Available online at www.sciencedirect.com
SCIENCE DIRECT®
Data & Knowledge Engineering 51 (2004) 189–222
DATA & KNOWLEDGE ENGINEERING
www.elsevier.com/locate/datak

A conceptual model of information supply ☆

B. van Gils *, H.A. Proper, P. van Bommel

University of Nijmegen, Sub-faculty of Informatics, Toernooiveld 1, Nijmegen 6525 ED, The Netherlands

Received 2 September 2003; received in revised form 14 January 2004; accepted 17 March 2004
Available online 18 May 2004

Abstract

In this paper we introduce a conceptual model for information supply which abstracts from enabling technologies such as file types, transport protocols and RDF and DAML + OIL. Rather than focusing on technologies that may be used to actually implement information supply, we focus on the question: what *is* information supply and how does it relate to the data (resources) found on the Web today. By taking a high level of abstraction we can gain more insight in the information market, compare different views on it and even present the architecture of a prototype retrieval system (*Vimes*) which uses *transformations* to deal with the heterogeneity of information supply.
© 2004 Elsevier B.V. All rights reserved.

Keywords: Information supply; Information market; Information retrieval

1. Introduction

In today's world *information* plays an increasingly important role. We rely on it in our day to day lives to make investment decisions (e.g. on the stock-market), to plan our holidays (using the website of a travel agency) and so on. With the apparent rise of the Web, a lot of this information is offered to us in digital form.

From a modeling perspective it makes sense to distinguish between *data* and *information* in this context. The resources that we can find on the Web today are *data resources*. Only when a data

☆ The investigations were partly supported by the Dutch Organization for Scientific Research (NWO). We would like to thank the reviewers for their useful suggestions which resulted in many improvements of this article.
* Corresponding author.
E-mail addresses: basvg@cs.kun.nl (B. van Gils), e.proper@acm.org (H.A. Proper), pvb@cs.kun.nl (P. van Bommel).

0169-023X/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.datak.2004.03.009

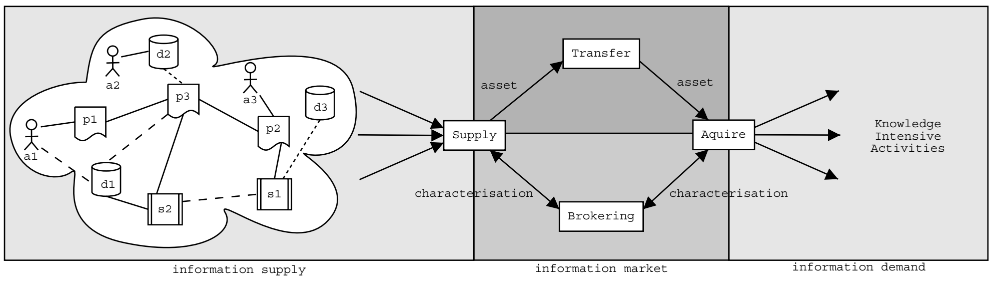

Fig. 1. The information market.

resource is relevant with regard to the information need (e.g. to fulfill some knowledge-intensive task) can we speak of information (see also Fig. 1). For example, it hardly makes sense to take a *random* resource and call it information!

1.1. Background

The amount of information that we have "under our fingertips" these days is vast, and still growing. In [1] it is even called an "explosion of on-line information". The huge number of resources available to us implies that it is sheer impossible for us to manually select those resources that are relevant to us in any given situation. We rely on automated search tools such as search engines on the Web, digital libraries and E-service repositories to assist us with our search.

We are, in a way, dependent on these tools to perform our day to day activities and expect them to "work well". This means, for example, that we expect a search engine to index everything on the Web, that we expect a digital library to have the perfect matching algorithm that finds exactly those resources in its database that are relevant to us and that an E-service repository finds exactly those services that conform to all our wishes in terms of security, speed, etc. In practice, this turns out to be difficult at least.

Another complicating factor is the fact that the way we use/browse the Web is changing as well: not only do we access the Web from a desktop computer, we also access it using mobile devices and handhelds such as a PDA or a mobile phone. This change also poses restrictions on the resources that we can handle: for example it seems hardly possible/does not make sense to send huge Word files to a user browsing the Web with a mobile phone using WAP.

We illustrate this situation by mapping it to the notion of an *information market* as illustrated in Fig. 1 [2]. The left side of the figure shows the data that is supplied via the Web. The different symbols indicate that there are different types of resources, ranging from HTML-files to online databases and even people (the implicit knowledge in people's head can also be seen as a resource which can be accessed by communicating with them). We define the totality of all resources available to us as *information supply*. The right-hand side of the figure, on the other hand, represents *information demand*, inspired by the fact that people need information to conduct activities for which they need information. The middle part of the figure represents the *brokering* between supply and demand, as well as the fact that resources should be *transported* to the user before they can be consumed. We define this to be the *information market*.

We expect the automated tools in the *information market* to support us in our search for information, taking into account the way we access the web (transport) as well as all aspects of our information need (brokering). In this article, we specifically focus on the supply side of the information market; i.e. *information supply*. In our view, one of the key challenges for automated tools supporting the information market, is their ability to deal with the heterogeneous nature of information supply:

- There are many different ways to represent information. For example using a webpage, a document, an image or some interactive *form*.
- There are many different *formats* that may be used to represent information on the Web. For example, using formats such as PDF, HTML, and GIF.

Some examples that illustrate this heterogeneous nature are:

- The resources that we find on the Web today are *heterogeneous* in nature [3–5]. Different types of resources exist on the Web, ranging from "passive" documents such as PDF and HTML files, to resources of a more dynamic nature, such as flight or train schedules, stock tickers, etc. This raise an important challenge: how can we uniformly access these heterogeneous resources? Is it possible to index and represent them in a uniform way? The search engine GOOGLE [1] is able to index many different file types. But what happens if it runs into a new, and unknown, file format?
  How should dynamic resources be accessed/indexed? Most traditional search engines, including GOOGLE, do not handle dynamic context very well. They merely index some of the dynamically generated web-pages. Another challenge is: is there a uniform way to specify what these resources are about, i.e. their *informatics* [6]?
- In [7], Tim Berners-Lee presents his vision on the *semantic web*. He states that "a new form of Web content that is meaningful to computers will unleash a revolution of new possibilities" and "For the semantic web to function, computers must have access to structured collections of information and sets of inference rules that they can use to conduct automated reasoning." Many researchers have taken up the gauntlet and developed semantically rich mark-up-languages (e.g. [8,9]), a framework to semantically describe/annotate resources (see e.g. [10–12]), etc. With these new technologies it should, for instance, be possible to express that a certain webpage can only be accessed at a fee. However, the technology to exploit these new semantic structures is still in a very early stage and despite the efforts of e.g. the World Wide Web Consortium [2] standardization is not yet achieved.
- Not long ago, the web consisted mainly of hyper-linked *documents*. Today, however, the Web also offers numerous ways to interactively obtain information. Examples in this area are newsgroups, mailing list, instant messaging protocols such as Jabber etc. This implies even that humans can, to a certain extent, also be seen as data resources: when you contact them (e.g. via

[1] http://www.google.com.
[2] http://w3c.org.

E-mail) they may provide you with the right information! In this scheme, humans can be contacted via their URI [13]. As [14] points out, this scheme is ambiguous because an URI can now be used to refer to two things: the webpage/E-mail address of a person, or to the person himself.

The evolving and multimedia nature of the Web has to be reflected in the tools that we develop and use to assist us in our search for information. This means that our tools have to be (re)designed to be able to deal with changes so that we do not have to re-engineer (parts of) our tools whenever a new file format arrives, or documents are produced in a new language/character set such as Chinese symbols.

1.2. Approach

Much research has been conducted in the areas of information modeling, representation and retrieval already. Fields of research in this area include library applications, relational databases, meta data activities, relevance ranking, mark-up languages, ontologies, conceptual information modeling, etc. An overview of these developments is given in Section 2. Each of these fields have their own perspective with regard to information supply, most of which are technology oriented in the sense that they often target at developing new standards or applications. We feel that a solid and complete view of information supply at the conceptual level is lacking still. [3] Apart from the 'traditional' reasons for domain modeling (see e.g. [15–17]), such a model can be used to:

- gain more insight into the workings of the information market, especially information supply;
- serve as a *reference model* in order to compare different views/perspectives on information supply;
- serve as a basis for a new way of thinking about/new architectures for retrieval on the Web.

In this article we aim to derive a *conceptual* model for information supply, aiming to provide a theoretical underpinning of information supply. In our approach we:

- will abstract from the medium on which data is represented (as well as other 'enabling technologies'). The fact that data is represented in a PDF file or a PS file is merely a parameter in our model;
- will not try to derive a new syntactical denotation such as RDF [10];
- will abstract from specific (organizational) settings in the sense that we are not making a new ontology with which what concept x means in setting y;
- will focus on the concepts and do not target at developing a specific application such as a Web-based retrieval engine.

[3] An overview of relevant approaches in literature is given in Section 2.

The basis for our model is the distinction between *data* resources that can be found on the Web (basically, anything which can be addressed by means of an URI [13]) and the *information* that is consumed by people. In this paper we present both an informal introduction to our model, as well as a formalization.

In Section 7 we illustrate how our model can be used in practice and present a novel retrieval architecture called *Vimes*. [4] The core of this architecture is the notion of *transformations* (see Section 6). Transformations add flexibility to our model in the sense that it allows us to transform resources from one format (e.g. HTML) to another (e.g. PDF). The need for such a mechanism was recognized in [19]:

> Today's Web is far from mature, with less than a decade of widespread use. It will evolve rapidly, adding functions and features. Maintaining the utility of older Web pages will probably mean migrating them to newer formats, which may not always provide easy ways to preserve the content of the older pages. Migration is a particularly difficult issue for pages that contain active elements composed of computer programs providing functions such as animation and custom user interactions. Preserving the functionality of such programs while support for the underlying software disappears will present a significant technical challenge. Another challenge is whether to support a large and increasing number of formats or to select a few favored formats and map all content to the selected formats. The convergence of media and equipment for audio, video, and textual materials over the next few years will only make these questions more pressing.

Note that our main contribution is the *conceptual model*. *Vimes* is used here purely to illustrate our ideas.

1.3. Overview

We start out by presenting an overview of relevant, related research in Section 2, concentrating on structured information access, information retrieval, the semantic web and information modeling techniques. In Section 3 we introduce our conceptual model using the ER-notation. In the section thereafter we formalize our model. Section 5 illustrates our model. Sections 6 and 7 introduce transformations and the *Vimes*-architecture respectively. The last section summarizes this paper and provides an outlook for future research.

2. Current work

In the previous section we explained the topic of this paper: deriving a conceptual model for information supply. We also briefly touched upon relevant work in literature. In this section we elaborate on this by giving a more elaborate overview [5] of related work and explaining how this relates to our work.

[4] See also [18] for an introduction to *Vimes*.

[5] Note that this overview is extensive, but not complete.

2.1. Structured data access

Computers have been used for storing and retrieving structured data for many years, most notably by using (relational) databases. In [20] it is even claimed that the history of database research has led to the database system becoming arguably the most important development in the field of software engineering.

