksler, W. D. Gray, and C. Kotfila. TreePlus: Interactive Exploration of Networks with Enhanced Tree Layouts. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1414–1426, 2006.

[178] V. B. Lenzi, R. Sprugnoli, and E. Pianta. Annotation of Semantic Relations in Patent Documents. In *Proceedings of the 5th International Conference on Generative Approaches to the Lexicon (GL2009)*, Pisa, Italy, 2009.

[179] Y. K. Leung and M. D. Apperley. A Review and Taxonomy of Distortion-Oriented Presentation Techniques. *ACM Transactions on Computer-Human Interaction*, 1(2):126–160, 1994.

[180] M. Ley and P. Reuther. Maintaining an Online Bibliographical Database: The Problem of Data Quality. In *EGC*, pages 5–10, 2006.

[181] W. Li, P. Eades, and N. Nikolov. Using spring algorithms to remove node overlapping. In *Proceedings of the Asia-Pacific Symposium on Information Visualisation (APVIS2005)*, pages 131–140, 2005.

[182] G. Luger. *Künstliche Intelligenz - Strategien zur Lösung komplexer Probleme*. Pearson Studium, 2001.

[183] C. Martínez. Insight into different types of patent families. OECD Working Paper Series, 2010.

[184] W. Müller and H. Schumann. VISUALIZATION METHODS FOR TIME-DEPENDENT DATA – AN OVERVIEW. In S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, editors, *Winter Simulation Conference*, 2003.

[185] S. Mukherjea, B. Bamba, and P. Kankar. Information Retrieval and Knowledge Discovery Utilizing a BioMedical Patent Semantic Web. *IEEE Transactions on Knowledge and Data Engineering*, 17(5):1099–1110, 2005.

[186] I. Niles and A. Pease. Towards a standard upper ontology. In *Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS'01)*, 2001.

[187] I. Niles and A. Pease. Linking Lexicons and Ontologies: Mapping WordNet to the Suggested Upper Merged Ontology. In *International Conference on Information and Knowledge Engineering (IKE'03)*, 2003.

[188] OASIS. Reference Model for Service Oriented Architecture 1.0. OASIS Standard, 2006. http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf.

[189] A. Panagiotidis, H. Bosch, S. Koch, and T. Ertl. EdgeAnalyzer: Exploratory Analysis through Advanced Edge Interaction. In *Proceedings of the 44th Hawaii International Conference on System Sciences (HICSS44)*, 2011.

[190] P. Parapatics and M. Dittenbach. Patent Claim Decomposition for Improved Information Extraction. In M. Lupu, K. Mayer, J. Tait, and A. J. Trippe, editors, *Current Challenges in Patent Information Retrieval*, volume 29 of *THE INFORMATION RETRIEVAL SERIES*, pages 217 – 239. Springer, 2011.

[191] I. Paterson, D. Smith, P. Saint-Andre, and J. Moffitt. XEP-0124: Bidirectional-streams Over Synchronous HTTP (BOSH). Draft Standard of the XMPP Standards Foundation, July 2010. http://xmpp.org/extensions/xep-0124.html.

[192] A. Pease, I. Niles, and J. Li. The Suggested Upper Merged Ontology: A Large Ontology for the Semantic Web and its Applications. In *Working Notes of the AAAI-2002 Workshop on Ontologies and the Semantic Web*, 2002.

[193] K. Perlin and D. Fox. Pad: an alternative approach to the computer interface. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '93, pages 57–64, New York, NY, USA, 1993. ACM.

[194] T. A. Phelps and R. Wilensky. The multivalent browser: a platform for new ideas. In *Proceedings of the ACM Symposium on Document Engineering*, pages 58–67, 2001.

[195] E. Pietriga. IsaViz: A Visual Environment for Browsing and Authoring RDF Models. In *Proceedings of the 11th World Wide Web Conference (Developer's day)*, 2002.

