Gerenciamento do Ciclo de Vida de Dados de Sensores: da Produção ao Consumo

Este exemplar corresponde à redação final da Tese devidamente corrigida e defendida por Gilberto Zonta Pastorello Jr e aprovada pela Banca Examinadora.

Campinas, 19 de dezembro de 2008.

Claudia M. Bauzer Medeiros (Orientadora)

Tese apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Doutor em Ciência da Computação.

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP

Bibliotecária: Miriam Cristina Alves – CRB8a / 5094

Pastorello Júnior, Gilberto Zonta

P776g Gerenciamento do ciclo de vida de dados de sensores: da produção ao consumo/Gilberto Zonta Pastorello Júnior -- Campinas, [S.P. :s.n.], 2008.

Orientadora : Claudia Maria Bauzer Medeiros

Tese (doutorado) - Universidade Estadual de Campinas, Instituto de Computação.

1. Metadados. 2. Serviços Web. 3. Software Middleware. 4. Aquisição de dados. I. Medeiros, Claudia Maria Bauzer. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Título em inglês: Managing the lifecycle of sensor data: from production to consumption.

Palavras-chave em inglês (Keywords): 1. Metadata. 2. Web services. 3. Software middleware. 4. Data acquisition.

Área de concentração: Bancos de dados

Titulação: Doutor em Ciência da Computação

Banca examinadora: Profa. Dra. Claudia Maria Bauzer Medeiros (IC-UNICAMP)
Prof. Dr. Altigran Soares da Silva (DCC-UFAM)
Prof. Dr. Antonio Alfredo Ferreira Loureiro (DCC-UFMG)
Prof. Dr. Edmundo Roberto Mauro Madeira (IC-UNICAMP)
Prof. Dr. Sandro Rigo (IC-UNICAMP)

Data da defesa: 19/12/2008

Programa de Pós-Graduação: Doutorado em Ciência da Computação

Folha com assinaturas da banca

Instituto de Computação
Universidade Estadual de Campinas

Gerenciamento do Ciclo de Vida de Dados de Sensores: da Produção ao Consumo

Gilberto Zonta Pastorello Jr[1]

Dezembro de 2008

[1]Trabalho desenvolvido com bolsa de doutorado FAPESP (processo número 2004/14052-3) 2005–2008.

À Chucrie Azure (*in memoriam*), de quem o
exemplo de dedicação, amor e carinho
nunca será esquecido.

Agradecimentos

À Profa. Claudia, minha orientadora de mestrado e doutorado, por orientar, educar, ensinar, aconselhar, encorajar, e principalmente por confiar em mim e em meu trabalho durante todo esse tempo. Minha eterna gratidão.

À minha família, que sempre soube me amar e me aceitar. Em especial: minha avó, Chucrie (*in memoriam*), quem sempre me ofereceu apoio incondicional, incentivou em todos os momentos da minha vida, e cujo exemplo de dedicação nunca será esquecido; minha tia Nazira, pela ajuda, pragmatismo e conselhos que me empurraram para frente incontáveis vezes; minha mãe, Taufica, que sempre ficou do meu lado, mesmo que eu estivesse errado; meu pai, Gilberto, por me incentivar a alcançar mais; minha irmã, Adriana, quem me ensinou a deixar de lado o que não é essencial; minha irmã caçula, Claudia, que, sempre teimosa, mostrou que teimosia não é sempre ruim; e, à Kelly, por toda paciência e compreensão em muitos momentos difíceis.

Aos meus amigos, sem quem nada disso valeria a pena. André, grande amigo e pesquisador que influenciou muito este trabalho. Celso, sua perspectiva inspiradora com relação à ciência e disposição para fazer as coisas acontecerem. Senra, sempre animado, sempre com um às na manga, e sempre pronto a ajudar as pessoas. Esse trio é difícil de bater! Jaudete, por ajudar a manter meu pensamento honesto no final deste trabalho. A essas pessoas agradeço ainda pela parceria e participação direta no desenvolvimento deste trabalho. Agradeço ainda aos amigos do LIS de muitas épocas: Alan, Arnaldo, Carla, Daniel, Evandro, Fábio, Leonardo, Lin, Maurício, Nielsen, e tantos mais, e principalmente ao Prof. Ricardo pelo exemplo e orientação; aos amigos da cc99: Alexandre, Rossi, Sandra, Sheila, Tomita, e muitos outros; e todos que fizeram desses últimos dez anos de Unicamp algo incomparável.

À Profa. Ítala, do CLECH–UNICAMP, minha orientadora de iniciação científica durante a graduação, que me guiou de maneira exemplar nos primeiros passos da pesquisa científica, influenciando de forma definitiva minha escolha pela carreira em pesquisa.

Ao Prof. Brenelli, da FCM–UNICAMP, por toda ajuda em aspectos intangíveis mas essenciais para o desenvolvimento deste trabalho.

