

Rodrigo Dias Arruda Senra

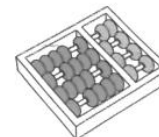
**“Organization is Sharing: From eScience to Personal
Information Management”**

***“Organização é Compartilhamento: de eScience para Gestão de
Informação Pessoal.”***

**CAMPINAS
2012**



University of Campinas
Institute of Computing



Universidade Estadual de Campinas
Instituto de Computação

Rodrigo Dias Arruda Senra

**“Organization is Sharing: From eScience to Personal
Information Management”**

Supervisor/Orientador(a): **Profa. Dra. Claudia Bauzer Medeiros**
Institute of Computing - UNICAMP


***“Organização é Compartilhamento: de eScience para Gestão de
Informação Pessoal.”***

PhD Thesis presented to the Post Graduate Program of the Institute of Computing of the University of Campinas to obtain a PhD degree in Computer Science.

THIS VOLUME CORRESPONDS TO THE FINAL VERSION OF THE THESIS DEFENDED BY RODRIGO DIAS ARRUDA SENRA, UNDER THE SUPERVISION OF PROFA. DRA. CLAUDIA BAUZER MEDEIROS
INSTITUTE OF COMPUTING - UNICAMP.

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Computação da Universidade Estadual de Campinas para obtenção do título de Doutor em Ciência da Computação.

ESTE EXEMPLAR CORRESPONDE À VERSÃO FINAL DA TESE DEFENDIDA POR RODRIGO DIAS ARRUDA SENRA, SOB ORIENTAÇÃO DE PROFA. DRA. CLAUDIA BAUZER MEDEIROS
INSTITUTO DE COMPUTAÇÃO - UNICAMP.


Supervisor's signature / Assinatura do(a) Orientador(a)

CAMPINAS
2012

FICHA CATALOGRÁFICA ELABORADA POR
MARIA FABIANA BEZERRA MULLER - CRB8/6162
BIBLIOTECA DO INSTITUTO DE MATEMÁTICA, ESTATÍSTICA E
COMPUTAÇÃO CIENTÍFICA - UNICAMP

Se59o Senra, Rodrigo Dias Arruda, 1974-
Organização é compartilhamento : de eScience para gestão de
informação pessoal / Rodrigo Dias Arruda Senra. – Campinas, SP :
[s.n.], 2012.

Orientador: Claudia Bauzer Medeiros.
Tese (doutorado) – Universidade Estadual de Campinas,
Instituto de Computação.

1. Banco de dados. 2. Banco de dados - Organização. 3.
Metadados. 4. Política ambiental. I. Medeiros, Claudia Maria
Bauzer, 1954-. II. Universidade Estadual de Campinas. Instituto de
Computação. III. Título.

Informações para Biblioteca Digital

Título em inglês: Organization is sharing : from eScience to personal
information management

Palavras-chave em inglês:

Databases

Databases - Organization

Metadata

Environmental management

Área de concentração: Ciência da Computação

Titulação: Doutor em Ciência da Computação

Banca examinadora:

Claudia Bauzer Medeiros [Orientador]

Ariadne Maria Brito Rizzoni Carvalho

André Santanchè

Cesar Augusto Camillo Teixeira

Ronaldo dos Santos Mello

Data de defesa: 10-12-2012

Programa de Pós-Graduação: Ciência da Computação

TERMO DE APROVAÇÃO

Tese Defendida e Aprovada em 10 de Dezembro de 2012, pela

Banca examinadora composta pelos Professores Doutores:



Prof^a. Dr^a. Ariadne Maria Brito Rizzoni Carvalho
IC / UNICAMP



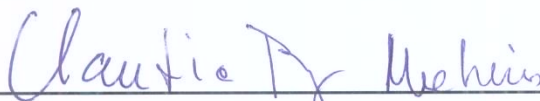
Prof. Dr. Ronaldo dos Santos Mello
INE / UFSC



Prof. Dr. Cesar Augusto Camillo Teixeira
DC / UFSCAR



Prof. Dr. André Santanchè
IC / UNICAMP



Prof^a. Dr^a. Claudia Maria Bauzer Medeiros
IC / UNICAMP

Organization is Sharing: From eScience to Personal Information Management

Rodrigo Dias Arruda Senra

December 10th, 2012

Examiner Board/*Banca Examinadora*:

- Profa. Dra. Claudia Bauzer Medeiros
Institute of Computing - UNICAMP (Supervisor/*Orientadora*)
- Profa. Dra. Ariadne Maria Brito Rizzoni Carvalho
Institute of Computing - UNICAMP
- Prof. Dr. André Santanchè
Institute of Computing - UNICAMP
- Prof. Dr. Cesar Augusto Camillo Teixeira
DC - UFSCAR
- Prof. Dr. Ronaldo dos Santos Mello
INE - UFSC

Abstract

Information sharing has always been a key issue in any kind of joint effort. Paradoxically, with the data deluge, the more information available, the harder it is to design and implement solutions that effectively foster such sharing. This thesis analyzes distinct aspects of sharing - from eScience-related environments to personal information. As a result of this analysis, it provides answers to some of the problems encountered, along three axes.

The first, *SciFrame*, is a specific framework that describes systems or processes involving scientific digital data manipulation, serving as a descriptive pattern to help system comparison. The adoption of SciFrame to describe distinct scientific virtual environments allows identifying commonalities and points for interoperation.

The second axe contribution addresses the specific problem of communication between arbitrary systems and services provided by distinct database platforms, via the use of the so-called *database descriptors* or DBDs. These descriptors contribute to provide independence between applications and the services, thereby enhancing sharing across applications and databases.

The third contribution, *Organographs*, provides means to deal with multifaceted information organization. It addresses problems of sharing personal information by means of exploiting the way we organize such information. Here, rather than trying to provide means to share the information itself, the unit of sharing is the organization of the information. By designing and sharing organographs, distinct groups provide each other dynamic, reconfigurable views of how information is organized, thereby promoting interoperability and reuse. Organographs are an innovative approach to hierarchical data management.

These three contributions are centered on the basic idea of building and sharing hierarchical organizations. Part of these contributions was validated by case studies and, in the case of organographs, an actual implementation.

Resumo

Compartilhamento de informação sempre foi um aspecto chave em qualquer tipo de esforço conjunto. Paradoxalmente, com o dilúvio de dados, a incremental disponibilização de informação tem dificultado o projeto e implementação de soluções que efetivamente estimulam o compartilhamento. Esta tese analisa aspectos distintos do compartilhamento - desde ambientes relacionados a eScience até informação pessoal. Como resultado desta análise, ela provê respostas para alguns dos problemas encontrados, ao longo de três eixos.

O primeiro, *SciFrame*, é um arcabouço específico para descrição de sistemas ou processos envolvendo manipulação de dados científicos no formato digital, servindo como um padrão que auxilia a comparação de sistemas. A adoção do SciFrame para descrição de ambientes científicos virtuais permite a identificação de pontos em comum e oportunidades de interoperabilidade.

O segundo eixo de contribuição contempla o problema da comunicação entre sistemas arbitrários e serviços oferecidos por bancos de dados, através do uso dos então chamados *descritores de bancos de dados* ou DBDs. Estes descritores contribuem para desacoplar aplicações dos serviços, melhorando portanto o compartilhamento entre aplicações e bancos de dados.

A terceira contribuição, *Organografos*, provê meios para a organização de informação multifacetada. Ela contempla problemas de compartilhamento de informação pessoal por intermédio da exploração da forma como organizamos tais informações. Neste caso, ao invés de tentarmos prover meios para o compartilhamento da informação propriamente dita, a unidade de compartilhamento é a própria organização da informação. Através do projeto e compartilhamento de organogramas, grupos distintos trocam entre si visões reconfiguráveis de como a informação está organizada, promovendo assim interoperabilidade e reuso. Organogramas são uma abordagem inovadora para o gerenciamento de dados hierárquicos.

Essas três contribuições estão centradas nas idéias básicas de construção e compartilhamento de informação organizada hierarquicamente. Parte destas contribuições foi validada por estudos de caso e, no caso de organogramas, por uma implementação de fato.

Acknowledgements

First and foremost I thank my advisor and awesome editor Claudia Bauzer Medeiros for being a source of inspiration for 20 years and beyond.

I thank my parents Magali and Nelson, for giving me the mental, emotional and physical resources anyone could ever ask for, and then give me some more. The small accomplishments I had so far, I owe you.

I thank my lovely wife Aline for showing me there is life and joy outside work, and for believing that I will have spare time after finishing this PhD.

I thank all LIS members from IC-Unicamp for their valuable comments and suggestions and their invaluable friendship.

I thank all my family and friends for sharing life with me.

This work was partially supported by the Microsoft Research FAPESP Virtual Institute (NavScales and eFarms projects), the Brazilian Institute for Web Sciences Research, CNPq (WebMAPS, BioCORE and MuZOO projects), PRONEX-FAPESP ¹, CAPES (AMIB project), as well as individual grants from CNPq.

¹Model and Methods in eScience for the Life and Agricultural Sciences

Contents

Abstract	vii
Resumo	viii
Acknowledgements	ix
1 Introduction	1
2 SciFrame: a conceptual framework to describe data sharing in eScience	5
2.1 Introduction	5
2.2 SciFrame: A Conceptual Model to describe Information Sharing	6
2.3 Case Study: Crop Monitoring in WebMAPS	9
2.3.1 Practical Pitfalls	10
2.4 Conclusions	11
3 Database descriptors: laying the path to commodity web data services	13
3.1 Introduction	13
3.2 Database Descriptors	15
3.2.1 Basic Definitions and Architecture	15
3.2.2 DBD Structure	16
3.2.3 DBD Representation	17
3.2.4 Matching DBDs	18
3.3 DBD Example	19
3.4 Use Case	20
3.5 Related Work	22
3.6 Conclusions and future directions	23
4 Evaluating, Reorganizing and Sharing Digital Information Hierarchies	24
4.1 Introduction	24
4.2 Limiting Factors Leading to Poor Hierarchical Organization	26

4.3	Related Work	27
4.3.1	Text Classification or Categorization	28
4.3.2	Text Clusterization	29
4.3.3	Information Extraction	29
4.4	Exploiting Hierarchies	30
4.5	Organographs	32
4.5.1	Applicability	32
4.5.2	Organograph Instantiation	34
4.5.3	Organograph Execution	35
4.6	Experiment: Using ACM's CCS98 to reorganize a private collection of papers .	36
4.6.1	The Organicer Platform	37
4.6.2	Transforming $H_{initial}$ into H_{mine}	38
4.6.3	Initial evaluation	39
4.6.4	Sharing the organization of H_{acm} – reorganization of H_{mine}	40
4.7	Conclusion	43
5	Conclusions	44
5.1	Contributions	44
5.2	Extensions	45
	Bibliografia	48

List of Tables

2.1	NDVI profile generation described with SciFrame	10
4.1	Statistics about the structure of the hierarchies computed by Organicer	40
4.2	Accuracy of f_{org} classification according to ACM DL	42

List of Figures

2.1	SciFrame - Scientific Digital Data Processing Framework	7
3.1	Database Descriptor Architecture	15
3.2	Feature DBD Example using annotations	19
3.3	Desiderata DBD Example	20
4.1	Specialized graph notation to describe hierarchies.	25
4.2	Framework for instantiation of f_{org}	34
4.3	Organograph Execution	36
4.4	Original paper collection $H_{initial}$ with ad-hoc organization	37
4.5	Collection H_{mine} (after transformation from $H_{initial}$)	38
4.6	Collection H_{acm} derived from ACM CCS98	39

Chapter 1

Introduction

eScience focuses in the computer simulations that accelerate scientific discovery, and the high performance distributed platforms these simulations run on. It now encompasses several branches of Computer Science. eScience environments are fed by sophisticated instruments that generate large volumes of complex and heterogeneous data at fast rates. Moreover, the ease to publish in the Web, associated with the vast amount of scientific data produced every day, caused an explosive growth in the amount of information available to scientists. Voluminous data with a fast growth rate are just part of the problem. Heterogeneity is frequently cited as one of the most complex problems in data sharing. As such, information representation and information sharing are important issues in eScience. Heterogeneity and volume are at the center of trying to allow people to work together. This is what motivates this work.

The goal of this thesis was to develop a theoretical framework to help solve some of these problems. SciFrame was conceived to meet interoperability and data management requirements that emerged during an effort to design and implement tools within the WebMAPS project [41]. WebMAPS was a multidisciplinary eScience effort involving computer scientists and experts on agricultural and environmental sciences to develop a platform for agro-environmental planning.

The starting point was the proposal of SciFrame or the *Scientific Digital Data Processing Framework*. It is a simplified and standardized vocabulary to describe the design patterns of digital data processing with three high-level abstractions: Interfacing, Data Management and Information Management. Although conceived independently, SciFrame is not an original initiative, it follows the steps of CLRC - Scientific Metadata Model [70] and myGrid [66]. The difference between SciFrame and these other projects relies on SciFrame's simplicity, being designed from a Computer Science perspective.

In an attempt to further specialize SciFrame, focusing in its Data Management component, we investigated how to define mechanisms through which applications couple with *Database Management Systems* (DBMSs) in the emergent scenario of cloud computing. This resulted in the proposal of *database descriptors* or DBDs. DBDs are artifacts to capture and match appli-

cations requirements and database capabilities, in order to allow dynamic binding of services or seamless data migration in the cloud. The concept of *database descriptors* was originally presented by Madnick and Wang [42] in 1988 to describe something similar to a (relational) DBMS feature set. We extend this concept for new kinds of DBMS and applications, accommodating it to the Web and Cloud Computing scenarios. The dynamic binding of services described in chapter 3 is a simplified version of an UDDI-like negotiation framework [14].

SciFrame and DBDs are complementary initiatives to describe information sharing between systems and processes through concept hierarchies, describing and organizing digital information hierarchically. Our subsequent investigation about organization of information led to questions, such as: Why do the organizational structures we create ourselves seem to become inadequate as time goes by? Why don't we share information organization (classification criteria) dissociated from content? How can we improve the way we organize information and what will be the impact on how we share information? These questions are all associated with the issue of collaboration and reuse of content, but also with content organization.

In order to attack these questions, we concentrated on issues related to organizing information through multi-faceted hierarchical categorization. As a result several limiting factors related to content organization were identified: the lack of evaluation tools for hierarchies with ad-hoc and implicit organization criteria; current mechanisms and methods lead to static and content-driven hierarchies instead of a more flexible dynamic and task-driven approaches; and finally the incapacity to share organization criteria and apply it automatically to reorganize different content to perform some recurring task. Based on that, *organographs* were proposed as a conceptual framework to transform implicit information organization criteria into explicit parameters, in the context of a particular task. Organographs can be used to evaluate, reorganize and share digital information hierarchies. DBDs can be used in organograph instances to provide loose and dynamic coupling with external data sources (i.e. hierarchies). Furthermore, SciFrame can be used to describe aspects of the transformation encapsulated in an organograph instance.

Given this scenario, the main contributions of this thesis are:

- the proposal of SciFrame: a conceptual framework to describe systems or processes involving scientific digital data manipulation;
- the proposal of Database Descriptors (DBDs): a conceptual framework to capture and match applications requirements to database capabilities, thus helping to commoditize data services in the cloud;
- the proposal of Organographs: a conceptual framework to transform implicit organization information criteria into explicit parameters, in the context of a particular task. Organographs can be used to evaluate, reorganize and share digital information hierarchies.

- practical validation of a major part of these concepts through the implementation of software tools and services.

This thesis is organized as a collection of papers that are most representative of the research developed. The rest of this text is organized as follows:

Chapter 2 is the paper *SciFrame: a conceptual framework to describe data sharing in eScience*, published in the Proceedings of the III Brazilian eScience workshop in 2009. This chapter provides an integrated perspective of efforts and phases involved in sharing of eScience data. The SciFrame model is presented through a case study of scientific data manipulation in WebMAPS. This study illustrates some typical problems related to data sharing, particularly the problem of distributing large datasets over the Web. The paper points out that SciFrame should be used as a design pattern, from which scientists can structure and describe their own eScience efforts.