[196] E. Pietriga, C. Bizer, D. Karger, and R. Lee. Fresnel - A Browser-Independent Presentation Vocabulary for RDF. In *Lecture Notes in Computer Science (LNCS 4273), Proceedings of the 5th International Semantic Web Conference (ISWC06)*, pages 158–171. Springer, November 2006.

[197] R. Polillo. Quality Models for Web [2.0] Sites: A Methodological Approach and a Proposal. In *Proceedings of the 11th international conference on Current Trends in Web Engineering*, ICWE'11, pages 251–265. Springer-Verlag, 2012.

[198] A. Potrich and E. Pianta. L-ISA: Learning Domain Specific Isa-Relations from the Web. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, May 2008.

[199] I. H. Puhlmann. D8.2 User Requirements. Technical report, PATExpert Consortium, 2006.

[200] Y. Qian. Untersuchung der Rekonstruierbarkeit partiell verschlüsselter semantischer Beschreibungen im Semantic Web. Studienarbeit, Universität Stuttgart, Institut für Intelligente Systeme, 2005.

[201] Y. Qiu and A. Elsayed. Semantic Structures as Cognitive Tools to Support Reading. In T. Bastiaens and S. Carliner, editors, *Proceedings of World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education 2007*, pages 7319–7324, Quebec City, Canada, October 2007.

[202] R. Rao and S. K. Card. The Table Lens: Merging Graphical and Symbolic Representations in an Interactive Focus+Context Visualization for Tabular Information. In *Conference Companion on Human Factors in Computing Systems*, 1994.

[203] RDF Working Group. Resource Description Framework. Webpage, last visit December 10, 2012. http://www.w3.org/RDF/.

[204] U. Reimer. *Einführung in die Wissensrepräsentation*. B. G. Teubner Stuttgart, 1991.

[205] A. Rind, W. Aigner, S. Miksch, S. Wiltner, M. Pohl, F. Drexler, B. Neubauer, and N. Suchy. Visually exploring multivariate trends in patient cohorts using animated scatter plots. In *Proceedings of the 2011th international conference on Ergonomics and health aspects of work with computers*, EHAWC'11, pages 139–148. Springer-Verlag, 2011.

[206] A. Rind, S. Miksch, W. Aigner, T. Turic, and M. Poh. VisuExplore: Gaining New Medical Insights from Visual Exploration. In *Proceedings of the 1st International Workshop on Interactive Systems in Healthcare*, WISH@CHI2010, pages 149–152, 2010.

[207] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, Feb. 1978.

[208] J. Roberts. State of the Art: Coordinated and Multiple Views in Exploratory Visualization. In *Proceedings of the 5th International Conference on Coordinated and Multiple Views in Exploratory Visualization (CMV 2007)*, pages 61–71. IEEE Computer Society, 2007.

[209] T. Ropinski, S. Oeltze, and B. Preim. Visual Computing in Biology and Medicine: Survey of Glyph-Based Visualization Techniques for Spatial Multivariate Medical Data. *Comput. Graph.*, 35(2):392–401, 2011.

[210] M. Rotard, M. Giereth, and T. Ertl. Integrating Wikipedia Previews into Web Pages. In *Proceedings of Wikimania 2006*, 2006.

[211] M. Rotard, M. Giereth, and T. Ertl. Semantic lenses: Seamless augmentation of web pages with context information from implicit queries. *Computers and Graphics*, 31(3):361–369, 2007.

[212] S. Russell and P. Norvig. *Artificial Intelligence - A Modern Approach*. Prentice Hall, 2nd edition, 2003.

[213] S. S. Sahoo, O. Bodenreider, P. Hitzler, A. P. Sheth, and K. Thirunarayan. Provenance Context Entity (PaCE): Scalable Provenance Tracking for Scientific RDF Data. In *Proceedings of the 22nd International Conference on Scientific and Statistical Database Management (SSDBM 2010)*, volume 6187 of *Lecture Notes in Computer Science*, pages 461–470. Springer, 2010.