Ao Prof. Altigran, do DCC-UFAM, por ter me recebido em Manaus e ajudado a conhecer vários pesquisadores de diversas áreas na cidade, expandindo a visão sobre tópicos de pesquisa que foram usados neste trabalho e, mais importante, que servirão como base para trabalhos futuros.

Aos professores do IC, que sempre instigaram a aprender e fazer funcionar o que foi aprendido. Aos funcionários do IC, que mesmo tendo que fazer malabarismo com tantas limitações conseguiram criar um ambiente produtivo e confortável no instituto. E a todos que de alguma forma fazem parte da Unicamp, um lugar que sempre me inspirou e estimulou, mesmo em uma simples caminhada pelo campus.

Em especial, agradeço aos membros das bancas examinadoras, tanto de qualificação quanto de defesa de tese, Professores Altigran, Ariadne, Beatriz, Edmundo, Islene, Loureiro, Mauro, Sandro, Wainer, por todos os questionamentos, sugestões e discussões que me ajudaram a crescer como pesquisador.

Finalmente, este trabalho de pesquisa seria muito mais difícil, ou mesmo impossível, sem o apoio das agências de fomento à ciência e tecnologia. Especificamente, este trabalho foi financiado por uma Bolsa de Doutorado da Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), através do processo de número 2004/14052-3 nos anos de 2005 a 2008. Outras fontes de suporte incluem os Projetos WebMAPS financiados pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e o Projeto eFarms financiado pelo Instituto FAPESP–Microsoft Research.

Resumo

Dispositivos sensores estão se tornando bastante disseminados e vêm sendo aplicados em diversos domínios, principalmente em pesquisa científica. Entretanto, o aumento no número e variedade desses sensores introduz problemas de gerenciamento dos dados gerados, tais como a disponibilização de dados de sensores em diferentes taxas ou resoluções temporais e espaciais. Este trabalho trata de quatro aspectos do gerenciamento de dados de sensores para aplicações científicas: (i) prover acesso homogêneo a dispositivos sensores heterogêneos e aos dados produzidos por eles; (ii) gerenciar a composição de operações aplicadas a dados de sensores; (iii) oferecer funcionalidades de pré-processamento de dados que sejam flexíveis e possam ser executadas antes da publicação dos dados; e, (iv) propagar e criar anotações válidas (metadados) associadas aos dados durante todo seu ciclo de vida. A solução proposta para o aspecto (i) envolve o encapsulamento uniforme de dados e software, através da extensão de uma tecnologia de componentes chamada *Componentes de Conteúdo Digital* (DCCs), oferecendo também a associação de anotações a esse conteúdo. Tendo esses componentes como base, a solução proposta para (ii) é baseada no uso de workflows científicos para coordenar a combinação de DCCs de dados e software. A solução proposta para (iii) considera a invocação de workflows armazenados no provedor de dados e a submissão de novas especificações de workflows para armazenamento e/ou execução. Além disso, a solução usa as anotações dos DCCs para enriquecer consultas e suas respostas. Finalmente, um mecanismo de propagação de anotações é proposto como solução para (iv). As contribuições desta tese são apresentadas em um *framework* para gerenciamento de dados de sensores, considerando aspectos de acesso, pré-processamento, publicação e anotações de dados.

Abstract

Sensing devices are becoming widely disseminated, being applied in several domains, noticeably in scientific research. However, the increase in their number and variety introduces problems on managing the produced data, such as how to provide sensor data at distinct rates or temporal resolutions for different applications, or how to pre-process or format the data differently for each request. This work is concerned with tackling four issues that arise in the management of sensor data for scientific applications: (i) providing homogeneous access to heterogeneous sensing devices and their data; (ii) managing the composition of operations applied to sensor data; (iii) offering flexible data pre-processing facilities prior to sensor data publication; and, (iv) propagating and creating valid data annotations (metadata) throughout the data life cycle. The proposed solution to issue (i) is to uniformly encapsulate both software and data by extending a component technology called *Digital Content Components* (DCCs), also allowing associated annotations. Using these components as a basis, the proposed solution to (ii) is to apply scientific workflows to coordinate the combination of data and software DCCs. The solution proposed to (iii) involves invoking and posting workflow specifications from the data provider as well as using the annotations on DCCs to enrich the queries and answers. Finally, an annotation propagation mechanism is proposed as a solution to (iv). Our contributions are presented within a framework for sensor data management, which unifies aspects of data access, pre-processing, publication and annotation.