Chapter 3 is the paper *Database descriptors: laying the path to commodity web data services*, published in the Proceedings of Engineering of Computer-Based Systems (ECBS) in 2010. This chapter introduces the concept of database descriptors (DBDs). DBDs are presented as the foundational bricks to build dataspaces, allowing applications to switch across DBMSs in a loosely coupled scenario. DBDs can thus contribute to help to commoditize data services in the cloud, by supporting dynamic switching between DBMSs and applications. DBDs can also be seen as a different way of tackling the information integration problem, from a connectivity point of view, in which applications and DBMSs can negotiate their coupling. As already mentioned in this chapter, DBDs can be seen as a solution within SciFrame to deal with data management issues.

Chapter 4 is the paper *Evaluating, Reorganizing and Sharing Digital Information Hierarchies*, that has been submitted for publication in the Journal on Data Semantics (JODS). This chapter presents organographs. It discusses how organographs can be used to evaluate, reorganize and share digital information hierarchies. Moreover, it describes a concrete organograph use case, where a personal collection of papers is reorganized according to ACM subject headings and evaluated before and after reorganization. We concluded the chapter showing that content organization can be shared, ad-hoc hierarchies can be refactored into more balanced hierarchical structures while preserving valid categorical relationships. As pointed out in the chapter, Organographs differ from proposals such as [25, 55] that are restricted to file manipulation.

Chapter 5 contains conclusions and some directions for future work.

Besides these publications chosen to compose the thesis text, the following papers are also associated with this work:

- *Organographs - Multi-faceted Hierarchical Categorization of Web Documents*. Rodrigo D. A. Senra, Claudia B. Medeiros. Proceeding of the 7th International Conference on

Web Information Systems and Technologies - WEBIST: 583-588 (2011). This paper introduces the concept of Organographs.

- *A standards-based framework to foster geospatial data and process interoperability*. Gilberto Z. Pastorello Jr., Rodrigo D. A. Senra, Claudia B. Medeiros. Journal of the Brazilian Computer Society 15(1): 13-25 (2009) This paper covers a few challenges met by the implementation of WebMAPS.
- *Bridging the gap between geospatial resource providers and model developers*. Gilberto Z. Pastorello Jr., Rodrigo D. A. Senra, Claudia B. Medeiros. Proceedings of the 16th International Conference on Advances in Geographic Information Systems - ACM SIGSPATIAL: 44 (2008) This paper deals with seamless composition of distributed data sources and processing solutions to leverage model development.
- *O projeto WebMAPS: desafios e resultados*. Carla G. N. Macário, Claudia B. Medeiros, Rodrigo D. A. Senra. Proceedings of 9th Brazilian Symposium on Geoinformatics - GeoInfo: 239-250 (2007). This paper presents results from the WebMAPS project.

Moreover, the following software tools were developed within the thesis:

- Paparazzi, a software tool to crawl and fetch remote sensed data automatically from NASA's Web portal. It was used to collect data for experiments that led to SciFrame, DBDs, and also used in other PhD and MSc research in LIS.
- WebMAPS, a portal capable of computing and rendering NDVI time profiles for given geographical regions. The underlying algorithms and implementation of WebMAPS helped the theoretical work of this thesis, and were also taken advantage of by other PhD and MSc students in LIS.
- the design and implementation of Organicer, a tool to validate the organographs concept.

Chapter 2

SciFrame: a conceptual framework to describe data sharing in eScience

2.1 Introduction

Computer Science has introduced a revolution in scientific research, and is recognized, nowadays, as being essential to the advance of science. The term *eScience* [21] was introduced in the end of the 90's. While it originally focused in the computer simulations that accelerate scientific discovery, and the high performance distributed platforms these simulations ran on, it now encompasses several branches of Computer Science. Indeed, these platforms are fed by sophisticated instruments – e.g., telescopes, satellites, medical devices – which generate large volumes of complex and heterogeneous data at fast rates. These data should be processed by scientists using suites of complex algorithms and computational tools, and novel visualization methods. Interpreted results are fed back to the network, to become part of eScience data available.

As such, information representation and information sharing are both essential components of eScience. In fact, the World Wide Web – the most visible face of the Internet – was motivated by the need to communicate information among researchers. However, the ease to publish in the Web, associated with the vast amount of scientific data produced every day, caused an explosive growth in the amount of information available to scientists.

Voluminous data with a fast growth rate are just part of the problem. Heterogeneity is frequently cited as one of the most complex problems in data sharing. In eScience, it is aggravated by the inherent multidisciplinary - besides the usual problems of variety in data acquisition, modeling, storage, processing and publication, all of which responsible for heterogeneity, there exists the issue that the scientists that participate in any given project have very distinct profiles and work contexts.

Another problem is how information is represented so that sharing can be facilitated. A white paper, a spreadsheet or a raster image are all valid representation formats, but not nec-

essarily self-sufficient or complete. For instance, a white paper may lack details about the raw data used in an experiment, a spreadsheet may not inform from where or when the data were gathered, or a raster image might omit details about the sensors used for data capture. Completeness criteria depend not only on data producers, but also on the consumer’s intent. This has prompted research on metadata, annotations and ontologies to enhance data characterization and provenance.

Sharing of data is just part of the problem – scientists also need to share *models*, which are defined in terms of sequences of operations, usually as scientific workflows – e.g., [48]. This paper does not directly cover workflows and models, concentrating on data aspects. We do, however, indicate several challenges associated with such workflows, which are closely connected with the second Grand Challenge of SBC – the management of models.

The goal of this paper is to exploit the many facets of the problem of sharing scientific digital data. This research is directly connected with the first Grand Challenge of SBC: management of large multimedia data volumes [44]. Our main contributions to the Challenge concern the proposal of a conceptual model to be used as background to support an integrated analysis of these issues. Moreover, SciFrame can be used as high-level design pattern from which scientists can structure and describe their own work. The use of this model is exemplified through a real-world case of scientific data sharing. We conclude the paper with references to research efforts that try to tackle some of these issues.

2.2 SciFrame: A Conceptual Model to describe Information Sharing

According to Longworth [39], the stages in human learning can be described by the following *information ladder* also known as the *DIKW model*: $Data \rightarrow Information \rightarrow Knowledge \rightarrow Understanding \rightarrow Insight \rightarrow Wisdom$.

While the rightmost stages belong to the domains of cognition, psychology and philosophy, the first three steps are directly related to the first SBC Grand Challenge, and to SciFrame. The terms *data*, *information* and *knowledge* have various definitions and can be used for overlapping concepts. However, in the context of SciFrame, we adopt the following definitions: **Data** is a structured collection of *typed values*, represented in *digital form*. The important distinction between *data* and *information* is that the latter has **explicit** semantics. **Information** is a set of inter-related data, *bound to semantics* and *useful for some purpose*. From a Semiotics point-of-view, data are symbols and information occurs when data are used to refer to something. **Knowledge** represents the *cognitive dimension* of the information’s consumer, linking information from the process domain to information present in the “out-of-process world”. From the DIKW model, knowledge is created by *using the information for action*.

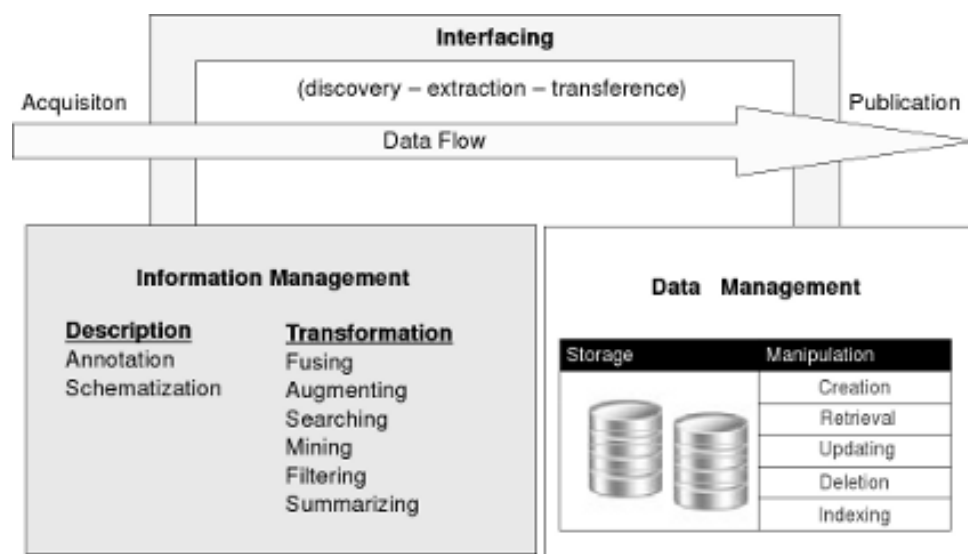


Figure 2.1: SciFrame - Scientific Digital Data Processing Framework

In this section we define a conceptual framework that describes systems or processes involving scientific digital data manipulation – *the Scientific Digital Data Processing Framework*, or **SciFrame** for brevity. SciFrame is depicted in Figure 2.1, and it is divided into three high-level abstractions: Interfacing, Data Management and Information Management. This overall structure and its elements are well known and compose a long-time adopted pattern. Nevertheless, this pattern lacked a cohesive definition in a standardized vocabulary, that we try to remedy here.

Let us consider an eScience process that involves some kind of scientific data manipulation. **Interfacing** defines the process boundaries and comprehends all digital data exchange between the process itself and external entities, either human or artificial. The *Interfacing* element has two subdivisions: *acquisition* and *publication*. **Acquisition** represents the obtention of data: in digital form, with a known structure, from a known source, through a given media. This stage represents the process input. **Publication** stands for suitable data representations that allow proper communication with external entities. Data are published in digital form, with a known structure, via a given media. This stage represents the process output. The *acquisition* and *publication* elements of a given eScience process are potentially independent from each other. When two processes are interacting, the acquisition element of one is coupled to the publication element of the other [53].

One way to define their data exchange pattern is to determine which role takes initiative in the transaction. A *PUSH* pattern occurs when the data provider initiates the data exchange, and

conversely a *PULL* pattern occurs when the data consumer takes the initiative. Data exchange can be divided into three stages: discovery, extraction and transference. **Discovery** represents the acquirer's concern with identifying suitable data publishers given a set of information needs. On the other hand, publishers are concerned with making themselves well-known to potential acquirers. **Extraction** represents the problem of extracting the information from a chosen data provider/publisher. **Transference** represents the problem of moving data from the publisher into the data management facilities of the consumer.

SciFrame makes the distinction between *Information Management* and *Data Management*. *Information Management* is responsible for higher-level information manipulation, analysis and synthesis. *Data Management* is responsible for lower-level data manipulation for persistence purposes.

The **Data Management** element is subdivided into *storage* and *manipulation* elements where: **Storage** is responsible for data persistence (caching inclusive). **Manipulation** provides support for the basic interactions with the storage element. These interactions are called CRUD, an acronym for the operations: create, retrieve, update and delete. We have also included the index interaction. Therefore, we shall refer to it as CRUDI.

In the context of *Information Management*, an eScience process can be examined and modeled according to two main axes: *data description* and *data transformation*. **Transformation** corresponds to a finite number of operations that the process applies to the stored data, in order to change its contents or structure. We present an informal set of definitions for the operations. **Augmenting** adds information to the data present. **Fusing** generates new information by coalescing part of the data present. **Filtering** decreases the amount of information by discarding data. **Summarizing** decreases the amount of data by classifying, clustering or generalizing data. **Searching** locates information inside data. **Mining** extracts unperceived information from data.

The last two, searching and mining, can be seen as idempotent transformations, considering that the status of the data is not altered. Transformation can occur at any time, but when it takes place immediately after acquisition, it is often called pre-processing. The *Description* element is orthogonal to the *Transformation* element. It corresponds to the gathering and organization of information about the process data elements, documenting their nature, structure and purpose. It also encompasses the roles of *annotation* and *schematization*. As an example, *provenance* is one of the most important types of description, fundamental in the eScience context to ensure the shared data elements' quality and usefulness.

We acknowledge that SciFrame requires a more complete and formal characterization of the interactions and dependencies of its constituent elements. However, due to restrictions in this paper's length, we chose to present a case study instead. The case study illustrates a typical example of a scientific application with a strong focus on information and data sharing, evidentiating the role of SciFrame as a generic pattern to describe eScience research.

2.3 Case Study: Crop Monitoring in WebMAPS

In order to illustrate SciFrame’s applicability, we present a real-world scenario from the WebMAPS eScience project [41]. This is a multidisciplinary effort involving computer scientists and experts on agricultural and environmental sciences to develop a platform for agro-environmental planning. The case study concerns an eScience process within WebMAPS, and shows that sharing scientific data presents challenges that are not found in other kinds of data sharing (e.g., in industrial or business contexts).

An important problem in agro-environmental planning is monitoring crop behavior. One of the earliest studies [72] about deriving crop condition from solar radiation has shown that there is a strong correlation between radar measurements (backscatter) and *leaf area index* (LAI). LAI determines the amount of energy available to the plant for photosynthesis which in turn drives the plant development and subsequent yield.

One of the tools used by experts to monitor crop behavior is based on *Normalized Difference Vegetation Index* (NDVI). Informally, this index represents the healthiness (“greenness”) of a given vegetation cover. It is computed as the difference between the red (RED) and near-infrared (NIR) bands of multispectral images, given by the formula: $NDVI = (NIR - RED) / (NIR + RED)$. There are several vegetation indexes proposed, such as: Perpendicular Vegetation Index [60], the Soil-Adjusted Vegetation Index (SAVI) [26], the Atmospherically Resistant Vegetation Index (ARVI) [31] and the Global Environment Monitoring Index [54]. Choosing amongst them is part of the problem – distinct scientists favor different indexes, which result in incompatible analyses. However, NDVI remains the most well-known and used index to detect live green plant canopies from multispectral remote sensing data.

One of the processes in WebMAPS corresponds to a tool that generates *NDVI profiles*. Each profile is a time series of average NDVI measurements, which represents the vegetation’s health (biomass status) in a particular region for a given time period (crop’s phenological cycle). NDVI profiles characterize the spatio-temporal behavior of specific crops. They allow experts to monitor the evolution of the crop, detect (and prevent) anomalies and forecast crop yield.

The management of spatio-temporal data series is a problem common to many eScience domains, which is one of the reasons for our choosing this case study. Table 2.1 gives an overview of the process that generates *NDVI profiles*, summarized under our SciFrame conceptual model. Some processing details were omitted, but it serves the purpose of illustrating SciFrame’s application.

For instance, although the NDVI formula is mathematically simple, satellite image pre-processing is complicated and requires extensive data correlation. Perturbing factors should be detected and mitigated in order to avoid negative influence in the computed NDVI. They include: (i) high level of water vapor and aerosols; (ii) soil moisture; (iii) angular geometry of illumination and observation at the time of the measurements; (iv) sensor-dependent data

Table 2.1: NDVI profile generation described with SciFrame

Data Management		
Storage		organize and persist input raster images and their textual metadata
		organize and persist composite NDVI profiles
Manipulation		index and fetch regions from images
Information Management		
Description		spatial regions of interest are described in Well-Known Text notation (WKT), raster images in HDF format have embedded textual metadata
Transformation		eliminate clouds by generating composite images (data gaps removal)
		detect and mitigate perturbing factors
		calculate NDVI time series
		filter out noise using HANTS (Harmonic Analysis of Time Series)
Interfacing		
Acquisition	Discovery	elect adequate remote sensing data products and providers available in the Web
	Extraction	identify a path to data products in the provider's Web portal
	Transference	download products (raw multispectral satellite images) via HTTP or FTP
Publication		publish NDVI profiles as 2D scatter plots (average NDVI vs time) in WebMAPS portal

calibration. These issues represent nested processes in NDVI profile generation. Though not described in this paper, we point out that they have a SciFrame's description of their own.