A theoretical perspective on information supply, from the perspective of structured data access, is provided by the relational algebra [21] and its 'real life' implementation SQL (Structured Query Language). SQL is the industry-standard language for creating, updating and querying relational database management systems. In other words, SQL is the de facto standard for querying *structured data* stored in relational database management systems (RDMS) such as ORACLE and INGRES. As such, relational algebra, and more specifically: SQL, provides a viewpoint on that part of information supply which is represented in terms of a relational database management systems.

A more recent approach is described in [22], where the focus shifts towards the relation between query languages and the Web. Data resources on the Web (typically HTML-documents) are less structured than, for example, a database schema. This has strong implications for the way we access them. The authors discuss two important features that distinguish the Web from traditional databases: the navigational nature of the Web and the lack of concurrency control. To overcome the problems associated with these two features, a new formalism for *query computation* is developed, which results in a *web calculus*. For this purpose, the web is modeled as a relational database containing a relation of *Node* objects, one per document, and a relation of *Link* objects, one per node-to-node link.

Besides relational databases, much research has been conducted in the area of object-oriented databases (OODB) also (see [23,24], and [25] for the differences between relational and object-oriented database systems). Even though RDBMS and OODB are two distinct strategies to provide means of storing and retrieving data, they are essentially *implementations* of the same perspective on information supply: try to structure and store data (resources) in such a way that they can effectively be retrieved and queried.

2.2. Information retrieval

In *information retrieval* (IR), the goal is to retrieve those documents/resources from a collection that are relevant with regard to a user query (which is presumed to represent the *information need* of the user [6]). More specifically, for each resource the *relevance* with regard to the query is computed after which the documents are ranked. Documents with a similarity value greater then a certain threshold are presumed to be relevant; documents with a lower similarity value are not. Depending on the way the documents are represented in the IR system, different ranking algorithms can be used (see e.g. [27–29]). Furthermore, the constraint that resources must adhere to a certain structure (such as a relational database schema) is relaxed.

In traditional IR, the retrievable resources consisted of textual documents only. Each document can be described in terms of a set of keywords (which are either assigned manually or extracted

[6] See e.g. [26] for a discussion on query formulation in IR.

from the document automatically). Queries are also in the form of several keywords and, optionally, uses boolean connectors such as `and`, `not` and `or`. Matching is done by finding those documents in which the keywords occur (or *not* occur, if the `not`-connector is used). A more advanced methodology is the *vector space model*, of which [30] attempts to give an overview. In the vector space model, both documents and queries are represented as vectors of keywords using a weighting scheme (such as the binary scheme in which a position in the vector gets a 1 if the word occurs in the document and a 0 if itdoes not or a more advanced scheme such as the *tf.idf* class of word weights). The matching algorithm simply measures the distance between the query vector and the document vectors: the closer a document vector is to the query vector the more relevant this specific document is with regard to the query and vice versa. As of recent, similar approaches have been developed for image retrieval, video retrieval, audio retrieval and even E-service discovery.

Another way of indexing/characterizing (textual) resources uses *index expressions*, which is an extension to the term phrases whereby the relationships between terms are modeled [31–33]. Consider the phrase *The attitude of students in universities to the war in Iraq*. In a ‘normal’ keyword-based approach, a representation of this phrase would contain (stemmed/normalized) words: {*attitude*, *student*, *university*, *war*, *Iraq*}. The representation in an index expression is much more semantically rich; it describes the meaning of the original sentence better: *attitudes of* (*students in* (*universities*)) *to* (*war in* (*Iraq*)). Fig. 2 graphically shows this index expression. With index expressions one achieves a mechanism for information disclosure and hence, the possibility for more accurate retrieval.

2.2.1. Retrieval on the Web

With the rise of the Web, the need for a more elaborate retrieval rose. Search engines such as GOOGLE attempt to index the heterogeneous resources found on the Web as good as possible. These days GOOGLE can index several formats found on the Web such as HTML, PDF, Word and also graphical formats such as PNG and JPEG. Furthermore, GOOGLE is esteemed for its elaborate ranking system called *PageRank* [34,35]. This ranking system is based on the notion that pages that are referred to by many other pages are probably ‘important’. More specifically: GOOGLE makes heavy use of the (link) *structure* present in hypertext for *indexing* and *raking* webpages [34] in the sense that webpages that are referred to a lot, are likely to be important. Hence, PageRank is designed to be an objective measure of its citation importance.

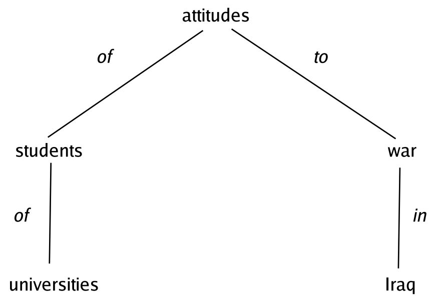

Fig. 2. Example index expression.

These Web-based search engines are generally perceived to work rather well. Often, however, these search engines tend to return too many (references to) potentially relevant resources. Users are still required to manually wade through large result sets in search of truly relevant assets. A second problem lies in the area of vocabulary that is used. Practical studies have shown that there is a critical mismatch between a user's vocabulary, and the vocabulary used on the Web [36]. Picking the right query terms depends on how intimate searchers are with the vocabulary used in documents they wish to retrieve, which is not always straightforward.

Recent approaches have tried to overcome problems related to the heterogeneity and diversity of websites spread all over the Web. For example, in [37] the *Hyperview* approach is discussed. In this approach, a *virtual website* is presumed to contain concentrated information that has been extracted, homogenized and combined from several underlying webpages, with the purpose to save users the trouble to search for, and browse through all these (underlying) pages. These three steps are treated uniformly as consecutive "views that map between different levels of abstraction", where the views are represented as graphs. Transitions from one view to another are achieved by means of graph transformations.

2.2.2. Digital Libraries

Digital Libraries (DL) also make use of retrieval techniques. A DL is a collection of services and the collection of information objects that support users in dealing with information objects, the organization and presentation of those objects available directly or indirectly via electronic/digital means [38]. Simply put, a DL provides users with an infrastructure including a bulk of digital resources (which may be heterogeneous) and services that are needed to access these resources. Functionality ranges from searching and browsing to transporting the resources to the user [39]. The fact that heterogeneity of resources poses constraints on how you access them is recognized in the DL community also. In [40], for example, it is stated that the architecture underlying the DL should be separated from the content stored in it; the architecture must provide a (uniform) way to specify characteristics of each type of resource. This general mechanism may be extended for specific types of resources.

As with traditional (bricks and mortar) libraries, digital libraries deal with a well-known collection of resources. This is quite different from the situation on the Web! From a metadata perspective it is easier to annotate the resources in such a controlled environment e.g. with *Dublin Core* [41–43] which, in a way, adds more structure to the library. For each resource in the library both the *aboutness* is captured (by using keywords, index expressions or other means) as well as data about the resource (such as the author, year of publication etcetera). From an information supply perspective, DL's attempt to advanced techniques to describe and characterize heterogeneous digital resources in a controlled environment.

2.3. Metadata

Metadata is data about data. It can be useful to know several things about your data like its structure, last modification date etcetera. A myriad of standards for metadata exist (see also Section 2.2.2), especially in the business domain. Examples are:

- CWM (Common Warehouse Metamodel) is a specification that describes metadata interchange among data warehousing, business intelligence, knowledge management and portal technologies. It provides a framework for "representing metadata about data sources, data targets, transformations and analysis, and the processes and operations that create and manage warehouse data and provide lineage information about its use" [44].
- UDDI (Universal Description, Discovery, and Integration) is a standard for locating web services by enabling robust queries against rich metadata. In summary, metadata about webservices are stored in a repositories. The information provided in a listing consists of three conceptual components: "white pages" of company contact information; "yellow pages" that categorize businesses by standard taxonomies; and "green pages" that document the technical information about services that are exposed [45,46].
- ebXML (Electronic Business using eXtensible Markup Language), is a modular suite of specifications that enables enterprises of any size and in any geographical location to conduct business over the Internet [47]. The ebXML specification provides a standard infrastructure for sending business messages across the internet.

In terms of the information market, these standards focus on describing (aspects of) information supply. Important insights about what information supply is can be gained by studying these standards and how they are applied in practice.

2.4. Semantic Web

After the apparent success of the Web, a new form of Web content that is meaningful to computers will unleash a revolution of new possibilities [7]. One of the problems with the current Web is that the available data (web content) was designed to be read by humans, rather than to be interpreted/manipulated meaningfully by computers. The vision of the semantic web is that the current Web must be extended such that the available data are given a well-defined meaning. By doing so, computers (and other agents such as hand-held devices, mobile phones etcetera) can cooperate to perform knowledge-intensive tasks:

> The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation. The first steps in weaving the Semantic Web into the structure of the existing Web are already under way. In the near future, these developments will usher in significant new functionality as machines become much better able to process and "understand" the data that they merely display at present (*taken from*: [7]).

There are several activities to be discerned in the semantic web community, such as *knowledge representation* and *ontologies*.

One of the enabling technologies behind the Semantic Web is RDF; the *Resource Description Framework* [10]. The broad goal of RDF is to define a mechanism for describing resources in terms of their metadata without defining their semantics. The basic object model of RDF consists of:

- all things described by RDF are *resources*. They are identified by an URI (see [13]);
- a specific characteristic that describes the resource is a *property*;
- a specific resource together with a named property plus the value for that property is a *statement*.

In other words, RDF can be used to describe resources in terms of (meta-data) characteristics.

The *Darpa Agent Markup Language* (DAML) and *Ontology Inference Language* (OIL) build on RDF and jointly provide a semantic markup language for resources on the Web [11,48]. The constructs of DAML are said to be rich enough to create ontologies and markup information that is machine readable and understandable. OIL, on the other hand, provides an inference layer on top of ontologies. The OWL *Web Ontology Language* [49], on the other hand, is intended to be used when the information contained in documents needs to be processed by applications, as opposed to situations where the content only needs to be presented to humans and is a revision of the DAML + OWL language.

In our approach we derive a conceptual model for information supply. Implementations of this conceptual model, can indeed employ RDF to describe instances by using a fixed set of allowable (names for) properties, most notably those that we introduce in our model.

2.5. Information modeling

Finally, conceptual information modeling techniques such as ER [50], EER [51], ORM [17], UML Class Diagrams [52] and their associated query/constraint languages such as RIDL [53], CADDY [54,55], LISA-D [56], ConQuer [57], can be used to provide a conceptual model of some given application domains. This conceptual model is usually formulated in terms of a set of entity and relationship types, describing the essence of the application domain.

Usually, the conceptual model is translated to some implementation model such as a relational database schema. In this case, the information supply consists of the representation of the state of affairs of some domain. For example, the state of affairs with regards to car-prices, airplane flights, enrollments of students, etcetera. The actual information then corresponds to the contents of the underlying relational database. The original conceptual model would allow us to view this relational database as a conceptual database, restating the contents of the relational database as a population of the relationship types as identified in the conceptual schema. In other words, a conceptual representation of the state of affairs in a domain. We use conceptual modeling techniques (most notably ER) to introduce our model.

3. A reference model for information supply

In this section we introduce our reference model. The goal of this section is to give a high-level overview of the model, whereas Section 4 presents the formalization of the model. The presented model displays the *essential* properties of information supply. We now first motivate our way of modeling.