Sumário

Lista de Figuras

Capítulo 1

Introdução

Dispositivos sensores vêm sendo utilizados em diversas áreas há décadas. Entretanto, nos últimos anos três aspectos mudaram radicalmente o cenário do uso de sensores. Primeiro, o número e a variedade desses sensores cresceu enormemente. Segundo, os tipos e domínios de aplicação que usam dados de sensores também têm crescido bastante. Por último, as tecnologias atuais para aquisição, armazenamento, manipulação e publicação desses dados são insuficientes para disponibilizar esses dados adequadamente.

As soluções clássicas para a construção de aplicações que usam dados de sensores envolvem controle total sobre o ciclo de vida desses dados, desde o sensor (produtor) até a aplicação (consumidor). Isso apresenta diversos desafios, já que o sensor pode estar geograficamente muito distante da aplicação. Além disso, aplicações diferentes podem querer usar dados de um sensor e/ou muitos sensores diferentes podem vir a prover dados para única uma aplicação. Com isso, há uma necessidade de uma solução mais flexível para conectar os dois extremos: produtor e consumidor, ou seja, sensores e aplicações. Além disso, o uso de sensores na Web abre um novo leque de problemas a serem resolvidos.

Os objetivos desta tese envolvem, portanto, prover soluções para problemas em gerenciamento de dados de sensores, considerando os seguintes aspectos:

- interoperabilidade na aquisição e publicação de dados;
- gerenciamento das operações de transformação aplicadas a dados de sensores, incluindo as diversas combinações dessas operações;
- publicação, incluindo acesso a dados e metadados; e,
- atualização de metadados simultaneamente a transformações sofridas pelos conjuntos de dados associados.

Para alcançar esses objetivos, nossa solução é baseada nas seguintes soluções. Primeiro, usar e estender um tipo especial de componente, *Componente de Conteúdo Digital*

(DCC), para encapsular conteúdo digital (dados, o acesso aos dados e conteúdo executável). Segundo, usando as interfaces uniformes desses DCCs, especificar e executar workflows científicos para melhorar o gerenciamento das operações de transformação de dados. Terceiro, prover acesso aos dados de sensores (quer armazenados, quer em tempo real), usando dois tipos de mecanismos de acesso: baseado em padrões e baseado em ontologias. Por último, a tese propõe um mecanismo para propagar as anotações associadas a conjuntos de dados ao mesmo tempo em que esses conjuntos passam por transformações.

As contribuições desta tese são as seguintes:

1. A caracterização do cenário atual para gerenciamento de dados de sensores, apresentada sob forma de camadas de processamento e representação de dados. Essa caracterização auxilia o entendimento do escopo de problemas e soluções na área;

2. Provimento de acesso uniforme a dados, fontes de dados e funções de processamento de dados de sensores através da adaptação de uma tecnologia de componentes (DCCs). Alem disso, essa tecnologia de componentes provê mecanismos para dados de sensores auto-descritivos. Este tipo de descrição é relevante para a publicação desses dados, pois permite à aplicação consumidora determinar se um conjunto de dados é adequado. Metadados, nesse caso, são tão importantes quanto os próprios dados.

3. A proposta da utilização de workflows científicos para o gerenciamento de operações de transformação de dados de sensores; e a proposta de categorias de workflows científicos para o gerenciamento de dados de sensores. Além disso, workflows científicos foram explorados também para estender a expressividade de consultas.

4. A análise de padrões Web aplicáveis ao gerenciamento de dados de sensores, tais como os propostos pelo OGC (*Open Geospatial Consortium*), e a comparação e tradução desses padrões em descrições baseadas em ontologias. Isso permite que aplicações usem descrições baseadas nos padrões ou baseadas em ontologias para fazer consultas, obtendo resultados com a mesma precisão.

5. A proposta de um mecanismo para a atualização progressiva de metadados ao mesmo tempo em que os dados que descrevem são modificados. Esse mecanismo gera automaticamente novos metadados, para serem associados aos dados de saída de uma dada operação de transformação.

O texto está dividido nos seguintes capítulos:

- **Capítulo 2.** Visão geral dos problemas e desafios de pesquisa associados ao trabalho;
- **Capítulo 3.** Descrição em camadas do cenário atual para gerenciamento de dados de sensores;
- **Capítulo 4.** Proposta de uma solução usando DCCs para prover acesso a dados de sensores e para encapsular operações de manipulação desses dados;
- **Capítulo 5.** Proposta da utilização de workflows científicos para gerenciar operações de manipulação de dados de sensores;
- **Capítulo 6.** Proposta da combinação de abordagens baseadas em padrões Web e ontologias para prover mecanismos de acesso a dados de sensores na Web e proposta de utilização de workflows científicos para pre-processar os dados antes de recebê-los;
- **Capítulo 7.** Proposta de um mecanismo de propagação de anotações para atualizar metadados ao mesmo tempo em que os dados sofrem alterações;
- **Capítulo 8.** Conclusões do trabalho e suas possíveis extensões.