2.3.1 Practical Pitfalls

Consider a scientist in charge of studying sugarcane crops in Ariranha County in São Paulo state (Brazil). The goal is to analyze sugarcane yields using year 2001 as benchmark, when it was the top producer county, yielding approximately 5.15 million tons/year. This scientist decided to use a NDVI profile as an estimator [11], based on previous studies of NDVI correlation to crop yields.

In order to do that, first of all, this scientist needs the georeferenced perimeter of every farm growing sugarcane in Ariranha County. Georeferencing means to establish an appropriate set of coordinates defining accurately the region's location on the Earth's surface. Here we face a common barrier in eScience – data availability. Georeferenced boundaries may be hard to obtain in practice, due to the lack of reliable boundary databases. This may be circumvented by a ground survey with GPS measurements, which requires the farmers agreement. We are not concerned here with confidentiality issues. Data privacy and security are valid open problems in eScience data sharing that are out of the focus of this proposal.

In addition to the spatial regions of interest, the scientist must collect remote sensed imagery covering the county's area (aprox. 133 km^2) during the target years. NASA's MODIS sensor is a reasonable data source. One of its derived products is "MOD13Q1 - Vegetation Indices 16-Day L3 Global 250m". This dataset is delivered by NASA already pre-processed, with 11 pre-calculated vegetation indexes, including NDVI. MOD13Q1 is distributed as files encoded in NASA's HDF-EOS format [57], each covering an area of $5,760,000 \text{ km}^2$ with average size

of 500 Mbytes.

This means that a 2-year long NDVI time series covering Ariranha County's area (532 pixels per MOD13Q1 image) requires 46 images. The data volume amounts to roughly 47.8 Kbytes. However, due to distribution granularity (in 500 Mb files), this scientist will have to download 23 Gbytes. Therefore, 99.99979% of the downloaded data is useless considering the target study. In the worst case where a single satellite image snapshot (swath) does not entirely contain the region of interest, the waste is doubled.

Moreover, to download the 46 files (92 in the worst case scenario), the scientist will have to fill out a NASA web form (maybe several times) specifying product, swath (region of interest) and time interval. The estimated delivery delay can range from a day to a week, depending on the available network throughput. Nevertheless, that is still a straightforward process. Each web portal providing remote sensed imagery implements a different acquisition workflow. Some portals demand a round of e-mail exchanges prior to data release. Other portals arrange the files in hierarchies, forcing the user to browse through several pages before reaching the target links.

After all data are obtained, there remains the issue of compatibility with the scientist's processing environment. For instance, computing time and storage space may be required to convert NASA's HDF-EOS format into more widespread input formats such as GeoTIFF [61], NetCDF [59], HDF4, or HDF5 [19].

Once the satellite data are converted and stored, several other issues remain. For instance, the acquired images may present gaps in the region of interest (e.g., clouds) that could be compensated by additional processing (e.g., by acquiring images from other providers, or executing complex data manipulation procedures). Image noise has to be taken into account.

Once all preprocessing is finished to the scientist's satisfaction, the profiles can be generated. Again, this presents many challenges. For instance, just as they may choose distinct vegetation indexes, research groups may adopt different procedures to generate profiles, which in turn may result in differences in profiles. Notice that each processing strategy chosen will compound the obstacles to sharing profile data. Hence, in order to share the published profiles with other groups, an appropriate description of the entire profile generation process must be provided – e.g., indicating the source images used, the scientific workflow selected to create the profile, and so on. This kind of discussion falls into the general problem of *provenance* in eScience.

These are just a small sample of problems involved in sharing eScience data.

2.4 Conclusions

The first Grand Challenge of SBC involves the management of multimedia data, which includes scientific data. There are many issues concerning the latter that need to be investigated using specific procedures, given some of their peculiarities. This paper is a step towards this direction, providing an integrated perspective of efforts and phases involved in sharing of eScience data.

The SciFrame model, conceived with this in mind, was presented through a case study of scientific data manipulation in WebMAPS. This study illustrates some typical problems related to data sharing, particularly the problem of distributing large datasets over the Web. We concluded the paper pointing out that SciFrame should be used as a design pattern, from which scientists could structure and describe their own eScience efforts.

Chapter 3

Database descriptors: laying the path to commodity web data services

3.1 Introduction

We are interested in supporting seamless switching between applications and DBMSs. In the context of this paper, applications are any software artifact, and databases refer to *Database Management Systems* (DBMS). When an application switches from using a DBMS to another, data may also have to be migrated and transformed. We attack the problem in two stages. This paper is concerned with the first stage - mechanisms to support dynamic coupling - and assumes that, once this is achieved, appropriate mechanisms will be devised to migrate data, when needed (the second stage).

Today, applications are still conceived to be tightly coupled to a given DBMS instance. Such a tight coupling is the most feasible solution to implement, since such systems differ in terms of model, operations and interface. For instance, an application written to use a relational database must be refactored to use a different DBMS. When the underlying data models are different, the way data is structured and handled is radically different, such as an XML storage or an Object-oriented database. Even if two DBMSs support the same model, they may differ on the capabilities supported, such as temporal or spatial facilities, and the DBMSs may offer a different feature set.

The term *feature set*, in this paper, refers to a set of properties that include: data model, functional capabilities, access methods and API, performance and configuration settings. Applications can only switch from one DBMS to another if the target DBMS offers a feature set compatible to what is required by the application.

Relational DBMS already achieved a good degree of interchangeability through successful standardization efforts such as *Open Data Base Connectivity* (ODBC) from the X/Open consortium, or ISO's ANSI-SQL proposed in 1989 (and revised in 1992). However, there is room for

improvement. The software industry continuously improves its products with extensions that transcend the relational model and fall out of the standardization efforts. As a consequence, applications that depend on non-standard extensions become enslaved to a particular product.

Success in designing and building DBMS database products will certainly introduce additional interoperability issues. For example, given a Web application in the emergent cloud computing scenario [5], as the user base demands scalability, the solution may be to switch from a single multi-purpose database to several special-purpose database-as-a-service (DaaS) approaches.

Several research efforts have been undertaken towards more flexible coupling between application requests for data and DBMS. Examples of such efforts include: n-tiered architectures, database federations, Web services and cloud computing. Each such initiative is based on some set of standards that determine, for example, how to invoke operations or how to encapsulate data.

There are two basic scenarios to be considered. In the first scenario, an application requests data from several DBMS, and *may need further data* from another DBMS of a different nature, i.e. with a widely different feature set. In the second scenario, the application wants to *switch* from the initial set of DBMS to another (potentially different) set of DBMS. Here, it may have to depend on additional preprocessing operations – e.g., data conversion and migration from the original set to a new one.

These two scenarios introduce many research challenges. For instance, how to choose an adequate DBMS amongst several vendors? If changes in DBMS involves data migration, what is the effort and schedule involved? Would there be any collateral effects due to compatibility mismatches? And the bottom line – could all of these questions be answered and the migration be carried out automatically and seamlessly?

In order to attack this problem we propose the use of *database descriptors* (DBDs), which are data structures that describe the feature set of a DBMS and the requirements applications have in terms of DBMS support. From a high level point of view, an application A can switch from DBMS X to DBMS Y if, desiderata DBD_A is compatible with feature DBD_Y .

In more detail, this paper proposes DBDs as a mechanism to describe the nature and capabilities of DBMS and application requirements. DBDs could be used to verify and validate the matching between application requirements and database capabilities, and ultimately be used as the foundation for dynamic negotiation and autonomous binding between applications and databases.

This paper is organized as follows. Section 3.2 introduces DBDs. Section 3.3 discusses situations in which they are needed. Section 3.4 provides a use case. Section 3.5 discusses a few major trends in related work. Section 3.6 concludes the paper.

3.2 Database Descriptors

The concept of *database descriptors* was originally presented by Madnick and Wang [42] in 1988 to describe something similar to a (relational) DBMS feature set. We extend this concept for new kinds of DBMS and applications, moreover accomodating it to the Web scenario.

The main goal behind constructing DBDs is to enforce a loose coupling between Applications and DBMS, that could help to: (i) ensure DBMS product/vendor independence, (ii) provide seamless cross-database migration, and (iii) support Applications and DaaS in the Cloud. Some of these goals depend on strategies to solve the schema integration problem. In this paper we are not focused on the data integration issue. Our main interest is to explore a mechanism that allows capability verification, validation and negotiation amongst applications and DBMS.

3.2.1 Basic Definitions and Architecture

We devise two types of *database descriptors*: desiderata descriptor and feature descriptor. The desiderata descriptor specifies what a client application needs (requirements) from a DBMS. The feature descriptor specifies the DBMS feature set. As we pointed out earlier, it refers to a set of properties that include: data model, functional capabilities, access methods and API, performance and configuration settings.

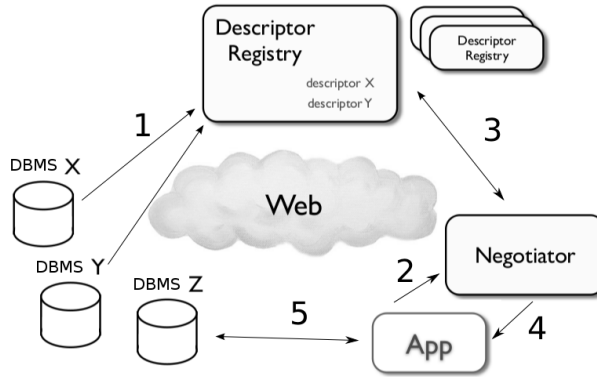


Figure 3.1: Database Descriptor Architecture

Assuming that DBDs are already available, there are many possible interaction patterns between applications and DBMSs. Figure 3.1 presents a generic architecture that serves as a reference for discussing interaction patterns.

The first step is taken when a given DBMS reifies its feature set as a feature descriptor. Considering that the feature descriptor is in digital form, it could be stored anywhere: as a file

in the filesystem, as data in some DBMS or published in a Web page. We advocate the creation of registries in the Web. A descriptor registry consists of a publicly accessible repository specialized in storing DBDs. For instance, a minimal registry could be materialized as a Web page with links to pure-XML pages describing DBDs. Step 1 in Figure 3.1 illustrates DBMSs X and Y registering DBD_X and DBD_Y in a given registry.

The second step is taken when an application produces a desiderata descriptor that reflects its expectations in terms of a DBMS. The purpose of the desiderata descriptor is to be matched against the feature descriptors found in a registry. One possible approach is to make applications themselves responsible for discovering registries and carrying out the descriptor matching process.

We have chosen to introduce another element in the architecture called the negotiator, who is responsible for mediating the negotiation process, in which applications "negotiate" switching across DBMSs. Therefore, the negotiator can be an independent software artifact (such as a server), dwelling on the Web, and shared by potentially many applications. On the other hand, the negotiator can be a software module (such as a code library) embedded in the application or in the descriptor registry. Although we will refer to negotiator as an entity independent from the application to highlight its role, the architecture proposed is generic and accommodates different implementations.

We will not explore in this paper the trade-offs among the options to implement negotiators, either embedded in the application, in the registry or as an independent external mediator. We leave this topic to be explored in the future.

Step 2 in Figure 3.1 depicts application App presenting its desiderata descriptor DBD_{App} to the negotiator. The negotiator should discover the available registries and run the descriptor matching algorithm (see Section 3.2.4) against them. This is represented by step 3. After the matching process, the resulting collection of feature descriptors is ranked by similarity with the desiderata descriptor and returned back to App in step 4. The process concludes with dynamically binding application and DBMS.

3.2.2 DBD Structure

The desiderata and the feature descriptors both share a common structure, composed by three distinct parts: metadata, dimensions, dimensional values.

The *metadata part* describes the descriptor itself, and we propose the adoption of a Dublin Core (DC) [1] subset. The following DC fields should be mandatory: identifier, format, date, creator, title and type.

The *dimensions part* are the DBMS properties described by the DBD, such as: connectivity, data model, type system, indexing resources, DDL/DML support, security, provenance, versioning, replication, scalability, etc. For each dimension mentioned in the DBD, there should

be an associated *dimensional value*, which composes the third part.

The desiderata DBDs could have additional dimensions that express limitations or trade-offs from the application perspective, for example: prioritize up-to-date information, prioritize low latency for data delivery, enforce size constraints for result sets, determine quality of data (e.g., accuracy or completeness). Other (non-functional) requirements include privacy, security, costs, legal issues.

Desiderata descriptors could also include a fourth algorithmic part that specifies criteria for matching against feature descriptors. This algorithmic part can be represented by code embedded in the DBD, or it could be just a textual reference to some algorithm well known by the negotiators (further explained in Section 3.2.4). Once more, the trade-offs derived from these implementation choices are left for the future.

3.2.3 DBD Representation

The wide spectrum of the dimensions exemplified in Section 3.2.2 suggests that the DBD representation format should be extensible. We assume that distinct DBD instances will have a different collection of dimensions. As a result, the representation format and the matching algorithms should cope with partial information and heterogeneous structures. Given these constraints, XML might be a sound technological choice for representing DBDs. XML satisfies the heterogeneity condition but it is not sufficient for DBD representation, as explained by Wilde et al [74].

We propose, therefore, that DBDs be represented by the semantic annotations of [40, 50].

Annotation Units. An annotation unit a is a triple $\langle s, m, v \rangle$, where s is the subject being described, m is the label of a metadata field and v is its value or description.

Annotation. An annotation A is a set of one or more annotation units.

Semantic Annotation Units. A semantic annotation unit sa is a triple $\langle s, m, o \rangle$, where s is the subject being described, m is the label of a metadata field and o is a term from a domain ontology.

Semantic Annotation. A semantic annotation SA is a set of one or more semantic annotation units.

In fact, annotation units describe data using natural language; semantic annotations use ontology classes and can be processed by a machine. Since semantic annotations rely on ontologies, they provide the necessary interoperability basis to accommodate the needs of DBDs. However, there still remains the need to define the notion of *compatibility*, which is discussed next.

3.2.4 Matching DBDs

An application can couple to a DBMS if the former's desiderata descriptor matches the latter's feature descriptor. In an ideal world, both DBDs should be the same (i.e., identical metadata, dimensions and values). However, this restricts application-DBMS coupling. Thus, we have to look for more flexible matching criteria – e.g., borrowing notions from (i) programming languages or from (ii) content-based retrieval mechanisms in image databases.

In the first case (i), we can use an analogy from interface matching, where a subroutine invocation (message in object-oriented jargon) should match one function or procedure (resp., method) considering its context (i.e., namespace) and signature (i.e., name and parameters). Similarly, we can consider DBDs to represent signatures, where the dimension values are ontology terms.

Moreover, if we use semantic annotations, then matching can be performed using ontological relations. Borrowing from Santanche [62], two DBD values (A and B) are considered equivalent if they refer to the same concept in an ontology (equal URIs), or if they point to two concepts related by OWL equality relationships (equivalentClass or sameAs). Moreover, A is said to be more general than B if A subsumes B and conversely B is more specific than A. For instance, if B is OWL subClass of A, or B is related with A through the partOf property (B partOf A), then A subsumes B. Consider A and B vertices of a graph, whose edges are properties. The subsumption relationship between A and B is a path formed by one or more edges. Therefore, the similarity rank value between A and B is inversely proportional to the number of edges which connect A and B in a subsumption relationship.

In the second case (ii), one can consider an analogy between DBDs and descriptors of images. From a high-level perspective, image similarity mechanisms are based on the notions of *feature descriptor* and *distance function* [67]. A feature descriptor is typically a set of values, organized according to some structure, that summarizes a given object (here, an image). Vectors are the most common structure used, and feature descriptors are therefore often called *feature vectors*. Two objects are considered similar if the distance between their descriptors is below some threshold. Similarity depends on the features selected – thus, distance functions are intimately associated with the algorithms used to create the vectors. Examples of distance functions involve *Manhattan* (also known as *L1*) and Euclidean – *L2*. If we borrow from this second kind of domain, then DBDs are our image feature descriptors and we can devise different distance functions (e.g., edit distance for each string in an annotation unit) to compare two DBDs.