Information supply can be modeled in several ways. For example, a technical operational model would be preferred when implementation aspects, such as database and network parameters, have to be emphasized. Alternatively, a logic-based model would be preferred when deductions need emphasis. A full information model in the style of e.g. ER and UML would be preferred when emphasis is on highly structured resources. However, our emphasis is on the description of basic properties of information supply, where information services are interrelated and may be associated with features and representations. We therefore give a high-level overview in terms of a global ER-like model, which on the one hand connects our approach to traditional information modeling techniques, and on the other hand allows for a straightforward formalization of the underlying typing mechanism.

This also sets the context for transformations, needed in heterogeneous environments such as the Web. Note that our reference model can be applied in a classical document retrieval context, as well as in database-driven Web access (see e.g. Section 7 about the *Vimes* architecture). Moreover, it can be an aid in new applications, such as publishing and distributing parts of existing databases that have not been connected to the Web yet.

3.1. Information services, features and representations

An important distinction is that of data versus information: data *becomes* information as soon as it is found to be relevant to a given information need. It is evident that data itself often does not help us. For example, if somebody gives you a piece of paper, containing the text "€23" without saying what the purpose of this text is, this does not mean anything to you. Therefore, we need data along with the meaning (semantics) of that data. This meaning could for example explain that the price of a given book is €23. Data with semantics is also called *information*. For more details about the distinction between data and information, the reader is referred to [17].

A similar, and equally important, distinction is that of an information service and the technology that was used to store it. Technology, in this context, is used in a broad sense. It can be paper, a database, a flat file, but also the knowledge in people's heads (to be accessed using e.g. a conversation). With this in mind, we view the Web as a landscape of inter-related information providing entities, which are at the technology level. The ability of these entities to—in terms of our supply-chain perspective—provide information to some consumer, is viewed as a *service* that the entities may provide. This is why we have chosen to use the term *information service* for the entities that make up information supply.

Obviously, there is a relation between information services and their underlying representations (the data resources on the Web). However, there is more to this relation than meets the eye. For example, consider a document on the Web that may have an abstract of the book *The color of magic*. Depending on one's perspective, this document is either 'full content', or 'an abstract'. To be able to model these facts, we introduce the notion of *features*.

More specifically: we have chosen to view an information service as an abstract entity, which may possess several *features*. Each feature is presumed to have some concrete underlying *representation* associated to it. To make this more concrete, consider the following two examples of information services and some of their features:

(1) A report discussing the potential impact which an eastward expansion of the Eurozone may have on the Euro/Dollar exchange rate, with features:
- A PDF file (the representation) with the feature *full-content*;
- an ASCII representation of the *abstract*;
- an XML table of *authors*.

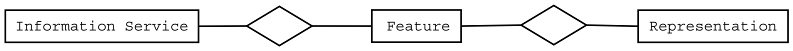

Fig. 3. Services, features and representations.

(2) A railway travel planner, providing information regarding desired railway trips, with features:
- An application (the representation) running on a PDA providing the *full-service*;
- a SOAP based web-service also providing the *full-service*,
- a set of *keywords* (for example: railway inquiries) about the information service,
- a WSDL [7] *description* of the information service, etc.

Note that for some information services, a "full content" version is not available; for example, it is highly unlikely that *everything* someone knows can be "dumped" (the term *memory dump* comes to mind) and stored. This leads to the situation as depicted in Fig. 3.

Note that even though Fig. 3 depicts *representation* as an 'atomic' object, we will discuss below how such objects may actually have a complex (such as an XML document) as well as a dynamic nature (such as an application).

3.2. Introducing relations

With this machinery, we are able to model information services, their representations and the fact that there are different 'views' on information services. That is not all there is to it however, since information services (especially on the Internet) have relationships with others. For example, a scientific paper may *refer to* other papers, a chapter is *part of* a book and a movie is *based on* a script/book. In case of the Web, these relations are usually *implemented* using hyperlinks; a mechanism which dates back to the notion of hypertext [59,60].

We view these relations to be binary, with a unique *source* and *destination*. For example, a situation where a scientific paper A refers to another paper B. In this case, A is the source, B is the destination and 'refers to' is the relation. This leads to the situation as shown in Fig. 4.

The following example illustrates the concepts introduced so far. Consider the research index *Citeseer* [61], an information service offering search capabilities over a database with scientific publications (hence, the feature is "full service"). One of the papers that is stored in its database is *Invading the Fortress: How to Besiege Reinforced Information Bunkers* [62], which is also an information service with feature "full content". *Citeseer* also stores an abstract for each paper. This abstract in itself is also an information service, with either "full content" or "abstract" as it's feature (depending on ones point of view). Obviously, several relations exist between the above mentioned information services (the source and target are not explicitly mentioned in the following):

[7] WSDL is the Web-Service Description Language, see e.g. [58].

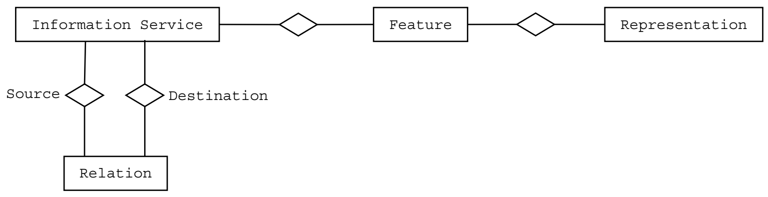

Fig. 4. Information services and relations.

- The relation between *Citeseer* and the two other information services is named "is stored in".
- The relation between the paper [62] and its abstract is named "is abstract of".

3.3. Introducing typing

On closer examination, some observations can be made with regards to the features of information services and the relations between information services. Different information services may have similar *features*, yet with differing representations. For example: 'title', 'authors', 'full-content', 'full-service', 'description', etc. In other words, features may be classified in terms of a *feature type*. Furthermore, features may be represented according to different formats, referred to as *representation types*. For instance: 'XML', 'LaTeX', 'Java' or 'PDF'. For resources of a static nature these types can be thought of as MIME-TYPES, a standard developed to—among other things—facilitate the uniform recognition and handling of different media types across applications. [8] There is more to representation types than initially meets the eye. For example, an ASCII file is quite different from a POSTSCRIPT file, which again are very different from a structured database, or a JAVA application. The way we will approach this diversity and heterogeneity is to treat representation types as abstract data types, which are represented as many-sorted algebra's providing abstractions of the underlying 'implementation details' [64]. It is not our intention to define a 'definite' set/taxonomy of representation types. With the abstract data type approach we aim to facilitate an open typing system in which new types can be introduced when and if needed. In a practical situation (such as the Vimes prototype, see Section 7), the syntactic definitions as well as their semantics can be denoted in terms of a language such as OWL [49] and/or DAML + OIL [48].

Note that such a strategy can deal with both static as well as dynamic resources. The approach as described in [64] actually uses many-sorted algebra's to formalize the behavior of objects as used in object-oriented approaches. We essentially view representations as statefull objects that provide 'methods' as defined in the many-sorted algebra associated to their representation type.

Finally, different semantic classes of relations between Information Services exist, for example: 'refers to', 'part of' and 'based on', etc. In other words, relations may be classified in terms of a *relation type*. In other words, a *typing mechanism* must be introduced. Using such mechanism would allow us to make very precise statements about (groups of) elements in the information space. The usual merits of introducing such a mechanism also apply.

[8] MIME is an acronym for *Multipurpose Internet Mail Extensions* and is defined in [63].

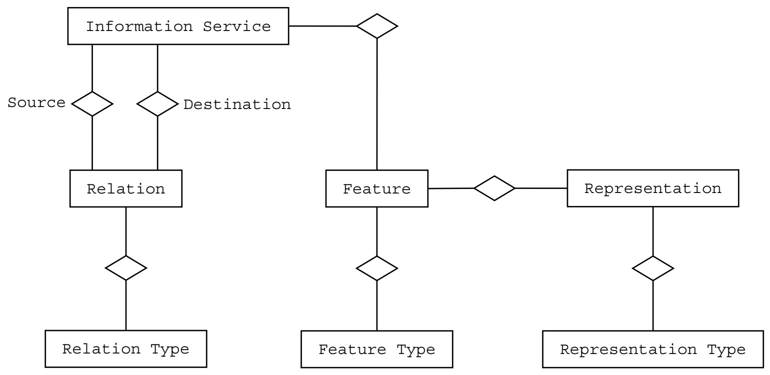

Fig. 5. Typing of features and representations.

When refining the situation as depicted in Fig. 4 with a typing mechanism for features, representations and relations, the ER model as shown in Fig. 5 results. Note that information services do not receive an explicit type! A typing mechanism for information service is really yet another *feature type*. The model represented in Fig. 5 was essentially derived by applying a domain modeling approach, such as ORM [17] to information supply in the context of the Web.

With the typing mechanism in place, the model is expressive enough to reason about the heterogeneity of information supply. Some early results, in particular in terms of transformations between representations, are reported in Section 6 and [65]. In the following section, this model is formalized, after which (in Section 5) a more detailed example is presented.

The model presented in Fig. 5 has been validated 'in the small' by applying a so-called population check [17] on some samples from the Web. Currently, see Section 7, a prototype is being developed to validate our model 'in the large'; i.e. the applicability of the presented model for information supply.

4. Formalization of the reference model

In this section we further explore the model that was derived in the previous section. We formalize the concepts that were introduced so far using descriptive mathematics (for a good introduction see e.g. [66]). Using this formalization we are able to prove some insightfull properties of the reference model. Section 6 builds further upon this formalization by adding transformations that can be applied to representations, enabling us to better deal with heterogeneity in information supply. We will further clarify our formalization by means of an example in the following section.

4.1. Descriptive elements

For the information landscape as we see it today, the information services, their relationships, features and representations that can be discerned are presumed to be contained in the sets: $\mathcal{IS}$, $\mathcal{RL}$, $\mathcal{FE}$, $\mathcal{RP}$ respectively. Because we consider them to be *elementary*, these sets must be disjoint:

Axiom 1 (*Disjoint elements*). $\mathscr{IS}$, $\mathscr{RL}$, $\mathscr{FE}$, $\mathscr{RP}$ are disjoint sets.

This axiom implies that an information service cannot be a feature at the same time. Or, that a relation cannot be a representation at the same time, etc. For example, the information service 'lord of the rings' is not the same thing as its representation as a movie `lotr.mpg`.

Information service may have several features associated to it. However, each feature has exactly one information service associated to it. This is modeled by the function $\mathsf{Service} : \mathscr{FE} \to \mathscr{IS}$. Furthermore, the representations that may be associated to a feature are modeled by the function $\mathsf{Representation} : \mathscr{FE} \to \mathscr{RP}$. Since we do not want two different features with exactly the same information service and representation associated to it, we define the combination of an information service and representation in Fig. 5 to be unique.

Axiom 2 (*Unique features*)

$$\forall_{f,g} \in \mathscr{FE}[\mathsf{Service}(f) = \mathsf{Service}(g) \wedge \mathsf{Representation}(f) = \mathsf{Representation}(g) \Rightarrow f = g]$$

The sources and destinations of the inter-service relationships in $\mathscr{RL}$ are presumed to be provided by the functions $\mathsf{Src}, \mathsf{Dst} : \mathscr{RL} \to \mathscr{IS}$ respectively. Even though the latter two functions may appear awkward at first sight, it actually makes sense to distinguish between the *source* and the *destination* of a relation between two information services since this allows us to introduce them explicitly in our model. For example, consider the relation 'is based on'. This relation allows us to express that book x is based on another book y. In this case, x is said to be the source of the relation, and y the destination.