Chapter 2

Problem Overview: The Sensor Data Life Cycle

2.1 Motivation

Sensing devices have been used in several areas for decades. However, in the last decade, three aspects have changed radically the scenario concerning this kind of device: (i) their number and variety has grown enormously; (ii) application types and domains that use sensor data are increasing; and, (iii) the possibility of making sensor data available, overwhelms the current technologies for acquisition, storage, manipulation and publication of these data.

The first aspect, caused mainly by the evolution of systems hardware, has enabled acquisition of previously unthinkable amounts of data on physical phenomena. Sensors are becoming ubiquitous in our daily lives and, what is more important to this work, in scientific research in general. Sensor systems can range from tiny powerless devices distributed as tags and used in large quantities to large and sophisticated equipment embarked on satellites. Figure 2.1, adapted from [45], depicts this spectrum of sensing devices.

While the devices at the left of the figure are usually sophisticated, large, expensive, use complex sensing and communication schemes and produce large quantities of data, at the right we have devices each of which is small, simple, cheap, with very limited capabilities and produces very little quantities of data. However, the amounts of data produced by the latter easily surpass the former because of their quantity. In the middle range lies the complex devices, which are intended to be used in large quantities but also have somewhat richer processing, storage and communication capabilities than those on the right.

The second aspect shows the impact of the evolution of these devices on the possible

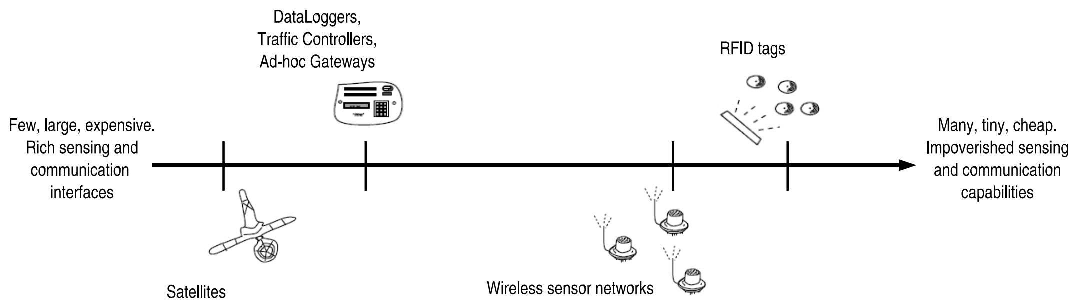


Figure 2.1: Sensing devices spectrum (from [45])

applications, spanning through different domains with very diverse sets of requirements and complexity. Applications for sensors can range from validation of experiments in astronomy and evaluation of global environmental conditions, to room confort control and body monitoring. There are nevertheless clear intersections on the data requirements of many such applications, with possibility of sharing the same sensor data sources. This factor is pushed even further by the third aspect, discussed next.

In this work, an application is the main consumer for sensor data. However, its actual implementation may be distributed, including code running within the sensors themselves. This aspect is detailed futher in Section 2.2.

The classic solution for building applications that use sensor data involves the total control of the data path from the sensor to the application. This is not only harder to achieve currently, because the sensor can be in a location far away from the application, but also undesirable: many different applications can profit from data of a sensor and also many different sensors can provide data to an application. Therefore, there is need for a flexible solution to connect these two extremes. Many problems arise when considering the Web as a means to bridging this gap, but also, many possibilities are opened. This thesis is focused on analysing these variables for composing a framework for sensor data management using the Web as a primary communication mechanism.

The current scenario for sensor data management and the impedance between how sensor data is acquired/stored and how the applications actually need them is further detailed in Chapter 3. Chapters 4, 5 and 6 consider, respectively, sensor data acquisition, manipulation and publication. Finally, Chapter 7 proposes a mechanism to automatically update metadata associated with sensor data along this management path.

2.2 Research Aspects

This thesis concerns many aspects of computer science research, within the scenario of sensor data management. Key issues considered were the following: data and process interoperability, data acquisition and publication, management of data transformation operations, component- and service-based technologies, process management and scientific workflows, web standards, and metadata evolution. These issues are briefly analysed in the following paragraphs, associating them to the relevant thesis chapter(s).

Sensors – data generation and processing. Sensors and applications have the two extreme roles in the sensor data lifecycle: producers and consumers, respectively. Although this distinction is useful for conceptually understanding the sensor data lifecycle, from the implementation point of view the disctiction between these extremes becomes blurred. A sensor, or any sensor data source, may offer several processing capabilities that might be not only useful but also essential to the development of an application. The processing capabilities of current sensing systems, especially sensor networks, leverage data collection and allow parts of an application to be implemented within the scope of the sensor system. In this case, the sensor extreme of the cycle is not only producing the data, but also helping the consumption; at the same time, the application is not running independently from the sensor data production, but in resonance with it.