We propose the adoption of the first definition, which borrows from interface matching in programming languages. Nevertheless, we point out that other matching mechanisms can be used.

3.3 DBD Example

Figure 3.2 exemplifies, from a high level point of view, a hypothetical DBD for an RDF DBMS. This example represents the feature DBD whose identifier is DBD1 (created in Dec 18, 2009 by Claudia). The dimensions and values indicate that the 2PL concurrency protocol is used, there is no versioning, storage uses RDF format, and the query language supported is RDQL. In order to represent DBD1 we have chosen the RDF data model rendered in XML markup language, which is acceptable in such a simple example. Posterior and more complete versions of DBD1 could be rendered in more expressive semantic languages, e.g. an OWL dialect (FULL/DL/Lite), depending on the need to express more accurately the identities, relationships and restrictions on the DBD dimensions.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:dbd="http://www.lis.ic.unicamp.br/purl/DBD*"
  <rdf:Description rdf:about="http://www.lis.ic.unicamp.br/purl/DBD/DBD1">
    <!-- metadata -->
    <dc:creator>Claudia Bauzer Medeiros</dc:creator>
    <dc:description>Hypothetical DBD for an RDF DBMS</dc:description>
    <dc:identifier>DBD1</dc:identifier>
    <dc:format>application/rdf+xml</dc:format>
    <dc:type>
      <rdf:Description>
        <dbd:type>Feature DBD</dbd:type>
      </rdf:Description>
    </dc:type>
    <dc:title>Descriptor of an RDF DBMS</dc:title>
    <dc:date>2009-12-18</dc:date>
    <dc:language>EN</dc:language>
    <!-- dimensions and values -->
    <dbd:concurrency>Two phase lock</dbd:concurrency>
    <dbd:versioning>unsupported</dbd:versioning>
    <dbd:storage>RDF triples</dbd:storage>
    <dbd:DML>
      <rdf:Bag>
        <rdf:li>RDQL</rdf:li>
        <rdf:li>SPARQL</rdf:li>
      </rdf:Bag>
    </dbd:DML>
  </rdf:Description>
</rdf:RDF>
```

Figure 3.2: Feature DBD Example using annotations

Figure 3.3 presents a desiderata descriptor that matches with the feature DBD exemplified in Figure 3.2. We point out that these examples are artificial, in the sense that feature and desiderata descriptors were both created by the same people. However, they serve the purpose of illustrating the look-and-feel of DBDs.

There are many other domains where DBDs are applicable. We just point out a few examples, to illustrate their utility. In *Online Transaction Processing Systems* (OLPTs) applications should require support for lots of small concurrent transactional workloads (e.g. debit/credit). In *Digital Libraries*, applications may demand for corpora text-indexing and, for more sophisticated libraries, multimedia indexing. In *Web Portals* there is an increasing demand for multimedia delivery (audio, image and video streaming) and the basic operation is to serve pages. *Scientific Grid* applications are interested in accessing voluminous data cubes and in number