[214] F. Schmedding. Content-sensitive User Interfaces for Annotated Web Pages. In *Proceedings of the GI Jahrestagung*, pages 3016–3025, 2009.

[215] H. Schmid. Probabilistic Part-of-Speech Tagging using Decision Trees. In *International Conference on New Methods in Language Processing*, 1994.

[216] I. Schröck. Interaktive 3D-Visualisierung von Patentgraphen. Diplomarbeit, Universität Stuttgart, Institut für Visualisierung und Interaktive Systeme, 2007.

[217] H.-J. Schulz. *Explorative Graph Visualization*. PhD thesis, Universität Rostock, 2010.

[218] H.-J. Schulz. Treevis.net: A Tree Visualization Reference. *IEEE Computer Graphics and Applications*, 31(6):11–15, 2011. http://www.informatik.uni-rostock.de/~hs162/treeposter/poster.html.

[219] M. Shaw. Toward Higher-Level Abstractions for Software Systems. *Data Knowl. Eng.*, 5(2):119–128, July 1990.

[220] Z. Shen, K.-L. Ma, and T. Eliassi-Rad. Visual Analysis of Large Heterogeneous Social Networks by Semantic and Structural Abstraction. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1427–1439, 2006.

[221] M.-J. Shih and D.-R. Liu. Patent Classification Using Ontology-Based Patent Network Analysis. In *Proceedings of the Pacific Asia Conference on Information Systems (PACIS)*, 2010.

[222] B. Shneiderman. Tree Visualization with Tree-Maps: A 2-D Space-Filling Approach. *ACM Transactions on Graphics*, 11(1):92–99, 1992.

[223] B. Shneiderman. The eye have it: a task by data type taxonomy for information visualizations. In *Proceedings of Visual Languages*, 1996.

[224] B. Shneiderman and M. Wattenberg. Ordered Treemap Layouts. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis)*, pages 73–78, 2001.

[225] B. Shniderman. Why not make interfaces better than 3D reality. *IEEE Computer Graphics and Applications*, 23(6):12–15, 2003.

[226] J. Sowa. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Brooks Cole Publishing Co., 2000.

[227] R. Spence. *Information Visualization: Design for Interaction*. Addison-Wesley, 2nd edition, 2007.

[228] A. Spoerri. Visual Mashup of Text and Media Search Results. In *Proceedings of the 11th International Conference on Information Visualization*, 2007.

[229] G. Starke and P. Hruschka. *Software-Architektur kompakt - angemessen und zielorientiert*. Spektrum, 2009.

[230] A. Stäbler. Representation and Transformation of Patent Data using Semantic Web Technologies. Diplomarbeit, Universität Stuttgart, Institut für Visualisierung und Interaktive Systeme, 2006.

[231] C. Stolte, D. Tang, and P. Hanrahan. Polaris: A System for Query, Analysis, and Visualization of Multidimensional Databases. *Commun. ACM*, 51(11):75–84, 2008.

[232] M. Stone, K. Fishkin, and E. Bier. The Movable Filter as a User Interface Tool. In *Proceedings CHI 94*, pages 306–312. ACM, 1994.

[233] S.-I. Suzuki. Introduction to Patent Map Analysis. Webpage, 2011. http://www.training-jpo.go.jp/en/uploads/text_vtr/pdf/Introduction%20to%20Patent%20Map%20Analysis2011.pdf.

[234] S. Taduri, G. Lau, K. H. Law, H. Yu, and J. P. Kesan. Developing an Ontology for the U.S. Patent System. In *Proceedings of the 12th Annual International Conference on Digital Government Research*, 2011.

[235] S. Taduri, G. Lau, K. H. Law, H. Yu, and J. P. Kesan. An Ontology to Integrate Multiple Information Domains in the Patent System. In *International Symposium on Technology and Society (ISTAS)*, 2011.