These mixed implementation strategies enable the creation of sensor data based applications that were unattainable before. As an example, an application for monitoring a given space based on data collected by a large number of sensors could only be deployed close to these sensors because of limited communication bandwidth. If the sensors themselves can selectively send data, based on the data's relevance to the application, the user end of the application can easily be deployed anywhere.

There is a set of processing functions that are more commonly used within sensor systems such as data fusion or consistency checking. These functions allow better use of the sensors' capabilities, such as energy management in sensor networks, and increase the variety of applications that can use these data.

Another issue concerns the time frames within which the data is need by an application. We consider two base classes under this perspective: streamed and stored data. Streamed data are often needed as soon as it is produced and is mainly used in monitoring, event detection and action triggering applications. Sensor data that is stored and used as a time-series has modeling and forecasting applications as their main consumers.

Data interoperability and associated issues. We classify data interoperability regarding sensor data in two separate but related categories: (i) data acquisition interoperability, which is related to methods used to access and load data from sensors and also from other sensor data providers (e.g., databases or data services); (ii) data publication interoperability, which deals with how to make sensor data from a repository more readily available and accessible.

The relationship between (i) and (ii) becomes clear considering that a data set acquired by and stored at one site `S1`, which is responsible for its publication, may be the same data set that will be loaded by another site S2, which must consider acquisition methods. If S1 uses access and representation mechanisms that S2 also understands, the communication between them is straightforward; if not, adaptations must be made in one or both of them so that data exchange can succeed.

These two categories correspond to two main aspects in data interoperability: data representation and data access/exchange mechanisms. Data representation goes well beyond choosing a common binary or textual format (i.e., syntactic interoperability). It also involves determining if a given data set is what was actually expected at both ends (i.e., semantic interoperability). Associating descriptions to a data set helps interpreting data and, thus, allows the evaluation of the data set appropriateness. These descriptions are usually known as metadata. Making the interpretation of metadata a computer processable task is the aim of standards and ontologies, which will be discussed later on in this Section. Data access mechanisms specify how to request a given data set from a data provider. It is usual for data access mechanisms to be the target of some process interoperability solutions.

There are many challenges concerning data interoperability for sensor data, such as: (i) finding standard ways for dealing with streamed data; (ii) solving temporal and spatial resolution incompatibilities; (iii) determining standard representations for large data sets; (iv) temporally and spatially cropping interest areas from data sets; and many others. All chapters in this thesis attack interoperability issues. This work directly contributes towards (i), and offers interfaces to support solutions to (ii)-(iv).

Process interoperability and associated issues. Process interoperability has to do with how two different and independent systems can communicate to be able to jointly carry out a task. One example is to use two such systems in conjunction to solve a simulation problem. Once finished, the first system has to somehow know that it is supposed to pass its results using some mechanism to the second system. Determining how they can accomplish that characterizes many research efforts on process interoperability. Object-oriented, component-based and service-based technologies are commonly known approaches to tackle process interoperability. The last two were more thoroughly explored in this work, notably in Chapters 4 and 6.

Component-based development advocates the construction of software as independent modules (components) with well defined interfaces and a common communication mechanism. Each component is the result of the functional decomposition of a system. Two components should not share state and should only communicate using messages. Service-based development, on the other hand, is more focused on platform and service location independence. The main differences from component-based to service-based development is that the latter has a clear distributed nature and centers around a given process of services. Components could also provide these features, but are not required to. The most common form of a service-based implementation involves Web services, which are basically services that adhere to Web standards for their description, communication and discovery.

Challenges in the area of component-based development include how to achieve effective separation of concerns, how to specify interfaces that are general enough for creating product lines, performing large-scale testing on sets of interdependent components, and creating processor parallelism-aware components. In service-based development, the solutions and research are mainly being developed within the context of the Web. The main challenges are related to the creation of standards for solving specific problems with services such as transactions, security, composition, authentication, etc.

Standards, ontologies and metadata. Besides standards for Web services, there are other standardization efforts aiming at improving interoperability of both data and processes. Among these, we highlight: (i) XML and XMLSchema, which form the basis for data representation formats with shared schemas specified for selected domains and/or applications; (ii) extensions to Web service descriptions, also with specific schemas for selected domains/applications. We point out two scientific domains that have widely known initiatives: the geospatial domain, mainly within the *Open Geospatial Consortium* (OGC) [75], and biodiversity, mainly within the *DarwinCore initiative* [33]. Both specify XML-based data formatting and metadata structures and fields; OGC also specified Web service extensions to deal with geospatial operations. This improves the semantic interoperability for both data and processes. Another initiative concerns health systems, and standards such as [100] proposed by the World Health Organization.