```

<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:dbd="http://www.lis.ic.unicamp.br/purl/DBD">
  <rdf:Description rdf:about="http://www.lis.ic.unicamp.br/purl/DBD/DBD1">
    <!-- metadata -->
    <dc:creator>Rodrigo Dias Arruda Senra</dc:creator>
    <dc:description>Desiderata DBD for an hypothetical application</dc:description>
    <dc:identifier>DBD2</dc:identifier>
    <dc:format>application/rdf+xml</dc:format>
    <dc:type>
      <rdf:Description>
        <dbd:Type>Desiderata DBD</dbd:Type>
      </rdf:Description>
    </dc:type>
    <dc:title>Desiderata descriptor of an hypothetical application</dc:title>
    <dc:date>2010-01-05</dc:date>
    <dc:language>EN</dc:language>
    <!-- dimensions and values -->
    <dbd:concurrency>Two phase lock</dbd:concurrency>
    <dbd:storage>RDF triple store</dbd:storage>
    <dbd:DML>RDQL</dbd:DML>
  </rdf:Description>
</rdf:RDF>

```

Figure 3.3: Desiderata DBD Example

crunching.

In each of these situations, applications frequently may have to migrate from one DBMS to another, and in all cases DBDs have to capture and express the applications' requirements and the DBMS capabilities in each of those domains.

3.4 Use Case

This section presents a hypothetical use case for DBDs. It is based on two real life projects in agro-environmental planning - the WebMAPS [51, 52] and the eFarms [40] projects, both conducted at the Laboratory of Information Systems of the Institute of Computing at UNICAMP.

The task to accurately create feature and desiderata DBDs is far from trivial. However, manually performing a compatibility analysis between an application and cloud DBMS services, and designing and executing a migration plan is not trivial either, not to mention time consuming. The purpose of this use case is to illustrate a concrete scenario where DBDs are useful, and show some of the difficulties involved in devising and applying DBDs. The DBD negotiation process presents its own set of relevant research challenges that fall out the scope of this use case discussion.

Consider a web application designed to support crop monitoring. This application handles three types of data: satellite images, farm/county geometries (coordinate sets), and time series of temperature and pluviosity measurements. Suppose the images and geometries are stored in the filesystem (as NetCDF and Shapefiles respectively), while the temporal series are stored in a relational database. The first prototype has been tested, and the application must now be released on the Web to end-users (farmers and agronomers). User demand is expected to scale from dozens to thousands of active sessions within a month's period.

Consider furthermore that we decided to adapt this application to use Internet-scale computing platforms, but we must choose from the available cloud computing storage services such as: Amazon's SimpleDB, S3 and Relational; Microsoft SQL Azure; or Google Data Services: Docs, Base and DataStore/BigTable.

The first challenge in this use case is to convey feature DBDs for these storage services – one just needs to compare such services to see that there are countless factors to take into account. There are, in fact, many dimensions to consider to properly describe each service – e.g., volume restrictions, pricing, scalability, data model, security. Comparing Amazon's SimpleDB and S3, both services provide: high availability, low latency, a key-value data model, a REST-based API and an access control lists (ACLs) security model. On the other hand, these two services differ in terms of volume restrictions and pricing. For example, S3 focuses on large raw data items (i.e. BLOBs), while SimpleDB focuses on small textual items (described by textual attributes) with implicit indexing. Considering that the pricing model mimics the data model and the latter is different amongst services, then the pricing model becomes harder to be compared automatically amongst services.

All of Google's Data Services provide: high availability, low latency and a REST-based API, though they differ in terms of data model, access restrictions and specific APIs. Google Docs is adequate to store textual documents and spreadsheets. Google Base is similar to Amazon's S3 – it is a web storage service for structured content as a set of descriptive attributes. Differently from S3, Google Base has no access restriction mechanisms, all items published are always publicly available. Google DataStore is a non-relational key-value-based web service DBMS that is part of the App Engine development stack, and it is more adequate for request-oriented applications (optimized for read operations). BigTable is a sparse, distributed, persistent multi-dimensional sorted map. It is the technology that lies underneath the DataStore service, and was built with access restrictions that prevent careless or malicious users from causing a query overload. For instance, no query can use an inequality operator ($<$, $<=$, $>=$, $>$, $!=$) on more than one property (a.k.a field) and the result sets are limited to 1000 entries.

Microsoft SQL Azure Database is a cloud-based relational database service built on SQL Server technologies. It provides a highly available, fault tolerant, scalable, multi-tenant database service. Amazon Relational Database Service (Amazon RDS) is a similar service, based on MySQL technology. These relational DBMS in the cloud are cost-efficient, resizable (in capacity), managed by the respective service providers (for time-consuming database administration tasks).

Another challenge in this use case is to materialize the desiderata DBD that describes our hypothetical crop monitoring web application's DBMS needs. There are two options: build a single desiderata DBD, or build three separate DBDs – one for each type of data handled by the application, assuming that it will be easier to find DBMSs that will handle some, but not all data types. In the former option, a single DBD would be expressing the wish to integrate different

data sources in a single DBMS. In the latter option, three separate desiderata DBDs might suggest that data sources will be kept within isolated repositories. Moreover, it is important to reify several application requirements, such as access patterns (read and write) and indexing needs, relationships between its data types, and data model used by the data structures to be persisted.

3.5 Related Work

As pointed out in the introduction there are several initiatives to foster interoperability between applications and DBMS, from n-tiered architectures, passing through database federations, and reaching web services and cloud computing.

DBDs provide a basis for self-describing DBMS, and as such can be seen as a means of structuring (unstructured) facilities of these DBMS. As such, they can be used within the so called UIMA (Unstructured Information Management Architecture) ¹. Unstructured information may be contrasted with the information in classic relational databases where the intended interpretation for every data field is explicitly encoded in the database by column headings – similar to a schema. Unstructured information represents the largest, most current and fastest growing source of knowledge available to businesses and governments worldwide. For unstructured information to be processed by applications that rely on specific semantics, it must be first analyzed to assign application-specific semantics to the unstructured content. The added headings structure provides an initial support to deriving such semantics. By the same token, DBDs provide a high level description of a DBMS (and of application requirements), in which attributes (dimensions) can be used to derive semantics.

We believe that DBDs are specially well suited to the cloud computing scenario [77]. The Open Cloud Manifesto [2] states that: (i) The challenges to cloud adoption are addressed through open collaboration and the appropriate use of standards. (ii) Cloud providers must not use their market position to lock customers into their particular platform. (iii) Cloud providers must use and adopt existing standards. In this scenario, DBMS can be seen as a special kind of provider within the cloud, and DBDs can describe their features, thereby enabling applications to look for the appropriate databases, with help from the repositories.

The Claremont Report on Database Research [4] states that a significant long-term research goal is to transition from managing schemata-based structured data to the managing of structured, semi-structured and unstructured data spread over many repositories in the enterprise and on the Web. This is referred to as the challenge of managing dataspace. Again, DBDs can be seen as a kind of high level descriptor of dataspace.

Federations and integration are two facets of the problem of enabling applications to access

¹<http://docs.oasis-open.org/uima/v1.0/os/uima-spec-os.html>

heterogeneous data sets. Federations [7, 24, 30] allow applications to access distinct DBMS via some kind of mediation layer, which concentrates the "intelligence" needed to transform an application request into a sequence of requests to federation members. The integration approach, on the other hand, assumes that data or schemata have to be adapted, in order to provide a unifying view to all applications. In the same vein, Haas has introduced a model for information integration [22, 23] that consists of four phases: Understanding, Standardization, Specification, Execution. The *Execution* phase is divided in: Materialization (ETL, replication), Virtualization (Federation), and Search. We believe that DBDs are orthogonal to all of these phases, meaning that they could be used to describe them as a whole or separately. Our initial focus is not on schema integration, nor data cleansing, nor self-tuning [36] DBMS – rather, DBDs offer a means for DBMS self-description. In this sense, they can be used by mediators in a federation.

Last but not least, web services [12, 14] are another means to allow flexibility in application execution. Web services can encapsulate a DBMS, and serve as a layer that receives requests for data and returns the appropriate data. In this sense, instead of looking for the appropriate feature DBD, an application could look for appropriate services, and request for data invoking these services, according to standard protocols (e.g., SOAP). Finding the appropriate services would also require looking for service directories (as opposed to looking for matching DBDs in a DBD registry). This offers the advantage of not requiring the specification of feature and desiderata DBDs, nor requires negotiation; on the other hand, this demands extending web service capabilities beyond those normally found – e.g., to accomodate versioning or concurrency requests.

3.6 Conclusions and future directions

This paper presented the concept of database descriptors (DBDs). DBDs could be the foundational bricks to build dataspace, becoming an indispensable tool to allow applications to switch across DBMSs, in a loosely coupled scenario.

DBDs can thus contribute to help to commoditize data services in the cloud, by supporting dynamic switching between DBMSs and applications. Moreover, they can also be seen as a different way of tackling the information integration problem, from a connectivity point of view, in which applications and DBMSs can negotiate their coupling.

This work is part of our initiatives towards interoperability in an eScience context. Future directions include: to create a catalog of concrete feature descriptors, to design a descriptor negotiation framework, to do a proof-of-concept implementation, with real DBMS products and services.

Chapter 4

Evaluating, Reorganizing and Sharing Digital Information Hierarchies

4.1 Introduction

The data deluge is a reality and is here to stay [8]. This presents challenges at several levels – from storage, to retrieval and visualization. This paper is concerned with the problem of digital information *organization*, as a means to cope with these issues. Knowing that *organization* is a heavily overloaded term, we define it as a pattern – the logical structure present in the way information is partitioned into components. In this paper, we restrict our analysis to the topic of *hierarchical organization* of digital information. Hierarchical organizations are a pervasive approach towards understanding and filtering information, inside and outside the digital world. We are particularly interested in the hierarchical organization of digital information, because it is a widespread pattern used to organize any kind of digital content: e.g., files, emails, bookmarks, databases, applications.

We define a *hierarchy* for organizing digital objects as a directed acyclic graph (DAG) $H(V, E)$. V is a finite set of nodes representing *information units* (IUs). E is a finite set of directed edges, where each edge $e_k \in E$ represents a relation of subordination between two nodes $v_i, v_j \in V$ so that $e_{ij} = \{v_i, v_j\} \equiv v_i \rightarrow v_j$. IU nodes can be any digital artifact performing either the role of *aggregator* $v^{agg} \in V$ (e.g. folder, mailbox, or web site) or the role of *content* $v^{cnt} \in V$ (e.g. file, email message or web page). Notice that an IU can assume either or both roles depending on the context of use. Figure 4.1 depicts the notation we adopted to describe hierarchies, making explicit the distinction between *aggregator* and *content* roles.

We furthermore define the *context* in which a hierarchy is created/used as a synonym of the task which a given user (or group) performs when organizing information. For instance, email messages are organized *by subject* for the task/context of “reading only priority topics”, or they are organized *by author* for the task/context of “gathering someone’s point of view”.

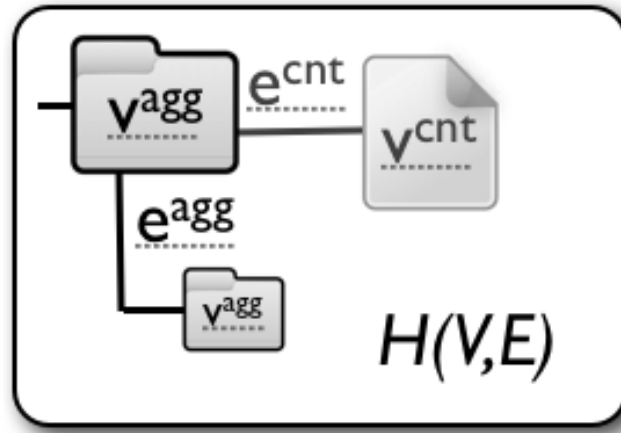


Figure 4.1: Specialized graph notation to describe hierarchies.

The utility of hierarchies to humans is restricted by our information processing limits. In 1956, George Miller published the famous 7 ± 2 rule [46], showing that our limits of absolute judgment (i.e. categorization) are far narrower than our limits on relative judgment (pairwise comparison). For instance, given a single sensory dimension (e.g. size) and a large set of objects, humans have an ample spectrum of discrimination (fine grain discretization) of these objects along that single dimension. On the other hand, when an object is given in isolation, we are capable to categorize it in approximately just 7 sensory dimensions.

Given these cognitive restrictions, the usefulness of hierarchies is evident: they reduce the information space into more general chunks. Any hierarchical *content management system* provides aggregators (e.g. folders and directories) so that users can build their own hierarchies. The problem is that even with the adoption of hierarchies, the digital information space we handle is unmanageable. Manual organization of our files and data is error-prone, not only because of their volume, but also because it is hard to balance between redundancy (i.e. copies created in distinct folders for easy access) and consistency (i.e. keep copies in sync). One explanation for this phenomenon is that as the volume of information grows, the quality of manually created static *ad hoc* hierarchies decreases. Tools and environments to help content organization ultimately lead to the same problems because they are configured and manipulated by us following the hierarchical paradigm. Even digital libraries, which support more flexible retrieval facilities, are not free from cluttered or faulty criteria imposed by digital librarians and developers.

Our contributions concerns the following factors related to hierarchical organizations: (i) content is innately *multi-faceted*. (ii) while building a hierarchy, the creation of a new aggregator node or a new edge (i.e. categorization of content) is guided by a particular content's facet

relevant for a given task; (iii) the semantics and rationale behind a given organization (i.e. the task the user had in mind while choosing content facets and structuring aggregators) is not usually made explicit and materialized, thus the organization's structure is not subject to sharing or reuse; (iv) the hierarchical organization of content does not change in response to context changes (i.e. when the user has to perform a different task than the one the hierarchy was created for). These will be examined in detail in section 4.2. The perception of these factors led us to reason about improving hierarchical organization, ultimately seeking more effective content management. Thus, the main contributions are the following:

- discussion of challenges concerning hierarchical organization of digital content, contrasting with related work – sections 4.2, 4.3 and 4.4
- formalization *organographs* – an approach to meet these challenges 4.5
- validation via an implementation, for a specific case study – 4.6.

We point out that the conceptual framework we propose in this paper is not restricted to textual content, but this paper is focused in this type of content.

4.2 Limiting Factors Leading to Poor Hierarchical Organization