[236] R. W. Taylor, M. Sarkar, M. Sarkar, M. H. Brown, and M. H. Brown. Graphical Fisheye Views of Graphs. In *Proceedings of CHI'92 Conference on Human Factors in Computing Systems*, pages 83–91. ACM Press, 1992.

[237] A. Telea. Combining Extended Table Lens and Treemap Techniques for Visualizing Tabular Data. In *Proceedings of the Eurographics / IEEE-VGTC Symposium on Visualization (2006)*, 2006.

[238] A. R. Teyseyre and M. R. Campo. An Overview of 3D Software Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 15(1):87–105, 2009.

[239] J. J. Thomas and K. A. Cook. *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. IEEE Press, 2005.

[240] R. Tolksdorf and E. P. B. Simperl. Towards Wikis as Semantic Hypermedia. In *WikiSym '06: Proceedings of the 2006 international symposium on Wikis*, pages 79–88, New York, NY, USA, 2006. ACM Press.

[241] C. Tominski, P. Schulze-Wollgast, and H. Schumann. 3D Information Visualization for Time Dependent Data on Maps. In *Ninth International Conference on Information Visualisation*, 2005.

[242] C. Tominski and H. Schumann. Enhanced Interactive Spiral Display. In *Proceedings of the Annual SIGRAD Conference, Special Theme: Interactivity*, pages 53–56. Linköping University Electronic Press, 2008.

[243] A. J. Trippe. Patinformatics: Tasks to Tools. *World Patent Information*, 25(3):211–221, 2003.

[244] F. van Ham and J. J. van Wijk. Beamtrees: Compact Visualization of Large Hierarchies. *Information Visualization*, 2(1):31–39, 2003.

[245] J. van Wijk and H. van de Wetering. Cushion Treemaps: Visualization of Hierarchical Information. In *INFOVIS*, pages 73–78, 1999.

[246] R. Vliegen, J. van Wijk, and E.-J. van der Linden. Visualizing Business Data with Generalized Treemaps. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):786–796, 2006.

[247] M. Völkel, M. Krötzsch, D. Vrandecic, H. Haller, and R. Studer. Semantic Wikipedia. In *Proceedings of the 15th International Conference on World Wide Web (WWW 2006)*, pages 585–594, New York, NY, USA, 2006. ACM Press.

[248] D. Vrandečić. *Ontology Evaluation*. PhD thesis, Fakultät für Wirtschaftswissenschaften des Karlsruher Instituts für Technologie (KIT), 2010.

[249] S. Vrochidis, S. Papadopoulos, A. Moumtzidou, P. Sidiropoulos, E. Pianta, and I. Kompatsiaris. Towards Content-Based Patent Image Retrieval: A Framework Perspective. *World Patent Information*, 32(2):94–106, 2010.

[250] W3C Recommendation. XML Path Language (XPath) Version 1.0, November 1999. http://www.w3.org/TR/xpath.

[251] W3C Recommendation. XML Encryption Syntax and Processing, December 2002. http://www.w3.org/TR/xmlenc-core/.

[252] W3C Recommendation. XPointer Framework, March 2003. http://www.w3.org/TR/xptr-framework/.

[253] W3C Recommendation. RDF Primer, February 2004. http://www.w3.org/TR/REC-rdf-syntax/.

[254] W3C Recommendation. RDF Test Cases, February 2004. http://www.w3.org/TR/rdf-testcases/.

[255] W3C Recommendation. RDF Vocabulary Description Language 1.0: RDF Schema, February 2004. http://www.w3.org/TR/rdf-schema/.

[256] W3C Recommendation. RDF/XML Syntax Specification (Revised), February 2004. http://www.w3.org/TR/rdf-syntax-grammar/.

[257] W3C Recommendation. Resource Description Framework (RDF): Concepts and Abstract Syntax, February 2004. http://www.w3.org/TR/rdf-concepts/.

[258] W3C Recommendation. XML Information Set (Second Edition), February 2004. http://www.w3.org/TR/xml-infoset/.

[259] W3C Recommendation. XML Schema Part 1: Structures Second Edition, October 2004. http://www.w3.org/TR/xmlschema-1/.