Collectively, $\mathscr{FE}$, $\mathscr{RP}$ and $\mathscr{RL}$ are referred to as *descriptive elements* since they collectively describe the information services that are available. Formally, we therefore introduce:

$$\mathscr{DE} \triangleq \mathscr{RL} \cup \mathscr{FE} \cup \mathscr{RP}$$

In this rest of this article, we will usually omit the adjective 'descriptive' and simply use *elements* or *model elements*.

4.2. Types

To model the typing of the descriptive elements, we presume to have a set $\mathscr{TP}$ of types. Let $\mathsf{HasType} \subseteq \mathscr{DE} \times \mathscr{TP}$ then be a relationship which provides the types of a given element $e \in \mathscr{DE}$. In other words, if $e\,\mathsf{HasType}\,t$, then element e is said to have type t. Consider for example the webpage $\underset{\smile}{e \text{ of}}$ a John Doe called `johndoe.html`. This webpage is a representation: $e \in \mathscr{RP}$. The type $t \in \widetilde{\mathscr{RP}}$ of this document is HTML. Hence in this case $e\,\mathsf{HasType}\,t$ reads: John Doe's webpage HasType HTML.

We presume that all elements (instances) in information supply are typed.

Axiom 3 (*Total typing*). $\forall_{e \in \mathscr{DE}} \exists_t [e \ \mathsf{HasType}\ t]$.

This axiom enforces that *every element* that we know about has a type. Conversely, we presume $\mathscr{TP}$ to only contain types that are actually 'used' and define:

Axiom 4 (*Total type usage*). $\forall_{t\in\mathcal{TP}}\exists_e[e\ \mathsf{HasType}\ t]$.

In order to talk about *all* elements of a given type, we define the population of a type as follows:

$$\pi(t) \triangleq \{e \mid e\ \mathsf{HasType}\ t\}$$

Let X be a set of elements, then $\widetilde{X}$ represents the set of types that are associated with these elements:

$$\widetilde{E} \triangleq \{t \mid \exists_{e\in E}[e\ \mathsf{HasType}\ t]\}$$

From this definition, and Axioms 3 and 4 it follows that every item has a type, and that $\widetilde{E}$ is the set of all existing types. Furthermore, the types of the information services, features and representations are presumed to be disjoint (recall Axiom 1):

Axiom 5 (*Disjoint types*). $\widetilde{\mathcal{RL}}$, $\widetilde{\mathcal{FE}}$ and $\widetilde{\mathcal{RP}}$ are disjoint.

We will refer to $\widetilde{\mathcal{RL}}$, $\widetilde{\mathcal{FE}}$ and $\widetilde{\mathcal{RP}}$ as *type classes*. The disjointness of the type classes allows us to define the class of types some given type belongs to:

$$\mathsf{TypeClass}(t) \triangleq T \quad \text{where } T \in \{\widetilde{\mathcal{RL}}, \widetilde{\mathcal{FE}}, \widetilde{\mathcal{RP}}\} \text{ such that } t \in T$$

TypeClass returns the actual set of types—which can be either $\widetilde{\mathcal{RL}}$, $\widetilde{\mathcal{FE}}$ or $\widetilde{\mathcal{RP}}$—to which a given type x belongs. For example, if $f \in \mathcal{FE}$ is the feature 'full content', then $\mathsf{TypeClass}(f)$ will return $\widetilde{\mathcal{FE}}$. Due to Axiom 5 it holds that for every t there is only *one* such T. Even more, due to Axiom 4 we have:

Corollary 1. $\forall_{t\in\mathcal{TP}}\exists_T[\mathsf{TypeClass}(t) = T]$.

This means that for *every* given type t, the actual type can be found using TypeClass. In other words, TypeClass is really a total function: $\mathsf{TypeClass} : \mathcal{TP} \to \wp(\mathcal{TP})$.

As a direct consequence from Axiom 5, we can also prove that a descriptive element can only have types that belong to a single type class:

Lemma 1. $\exists_e[e\ \mathsf{HasType}\ t \wedge e\ \mathsf{HasType}\ u] \Rightarrow \mathsf{TypeClass}(t) = \mathsf{TypeClass}(u)$.

Proof. Let $t, u \in \mathcal{TP}$ such that $\exists_e[e\ \mathsf{HasType}\ t \wedge e\ \mathsf{HasType}\ u]$.

From the definition of $\mathcal{DE}$ follows:

$$\exists_{E\in\{\mathcal{RL},\mathcal{FE},\mathcal{RP}\}}\exists_{e\in E}[e\ \mathsf{HasType}\ t \wedge e\ \mathsf{HasType}\ u]$$

From the definition of $\widetilde{E}$ follows: $\exists_{E\in\{\mathcal{RL},\mathcal{FE},\mathcal{RP}\}}[t, u \in \widetilde{E}]$.
In other words: $\exists_{T\in\{\widetilde{\mathcal{RL}},\widetilde{\mathcal{FE}},\widetilde{\mathcal{RP}}\}}[t, u \in T]$.
Due to Axiom 5 we know that this T is unique:

$$\exists!_{T\in\{\widetilde{\mathcal{RL}},\widetilde{\mathcal{FE}},\widetilde{\mathcal{RP}}\}}[t, u \in T]$$

From the definition of TypeClass finally follows:

$$\mathsf{TypeClass}(t) = \mathsf{TypeClass}(u) \qquad \square$$

4.3. Type relatedness

Types may be related to each other. Traditionally [56] types are regarded as being related if their populations may overlap. The relatedness of types may be due to several reasons, for instance, sub-typing. However, other reasons may exist as well [56].

Formally, type relatedness is modeled as a relation: $\sim \subseteq \mathcal{TP} \times \mathcal{TP}$. Type relatedness is reflexive and symmetric:

Axiom 6 (*Reflexive*). If $t \in \mathcal{TP}$, then $t \sim t$.

Axiom 7 (*Symmetric*). If $t \sim u$, then $u \sim t$.

As type relatedness expresses the fact that populations of types may overlap, the fact that a specific element has multiple types associated can be treated as evidence for type relatedness:

Axiom 8 (*Evidence for type-relatedness*). $e\ \mathsf{HasType}\ t \wedge e\ \mathsf{HasType}\ u \Rightarrow t \sim u$.

An immediate result of the above axiom is:

Lemma 2. $t \nsim u \Rightarrow \pi(t) \cap \pi(u) = \emptyset$.

Proof. Let $t, u \in \mathcal{TP}$ such that $t \nsim u$.

Let us assume $\pi(t) \cap \pi(u) \neq \emptyset$. Thus, there must be an $x \in \pi(t) \cap \pi(u)$.
On the basis of this assumption and the definition of π we know that
$x\ \mathsf{HasType}\ t \wedge x\ \mathsf{HasType}\ u$.
From Axiom 8 it follows that $t \sim u$, which contracts with $t \nsim u$.
Hence, the assumption that $\pi(t) \cap \pi(u)$ is *not* empty does not hold. □

4.4. Sub-types

A specific class of type relatedness is concerned with specializations between types. For example, a representation of type XML is also a representation of type ASCII. We therefore introduce the relationship $\mathsf{SubOf} \subseteq \mathcal{TP} \times \mathcal{TP}$ to cater for such sub-typing relationships, with the intuition that if $t\ \mathsf{SubOf}\ u$, we consider t to be a sub-type of u.

Sub-typing is transitive and irreflexive:

Axiom 9 (*Transitive sub-typing*). $t\ \mathsf{SubOf}\ u \wedge u\ \mathsf{SubOf}\ v \Rightarrow t\ \mathsf{SubOf}\ v$.

Axiom 10 (*Irreflexive sub-typing*). $\neg(t\ \mathsf{SubOf}\ t)$.

Consider for example the situation where $t = \text{ASCII}$, $u = \text{SGML}$ and $v = \text{XML}$. An SGML document is—by definition—also valid ASCII. Furthermore, XML is defined as a subset of SGML. From

Axiom 9 it follows that an XML document is also an ASCII document. Furthermore, from Axiom 10 it follows that ASCII is not a subset of ASCII.

Typing of elements is inherited from sub-type to super-type:

Axiom 11 (*Inheritance of sub-types*). e HasType $t \wedge t$ SubOf $u \Rightarrow e$ HasType u.

Sub-typing and type relatedness influence each other. Firstly, type relatedness is preserved by the sub-typing hierarchy:

Axiom 12 (*Preservation of type relatedness*). $t \sim u \wedge u$ SubOf $v \Rightarrow t \sim v$.

Type related sub-types must have inherited their type relatedness from some supertype. More precisely, if u is a sub-type that is type related to some type t, then u must have a supertype that was already type related to t. Formally this leads to:

Axiom 13 (*Origination of type relatedness*). $t \sim u \wedge \exists_v[v$ SubOf $u] \Rightarrow \exists_v[t \sim v \wedge v$ SubOf $u]$.

To illustrate the above axiom, consider the following (fictive) situation: there are three types, SGML, XML and ASCII. Furthermore, XML is a sub-type of SGML, and ASCII and XML are type related. This is graphically depicted in Fig. 6(a). Axiom 13 enforces that ASCII and SGML must also be type related. Fig. 6(b) illustrates this. For a more detailed discussion on an abstract theory for type relatedness and inheritance, see [67]. An immediate consequence of the above axioms is:

Corollary 2. t SubOf $u \Rightarrow t \sim u$.

This axiom states that if some document e (e.g. a webpage in HTML) has a certain type (in this case HTML), and if this type has a 'supertype' (e.g. ASCII in this case), it follows that this document is also of this supertype. Furthermore, it can be proven that sub-typing obeys the partioning of types:

Lemma 3. tSubOf$u \Rightarrow$ TypeClass(t) = TypeClass(u).

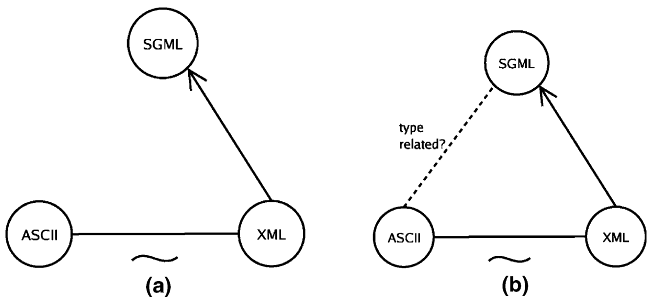

Fig. 6. Illustrating Axiom 13.

Proof. Let t SubOf u.

From Axiom 4 follows: $\exists_e[e\ \mathsf{HasType}\ t]$.
Using Axiom 11 we immediately have:

$\exists_e[e\ \mathsf{HasType}\ t \wedge e\ \mathsf{HasType}\ u]$

From Lemma 1 then follows: $\mathsf{TypeClass}(t) = \mathsf{TypeClass}(u)$. □

For example, if t is XML, u is ASCII, and e is an XML-document (containing e.g. sales figures of a company in a certain month). Then Lemma 3 states that both XML and ASCII have the same *typeclass*; in this case $\widetilde{\mathscr{RP}}$.