We point out some factors that are responsible for poor hierarchical organization. First, the categorization principle behind a given hierarchical organization is often hidden. In other words, does not exist *explicit* metadata associated to a given hierarchy that: (i) explains the existing aggregators (i.e. defining hierarchical levels) and their subordination relationships; (ii) states categorization criteria indicating to which hierarchical level(s) a content belongs; (iii) serves as input for coherence validation mechanisms or automatic categorization tools. For example, in our email or filesystems, folders are created manually in an *ad hoc* fashion with just a label to serve as a clue for the categorical intent of such aggregator. At creation time, a single folder label seems sufficient as a categorization dimension because we are immersed in some task (e.g. separate professional emails from personal ones). Since content is inherently multi-faceted, the act of placing an IU (e.g. file) inside an aggregator node (e.g. folder) is equivalent to choosing one facet of a multi-faceted object because it is relevant in the context of the current task.

That leads to the second factor, that we often mistake the transient relationship between information unit and aggregator for a permanent bond between content and category. The very nature of categorization is to abstract away many facets (i.e. properties) in favor of a few facets that together reduce the uncertainty among alternatives. Through categorization, we build abstractions from concrete IUs, by filtering out some facets and making others more evident.

This filtering process is guided by the relevance of the chosen facets for some task we have in mind. Therefore, we believe computational systems for information organization should shift the content-driven approach to a task-driven one. Traditional relational database management systems are good examples of the content-driven approach, where the database schema (i.e. content structural organization) is the first-class citizen and queries (i.e. context) are an afterthought. In opposition, NoSQL databases are an example of a context-driven approach, where the rigidity of structure is sacrificed in favor of task flexibility.

Third, aggregators' relations (i.e. the set of edges E) are static. Suppose a person has a body of information organized hierarchically. There comes a time when this person is faced with a new task, for which different facets would provide better filtering to build adequate abstractions. At this time, the hierarchical organization built previously for a prior task may be inadequate for the new task's information retrieval purposes. For example, instead of separating professional from personal email, we may want to cluster all messages from someone who is both a close friend and co-worker.

Another factor is the mixing of *subsumptive containment* with *compositional containment* in the same hierarchy. In subsumptive containment, IUs are subordinate to each other from general-to-specific or is-A relationships. In compositional containment, IUs are subordinate to each other from part-of or has-A relationships. This is one of the factors that leads to involuntary content duplication within a single hierarchy, because the same content IU might be subordinated to different aggregators – one subsumptive and the other compositional.

4.3 Related Work

As mentioned before, our proposal applies to any kind of digital content, but this paper concentrates on exploiting textual content. In this context, our research is connected to architectures and algorithms for text categorization, clusterization and information extraction. Our survey in this section serves as a basis for understanding our proposal and its implementation. According to Qi's taxonomy of classification problems [56], our research interests lie within hierarchical, multi-class, topical (subject) and functional classifications. Text categorization, text clustering and information extraction are the foundations upon which we seek to evaluate, reorganize and ultimately share digital hierarchical information. In a recent article, Blei [9] confirms the up-to-dateness of our goals, in which he proposes probabilistic topic models to enhance the way one organize, browse and understand information. Blei uses *latent dirichlet allocation* to compute hidden variables (i.e. topic structure) from the observed variables (i.e. words of the documents), based on the *bag-of-words* assumption where the order of the words is unimportant.

Our ultimate goal is to convey shareable organization criteria. Popitsch et al [55] propose an ad-hoc file sharing based on Linked Data principles, allowing users to interlink, annotate and browse files mounted from multiple file systems as web resources. Fernandes et al. [17]

describes an ontology-based knowledge system for creating and sharing users' personal objects, where users specify a local ontology to organize and publish their personal objects in community of practices. Hua et al. [25] propose SmartStore – a decentralized semantic-aware metadata organization, which exploits semantics of files' metadata to aggregate correlated files into semantic-aware groups by means of information retrieval tools. Spyns et al. [69] describe approaches to ontology engineering, as related to the description of knowledge about a given domain. In addition, research about folksonomies and social tagging [35] is also relevant to our work, given the effectiveness of collaborative tagging systems for describing resources [75].

4.3.1 Text Classification or Categorization

Text classification or categorization is a sub-domain of information retrieval, whose literature dates back to the '60s. It consists in the activity of labeling natural language texts with thematic categories from a predefined set. In the first 30 years, the main approach to text categorization was based on the *knowledge engineering* paradigm, where expert classification knowledge is manually encoded in a set of rules. From the '90s until now, the *machine learning* paradigm gained increasing popularity. Machine learning consists in a general inductive process to automatically build text classifiers by learning features of interest from a pre-classified document set.

Schutze et al. [63] demonstrated that statistical classifiers could perform better than relevance feedback, a result that fostered automated text classification. Later, Sebastiani [64] presented approaches to automated text classification that fall within the cost-effective machine learning paradigm, also discussing the important sub-problems of document representation, classifier construction, and classifier evaluation. On the issue of performance, Sebastiani concluded that: "Boosting-based classifier committees, support vector machines, example-based methods, and regression methods deliver top-notch performance." in comparison to batch linear classifiers (e.g. Rocchio [29]) and probabilistic classifiers (e.g. Naive Bayes [43]). These results were confirmed by similar experimental findings reported by Yang and Liu [76]. In Sebastiani's survey, we single out the work of [16], [34] and [73], because they explored the hierarchical structure of the category set.

Kiritchenko et al. [32] dealt with hierarchical categorization, and introduced the notion of consistent classification. Gates and Teiken [20] described a system for the construction of taxonomies which yielded high accuracy for automated categorization systems. Dekel et al. [15] formulated the hierarchical classification task as an optimization problem with varying margin constraints, and described new online and batch algorithms for solving it.

Pant and Srinivasan [49] surveyed categorization approaches that could be applied to topical web crawling. Qi et al. [56] provided a survey on the more recent and specific problem of web page classification, that posed different choices on the matter of document representation.

4.3.2 Text Clusterization

In the field of Text Clusterization, we are interested in *conceptual clustering* as a means towards evaluation and reorganization of digital information hierarchies – see section 4.5. As defined by Michalski [45], conceptual clustering is an unsupervised machine learning task where a set of object descriptions are grouped on clusters by an evaluation function. Each of these clusters should aggregate objects that fit the same conceptual description, thus becoming a mechanism for data summarization. Real world approaches to conceptual clustering made the assumption that not all inputs would be available a priori. That led to the development of incremental object assimilation as described by Fisher [18]. Incremental algorithms consume a stream of objects, always adapting to changes in context.

Moreover, the conceptual clustering task differs from classification because besides discovering an appropriate class for an uncategorized object, it must also discover an appropriate concept for the class. Cluster quality depends on the adequacy and coverage of the concept assigned to a cluster in relation to the objects that belong to the cluster. This differs from a *numerical taxonomy* [68], where cluster quality is only based on the nature of the objects and their similarities. Therefore, solutions for conceptual clustering might be used to evaluate and reorganize hierarchies automatically.

Jain et al. [28] introduce the five relevant aspects of any clustering technique, namely: data representation, distance measure, cluster construction algorithm, data abstraction and evaluation criteria. Mishra [47] provides a more in depth description of algorithms for classical clustering objectives. Typical clustering methods are: self-organizing maps [33], agglomerative-divisive hierarchical clustering [38] and partition-based clustering [6]. Bloehdorn et al. [10] discuss how ontologies can be used to improve results on text clustering and classification tasks.

4.3.3 Information Extraction

Information Extraction (IE) plays a key role in the reduction of dimensionality of textual document representation prior to classification or clustering. The literature about IE is extensive, being present in areas such as natural language processing, machine learning, databases and ontologies. *Named entity recognition* is a subproblem of IE in which specific parts of free text should be located and categorized. Examples of named entities are person, organization, date, phone number, and currency. The usual approach for named entity extraction is to code *wrappers*, specialized software that is either rule-based or mining-based. In the former, a domain expert creates a set of explicit rules (e.g. regular expressions) to identify patterns that match with named entities. In the latter, a supervised machine-learning algorithm is trained over an annotated dataset to recognize the same patterns.

Laender et al. [37] presents a short survey of web extractors, introducing a taxonomy for classifying the studied tools according to the technique employed for wrapper generation.

Crescenzi et al. [13] present an unsupervised learning algorithm for wrapper generation restricted to recognize prefix markup languages usually found in HTML pages. Irmak et al. [27] propose a novel framework for the detection of semi-structured entities (e.g. phone, date and time) over 5 different languages with high precision and recall. Turmo et al. [71] present an in depth survey about adaptive IE, concerned with applying machine learning to increase domain independence for IE applications.

4.4 Exploiting Hierarchies

Our methodology and framework support three important aspects of hierarchical organization: (i) they allow the dynamic reconfiguration of a hierarchy, reorganizing aggregators on the fly, according to the user's work context; (ii) they allow users to explicitly define their content clustering criteria, thereby rendering hierarchies shareable and reusable; (iii) they provide evaluation metrics whereby the coherence and consistency of a hierarchy can be measured and distinct hierarchical organizations can be compared, so users can evaluate the suitability of a given organization to their goals. Let us now examine in more detail each of these aspects.

Our first goal is to allow users to *evaluate the coherence and suitability of their hierarchically organized content*. In order to achieve this, we need to define coherence and suitability. We say that a hierarchy $H(V, E)$ is coherent if every edge $e \in E$ satisfies a predicate that represents explicitly one of the categorization criteria for a given task. The task T defines the context under which hierarchy H was created.

Let $e_{ij}^{cnt} \in E$ denote an edge that links an aggregator node $v_i^{agg} \in V$ to a content node $v_j^{cnt} \in V$, and $e_{kw}^{agg} \in E$ denote an edge that links two aggregator nodes $v_k^{agg}, v_w^{agg} \in V$. Then, the validity of the relations (i.e. of the set of edges E) is defined by two boolean predicates:

- a **categorical predicate** $F_{Cat}(e_{ij}^{cnt}) \equiv F_{Cat}(v_i^{agg}, v_j^{cnt})$ returns true if the content node v_j^{cnt} is a child of the aggregator node v_i^{agg} , under the corresponding categorization criterion or false otherwise.
- a **hierarchical predicate** $F_{Hil}(e_{ik}^{agg}) \equiv F_{Hil}(v_i^{agg}, v_k^{agg})$ returns true if there is a direct subordination relation (e.g. compositional such as holonymy/meronymy, subsumptive such as hyperonym/hyponym) between nodes v_i^{agg} and v_k^{agg} , otherwise returns false.

Thus, in the scope of a given task T , a hierarchy H is *coherent* if all edges are valid, i.e., the pre-defined predicates F_{Cat} and F_{Hil} are true for all edges in the DAG $H(V, E)$. If any of these predicates fails, then the hierarchy is said to be *incoherent*.

A hierarchy H is *suitable* for a given task T if:

1. it is coherent;

2. it is stable: every time T is performed, there are no changes in V^{agg} (i.e. the set of all aggregator nodes) and no changes in E^{agg} , though changes in leaves V^{cnt} (i.e. the set of all content nodes) and their respective edges E^{cnt} are allowed;
3. there is strong similarity between siblings, i.e. given a similarity function $\sigma(v_i, v_j)$ in the context of the task T , if v_i and v_j are siblings then $\sigma(v_i, v_j) > \epsilon$, where ϵ is a lower threshold to consider two nodes similar;
4. there are no unintentional duplicate nodes;
5. cardinalities in H lie within cognitive limits (the 7 ± 2 rule [46] is a suggestion, actual limits should be user configurable), meaning that there are not too many siblings under the same v^{agg} nor too long paths from the root to any given node.

The result of the evaluation process can be used to improve H without changing V^{cnt} . Moreover, it can be used to check the suitability of H to a task T .

Our second goal is to allow users to *dynamically reconfigure hierarchies* in response to task switching. In this case, users are concerned with their own organizations for their own purposes, without sharing in mind. Regular filesystem folders, application menus and navigational hyperlink structure of web sites are examples of different degrees of rigidity and immutability. For example, a researcher's email database can be used to: elaborate a project proposal, compile the biography of a colleague, re-engineer a course's pedagogical program or compile material for a book. Each of these tasks can be performed upon the same corpus of information units (i.e. content nodes), and yet each task can profit from a different hierarchical organization. Why should one task be privileged in detriment of others? Instead of creating a single concrete hierarchy, we seek to organize content in a way that dynamically specified hierarchies (e.g. as 'views' in relational databases) can be built on-the-fly for the task in context. This means re-defining the sets E^{agg} and E^{cnt} . This reorganization may also change V^{agg} by discarding or introducing nodes. Usually V^{cnt} remains the same, but may be reduced in the process due to discarded content nodes irrelevant to the present task.

Our third and last goal is to *allow users to share the way they organize information* by making their (now multiple) hierarchization criteria explicit. Three aspects need to become explicit: (i) F_{Cat} that represents the relation between aggregator nodes and their subordinate content nodes (i.e. the set E^{cnt}); (ii) F_{Hil} that represents the relations of subordination in a hierarchy of aggregator nodes (i.e. the set E^{agg}); (iii) and an identifier for the task T that acts as a namespace contextualizing the relations (i) and (ii). If these three aspects can be specified unambiguously, we have a digital artifact that is explicit and shareable.

4.5 Organographs

An ‘organograph’ [65] is an artifact to make explicit *how to organize information* in the context of a particular task. In more detail, an organograph f_{org} is a graph transformation function that takes an input hierarchy $H_{in}(V, E)$ and produces an output hierarchy $H_{out}(V', E')$.

4.5.1 Applicability

There are three scenarios to apply organographs: evaluation, reorganization and sharing. Consider two acquaintances: Alice and Bob. In the first scenario, Alice evaluates if some hierarchical collection of hers (H_{alice}) is indeed well-organized with respect to a task. By well-organized we mean: $H_{alice}(V, E)$ adheres to the organizational criteria Alice had in mind, the structure of H_{alice} lies within reasonable cognitive limits, similar content is grouped together, any duplication in V scattered around H_{alice} is intentional, not accidental.

In the second scenario, Alice is satisfied with H_{alice} to perform task T_1 . However, when Alice needs to perform some other task T_2 , she finds out that H_{alice} is unsuitable because $(V_{in}^{agg}, E_{in}^{agg})$ do not provide adequate clustering of V_{alice}^{cnt} for the exploratory needs of T_2 .