Another step towards leveraging semantic interoperability is given by semantic oriented technologies such as ontologies and standards from the Semantic Web initiative. Ontologies are basically models of vocabularies shared by a given community in a specific domain. The Semantic Web initiative has proposed standards for representing ontologies such as RDF and OWL. RDF is a framework with a model that uses triples of the form <*subject, predicate, object*> to describe relationships (*predicate*) among terms (*subject* and *object*). It allows virtually any model to be represented in it and also has an XML representation. OWL is a set of vocabularies of basic or general terms for RDF. It has three representation levels, each allowing more flexible and more complex inferences to be made.

The potential for automation in processing metadata can be improved by using references to ontologies to compose metadata fields, structure and even some values. Every computer that manipulates data described via references to ontology terms can have access, through the Web, to the original ontologies, therefore being able to determine the adequacy of the data set at hand.

Concerning standards, the main challenge is how to create and disseminate their use in many domains with underdeveloped interoperability. Ontologies in Computer Science have led to many research efforts, mainly involving knowledge representation techniques, their application in data and systems modeling, the management of multiple ontologies (including how to integrate them), and inference development based on models represented in ontologies.

The thesis uses standards and ontologies as a means to support interoperability and enhance queries on sensor data. For instance, when using a Web service for accessing data sets, the semantics (or intended interpretation) of an output parameter of that service may be unclear for an application using that service for the first time. Using standards (such the ones from OGC [75]) or references to ontologies (such as NASA's SWEET [93]) to describe the service and its interfaces makes the semantics of these interfaces clearer.

Digital Content Components. In this work, we adopted and extended a component-based solution as the first step towards solving data and process interoperability issues. The technology, called *Digital Content Components* (DCCs) [98, 99], uses metadata with references to ontologies. Its main characteristics are: (i) being able to uniformly encapsulate both executable (programs, processes, etc.) and non-executable (data sets) content; (ii) having a context description for its content, using references to ontologies; (iii) having descriptions of interfaces to operations, also with references to ontologies; (iv) being independent of platform or programming environments. *Semantic Web services*, i.e., Web services described with references to terms of ontologies, could also be used instead of DCCs to attack data and process interoperability. Services have, however, two disadvantages: they require a Web environment to be executed while DCCs do not, and they do not allow direct encapsulation and packaging of data sets.

The other approach that could be used to achieve interoperability among data sources and improved handling of data heterogeneity is the use of Semantic Web services. However, two factors weight in favor of adopting DCCs: (i) the underlying communication mechanism for DCCs does not require a Web environment as is the case for Semantic Web services, lessening the impact on communication preformance; and, (ii) DCCs natively allow encapsulation of passive content, i.e., data, while this is not a feature for which Semantic Web services were conceived for. Furthermore, DCCs are constructed to be compatible with Semantic Web services, making these two approaches largely compatible. Indeed, the interface description for the DCCs is based on WSDL and OWL-S. The former is a widely used standard for describing Web services interfaces and latter is the most developed proposal for Semantic Web services.

Regarding sensor data representation, we used DCCs to encapsulate and describe data sets. Other components allowed access database management systems and data providing services. This also included the development of components to access and load data from sensor devices directly. With a common interface, these components allowed very diverse data sources to be accessed uniformly. These aspects are detailed in Chapter 4.

We used the same principle regarding data publication. A specialized DCC hierarchy was described to take care of data publication. These components were adapted to provide access to the data following Web standards, such as the *Web Feature Service* (WFS) and *Web Map Service* (WMS) from OGC. Actually, we specified mappings from Web standards to ontologies to guarantee that one such DCC could answer queries with the same results either way (i.e., using standards or ontologies). These results are covered by Chapter 6.

Scientific Workflows. The need for process composition leveraged the use of scientific workflows for dealing with the issue of process management. Sensor data usually must go through a lot of transformations before being finally consumed by an application. The number of operations and all their possible combinations makes it hard to manage the transformations. By using scientific workflows we not only allow a user to clearly visualize and modify the process as a whole, but also have control over documentation of the execution. Given the dynamic nature of sensors and observed phenomena, workflows themselves can change dynamically. Transformation operations and their coordination using scientific workflows is discussed in more detail in Chapter 5. In Chapter 6 there is also a discussion on how workflows can be applied to allow flexible combination of general transformation operations within a query.

Metadata evolution. The last research aspect has to do with metadata evolution. As the sensor data goes through a series of transformations, their associated metadata becomes less accurate or even totally wrong. At each step, new metadata should be associated to a transformed data set in order to maintain a correct description. This problem is known as *metadata evolution* or *annotation propagation*, considering metadata items to be annotations. This problem is usually dealt with manually and only at significant milestones in the data transformation processes. This is an error prone and tedious process, besides the potential loss of descriptions along the transformation process. In Chapter 7 we propose a mechanism for automating the propagation of annotations to data sets by means of what we call *semantic annotations*, in which annotations have references to ontologies.