Recall from the previous section that *representation types* play a special role in abstracting from the heterogeneity of representations. In line with [64], we presume that these types actually have the form of a many sorted algebra. Formally, if $r \in \widetilde{\mathscr{RP}}$ is a representation type, then Σ_r is presumed to be the signature of the many sorted algebra that is associated to r. Usually, the signature of r will be of the form

$$\Sigma_r = \langle S; f_1, f_2 \ldots \rangle$$

where S is the carrier set—the set of values/types that are already known, e.g. *primitives* in the JAVA programming language—and $f_1, f_2 \ldots$ are functions. The domain of these functions correspond to tuples with elements from S, and the range corresponds to elements from S [64]. Consider for example the case where r is the type ASCII, then Σ_{ASCII} is the signature corresponding to the type ASCII. For this type, the carrier set has two elements, $\mathbb{N}$ (all natural numbers) and `Char` (all available characters available in the character set). Furthermore, the signature holds two functions, $\mathsf{Char} : \mathbb{N} \to \mathtt{Char}$ (takes a number $n \in \mathbb{N}$ as parameter and returns the nth character from an ASCII document), and $\mathsf{Len} : \to \mathbb{N}$ (returns the length of an ASCII document). In summary, the signature for ASCII is:

$$\Sigma_{\text{ASCII}} = \langle \{\mathbb{N}, \mathtt{Char}\}; \mathsf{Char} : \mathbb{N} \to \mathtt{Char}, \mathsf{Len} :\to \mathbb{N} \rangle$$

In the case of dynamic resources, the state of the resource needs to be added to the signature of the algebra. As an example of a dynamic resource, let us consider a weather forecasting application. Let Σ_{FC} represent the signature corresponding to the weather forecasting applications FC. Let *Loc* be some domain of locations on earth (for instance GPS coordinates) and let FCState represent the state of the application. The signature for this application could then be:

$$\Sigma_{\text{FC}} = \langle \{\text{FCState}, Loc, \mathbb{N}, \text{ASCII}\}, \text{TodaysForecast} : \text{FCState} \times Loc \times \mathbb{N} \to \text{ASCII} \rangle$$

Note that setting the actual weather parameters, such as air pressure, temperatures, wind speed, etc., by means of which the application may compute the weather forecast are left out of this signature since this signature focuses solely on the information *supply* perspective.

Summary of the elementary concepts:

$$\langle \mathscr{IS}, \mathscr{RP}, \mathscr{FE}, \mathscr{TP}, \Sigma, \mathsf{HasType}, \mathsf{Service}, \mathsf{Representation}, \sim, \mathsf{SubOf} \rangle$$

5. Example

A lot of things are *art*, such as photographs, paintings and sculptures. In the (concise) Oxford English Dictionary, it is described as:

> The expression or application of creative skill and imagination, especially through a visual medium such as painting or sculpture and works produced in this way.

The nice thing about art, is that it is everywhere. It is not just museums that have art. Many people have paintings in their homes, or photographs, or something else they consider to be art. In this example we will use the information system of a museum as a starting point. In this case, each item of art is presumed to be an information service, whereas the mentioned items that are stored in the system are the representations. Consider an exposition about the artist *Lou Reed* with the following items:

IS1 a photograph made by the photographer *Anton Corbijn* (London, 1983);
IS2 the album *New York* (1989);
IS3 a poster about Lou Reed by *Andy Warhol*, using the picture by Anton Corbijn;
IS4 the song *Take a walk on the wild side*, taken from IS2.

In other words, there are four *information services*: $\mathscr{IS} = \{IS1, IS2, IS3, IS4\}$. The system of the museum stores the following information about each of the pieces in the collection:

- a description in HTML that also explains the origin (is it inspired by another work or art, where was it made and why, for what price it was acquired);
- for graphical art pieces, a photograph of the item which is included in the HTML description;
- for audible pieces of art, an MP3 fragment is stored and included in the HTML description;
- a tuple in a database with the artist, the date of creation, etcetera. This tuple is based on the HTML description.

The information services *IS1* and *IS3* have three representations; a HTML description, a photograph (included in the HTML description) and finally a tuple in the database. The information services *IS2* and *IS4* have three representations also; the HTML description, an MP3 fragment and a tuple in the database. We assume that the MP3 fragment associated to *IS2* is the same as the one associated to *IS4*. In other words, the fragment that describes the album happens to be the one song that we also recognize as an information service. Since each information service has three representations associated to it and one of the representations is connected to two information services, there are 11 representations in this case: $\mathscr{RP} = \{Rep1, \ldots, Rep11\}$.

Recall that *features* are 'labels' that are used to connect information services to representations. The features are typed using *feature types*. An example of a feature is "the description in HTML of the photograph that Anton Corbijn made of Lou Reed in 1983". In this case, the feature type is *description*. Another example of a feature is "the full-content representation in EPS of the photograph that Anton Corbijn made of Lou Reed in 1983". The feature type, in this case, is *full-content*. The third feature associated to this information service is "the tuple that describes the

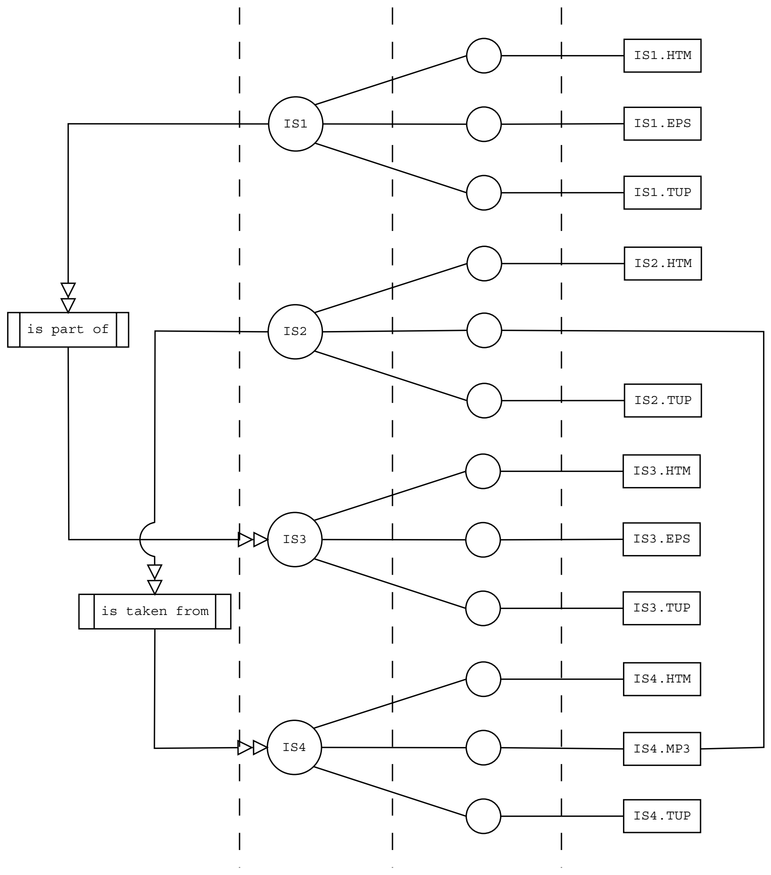

Fig. 7. Illustrating the example (partial).

photograph that Anton Corbijn made of Lou Reed in 1983". This tuple, with the fields *title*, *artist*, *year of creation*, *date of purchase*, *price in dollars* and *remarks* is:

```
('Lou Reed', 'Corbijn, Anton', '1983', '01-Feb-2002', '2500', 'black and white
photograph')
```

The feature type, in this case, is *description*.

There are also two relations to be discovered; "*IS1 is a part of IS3*" and "*IS4 is taken from IS2*". Hence, the two relations *RL1* and *RL2* respectively.

Furthermore $\mathsf{Src}(RL1) = IS1$ and $\mathsf{Dst}(RL1) = IS3$ and also $\mathsf{Src}(RL2) = IS4$ and $\mathsf{Dst}(RL2) = IS2$.

Fig. 7 illustrates the example. The labels for the features, as well as the featuretypes and relationtypes are left out. The double-headed arrows on the left indicate the 'direction' of the relations.

6. Transformations

An important question—that we will attempt to answer in this section—is whether two representations that are associated to a single information service carry the *exact* same information?

And what does the answer to this question imply for the features (and their types)? Can a representation, perhaps, be *derived* from others? An example to clarify this: suppose that we have some art object (e.g. Victory Boogie Woogie), represented as an EPS file on the Web. Unfortunately, we do not have a viewer for EPS files. Instead of EPS, we would like another format such as e.g. JPEG, or a very precise description of this painting in ASCII. The question is: can we somehow generate these representations?

In order to deal with this, we introduce the notion of *Transformations*. In line with the definition in the dictionary, we define a transformation as follows: by means of a transformation, a representation can be transformed into another. Some examples of transformations are:

- A $\LaTeX$ document can be transformed into a PDF document. After the transformation there is a new representation, attached to the same information service. This implies that the feature must have changed as well. Furthermore, the new representation has a different representation type.
- A long article on an art exhibition can be transformed into an abstract. [9] In this case the new representation also belongs to the same information service by means of a new feature. However, instead of the representation type, it's the feature type that has changed this time.
- A description of a webservice in WSDL can be transformed into a keyword list in ASCII. In this example both the representation type and the feature type have been altered by the transformation.
- An photograph in EPS can be transformed into another EPS file with a *lower resolution*. Neither the representation type, nor the feature type have changed in this example.

In Section 6.1 we formally introduce transformations, and extend the theory from the previous sections. Furthermore, we define a *taxonomy* of transformation functions in Section 6.2. Since this taxonomy is still a topic for further research, we only present an outline of it. In Section 6.3 we explain one special type of transformations, most notably the type neutral transformations.

6.1. Formal definition of transformation functions

We model the set of transformation functions $\mathcal{TR}$ to be all transformation functions with signature $T : \mathcal{RP} \rightarrowtail \mathcal{RP}$. Simply put, a transformation function produces a new representation. We require this new representation to belong to *the same* information service as the original representation, even though they are associated to different representations.

Axiom 14 (*$\mathcal{IS}$- neutral transformations*). If $T \in \mathcal{TR}$ and $f \in \mathcal{FE}$, then

$$T(\mathsf{Representation}(f)) = r' \quad \Rightarrow \quad \exists_{f'}[\mathsf{Representation}(f') = r' \wedge \mathsf{Service}(f) = \mathsf{Service}(f')]$$

Fig. 8 graphically depicts this axiom. Recall from Axiom 2 that if two features that point to the same information service as well as the same representation, then these two features must be the same. Using this, we can refine Axiom 14. We can now prove that there is only one feature f' that satisfies $\mathsf{Representation}(f') = r \wedge \mathsf{Service}(f) = \mathsf{Service}(f')$.

[9] See e.g. [68] to see how *lexical chains* can be used for text summarization.

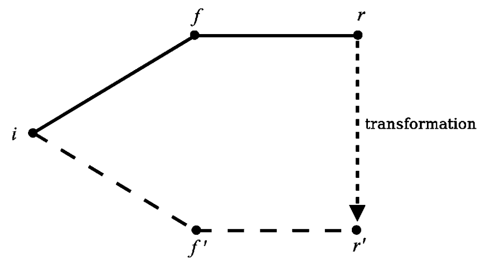

Fig. 8. Transformations are $\mathscr{IS}$-neutral.