In the third scenario, suppose Alice and Bob are both researchers with intersecting interests. Bob gives Alice free access to his content collection H_{bob} , but Alice wants to reorganize Bob’s content collection V_{bob}^{cnt} using $(V_{alice}^{agg}, E_{alice}^{agg})$ and then browse it as if it was her own.

Evaluation

In the first scenario, we are interested in evaluating the organizational quality of H_{in} .

As an example, let MAX_SIBLINGS and LONGEST_PATH respectively be upper thresholds for the outdegree of an aggregator node and the length from the root to the farthest leaf in a hierarchy. These constraints are inspired by the 7 ± 2 cognitive rule [46]. Let $0 \leq \sigma(v_i^{cnt}, v_j^{cnt}) \leq 1$ be a function that computes the similarity of two content nodes in a hierarchy. Let ϵ be a lower bound threshold of similarity for which if $\sigma(v_1, v_2) \geq \epsilon$ then v_1 and v_2 are considered similar. The evaluation of $H_{in}(V, E)$ could indicate the following issues in need of repair:

- $\exists v \in V^{agg}$ where $outdegree(v) > MAX_SIBLINGS$;
- $\exists path(root, v^{agg})$ where $length(p) > LONGEST_PATH$;
- $\exists v_i^{cnt}, v_j^{cnt}$ for which $siblings(v_i, v_j) \wedge (\sigma(v_i, v_j) < \epsilon)$;
- $\exists \sigma(v_i^{cnt}, v_j^{cnt}) > \epsilon$ then v_i^{cnt} and v_j^{cnt} are considered duplicates, even if they are not bit-by-bit identical.

We point out that intentional duplicates are not considered an organizational error. In case of poor evaluation results, an appropriate f_{org} can be tailored to transform H_{in} into a different H_{out} that yields better evaluation results for the same criteria.

Reorganization

In the second scenario, consider a hierarchy H_{in} built for task T_1 that needs to be *reorganized* or *refactored* to become suitable to perform task T_2 . So, it suffices to construct the organograph f_{org} that captures the needs of T_2 , where $H_{out}(V', E') = f_{org}(H_{in}(V, E))$ is the desired refactored hierarchy. The challenge is to translate semantic aspects of T_2 into the mechanisms available for specifying an organograph that will effectively transform H_{in} into H_{out} .

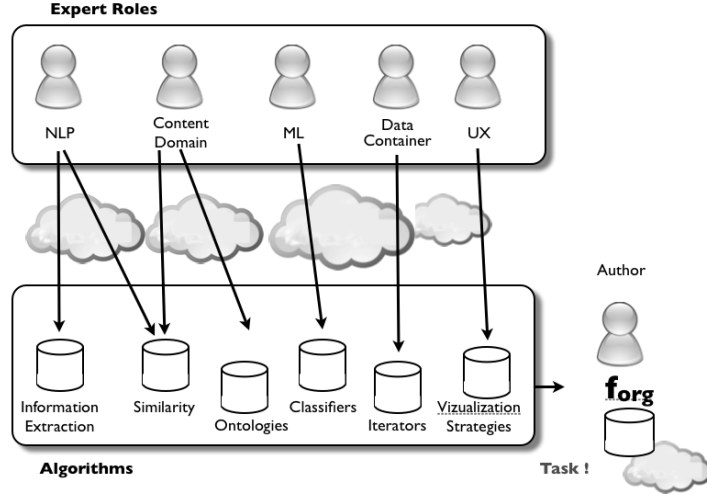
We illustrate this with a concrete example: a collection of papers (i.e. articles) is organized by *topic* \rightarrow *author* \rightarrow *year* \rightarrow *papers*. This hierarchy is useful to the task of identifying the recent publications of some author under a given topic. Now, suppose the new task is to find what a given author is recently writing about. It would be more appropriate to have the collection reorganized as *author* \rightarrow *year* \rightarrow *topic* \rightarrow *papers*. In this particular example, f_{org} preserves the nodes but defines a new set of edges while transforming $H_{in} \rightarrow H_{out}$. The transformation encapsulated in f_{org} can express any composition of basic operations on nodes and edges, such as to preserve, discard or create them. The portion of H_{in} that is preserved or changed in H_{out} is dictated by the task transition $T_1 \rightarrow T_2$.

Sharing

In the third case, two or more groups of people that need to perform the same task T_3 share either the structural organization (H^{agg}) or content (H^{cnt}). Sharing can be seen as a particular case of reorganization, where $(V_{out}^{agg}, E_{out}^{agg})$ comes from a different collection than the one that provided V_{out}^{cnt} .

In other words, consider two distinct user groups, where $Group_1$ owns $H_1(V_1, E_1)$ and $Group_2$ owns $H_2(V_2, E_2)$. Now, $Group_1$ wants access to V_2^{cnt} . Sharing means that $Group_1$ needs to create a graph H_{out} to access V_2^{cnt} . The organograph to be constructed needs to consider both V_2^{cnt} and aggregates E_1^{agg} and V_1^{agg} , and construct E_{out}^{cnt} , using a categorization algorithm such as the ones discussed in Section 4.3.

Section 4.6 provides a concrete detailed example to illustrate this scenario. Notice that if the groups share the same unchanged collection (i.e. both structural organization and content $H(V, E)$ are shared), then organographs are not necessary because no reorganization takes place (i.e. f_{org} is an identity transformation).

Figure 4.2: Framework for instantiation of f_{org}

4.5.2 Organograph Instantiation

In order to apply organographs to transform hierarchies, we need to describe their *instantiation* and *execution*.

The instantiation of an organograph consists of defining a *name*, a *purpose* and a *transformation*. The *name* can be any textual label; ideally, the name should be an Uniform Resource Identifier (URI), solving the problems of protocol, unicity, location and identification. The *purpose* is a textual description documenting for what tasks this organograph is suitable, and optionally containing semantic annotations and markup to be ‘understandable’ by both humans and machines. The *transformation* f_{org} comprises F_{Hil} and F_{Cat} , thereby also including navigational and feature extraction mechanisms.

F_{Hil} defines how to build E_{out}^{agg} , establishing the structural relationships for V_{out}^{agg} . In a valid organograph, F_{Hil} must cover every $v_i^{agg} \in V_{out}^{agg} \subset H_{out}$.

F_{Cat} establishes how to build E_{out}^{cnt} . F_{Cat} can be defined by choosing a combination of suitable pre-existing classifiers and information extraction algorithms. These information extractors will be used to single out the relevant facets from V_{in}^{cnt} . F_{Cat} may not consider all content nodes in H_{in} – i.e., the organograph will select the information relevant for the task at hand. The classifiers will categorize those extracted facets into appropriate categories that correspond one-to-one with V_{out}^{agg} nodes. If the nature of the chosen classifiers is supervised, then the respective knowledge derived from training sets must also be specified and encapsulated in the organograph. This feature allows users to share their categorization schemes by using their

own collections as examples, not necessarily sharing their content as well.

As a result of this creation process, the actual instantiation of f_{org} can be defined as a 4-tuple $\langle \text{URI, description, } F_{Cat}, F_{Hil} \rangle$. This tuple represents the core structure of an organograph instance. For the sake of simplicity, we assume that f_{org} embeds a proper traversal mechanism (iterator) suitable to traverse the input hierarchy (H_{in}) given its location (i.e. URI). This traversal parameter can be factored out in a concrete implementation, leading to greater flexibility and decoupling the nature of the transformation from the content itself. There are many possible choices for the actual specification of an organograph, such as: XML, RDF or even a new domain specific language (DSL). Discussing tradeoffs and suitability of each choice lies outside the scope of this paper.

Figure 4.2 portrays a possible set of experts and algorithms needed to create organographs. It shows that in several cases many experts must combine suites of algorithms to be invoked in the required transformations. Such diversity of roles and algorithms is an evidence of how regular users could benefit from organographs instead of performing ad-hoc and manual categorizations (where a single person ought to perform all roles simultaneously). One benefit of using organograph instances is to empower users with this collective expertise applied to their own organizational needs. For instance, experts in Machine Learning (ML) provide classifier algorithms to be used in the definition of F_{Cat} . Domain experts provide ontologies from which F_{Hil} and V_{out}^{agg} can be derived. Specialists in Natural Language Processing (NLP) provide information extraction tools, and the algorithms to compose the similarity comparison function σ . Moreover, crawlers and iterators are required to render content uniformly accessible, considering that it can be stored in all sorts of repositories or databases. The final role is the organograph creator, who assembles several of these components into f_{org} to transform $H_{in} \rightarrow H_{out}$ in the context of task T .

4.5.3 Organograph Execution

The execution of an organograph instance is the act of transforming one hierarchy into another. The input hierarchy H_{in} is specified by a URI, solving the location problem for local and remote hierarchies transparently. Notice that H_{in} can be any digital artifact for which the concepts of aggregators and content nodes can be mapped forming a DAG, such as: a local filesystem subtree, a remote filesystem storage in the cloud, a Website, a mailbox, a collection in a NoSQL database. The choice of *traversal mechanism* is not only a requirement for the organograph to be able to browse the input hierarchy H_{in} , it is also a means for the organograph user to define the information subset of interest.

The output hierarchy H_{out} produced by the execution of f_{org} must be published somewhere. Therefore, we adopt another URI parameter to specify the publish location of H_{out} .

All of those presuppose the existence of an execution engine, capable of: (i) understanding

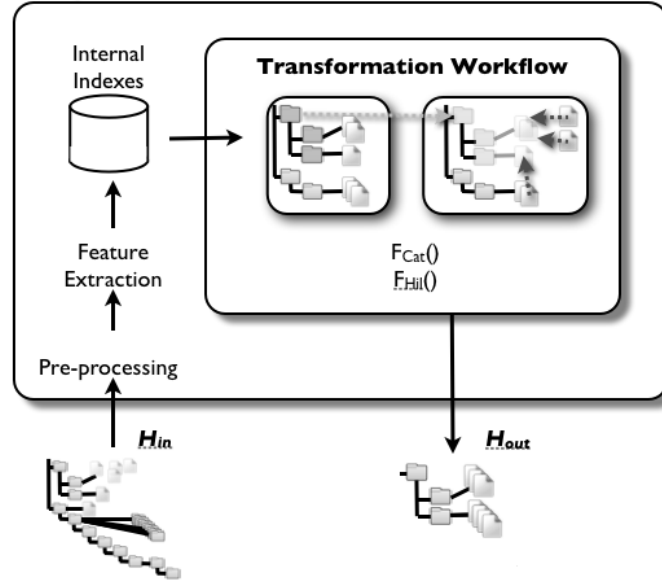


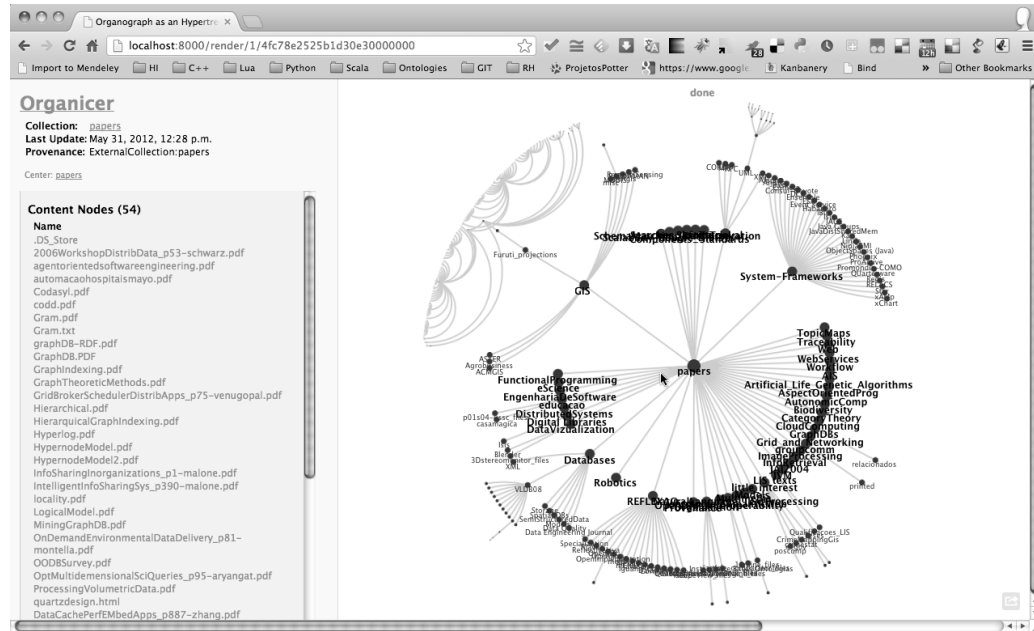
Figure 4.3: Organograph Execution

the specification of the organograph; (ii) traversing H_{in} through the given traversal mechanism; (iii) performing the transformations specified by the algorithms encapsulated in f_{org} ; (iv) exporting H_{out} to the desired output URI;

Figure 4.3 sketches the execution of an organograph. First, the user chooses a source hierarchy H_{in} . A proper iterator is used to visit all nodes belonging to V_{in}^{cnt} , each of which is assigned a unique ID. Information extractors select only the facets considered relevant in the context of task T . Next, F_{Hil} defines the edges E_{out}^{agg} that connect V_{out}^{agg} , and F_{Cat} defines the edges E_{out}^{cnt} . The execution of f_{org} is depicted as a workflow that produces as a result the output hierarchy H_{out} . This workflow will invoke the appropriate algorithms, as defined by the experts – see Figure 4.2

4.6 Experiment: Using ACM's CCS98 to reorganize a private collection of papers

We now present an experiment that illustrates the use of organographs as a means to convey reusable organizations. Our goal was to test the organograph conceptual framework through a concrete implementation. We started from an input collection of computer science papers

Figure 4.4: Original paper collection $H_{initial}$ with ad-hoc organization

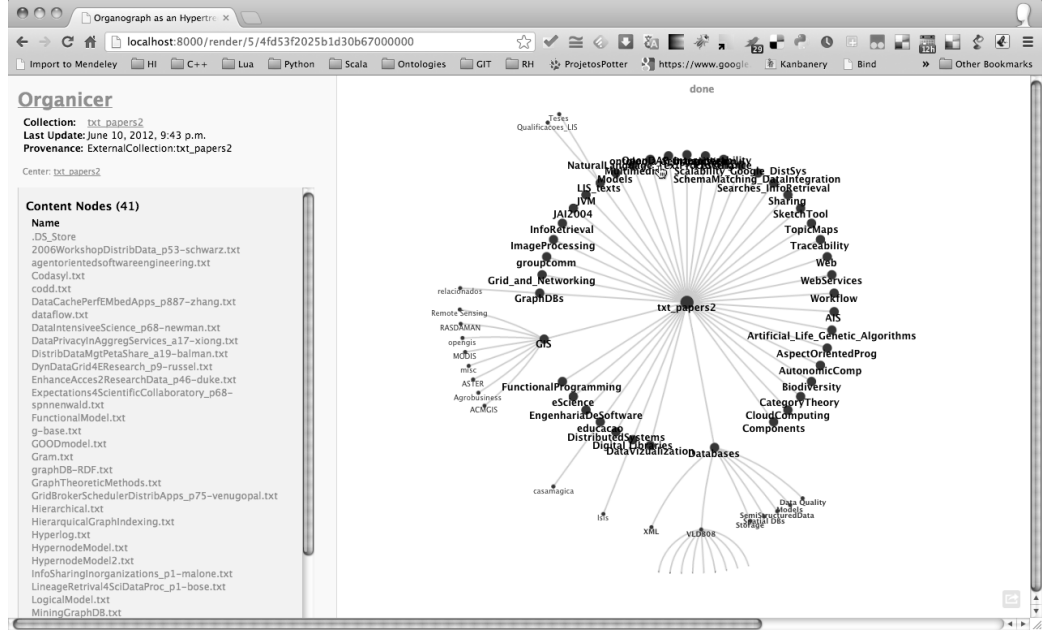
($H_{initial}$), gathered and organized¹ by the first author of this paper in an *ad hoc* fashion. This collection was transformed into another collection H_{mine} , where all content nodes (scientific papers) were transformed into a uniform textual format. Our goal is to reorganize the papers from H_{mine} (i.e. V_{mine}^{cnt}) by topic, according to the ACM Computing Classification System – CCS98 [3]. Let H_{acm} denote ACM CCS98; the output reorganization will reuse just V_{acm}^{agg} and E_{acm}^{agg} .

In order to conduct this experiment, we developed a tool, which we called *Organicer*, that supports visualization of hierarchies, their evaluation (by computing several metrics on a hierarchy) and the construction of organographs (by offering a suite of different algorithms that can be invoked and composed to execute f_{org}).

4.6.1 The Organicer Platform

We have implemented a web-based software tool, called *Organicer*, that served as platform to perform the experiment. Organicer allows the user to fetch digital collections from URIs, and execute organographs to reorganize and visualize them. The tool has a plugin-based architecture to ease the development of extensions, such as new collection iterators, machine learning

¹The input collection was organized prior and unaware of this experiment.

Figure 4.5: Collection H_{mine} (after transformation from $H_{initial}$)

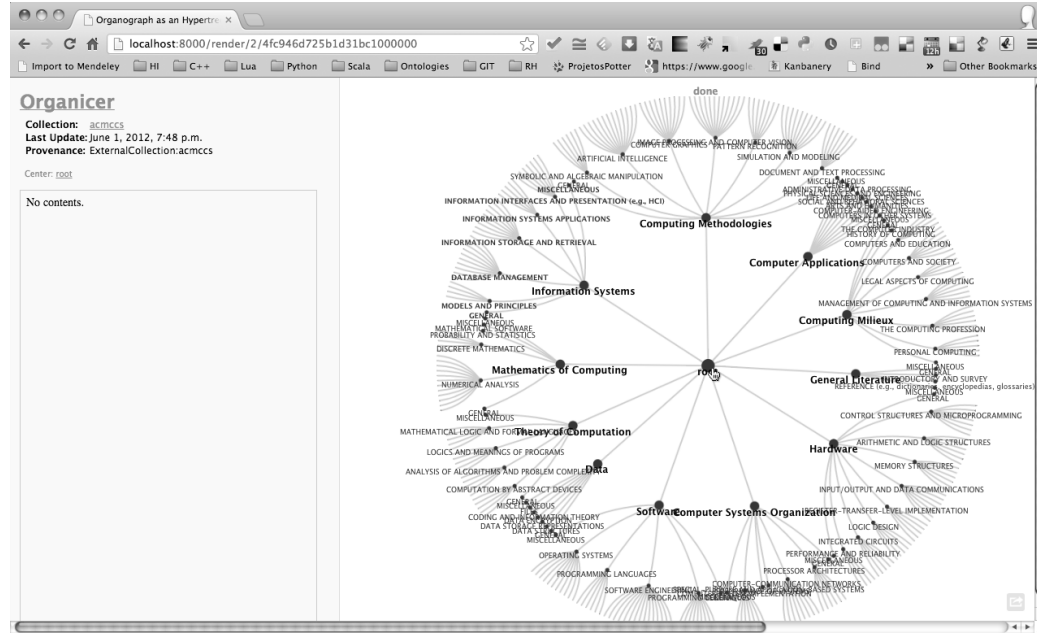
algorithms and visualization renders.

Organicer is implemented in the Python programming language, and reuses external modules (provided by third parties) for: (i) numerical computations, plotting and statistics (NumPy, SciPy and Matplotlib); (ii) natural language processing (NLTK); (iii) topic modeling algorithms (LSI and LDA in Gensim [58]); (iv) visualization algorithms (InfoViz and D3 javascript toolkits);

In this experiment, Organicer used two iterators: one for the local file system holding the input collection of papers, and another to fetch and parse CCS98 definition directly from the Web. Using the plugin architecture we foresee the creation of iterators to allow the reorganization of web pages and files stored in the cloud (e.g. data containers such as: Google Docs/Drive, Dropbox, Evernote, Delicious).

4.6.2 Transforming $H_{initial}$ into H_{mine}

The original paper collection $H_{initial}$, depicted in figure 4.4, contained $|V^{agg}| = 420$ directories and $|V^{cnt}| = 1739$ pdf files. It was transformed into H_{mine} through an organograph that converted $V_{pdf}^{cnt} \rightarrow V_{txt}^{cnt}$, and preserved V^{agg} , E^{agg} , E^{cnt} if possible. Invalid content (i.e., files that could not be converted) were discarded, thereby eliminating edges from the original hier-

Figure 4.6: Collection H_{acm} derived from ACM CCS98

archy. Notice that this kind of reorganization only eliminates edges and content leaves from a hierarchy, it does not really re-organize the collection.

Three conversion mechanisms (pdftotext, pdfbox, pypdf) were evaluated, they had the same performance, failing to convert 416 pdf nodes (24% of original V^{cnt}) - the set of discarded nodes became $V_{discarded}^{cnt}$. This set was linked to a subset of E^{cnt} that was also removed. Transitively, the set $E_{discarded}^{cnt}$ was linked to a subset V^{agg} representing empty directories that were removed as well. The result, hierarchy H_{mine} , depicted in figure 4.5, contained $|V_{mine}^{agg}| = 72$ and $|V_{mine}^{cnt}| = 1323$. All 1323 V_{mine}^{cnt} nodes were manually annotated to reliably identify the the paper. These annotations would be used later in the experiment to cross-validate the classification.

4.6.3 Initial evaluation

By choosing CCS98 we gained a well-thought hierarchical content organization that is accepted by the scientific computing community, and suitable for the task of topical classification of the computing literature. Moreover, we can use the ACM classification (i.e. index terms) from the ACM Digital Library service, and cross-validate the performance of the classification algorithm encapsulated in the organograph.

Table 4.1: Statistics about the structure of the hierarchies computed by Organicer

	H_{mine}	H_{acm}
$ V^{agg} $: total aggregator nodes	72	369
$ V^{cnt} $: total content nodes	1323	1323
$\max E^{agg} $	42	15
$\text{mean } E^{agg} (\mu)$	0.98	0.99
$\text{std. deviation } E^{agg} (\sigma)$	5.13	2.66
longest path in $H(V, E)$	4	4
μ : mean path in $H(V, E)$	2.35	3.46
σ : std. deviation path	0.84	0.92

Table 4.6.3 compares the structural statistics between H_{mine} and H_{acm} , showing that ACM CCS98 provides a more balanced content distribution. Organicer allows visual inspection of these results by rendering the hierarchies in each stage with an hyperbolic tree layout. Figure 4.4 represents the original paper collection. Figure 4.5 shows H_{mine} . Figure 4.6 shows the aggregation nodes of H_{out} and H_{acm} – both have the same visual representation, because we did not plot V^{cnt} nor E^{cnt} .

4.6.4 Sharing the organization of H_{acm} – reorganization of H_{mine}

The experiment consisted in reorganizing the *ad hoc* H_{mine} paper collection by topic, according to the ACM CCS98, represented by H_{acm} . To do this, we built an organograph f_{org} to transform H_{mine} into H_{out} , satisfying the following: $V_{out}^{agg} \subseteq V_{acm}^{agg}$ and $E_{out}^{agg} \subseteq E_{acm}^{agg}$ and $V_{out}^{cnt} \subseteq V_{mine}^{cnt}$.

The code in Algorithm 1 gives an overview of the organograph used to construct this experiment. We have used the Python language syntax because it is terse and resembles pseudocode. Lines 2-18 define the organograph f_{org} , while lines 20-22 represent the organograph execution. Line 5 obtains the definition of ACM CCS98. Lines 6-11,17 define the F_{Cat} component, while line 15 defines the F_{Hil} .

We point out that this example shows one possible materialization of the organograph built for this specific experiment. Other implementation materializations can be conceived, though alternatives for representation and coding of organographs are not discussed here.

f_{org} was built to apply a *Naive Bayes* classifier over the titles in H_{mine} papers.

Our approach tried to imitate what a user does while fetching a document from the web and immediately save it in a local folder. Our assumption for this scenario is that many classification actions performed by users are carried out without deep analysis of the paper contents, and often just the title is examined. The classifier considered 11 classes corresponding to the 11 first-level topics from ACM CCS98. The training set for each class (e.g. *B. Hardware*) was the union of labels from its children nodes in H_{acm} . The leaves in H_{acm} contained additional classification

Algorithm 1 Possible materialization of f_{org}