[260] W3C Recommendation. SPARQL Protocol for RDF, January 2008. http://www.w3.org/TR/rdf-sparql-protocol/.

[261] W3C Recommendation. SPARQL Query Language for RDF, January 2008. http://www.w3.org/TR/rdf-sparql-query/.

[262] W3C Recommendation. SPARQL Query Results XML Format, January 2008. http://www.w3.org/TR/rdf-sparql-XMLres/.

[263] W3C Recommendation. XML Signature Syntax and Processing (Second Edition), June 2008. http://www.w3.org/TR/xmldsig-core/.

[264] W3C Recommendation. OWL 2 Web Ontology Language Document Overview, October 2009. http://www.w3.org/TR/owl-overview/.

[265] W3C Recommendation. OWL 2 Web Ontology Language Profiles, October 2009. http://www.w3.org/TR/owl2-profiles/.

[266] W3C Recommendation. OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax, October 2009. http://www.w3.org/TR/owl2-syntax/.

[267] W3C Recommendation. Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification, 2011. http://www.w3.org/TR/CSS2.

[268] W3C Recommendation. Scalable Vector Graphics (SVG) 1.1 (Second Edition), August 2011. http://www.w3.org/TR/SVG11/.

[269] W3C Recommendation. RDFa Core 1.1, June 2012. http://www.w3.org/TR/rdfa-core/.

[270] W3C Team Submission. Turtle - Terse RDF Triple Language, March 2011. http://www.w3.org/TeamSubmission/turtle/.

[271] W3C Working Group Note. Defining N-ary Relations on the Semantic Web, April 2006. http://www.w3.org/TR/swbp-n-aryRelations/.

[272] W3C Working Group Note. Web Integration Compound Document (WICD) Core 1.0, August 2010. http://www.w3.org/TR/WICD/.

[273] X. Wan, J. Gao, M. Li, and B. Ding. Person resolution in person search results: WebHawk. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 163–170, 2005.

[274] L. Wanner, S. Brügmann, J. Codina, B. Diallo, E. Escorsa, M. Giereth, Y. Kompatsiaris, S. Papadopoulos, E. Pianta, G. Piella, I. Puhlmann, G. Rao, M. Rotard, P. Schoester, L. Serafini, and V. Zervaki. Towards Content-Oriented Patent Document Processing. *World Patent Information*, 30(1):21–33, March 2008.

[275] L. Wanner, S. Brügmann, B. Diallo, M. Giereth, Y. Kompatsiaris, E. Pianta, G. Rao, and P. Schoester. PATExpert: Semantic Processing of Patent Documentation. In *Poster and Demo Proceedings of the 1st International Conference on Semantic and Digital Media Technologies*, volume 233. CEUR-WS, 2006.

[276] M. Ward, G. Grinstein, and D. Keim. *Interactive Data Visualization: Foundations, Techniques, and Applications*. A K Peters Ltd, 2010.

[277] C. Ware. Designing with a 2 1/2D Attitude. *Information Design Journal*, 10(3):255–262, 2001.

[278] D. Weitzner, J. Hendler, T. Berners-Lee, and D. Connolly. Creating a Policy-Aware Web: Discretionary, Rule-based Access for the World Wide Web. In E. Ferrari and B. Thuraisingham, editors, *Web and Information Security*. Hershey, PA, 2004.

[279] R. Wettel and M. Lanza. Program Comprehension through Software Habitability. In *Proceedings of the 15th International Conference on Program Comprehension (ICPC 2007)*, pages 231–240. IEEE Computer Society, 2007.

[280] WIPO. Intellectual Property Handbook: Policy, Law and Use. Publication No.489 (E), 2004. http://www.wipo.int/about-ip/en/iprm/index.htm.

[281] WIPO. World Intellectual Property Indicators (2011 Edition). Publication No. 941E/2011, 2011. http://www.wipo.int/export/sites/www/freepublications/en/intproperty/941/wipo_pub_941_2011.pdf.