Lemma 4. *If* $T \in \mathscr{TR}$, $f \in \mathscr{FE}$, *then*

$$T(\mathsf{Representation}(f)) = r' \quad \Rightarrow \quad \exists!_{f'}[\mathsf{Representation}(f') = r' \wedge \mathsf{Service}(f) = \mathsf{Service}(f')]$$

Proof. Let $T \in \mathscr{TR}$, $f \in \mathscr{FR}$, $r' \in \mathscr{RP}$ such that $T(\mathsf{Representation}(f)) = r'$.

From Axiom 14 it follows that there must exist a feature $f' \in \mathscr{FE}$ such that $\mathsf{Representation}(f') = r'$ and $\mathsf{Service}(f) = \mathsf{Service}(f')$.
Assume there is a second feature $f'' \in \mathscr{FE}$, such that $f' \neq f''$ and $\mathsf{Representation}(f'') = r' \wedge \mathsf{Service}(f) = \mathsf{Service}(f'')$.
In this case we would have: $\mathsf{Representation}(f') = \mathsf{Representation}(f'') \wedge \mathsf{Service}(f') = \mathsf{Service}(f'')$. From Axiom 2 it follows that these two features must be equal: $f' = f''$. □

With this result, we can associate a transformation function between features based on the transformations between representations. Let $T \in \mathscr{TR}$, then we define $\widehat{T} : \mathscr{FE} \rightarrowtail \mathscr{FE}$ as follows:

$$\widetilde{T}(f) \triangleq g \quad \text{such that:} \ g \in \mathscr{FE} \wedge \mathsf{Service}(f) = \mathsf{Service}(g) \wedge T(\mathsf{Representation}(f)) = \mathsf{Representation}(g)$$

From Lemma 4 follows that this g is indeed unique.

6.2. Taxonomy of transformations

In the previous section we explained that transformations operate on *representations*, and that this may have an impact on the feature and/or representation type. In this section we look at the effect that a transformation has on these types. For this purpose, we ignore the difference between feature types and representation types. Each of the classes can therefore be further split up in a case for feature types, and one for representation types. This is illustrated in the following section. The purpose of developing such a taxonomy is to gain further insight in the effects of transformations. Knowing if a transformation has a (negative) informational effect or not may be very important from a user point of view: the tradeoff between the (informational) quality of a representation and (for example) its representation type may not be an easy one!

Recall that representations and features can have more than one type. Assume that $\tau : \mathscr{RP} \rightarrow \wp(\mathscr{TP})$ returns all the types for a given feature or representation. We base our

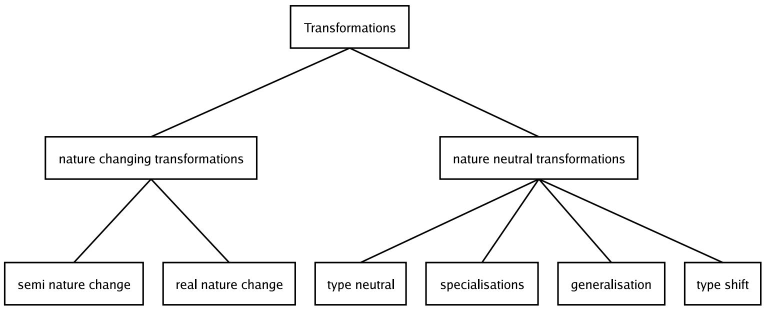

Fig. 9. Taxonomy of transformations.

taxonomy on the question: what happens to the types if a transformation is performed on a given representation. Fig. 9 is a graphical representation of the taxonomy:

Nature changing transformations: This type of transformations have a big impact on the types. The set of types associated to the original representation may not overlap with the set of types associated to the representation that results after a transformations:

$$\forall_{x \in \mathsf{Domain}(T)}[\tau(x) \cap \tau(T(x)) = \emptyset]$$

This type of transformations comes in two flavors:

Semi-nature change. If a transformation function is semi-nature changing, an additional rule must hold: there must be at least one type in the set associated to the input representation that is type-related to a type in the set associated to the resulting representation:

$$\forall_{x \in \mathsf{Domain}(T)}[\tau(x) \cap \tau(T(x)) = \emptyset \wedge \exists_{a \in \tau(x), b \in \tau(T(x))}[a \sim b]]$$

Real nature change. A transformation function is said to result in a real nature change if there are no elements in the set of types associated to the input representation that are type related to an element in the set of types after the transformation:

$$\forall_{x \in \mathsf{Domain}(T)}[\tau(x) \cap \tau(T(x)) = \emptyset \wedge \forall_{a \in \tau(x)}[\neg \exists_{b \in \tau(T(x))}[a \sim b]]]$$

Nature neutral transformations: This class of transformations is the counterpart of the nature changing transformations. In this case, the effects of the transformations on the types are less drastic; the two sets of types overlap:

$$\forall_{x \in \mathsf{Domain}(T)}[\tau(x) \cap \tau(T(x)) \neq \emptyset]$$

This class of transformations, again, comes in several flavors:

Type neutral. A transformation is neutral with regard to its types if the input representation and output representation have the same set of types attached to them:

$$\forall_{x \in \mathsf{Domain}(x)}[\tau(x) = \tau(T(x))]$$

Specialization. The result of a transformation is specialization if the set of types of the input representation is a proper subset of the set of types associated to the output representation.

That is, by means of the transformation, we move to a more specific type which is added to the set of types:

$$\forall_{x \in \mathsf{Domain}(x)}[\tau(x) \subset \tau(T(x))]$$

Generalization. This set of transformations is the opposite of specialization. A transformation is said to be a generalization if we move to a less specific type. That is, a type is removed from the set of types associated to a representation:

$$\forall_{x \in \mathsf{Domain}(x)}[\tau(x) \supset \tau(T(x))]$$

Type shift. In order to explain this concept we introduce the following notation. Assume that x and y are sets:

$$x \between y \triangleq \begin{cases} x \cap y & \neq \emptyset \wedge \\ x - y & \neq \emptyset \wedge \\ y - x & \neq \emptyset \end{cases}$$

In other words, the intersection of x and y, as well as the differences $x - y$ and $y - x$ must be non-empty. The result of a transformation is said to be a type shift if

$$\forall_{x \in \mathsf{Domain}(x)}[\tau(x) \between \tau(T(x))]$$

6.3. Type neutral transformations

We explained that transformation functions transform one representation into the other. Furthermore, a new feature connects the new representation to the same information as the original representation was associated to (see Fig. 8). Two remaining questions are:

(1) Has the *representation type* changed?
(2) Has the *feature type* changed?

In line with these two questions we can define the following classes of transformations:

Representation type neutral. A transformation $T \in \mathscr{TR}$ is neutral with regards to representation types iff:

$$\forall_{r \in \mathsf{domain}(T), t \in \mathscr{TP}}[r \ \mathsf{HasType}\ t \iff T(r)\ \mathsf{HasType}\ t]$$

Note that the resulting representation must have the same *set* of representation types as the original representation. In order to understand why this must be true, recall from Section 4.2 that HasType is a relation, and that we allow types to be related to each other. For example, the representation type XML is a sub-type of SGML, which can be considered to be a sub-type of ASCII. If we were to transform a representation from XML to ASCII by simply removing all the XML-tags, then the resulting representation would only have the type ASCII. This transformation is *not* neutral with regard to its representation type.

Feature type neutral. A transformation $T \in \mathscr{TR}$ is neutral with regards to feature types iff

$$\forall_{f \in \mathsf{domain}(\hat{T}), t \in \mathscr{TP}}[f\ \mathsf{HasType}\ t \iff \widetilde{T}(f)\ \mathsf{HasType}\ t]$$

Similar to representation types, feature types can be interrelated. As a result, we say that a transformation is neutral with regard to feature types, if the feature that results after a transformation has the same set of types as the original feature.

To illustrate the transformations that are neutral with regard to their representation type we consider the following situation: the information service, in this case, is a 300-page report with a rather vague title. There is only one feature ("report with number RF-4476 represented in ASCII") and one representation (`rf-4476. txt`). As a manager, we have very little time and since we are not sure what the report is all about, and wish to see a list with the 10 most important keywords before spending any time on reading it. In order to achieve this, we transform this representation into another ASCII-file. The new representation has the same representation type still (ASCII), but the feature type has changed from "full-content" to "keyword-list".

As an example of the feature type neutral transformations think of the following. A company has moved from the Microsoft Windows platform to Linux all together e.g. for security reasons. As a result, however, some file formats (such as Microsoft Access) can no longer be opened. One of the suppliers sends such a file every month with special offerings and such. To be able to see these offerings, a transformation is applied to this representation. With this transformation, the representation with type "MS-ACCESS" is transformed into another representation with type "ANSII-SQL". This transformation is neutral with regard to the feature type, since this type does not change. However, it is not neutral with regard to the representation type since this *does* change.

It is also possible that a transformation function is neutral with regard to both feature type and representation type. Consider for example the situation where the information service is, again, the photograph that Anton Corbijn made of Lou Reed. The feature is 'description of the photograph in PDF'. The feature type in this is case is 'description'. Furthermore, the representation is a file called `aboutRead. pdf` with type PDF. Using a transformation that optimizes PDF for usage on the Web, we transform this representation into a new PDF-file and associate a new feature to it. In this case, neither the feature type (which still is 'description') and the representation type (PDF) have changed.

The last class of transformations that we consider is neither neutral with regard to feature types, nor with regard to representation types. Hence, in this case, both the feature type and the representation type change with the transformation. For example, the information service is the (already mentioned) photograph that Anton Corbijn made of Lou Reed. The feature we consider is 'full content representation in EPS'. The feature type is 'full content'. The representation, in this case, is `lou-reed. eps` with type EPS. We perform a combined transformation by making a cut-out of the original photo (we preserve only the eyes) and store it in a JPEG file.

7. *Vimes*

In the previous sections we introduced a conceptual model for information supply and introduced the notion of transformations which allows us to deal with heterogeneity of representations and features. In this section we illustrate how these transformations can be used in an information

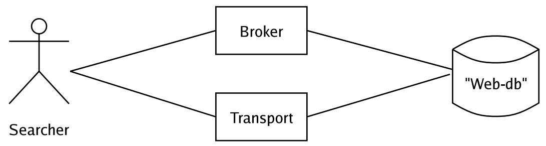

Fig. 10. Logical view on *Vimes*.

retrieval (IR) setting by presenting the *Vimes*[10] architecture. We will do so by using the transformations as a means to materialize a broader notion of relevance.

One of the basic functions of any information retrieval (IR) system is *relevance ranking*: the (characterizations of) resources are ranked such that the resources that are "most relevant" are listed first, and the ones that are least relevant are listed last. In [69] an overview is given of metrics that are used to determine the relevancy of a Web-document with regard to a query. Furthermore, it is pointed out that relevancy involves more than *topical relevance*; other attributes of resources (such as its quality and price, but also its feature- and representation type) are important as well.