2.3 Objectives and Contributions

The main goal of this thesis is to provide solutions to problems in sensor data management. To achieve that, we focused on advancing the following aspects:

- interoperability issues on data acquisition and publication;
- management of many possible transformation operations on sensor data, including different combinations of such operations;

- publication issues, including data and metadata accessibility; and,
- updating metadata along with data transformations.

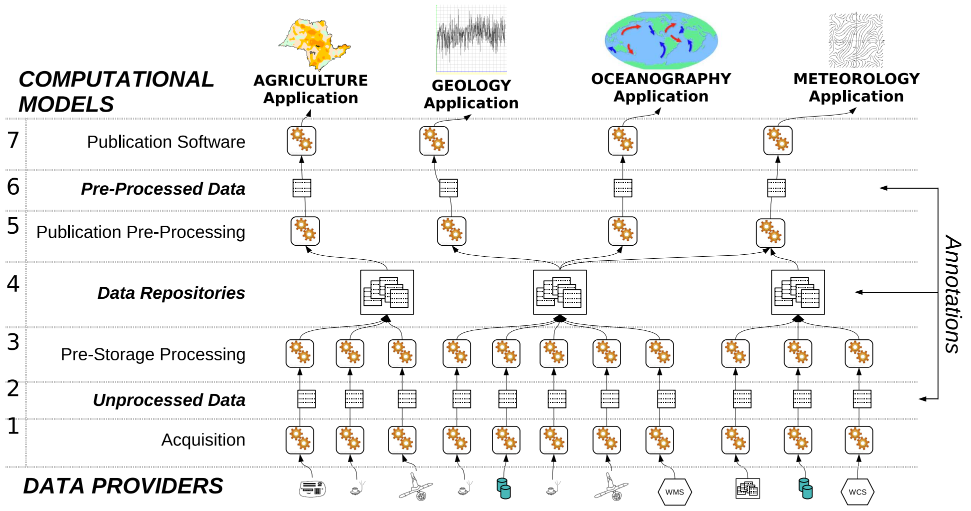


Figure 2.2: Sensor data usage scenario.

These aspects are better understood when considering all the phases of sensor data management. Such phases are depicted in Figure 2.2 (from Chapter 3). They provide the basic organization through which process interoperability problems at several levels can be treated. At the bottom and topmost levels lie the data providers and the applications, respectively, the producers and the consumers of this data management life cycle. Layer 1 (acquisition) has the acquisition software, which tries to interface with the several sensor data providers. Layer 2 (unprocessed data) has the data as they were acquired from the data sources. Layer 3 (pre-storage processing) transforms the data into a common format, possibly with some checking on the quality and suitability of the data sets. Layer 4 (data repositories) holds the storage and forwarding mechanisms for accessing the acquired data. Layer 5 (publication pre-processing) tailors the data to comply with the applications' requirements. Layer 6 (pre-processed data) has the data with format and content appropriate to the applications. Layer 7 (publication software) interfaces with the applications, providing the access mechanisms. The data descriptions, i.e., metadata or annotations, are present in all the data layers (2, 4, and 6), being used to interpret, index and search the data sets. Processing in layers 3 and 5 can be executed using workflows for specific purposes. Models running at consumer level also often rely on workflow execution. Not all data go through the entire cycle – e.g., repositories may be bypassed in stream sensor data processing.

Towards achieving our goals, our solution relates to the research aspects previously presented in the following way. DCCs are used to encapsulate data, access to data sources and executable content. Using the DCCs' uniform interfaces, we can specify and execute scientific workflows. At the top, publication software provide two kinds of access mechanism: standards-based and ontologies-based. After each transformation operation is applied to a data set, our annotation propagation mechanism generates new annotations that are associated with the output data set.

The contributions of this work are the following:

1. The characterization of the current sensor data management scenario, divided into data processing and data representation stages. This helps understanding the scope of both problems and solutions in this area. Previous viewpoints placed software for sensor data management as an end-to-end process, with the need for interfaces only for acquisition and publication [2, 105, 109]. Our scenario highlights the advantages of keeping intermediate processing steps apart, with clear interfaces between these steps, identifying points of interoperability at each stage. Moreover, it emphasizes the importance of metadata throughout the whole process. This contribution is described mainly in Chapter 3;

2. The uniform encapsulation of data, data sources and transformation operations, obtained through adaptation of a component technology (DCCs) to work as a middleware for sensor data management. The need for such extension, which focuses on self-describing data, was identified after an analysis of storage approaches for sensor data. The focus on the data description is relevant for publication of these data: if consumer applications cannot determine if a data set is adequate, it cannot be used. Thus, metadata are as important as the described data set itself. Current solutions for these issues are focused on specific platforms and/or protocols [1, 9, 67, 101, 115], not fully considering data heterogeneity and process interoperability [43]. Our solution deals with these two problems using an extension of DCCs. Chapter 4 presents the details on this part;