[282] WIPO. Intellectual Property Statistics. Webpage, last visit December 10, 2012, 2012. http://www.wipo.int/ipstats/en/.

[283] WIPO Standard. ST.32: Recommendation for the Markup of Patent Documents Using SGML, November 1995. http://www.wipo.int/export/sites/www/standards/en/pdf/03-32-01.pdf.

[284] WIPO Standard. ST.16: Identification of different kinds of patent documents, May 1997. http://www.wipo.int/export/sites/www/standards/en/pdf/03-16-01.pdf.

[285] WIPO Standard. ST.1: Recommendation concerning the minimum data elements required to uniquely identify a patent document, May 2001.
http://www.wipo.int/export/sites/www/standards/en/pdf/03-01-01.pdf.

[286] WIPO Standard. ST.6: Numbering of Published Patent Documents, December 2002.
http://www.wipo.int/export/sites/www/standards/en/pdf/03-06-01.pdf.

[287] WIPO Standard. ST.36: Processing of Patent Information using XML, November 2007.
http://www.wipo.int/export/sites/www/standards/en/pdf/03-36-01.pdf
Supplementary Material http://www.wipo.int/standards/en/xml_material/st36/.

[288] WIPO Standard. ST.9: Recommendation Conerning Bibliographic Data on and Relating to Patents and SPCs, February 2008.
http://www.wipo.int/export/sites/www/standards/en/pdf/03-09-01.pdf.

[289] WIPO Standard. ST.3: Two-letter Codes for the Representation of States, other Entities and Organizations, March 2011.
http://www.wipo.int/export/sites/www/standards/en/pdf/03-03-01.pdf.

[290] J. A. Wise, J. J. Thomas, K. Pennock, D. Lantrip, M. Pottier, A. Schur, and V. Crow. Visualizing the non-visual: spatial analysis and interaction with information from text documents. In *Proceedings of the 1995 IEEE Symposium on Information Visualization*, INFOVIS '95, pages 51–, Washington, DC, USA, 1995. IEEE Computer Society.

[291] P. C. Wong and R. Bergeron. 30 Years of Multidimensional Multivariate Visualization. In *Scientific Visualization*. IEEE Computer Society, 1997.

[292] P. C. Wong, H. Foote, G. C. Jr., P. Mackey, and K. Perrine. Graph Signatures for Visual Analytics. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1399–1413, 2006.

[293] P. C. Wong, H. Foote, P. Mackey, K. Perrine, and G. C. Jr. Generating Graphs for Visual Analytics through Interactive Sketching. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1386–1398, 2006.

[294] D. Yang. Interaktive Visualisierung von Patentsuchanfragen. Diplomarbeit, Universität Stuttgart, Institut für Visualisierung und Interaktive Systeme, 2007.

[295] P. T. Zellweger, B. W. Chang, and J. D. Mackinlay. Fluid links for informed and incremental hypertext browsing. In *Extended Abstracts on Human Factors in Computing Systems*, pages 7–8. ACM Press, 1999.

[296] J. Zhao, P. Forer, and A. S. Harvey. Activities, Ringmaps and Geovisualization of Large Human Movement Fields. *Information Visualization*, 7(3):198–209, 2008.

[297] H. Zhou, X. Yuan, H. Qu, W. Cui, and B. Chen. Visual Clustering in Parallel Coordinates. *Computer Graphics Forum*, 27(3):1047–1054, 2008.

[298] M. X. Zhou and S. K. Feiner. Visual Task Characterization for Automated Visual Discourse Synthesis. In *Conference on Human Factors in Computing Systems*, CHI '98, pages 392–399. ACM, 1998.

[299] Y. Zhou. Entwicklung eines Web-Service-orientierten Verfahrens zum kontrolierten Zugriff auf RDF-Graphen. Diplomarbeit, Universität Stuttgart, Institut für Intelligente Systeme, 2005.