```

01 @organograph
02 def forg_ccs98(self, input, output):
03     self.id = new_uuid() # ff7d8e21-4226-11e2-b2f1-109add6b426c
04     self.description = "docs by ACM CCS98"
05     ccs98 = extract("http://www.acm.org/about/class/1998/ccs98.xml")
06     trainset = []
07     for category, words in nlp_clean_titles(ccs98.Vcnt.paths):
08         for w in words:
09             trainset.append((make_feature(w), category))
10
11     classifier = NaiveBayes(trainset)
12
13     out = collection(output)
14     # FHil - implicitly defines out.Vagg
15     out.Eagg = ccs98.Eagg.Level[:1]
16     # FCat - implicitly defines out.Vcnt
17     out.Ecnt = classifier.classify(input)
18     return out
19
20 # Organograph Execution
21 Hin = collection("file:///some/local/dir/docs")
22 Hout = forg_ccs98(input=Hin, output="rodsenra@dropbox:/any_folder")

```

Table 4.2: Accuracy of f_{org} classification according to ACM DL

Class	-self	-self-A	-self-A-D-I-G-K
A. General Literature	0.39	-	-
B. Hardware	0.37	0.38	0.72
C. Computer Systems Organization	0.37	0.39	0.72
D. Software	0.38	0.42	-
E. Data	0.37	0.46	0.72
F. Theory of Computation	0.40	0.41	0.75
G. Mathematics of Computing	0.37	0.39	-
H. Information Systems	0.28	0.29	0.11
I. Computing Methodologies	0.34	0.35	-
J. Computer Applications	0.37	0.39	0.72
K. Computing Milieux	0.34	0.39	-
All	0.37	0.39	0.72

terms that were used to train the classifier.

However, we still needed to validate if the E_{out}^{cnt} computed by f_{org} represented a correct categorization. In order to do that, we wrote a web crawler to fetch from the ACM Digital Library all relevant metadata (namely, classification terms) from each paper in H_{mine} , given its title as input. Therefore, we used ACM DL’s own classification scheme to automatically evaluate the accuracy of the classification performed by f_{org} .

The experiment was configured with $V_{mine}^{cnt} = 1323$, for which the crawler succeeded to retrieve only 520 indexed papers (39%) from ACM DL. Table 4.2 presents three scenarios for variations in f_{org} . Each value represents the classification accuracy calculated as the ratio between successful classifications divided by the total number of documents. For example, Column 1 shows that the total accuracy of the experiment was 0.37 but it increased to 0.40 by removing the class F. Column 2 removes class A for all test cases increasing the accuracy in all of them. Column 3 represents an improvement with total accuracy of 0.72 when some ambiguous classes (i.e. A,D,I,G and K) are removed. Notice that in ACM DL, many documents are classified in multiple classes, thus removing ambiguous classes did not decreased the total number of documents classified. The results in table 4.2 show that we could build organographs with a simple classification scheme, and use them to identify classes that mostly contributed to misclassifications. The removal of such ambiguous classes resulted in quantitative improvements in classification accuracy.

Some limiting factors to increase accuracy in this experiment were: the small training set, the classification algorithm, the feature set (considering document titles), and finally high semantical overlap in the classes. Nevertheless, our goal with this experiment was to validate the idea of browsing some collection (V_{mine}^{cnt}) with a different organization structure (V_{acm}^{agg}).

4.7 Conclusion

In this paper we have modeled hierarchies of digital information as DAGs with two types of nodes and edges: aggregators (V^{agg}, E^{agg}) and contents (V^{cnt}, E^{cnt}). We used this notation to describe problems related to evaluation, reorganization and sharing of such hierarchies.

We have identified several limiting factors related to content organization: there is a lack of evaluation tools for hierarchies with ad-hoc and implicit organization criteria; current mechanisms and methods lead to static and content-driven hierarchies instead of a more flexible dynamic and task-driven approaches; and finally the incapacity to share organization criteria and apply it automatically to reorganize different content to perform some recurring task.

We presented *organographs* as a conceptual framework to transform implicit organization information criteria into explicit parameters, in the context of a particular task. We discussed how organographs can be used to evaluate, reorganize and share digital information hierarchies. Furthermore, we presented a concrete organograph use case, where a personal collection of papers is reorganized according to ACM subject headings and evaluated before and after reorganization. Our results show that content organization can be shared, and *ad-hoc* hierarchies can be refactored into more balanced hierarchical structures while preserving valid categorical relationships between content and aggregator nodes. This paper concentrated on (re)organizing and sharing content by managing the hierarchies that describe it. Even though all examples given in this text concentrate on filesystem hierarchies, aggregators also represent any other metaphor for aggregative (eg. containment) or subsumptive (eg. generalisation) hierarchical connector.

Future research steps include: enhance the palette of components from which to build organographs; lower the barrier of organograph specification and execution to be accessible to non-technical users; and test the concept with different media (i.e. images, audio, video). Moreover, we seek validation of organographs in a wider audience. We hope to achieve the latter goal making our tools Web available and gathering feedback from the community.

Chapter 5

Conclusions

5.1 Contributions

This work touched several Computer Science fields, all of which try to attack *database interoperability* and *data integration in eScience*, crossing the border into *content management* and *personal information management*. All results had the same backbone: sharing as a means to transform data into information and then into knowledge.

SciFrame presented an integrated perspective of efforts and phases involved in sharing of eScience data, as well as an uniform vocabulary to describe the concepts involved. We used SciFrame to describe WebMAPS – a case study in scientific data sharing, with emphasis on the problem of distributing large datasets over the Web. Our recommendation to data dissemination services was the adoption of RESTful oriented services that allow fine grained data distribution as well as bulk transferences.

The search for interoperability motivated us to find solutions to the problem of binding applications to data services in the cloud. Our approach was to borrow from the field of *image processing* the idea of *descriptors*, which were adapted into a mechanism to match application requirements with data services capabilities. However, the proposed mechanism still lacks experimental validation.

The perception that *inter-people information sharing* simplified to *oneself information sharing* matches the problem of *personal information organization* led to the proposal of *organographs*. Organographs transform implicit organization information criteria into explicit parameters that can be changed according to task context changes. We performed experimental validation of organographs by developing *Organicer* – an organograph-based extensible platform that can be used to evaluate, reorganize and share digital information hierarchies. We tested Organicer with a concrete example, where a personal collection of papers is reorganized according to ACM subject headings and evaluated before and after reorganization. Our results show that content organization can be shared, ad-hoc hierarchies can be refactored into more balanced hierarchi-

cal structures while preserving valid categorical relationships between content and aggregator nodes.

Our main contributions were:

- SciFrame – a conceptual framework to describe and compare systems or processes involving scientific digital data manipulation;
- the proposal of Database Descriptors (DBDs) – a conceptual framework to capture and match application requirements to database capabilities, helping the coupling between application and data services in the cloud;
- the design of Organographs – a conceptual framework to make organization information criteria explicit and bound to a particular task; Organographs can be used to evaluate, reorganize and share digital information hierarchies. DBDs can be used in organograph instances to provide loose and dynamic coupling with external data sources (i.e. hierarchies). SciFrame can be used to describe aspects of the transformation encapsulated in an organograph instance.
- development of software tools and algorithms (in Organicer, WebMAPS and Paparazzi) that supported experimentation and validation for some of the proposed ideas.

5.2 Extensions

There are many possible extensions to this work. From a theoretical point of view, examples are:

- SciFrame: formalize the description of SciFrame as a traditional design pattern, enhancing the details about the *information management* operations. Explore if there is an alignment between the concepts present in SciFrame with corresponding concepts in CLRC [70].
- DBDs: compile a comprehensive *vade mecum* of database capabilities to enrich DBDs expressivity, and explore comparison algorithms that rank DBDs according to a prioritized set of capabilities. Another extension is to formalize DBDs, perhaps exploring the notion of semantic annotations.
- Organographs: draw a parallel between the underlying mathematical model of Organographs and Category Theory, and formalize a language to express instances of Organographs. Formal representations of f_{org} for particular domains should also be exploited.

From a practical point of view, we suggest:

- SciFrame: create an online catalog of eScience systems using SciFrame's schema. Design and implement metrics to evaluate the advantages of using SciFrame.
- DBDs: conduct experiments with concrete data services using DBDs to rebind application to services through dynamic negotiation.
- Organographs: validate Organographs with other media types, such as: audio, images and video. An example of such an extension would be to take advantage of image descriptors in organizing images as content. Moreover, enhance the palette of components from which to build organographs by incorporating other information extractor and classifier algorithms, and exploit user interactions with the system.

...

Bibliography

- [1] Dublin core metadata initiative. <http://dublincore.org/>, 2009.
- [2] <http://www.opencloudmanifesto.org/openWeb>, 2009.
- [3] ACM CCS. Acm’s computing classification system (ccs). <http://www.acm.org/about/class/1998>, 2010.
- [4] R. Agrawal, A. Ailamaki, P. A. Bernstein, E. A. Brewer, M. J. Carey, S. Chaudhuri, A. Doan, D. Florescu, M. J. Franklin, H. Garcia-Molina, J. Gehrke, L. Gruenwald, L. M. Haas, A. Y. Halevy, J. M. Hellerstein, Y. E. Ioannidis, H. F. Korth, D. Kossmann, S. Madden, R. Magoulas, B. C. Ooi, T. O’Reilly, R. Ramakrishnan, S. Sarawagi, M. Stonebraker, A. S. Szalay, and G. Weikum. The claremont report on database research. *SIGMOD RecM*, 37(3):9–19, 2008.
- [5] M. Armbrust, A. Fox, R. Griffith, A. D Joseph, R. H Katz, A. Konwinski, G. Lee, D. A Patterson, A. Rabkin, I. Stoica, et al. Above the clouds: A berkeley view of cloud computing. *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28*, 2009.
- [6] L.D. Baker and A.K. McCallum. Distributional clustering of words for text classification. In *ACM SIGIR ’98: Proc. of the 21st Annual Inter. Conf. on Research and Development in Information Retrieval*, pages 96–103. ACM, 1998.
- [7] S. Berger and M. Schrefl. From federated databases to a federated data warehouse system. *Hawaii Inter. Conf. on System Sciences*, 0:394, 2008.
- [8] F. Berman. Got data?: a guide to data preservation in the information age. *Commun. ACM*, 51:50–56, 2008.
- [9] D. M. Blei. Probabilistic topic models. *Commun. ACM*, 55(4):77–84, 2012.
- [10] S. Bloehdorn, P. Cimiano, and A. Hotho. Learning ontologies to improve text clustering and classification. *From Data and Information Analysis to Knowledge Engineering*, pages 334–341, 2006.

- [11] T.N. Carlson and D.A. Ripley. On the relation between NDVI, fractional vegetation cover, and leaf area index. *Remote Sensing of Environment*, 62(3):241–252, 1997.
- [12] E. Cerami. *Web Services Essentials*. O'Reilly & Associates, Inc., 2002.
- [13] V. Crescenzi and G. Mecca. Automatic information extraction from large websites. *Journal of the ACM (JACM)*, 51(5):731–779, 2004.
- [14] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana. Unraveling the web services web: An introduction to soap, wsdl, and uddi. *IEEE Internet Computing*, 6:86–93, 2002.
- [15] O. Dekel, J. Keshet, and Y. Singer. Large Margin Hierarchical Classification. *Journal of the American Statistical Association*, 104(487):1213, 2004.
- [16] S. Dumais and H. Chen. Hierarchical classification of web content. In *ACM SIGIR '00: Proc. of the 23rd Annual Inter. Conf. on Research and Development in Information Retrieval*, pages 256–263. ACM, 2000.
- [17] A. Fernandes, A. M. de C. Moura, and F. Porto. An ontology-based approach for organizing, sharing, and querying knowledge objects on the web. In *DEXA '03: Proc. of the 14th Inter. Workshop on Database and Expert Systems Applications*, pages 604–609. IEEE, 2003.
- [18] D. H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2(2):139–172, 1987.
- [19] M. Folk, R.E. McGrath, and K. Yang. Mapping HDF4 Objects to HDF5 Objects. *National Center for Supercomputing Applications, University of Illinois, July*, 2002.
- [20] S.C. Gates, W. Teiken, and K.S.F. Cheng. Taxonomies by the numbers: building high-performance taxonomies. In *Proc. of the 14th ACM Inter. Conf. on Information and Knowledge Management*, pages 568–577. ACM, 2005.
- [21] V. Getov. e-science: the added value for modern discovery. *Computer*, 41(11):30–31, 2008.
- [22] L. M. Haas. Beauty and the beast: The theory and practice of information integration. *Database Theory–ICDT 2007*, pages 28–43, 2007.
- [23] L. M. Haas, M. Hentschel, D. Kossmann, and R. J. Miller. Schema and data: A holistic approach to mapping, resolution and fusion in information integration. In *Conceptual Modeling-ER*, pages 27–40, 2009.

- [24] D. Heimbigner and D. McLeod. A federated architecture for information management. *ACM Transactions on Information Systems (TOIS)*, 3(3):253–278, 1985.
- [25] Y. Hua, H. Jiang, Y. Zhu, D. Feng, and L. Tian. Semantic-aware metadata organization paradigm in next-generation file systems. *IEEE Transactions on Parallel and Distributed Systems*, 23(2):337–344, 2012.
- [26] A. R. Huete. A soil-adjusted vegetation index (savi). *Remote Sensing of Environment*, 25:295–309, 1988.
- [27] U. Irmak and R. Kraft. A scalable machine-learning approach for semi-structured named entity recognition. In *Proc. of the 19th Inter. Conf. on World Wide Web*, pages 461–470. ACM, 2010.
- [28] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, 1988.
- [29] T. Joachims. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In *Machine Learning Inter. Workshop*, pages 143–151, 1997.
- [30] D. Jonscher and K. R. Dittrich. An approach for building secure database federations. In *VLDB '94: Proc. of the 20th Inter. Conf. on Very Large Data Bases*, pages 24–35. Morgan Kaufmann Publishers Inc., 1994.
- [31] Y. J. Kaufman and D. Tanre. Atmospherically resistant vegetation index (arvi) for eos-modis. In *Proc. IEEE Int. Geosci. and Remote Sensing Symp*, pages 261–270, 1992.
- [32] S. Kiritchenko, S. Matwin, R. Nock, and A. F. Famili. Learning and Evaluation in the Presence of Class Hierarchies : Application to Text Categorization. In *Proc. of the 19th Canadian Conf. on Artificial Intelligence*, 2006.
- [33] T. Kohonen, S. Kaski, K. Lagus, J. Salojarvi, J. Honkela, V. Paatero, and A. Saarela. Self organization of a massive document collection. *Neural Networks, IEEE Transactions on*, 11(3):574–585, 2000.
- [34] D. Koller and M. Sahami. Hierarchically classifying documents using very few words. In *ICML '97: Proc. of the 14th Inter. Conf. on Machine Learning*, pages 170–178. Morgan Kaufmann, 1997.
- [35] C. Köörner, D. Benz, A. Hotho, and M. Strohmaier. Stop thinking, start tagging: tag semantics emerge from collaborative verbosity. In *Proc. of the 19th Inter. Conf. on World Wide Web*, pages 521–530. ACM, 2010.

- [36] V. Kriakov, G. Kollios, and A. Delis. Self-tuning management of update-intensive multi-dimensional data in clusters of workstations. *The VLDB Journal*, 18(3):739–764, 2009.
- [37] A. H. F. Laender, B. A. Ribeiro-Neto, A. S. da Silva, and J. S. Teixeira. A brief survey of web data extraction tools. *ACM Sigmod Record*, 31(2):84–93, 2002.
- [38] J. Liu, S. Yu, and J. Le. Dynamic mining hierarchical topic from web news stream data using divisive-agglomerative clustering method. In *PAKDD'05: Proc. of the 9th Pacific-Asia Conf. on Advances in Knowledge Discovery and Data Mining*, pages 826–831. Springer-Verlag, 2005.
- [39] N. Longworth and W. K. Davies. *Lifelong Learning*. Kogan Page Ltd, 1996.
- [40] C. G. N. Macário and C. B. Medeiros. Specification of a framework for semantic annotation of geospatial data on the web. *SIGSPATIAL Special*, 1(1):27–32, 2009.
- [41] C. G. N. Macário, C. B. Medeiros, and R. D. A. Senra. O projeto webmaps: desafios e resultados. In *Geoinfo '07: IX Brazilian Symposium on GeoInformatics*, pages 239–250. INPE, 2007.
- [42] S. E. Madnick and Y. R. Wang. Evolution towards strategic applications of databases through composite information systems. *Journal of Management Information Systems*, 5(2):5–22, 1988.
- [43] A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. In *AAAI '98: Workshop on Learning for Text Categorization*, volume 752, pages 41–48, 1998.
- [44] C. B. Medeiros. Grand research challenges in computer science in brazil. *IEEE Computer*, 41(6):59–65, 2008.
- [45] R. S. Michalski. Knowledge Acquisition Through Conceptual Clustering: A Theoretical Framework and an Algorithm for Partitioning Data into Conjunctive Concepts. *Journal of Policy Analysis and Information Systems*, 4(3):219–244, 1980.
- [46] G. A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *The Psychological Review*, 63(2):81–97, 1956.
- [47] N. Mishra and R. Motwani. Introduction: Special Issue on Theoretical Advances in Data Clustering. *Machine Learning*, 56(1-3):5–7, 2004.
- [48] F. T. Oliveira, L. Murta, C. Werner, and M. Mattoso. Using provenance to improve workflow design. In *IPAW '08: 2nd Inter. Provenance and Annotation Workshop*, pages 136–143, 2008.

- [49] G. Pant and P. Srinivasan. Learning to Crawl : Comparing Classification Schemes. *ACM Transactions on Information Systems*, 23(4):430–462, 2005.
- [50] G. Z. Pastorello, Jr, J. Daltio, and C. B. Medeiros. A mechanism for propagation of semantic annotations of multimedia content. *Journal of Multimedia*, 5(4):332–342, 2010.
- [51] G. Z. Pastorello, Jr, R. D. A. Senra, and C. B. Medeiros. Bridging the gap between geospatial resource providers and model developers. In *GIS '08: Proc. of the 16th ACM SIGSPATIAL Inter. Conf. on Advances in Geographic Information Systems*, pages 1–4. ACM, 2008.
- [52] G. Z. Pastorello, Jr, R. D. A. Senra, and C. B. Medeiros. A standards-based framework to foster geospatial data and process interoperability. *Journal of the Brazilian Computer Society*, 15(1):13–26, 2009.
- [53] G. Z. Pastorello Jr, R. D. A. Senra, and C. B. Medeiros. Bridging the gap between geospatial resource providers and model developers. In *GIS '08: Proc. of the 16th ACM SIGSPATIAL Inter. Conf. on Advances in Geographic Information Systems*, pages 1–4. ACM, 2008.
- [54] B. Pinty and M. M. Verstraete. GEMI: a non-linear index to monitor global vegetation from satellites. *Plant Ecology*, 101(1):15–20, 1992.
- [55] N. Popitsch and B. Schandl. Ad-hoc file sharing using linked data technologies. In *PSD '10: Proc. of the Inter. Workshop on Personal Semantic Data*, 2010.
- [56] X. Qi and B. D. Davison. Web page classification. *ACM Computing Surveys*, 41(2):1–31, 2009.
- [57] J. J. Qu, W. Gao, M. Kafatos, R. E. Murphy, and V. V. Salomonson. *Earth Science Satellite Remote Sensing*, volume 2, chapter The NASA HDF-EOS Web GIS Software Suite (NWGISS), pages 245–253. Springer, 2007.
- [58] R. Řehůřek and P. Sojka. Software Framework for Topic Modelling with Large Corpora. In *LREC '10: Proc. of the Workshop on New Challenges for NLP Frameworks*, pages 45–50. ELRA, 2010.
- [59] R. Rew and G. Davis. NetCDF: an interface for scientific data access. *Computer Graphics and Applications, IEEE*, 10(4):76–82, 1990.
- [60] A. J. Richardson and C. L. Wiegand. Distinguishing vegetation from soil background information (by gray mapping of Landsat MSS data). *Photogrammetric Engineering and Remote Sensing*, 43:1541–1552, 1977.

- [61] N. Ritter and M. Ruth. GeoTIFF Format Specification GeoTIFF Revision 1.0, 1995.
- [62] A. Santanchè. *The Fluid Web and Digital Content Components: from a document-centered to a content-centered view*. PhD thesis, Institute of Computing - Unicamp, 2006.
- [63] H. Schütze, D. A. Hull, and J. O. Pedersen. A comparison of classifiers and document representations for the routing problem. In *ACM SIGIR '95: Proc. of the 18th Annual Inter. Conf. on Research and Development in Information Retrieval*, pages 229–237. ACM, 1995.
- [64] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys (CSUR)*, pages 1–47, 2002.
- [65] R. D. A. Senra and C. B. Medeiros. Organographs - multi-faceted hierarchical categorization of web documents. In *WEBIST '11: Proc. of the 7th Inter. Conf. on Web Information Systems and Technologies*, pages 583–588, 2011.
- [66] N. Sharman, N. Alpdemir, J. Ferris, M. Greenwood, P. Li, and C. Wroe. The mygrid information model. In *UK e-Science programme All Hands Conference*, 2004.
- [67] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *Transactions on Pattern Analysis and Machine Intelligence*, 22:1349–1380, 2000.
- [68] P.H.A. Sneath and R.R. Sokal. *Numerical Taxonomy. The Principles and Practice of Numerical Classification*. Freeman, 1973.
- [69] Peter Spyns, Robert Meersman, and Mustafa Jarrar. Data modelling versus ontology engineering. *SIGMOD Rec.*, 31(4):12–17, December 2002.
- [70] S. Sufi and B. Mathews. Clrc scientific metadata model: Version 2. *Final report. Council for the Central Laboratory of the Research Councils. Report No: DL-TR-2004-001*, 2004.
- [71] J. Turmo, A. Ageno, and N. Català. Adaptive information extraction. *ACM Computing Surveys (CSUR)*, 38(2):4, 2006.
- [72] F.T. Ulaby. Radar response to vegetation. *IEEE Transactions on Antennas and Propagation*, AP-23(1):36–45, 1975.
- [73] A.S. Weigend, E.D. Wiener, and J.O. Pedersen. Exploiting hierarchy in text categorization. *Information Retrieval*, 1(3):193–216, 1999.
- [74] E. Wilde and R. J. Glushko. Xml fever. *Commun. ACM*, 51(7):40–46, 2008.

- [75] J. Xu, C. Dichev, and A. Esterline. On the Effectiveness of Collaborative Tagging Systems for Describing Resources. In *WRI '09: Proc. of the World Congress on Computer Science and Information Engineering*, volume 4, pages 467–471. IEEE Computer Society, 2009.
- [76] Y. Yang and X. Liu. A re-examination of text categorization methods. In *ACM SIGIR '99: Proc. of the 22nd Annual Inter. Conf. on Research and Development in Information Retrieval*, pages 42–49, 1999.
- [77] L. Youseff, M. Butrico, and D. Da Silva. Toward a unified ontology of cloud computing. In *GCE '08: Proc. of the Grid Computing Environments Workshop*, pages 1–10. IEEE, 2008.