7.1. Architectural overview

From a user-perspective, the Web (and thus, information supply as defined in previous sections) can be seen as a large database with "information" in the form of resources. In the *Vimes* architecture, the entries in the database are the population of our reference model, i.e. we characterize the Web in terms of our reference model. The reference model is represented directly in terms of the database tables. This allows us to search in this database with queries like: "give me all representations about *Semantic Web* that refer to the website of W3C on RDF".

The *logical view* on searching the Web with our prototype, *Vimes*, is represented in Fig. 10: The "Web-db" component in this logical view needs further discussion. This "database for the Web" receives, in terms of our model, two kinds of requests:

(1) *get*-requests return a specific data resource, if available. These pertain to the *transport* part of the information market.
(2) *query*-requests show a list of all data resources that are *apt*. These pertain to the *brokering* component of the information market.

Fig. 11 zooms in on this database. The cloud at the bottom represents the Web while the *extensional database* represents the part of the Web that we "know about". I.e. it represents the part of the Web that is characterized in terms of our meta-model. We do not intend to develop and implement yet another strategy for estimating the *topical relevance* of representations. For this, we use GOOGLE as a plug-in.

Furthermore, the transformations, as described in Section 6, provide a way to broaden our knowledge. In general, one can distinguish between an extensional database and intentional

[10] For a more elaborate introduction see [18].

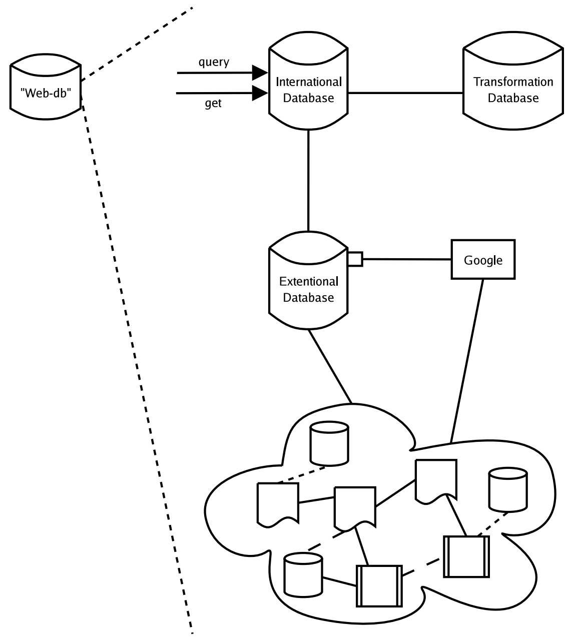

Fig. 11. Implementation view on *Vimes*.

database [21,70]. The extensional database corresponds to the a set of basic facts known about the world, whereas the intentional database represents the facts that may be derived from the extensional database by applying inference rules. The transformations can be regarded as inference rules on the extensional database (information supply as we know it), resulting in a larger intentional database. In summary: *Vimes* offers users the possibility to search the Web in terms of our meta-model. For topical relevance, though, we use GOOGLE.

To be able to deal with this broad definition of relevance (i.e. *aptness*) in practice, we introduce a modular architecture for retrieval with the following components:

Transformation broker. Using the broad definition of relevance, resources have to be *topically relevant* as well as conform to other criteria such as a proper representation type and/or feature type. If these types are 'wrong' then this may be corrected by applying a transformation to the representation. The transformation broker does this for us.

Vimes-broker. The main interface for users seeking information. It will interact with the profile-repository, the transformation broker and search engines on the Web. [11]

Finally, to bring the architecture to live, we provide a sample session of what the retrieval process may look like, when using a system based on this architecture. Let us assume that a user contacts the *Vimes*-broker and enters the following query into the system

[11] Essential to our architecture is the broker's ability to interact with online search engines such as Yahoo and GOOGLE to determine which representations are topically relevant.

Give me all representations about RDF that refer to the Website of the W3C on this topic: http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/. Present the results in PDF format.

Our broker dissects this query into the following elements:

- the *topic* of the query is RDF;
- only representations with a *Relation* to a specified Website are allowed;
- only allow the PDF format.

At this point, the system will use GOOGLE to find those representations that are topically relevant. After this step is completed, the system will verify which of these representations actually have a relation to the specified website. Before presenting the list of representations that conform to these two constraints, representations may need to be transformed to PDF. Once this is done the final list of *apt* representations is presented to the user.

8. Conclusions and future research

With the apparent rise of the Web, huge amounts of information are available to us these days. But it does not stop there. More and more applications are connected to the Web so that people can access them from their homes. Examples of these are Airline reservation services, Library applications and so on. The amount of information available to us today is so overwhelming that the term *information overload* is indeed an apt description.

With the apparent rise of the Web, much research has been invested in making sure we can *find* the information we need in an efficient manner. In this respect, fields of research are information retrieval, library applications, meta-data efforts, (relational) databases and so on. After carefully studying these fields (Section 2), we feel that these approaches are primarily implementation oriented (in the sense that they do not abstract from underlying technological considerations) or are rather exclusively geared towards a specific class of information supply such as structured information, textual information, etc.

In order to overcome this deficiency, we introduced our view on information supply in the form of a conceptual model. Section 3 informally introduces the model, which is based on the notion that the actual things that we see on the Web are *representations*. Since several representations can convey the same information, we introduced the notion of *information services*. Information service, in a way, is the information decoupled from the medium it is represented on. It is apparent that a relation exists between information services and their underlying representations. We characterize this information with *features*. For example, with a feature we can express the fact that one representation represents the 'full content' of an information service, whereas another representation is 'merely an abstract'. In Section 4 we formalized the reference model.

With the model in place, we introduced the notion of *transformations* in Section 6: by means of a transformation, a representation can be transformed into another. Transformations allow us to make sure the user is presented with the information he wants, in the form he wants and in the format that he wants. Suppose a relevant information service is found, but the feature type is ‘wrong’ (we want an abstract, not the full text) and the representation type is wrong (we want a PDF-file instead of a Word-document). With transformations we can ‘solve’ this problem easily.

Coming up with a nice model is one thing. Testing whether it works in practice, though, is another. We introduced the architecture of *Vimes*, a prototype retrieval system which deals with heterogeneity of representations and feature types using transformations.

Even though our model seems solid as well as promising (with regard to e.g. its application in the field of information retrieval), much more work needs to be done. Firstly, we need to investigate our typing mechanism more thoroughly. More specifically, we need to come up with a taxonomy of types as well as a way to describe them. Work along these lines is in progress.

We also need to describe a constraint and query language. With such a language we are able to reason about the instances in the population of our model and specify constraints on them. An example of such a constraint could be that for each information service there must be a representation such that the associated feature type is “full service” (which, in the case of text, will be “overloaded” with full-text). This language must be flexible enough to enable us to reason about (the results of) transformations as well.

The transformations in itself also need further investigation. In this article we have presented the taxonomy of transformations and worked out the details of one specific class within this taxonomy. Describing the other classes of transformations is part of future research still.

Last but not least, we want to practically apply our model in the field of information retrieval. In the (near) future, we will also implement the *Vimes* retrieval-system.

References

[1] M. Sahami, S. Yusufali, M.Q. Baldonado, Sonia: a service for organizing networked information autonomously, in: I. Witten, R. Akscyn, F.M. Shipman (Eds.), Proceedings of DL-98, 3rd ACM Conference on Digital Libraries, ACM Press, New York and Pittsburgh, 1998, pp. 200–209.

[2] S. Hoppenbrouwers, H. Proper, Knowledge discovery, de zoektocht naar verhulde en onthulde kennis (in Dutch), DB/Magazine 7 (1) (1999) 21–25.

[3] M. Day, Resource discovery, interoperability and design preservation, some aspects of current metadata research and development, VINE (2000) 35–48.

[4] S. Weibel, J. Godby, E. Miller, R. Danierl, Metadata workshop report, Dublin, OH, March 1995.

[5] M.P. Papazoglou, H.A. Proper, J. Yang, Landscaping the information space of large multi-database networks, Data Knowledge Engineering 36 (3) (2001) 251–281.

[6] H.A. Proper, P.D. Bruza, What is information discovery about? Journal of the American Society for Information Science 50 (9) (1999) 737–750.

[7] T. Berners-Lee, J. Hendler, O. Lassila, The semantic web, a new form of web content that is meaningful to computers will unleash a revolution of new possibilities, Scientific American 284 (5) (2001) 34–43.

[8] J. Heflin, J. Hendlef, S. Luke, Shoe: a knowledge representation language for internet applications, Technical Report DAAL01-97-K0135, Institute for Computer Studies, University of Maryland, supported by the Army Research Laboratory, 1999.

[9] J. Heflin, J. Hendler, Searching the web with shoe, in: Artificial Intelligence for Web Search, papers from the AAAI Workshop WS-00-01, AAAI Press, 2000, pp. 35–40.

[10] O. Lassila, R.R. Swick, Resource description framework (rdf) model and syntax specification, Recommendation, W3C, <http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/>, last checked: 29-July-2003, February 1999.

[11] oasis-open, <http://www.oasis-open.org/cover/daml.html>, DARPA Agent Markup Language (DAML), last checked: 29-July-2003, May 2003.

[12] I. Horrocks, A Denotational Semantics for Standard OIL and Instance OIL, University of Manchester, Department of Computer Science, UK, last checked: 29-July-2003, November 2000.

[13] T. Berners-Lee, Universal resource identifiers in www, Technical Report RFC 1630, IETF Network Working Group, http://www.ietf.org/rfc/rfc1630.txt, last checked: 13-August-2002, June 1994.

[14] S. Pepper, S. Schwab, Curing the web's identity crisis, Technical Report, Ontopia, 2003. Available from <http://www.ontopia.net/topicmaps/materials/identitycrisis.html>.

[15] Information processing systems—concepts and terminology for the conceptual schema and the information base, ISO/TR 9007:1987 (1987). Available from: http://www.iso.org.

[16] E. Yourdon, Modern Structured Analysis, Prentice-Hall, Englewood Cliffs, NJ, 1989.

[17] T. Halpin, Information Modeling and Relational Databases, From Conceptual Analysis to Logical Design, Morgan Kaufman, 2001, ISBN 1-55860-672-6.

[18] B.V. Gils, E.D. Schabell, User-profiles for information retrieval, in: Proceedings of the 15th Belgian–Dutch Conference on Artificial Intelligence (BNAIC'03), Nijmegen, 2003.

[19] Committee on Information Technology Strategy for the Library of Congress, Washington, DC 20055, LC21, A Digital Strategy for the Library of Congress, http://books.nap.edu/html/lc21/, last checked: 05-May-2003, 2000.

[20] T. Connolly, C. Begg, Database Systems, a Practical Approach to Design, Implementation and Management, 2nd ed., Addison-Wesley, 2002, ISBN 0-201-70857-4.

[21] J. Ullman, Principles of Database and Knowledge-base Systems, Computer Science Press, Rockville, MD, 1989, ISBN 071678162X.

[22] A.O. Mendelzon, T. Milo, Formal models of web queries, in: Proceedings of 16th Symp. on Principles of Database Systems—PODS 97, 1997, pp. 134–143.

[23] W.J. Premerlani, J.E. Rumbaugh, M.R. Blaha, T.A. Varwig, An object-oriented relational database, Communications of the ACM 33 (11) (1990) 99–109.

[24] M. Zand, V. Collins, D. Caviness, A survey of current object-oriented databases, ACM SIGMIS Database 26 (1) (1995) 14–29.

[25] K.E. Smith, S.B. Zdonik, Intermedia: a case study of the differences between relational and object-oriented database systems, in: Conference Proceedings on Object-Oriented Programming Systems, Languages and Applications, ACM Press, Orlando, FL, 1987, pp. 452–465.

[26] R.W.V.D. Pol, Knowledge-based query-formulation in information retrieval, Ph.D. thesis, University of Maastricht, 2000, ISBN 90-801577-4-9.

[27] G. Salton, M.J. McGill, Introduction to Modern Information Retrieval, McGraw-Hill, New York, 1983.

[28] C.V. Rijsbergen, Information Retrieval, 2nd ed., Butterworths, London, UK, 1979.

[29] R. Baeza-Yates, B. Ribeiro-Neto, Modern Information Retrieval, Addison-Wesley, 1999, ISBN 0-201-39829-X.

[30] H. Paijmans, Explorations in the document vector model of information retrieval, Ph.D. thesis, Tilburg University, Tilburg, The Netherlands, 1999, ISBN 90-361-0024-0.

[31] P.D. Bruza, Hyperindices: a novel aid for searching in hypermedia, in: A. Rizk, N. Streitz, J. Andre (Eds.), Proceedings of the European Conference on Hypertext—ECHT 90, Cambridge University Press, Cambridge, UK, 1990, pp. 109–122.

[32] P.D. Bruza, T.P.V.D. Weide, Two level hypermedia—an improved architecture for hypertext, in: A. Tjoa, R. Wagner (Eds.), Proceedings of the Data Base and Expert System Applications Conference (DEXA 90), Springer-Verlag, Vienna, Austria, 1990, pp. 76–83.

[33] P. Bruza, Stratified information disclosure: a synthesis between hypermedia and information retrieval, Ph.D. thesis, University of Nijmegen, Nijmegen, The Netherlands, 1993.

[34] S. Brin, L. Page, The anatomy of a large-scale hypertextual web search engine, Computer Networks and ISDN Systems 30 (1–7) (1998) 107–117.

[35] L. Page, S. Brin, R. Motwani, T. Winograd, The pagerank citation ranking: bringing order to the web, Technical Report, Stanford Digital Library Technologies Project, 1998.
[36] R. Schatz et al., Interactive term suggestion for users of digital libraries, in: First ACM International Conference on Digital Libraries, Bethesda, MD, 1996, pp. 126–133.
[37] L. Faulstich, The hyperview approach to the integration of semistructured data, Ph.D. thesis, Institute of Computer Science, Free University Berlin, partial, grabbed from Citeseer, 2000.
[38] B.M. Leinder, The scope of Digital Library, <http://www.dlib.org/metrics/public/papers/dig-lib-scope.html>, last checked: 01-August-2003, October 1998.
[39] L. Feng, J. Hoppenbrouwers, M. Jeusfeld, Towards knowledge-based digital libraries, SIGMOD Record 30 (1) (2001) 41–46.
[40] W.Y. Arms, Key concepts in the architecture of the digital library, Technical Report, Corporation for National Research Initiatives, d-Lib Magazine, July 1995.
[41] OASIS, <http://xml.coverpages.org/sgml.html>, Standard Generalized Markup Language (SGML), last checked: 11-May-2003, July 2002.
[42] Information Processing—Text and Office Systems—Standard General MarkUp Language (SGML), 1986, ISO 8879:1986. URL <http://www.iso.org>.
[43] S. Weibel, J. Kunze, C. Lagoze, M. Wolf, Dublin core metadata for resource discovery, Technical Report RFC 2413, Internet Engineering Task Force (IETF), <http://ww.ietf.org/rfc/rfc2413.txt>, last checked: 15-May-2003, 1998.
[44] Object Management Group (OMG), <http://www.omg.org/technology/cwm/>, Common Warehouse Metamodel (CWM) metamodel, version 1.0 Edition, last checked: 28-January-2004, February 2001.
[45] K. Januszewski, E. Mooney, UDDI Version 3 Features List, Oasis, <http://www.uddi.org/pubs/uddi_v3_features.htm>, last checked: 28-January-2003, 2002.
[46] The stencil group, <http://www.uddi.org/pubs/the_evolution_of_uddi_20020719.pdf>, The Evolution of UDDI, last checked: 28-January-2003, UDDI.org White Paper, July 2002.
[47] OASIS, http://www.ebxml.org/geninfo.htm, ebXML—Enabling a global electronic market, last visited: 28-January-2004, 2003.
[48] D. Connolly, F. van Harmelen, I. Horrocks, D.L. McGuinness, L.A. Stein, I. Lucent Technologies, DAML + OIL Reference Description, W3C, <http://www.w3.org/TR/daml+oil-reference>, last checked: 07-August-2003, December 2001.
[49] D.L. McGuinness, F. van Harmelen, OWL Web Ontology Language Overview, W3C Proposed Recommendation, W3C, <http://www.w3.org/TR/2003/PR-owl-features-20031215/>, December 2003.
[50] P. Chen, The entity-relationship model: towards a unified view of data, ACM Transactions on Database Systems 1 (1) (1976) 9–36.
[51] M. Gogolla, An Extended Entity-Relationship Model: Fundamentals and Pragmatics, in: Lecture Notes in Computer Science, vol. 767, Springer-Verlag, Berlin, 1994.
[52] G. Booch, J. Rumbaugh, I. Jacobson, The Unified Modelling Language Used Guide, Addison-Wesley, Reading, MA, 1999.
[53] R. Meersman, The RIDL conceptual language, Research Report, International Centre for Information Analysis Services, Control Data Belgium, Inc., Brussels, Belgium, 1982.
[54] U. Hohenstein, G. Engels, SQL/EER-syntax and semantics of an entity-relationship-based query Language, Information Systems 17 (3) (1992) 209–242.
[55] U. Hohenstein, G. Engels, Formal semantics of an entity-relationship-based query language, in: Proceedings of the Ninth International Conference on the Entity-Relationship Approach, Lausanne, Switzerland, 1990.
[56] A.T. Hofstede, H. Proper, T.V.D. Weide, Formal definition of a conceptual language for the description and manipulation of information models, Information Systems 18 (7) (1993) 489–523.
[57] A. Bloesch, T. Halpin, ConQuer: a conceptual query language, in: B. Thalheim (Ed.), Proceedings of the 15th International Conference on Conceptual Modeling (ER'96), Lecture Notes in Computer Science, vol. 1157, Springer-Verlag, Cottbus, 1996, pp. 121–133.
[58] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana, Web services description language (wsdl) 1.1, W3c note, World Wide Web Consortium, <http://www.w3.org/TR/wsdl>, last checked: 19-May-2003, March 2001.

[59] J. Conklin, Hypertext: an introduction and survey, IEEE Computer 20 (9) (1987) 17–41.
[60] V. Bush, As We May Think, The Atlantic Monthly.
[61] Research Index Citeseer, <http://citeseer.nj.nec.com>, last checked: 19-May-2003, 1997.
[62] J. Hoppenbrouwers, H. Paijmans, Invading the fortress: how to besiege reinforced information bunkers, in: Advances in Digital Libraries, 2000, pp. 27–38.
[63] N. Borenstein, N. Freed, Mime: multipurpose internet mail extensions, Technical Report RFC 1341, IETF Network Working Group, <http://www.ietf.org/rfc/rfc1341.txt>, last checked: 19-May-2003, June 1992.
[64] K. Bruce, P. Wegner, An algebraic model of subtype and inheritance, in: F. Bancilhon, P. Buneman (Eds.), Advances in Database Programming Languages, Frontier Series, Addison-Wesley, ACM Press, Reading, MA, 1990, pp. 75–96.
[65] B.V. Gils, H.A. Proper, P. van Bommel, T.V.D. Weide, Transformations in information supply, Technical Report, Nijmegen Institute for Information and Computing Science, University of Nijmegen, Nijmegen, The Netherlands, March 2004.
[66] W.K. Grassman, J.-P. Tremblay, Logic and Distrete Mathematics, Prentice-Hall, Upper Saddle River, NJ, 1996, ISBN 0-13-501206-6.
[67] E. Proper, A theory for conceptual modelling of evolving application domains, Ph.D. thesis, University of Nijmegen, Nijmegen, The Netherlands, 1994, ISBN 90-9006849-X.
[68] R. Barzilay, M. Elhadad, Using lexical chains for text summarization, in: Proceedings of the Intelligent Scalable Text Summarization Workshop (ISTS'97), Madrid, Spain, 1997.
[69] D. Dhyani, W.K. Ng, S.S. Bhowmick, A survey of web metrics, ACM Computing Surveys (CSUR) 34 (4) (2002) 469–503, iSSN 0460-0300.
[70] C.J. Date, An Introduction to Database Systems, 8th ed., Addison-Wesley, 2003, ISBN 0-321-18956-6.

B. van Gils received his Master's degree from the University of Tilburg (Tilburg, the Netherlands) at March 1st, 2002. The topic of his master thesis was "Application of Semantic Matching in Enterprise Application Integration". He currently works on his doctoral thesis. His research interests are Information Retrieval, Web information systems and (information) modeling and more specifically "aptness on the information market".

H.A. (Erik) Proper is a Professor at the University of Nijmegen, The Netherlands. He has co-authored several journal papers, conference publications and books. His main research interests include system theory, system architecture, business/IT alignment, conceptual modeling, information retrieval and information discovery. He received his Master's degree from the University of Nijmegen, The Netherlands in May 1990, and received his Ph.D. (with distinction) from the same University in April 1994. In his Doctoral thesis he developed a theory for conceptual modeling of evolving application domains, yielding a formal specification of evolving information systems. After receiving his Ph.D., he became a senior research fellow at the Computer Science Department of the University of Queensland, Brisbane, Australia. During that period he also conducted research in the Asymetrix Research Lab at that University for Asymetrix Corp, Seattle, Washington. In 1995 he became a lecturer at the School of Information Systems from the Queensland University of Technology, Brisbane, Australia. During this period he was also seconded as a senior researcher to the Distributed Systems Technology Centre (DSTC), a Cooperative Research Centres funded by the Australian government. From 1997 to 2001, he worked in industry. First as a consultant at Origin, Amsterdam, The Netherlands, and later as a research consultant and principal scientist at the Ordina Institute for Research and Innovation, Gouda, The Netherlands. In June 2001, he returned to academia, where he became an adjunct Professor at the University of Nijmegen, on the subject of "Architecture-Driven Information Systems Engineering". In September 2002, he obtained a full-professorship at the University of Nijmegen on the field of "Information Systems Science (Informatiekunde)".

Patrick van Bommel received his Masters degree in Computer Science in 1990, and the degree of Ph.D. in Mathematics and Computer Science, from the University of Nijmegen, the Netherlands in 1995. He is currently assistant professor at the University of Nijmegen. His main research interests include information modeling and information retrieval. In information modeling, he particularly deals with modeling techniques for information systems, data models for hyperdocuments and web sites, equivalence and transformation of data models, the transformation of conceptual data models into efficient database representations, and the transformation of database populations, operations and integrity constraints. In information retrieval his main research interests include document data modeling, WWW based retrieval/filtering, document characterization languages, and digital libraries.