3. The proposal of scientific workflows for managing the application of transformation operations to the sensor data, and proposal of specific categories of scientific workflows for sensor data management. Each of these categories takes care of a particular aspect of sensor data management. Current approaches in this field involve specialized middleware and the development of custom software [9, 43]. Our solution aims at a more general middleware, allowing customization through new scientific workflows. This contribution is discussed in Chapter 5. We also considered scientific workflows for expanding query expressiveness, allowing an application to tailor the data to its needs before transmission – these issues are presented in Chapter 6. This second aspect extends current solutions by offering flexible customization on the pre-processing of data with the use of general processing functions within workflows [66, 70] instead of pre-determined operations of specific languages or protocols. With this, our solution provides uniform methods for accessing both data and processes.

4. The analysis of Web standards applicable to sensor data management and their comparison and translation to ontology-based description. This allows applications to use either standards or ontology-based descriptions to select and acquire sensor data. Current solutions only allowed the use on one of these approaches at a time [79, 93], while ours allows using either in a consistent fashion. Chapter 6 shows these results;

5. The proposal of a mechanism for updating metadata throughout the data transformation processes. This mechanism, presented in Chapter 7, automatically generates new metadata to be associated to the output of a transformation operation. This problem was only partially identified in current literature [12, 13]. Our work presents a general definition for the problem, characterizing a few associated challenges. Our solution is also general, in the sense that any operation can be considered as susceptible to generate new metadata, not only pre-defined operations from a query language as previously done by others [12, 13].

2.4 Thesis Organization

Each chapter of this thesis corresponds to a paper that has been published or submitted to publication. All the papers were written in collaboration with colleagues and the supervisor of this work, Claudia Bauzer Medeiros.

Chapter 3, "*A standards-based framework to foster geospatial data and process interoperability*," corresponds to [89], submitted to the **Journal of the Brazilian Computer Society** (JBCS), and was developed in collaboration with Rodrigo Dias Arruda Senra. It extends a poster paper presented in ACM-GIS2008 [90], and discusses the scenario for geospatial data management with a specific focus upon sensor data. Such scenario is divided into seven management layers (see Figure 2.2) with clearcut interfaces between them and modularization of responsibilities, considering the full data path from data providers to computational models (or applications) that consume the data. This organization by processing stages allow more thorough analyses of particular issues of data management, by providing a clear separation between data representation and processing steps, often mingled in related work. The chapter also discusses how Web standards are useful in solving interoperability problems in all the layers, particularly at the outside layers. The framework depicted is demonstrated in a case study for the WebMAPS agriculture project [50], which used the proposals presented.

Chapter 4, "*Accessing and Processing Sensing Data*," corresponds to [88], published on the *Proceedings of the 11th IEEE International Conference on Computational Science and Engineering* (CSE2008), pp 353–360, and was written in collaboration with André Santanchè. It presents the basics on extending and applying DCCs for sensor data acquisition and processing. The extension shows a new hierarchy of DCCs for dealing with individual sensor devices, sensor networks, middleware for sensor devices, sensor data, and several data transformation operations. Each of these resources is classified and has its solution described in terms of the used DCCs. The chapter also discusses an implementation experiment using two different sensor platforms as a basis.

Chapter 5, "*Applying Scientific Workflows to Manage Sensor Data*," corresponds to [86], published on the *Proceedings of the 1st e-Science Workshop* (e-Science2007), pp. 09–18, which took place in conjunction with the *22nd Brazilian Symposium on Databases* (SBBD2007), and was written in collaboration with André Santanchè. It briefly describes the layers for sensor data management, with a focus on implementation aspects, contrasting with the layers in Chapter 3, which are functional in nature. The paper also analyses the use of DCCs to access sensors, operations and sensor data, focusing on how DCC uniform interfaces can be used to describe and execute scientific workflows. These kind of workflows are more suited for sensor data management since they have characteristics such as adaptability and focus on data flows, thus offering a flexible and extensible solution for sensor data acquisition and pre-processing. The chapter also presents a classification of scientific workflows in categories that have clear roles in different aspects of sensor data management, such as validation and publication of sensor data.

Chapter 6, "*Sensor Data Publication on the Web for Scientific Applications*," corresponds to [84], published on the *Proceedings of the 4th International Conference on Web Information Systems and Technologies* (WEBIST2008), pp. 137–142, and was developed in collaboration with Luiz Celso Gomes Jr and André Santanchè. It discusses in more detail sensor data publication issues, mainly considering the differences in representation and access of sensor data between approaches using Web standards and semantic descriptions. Both solutions have their strong and weak aspects, so that the better approach would be to use both in a consistent fashion. This is done in our solution by mapping ontology terms into parts of the Web standards, using mapping rules stored within the DCCs. This enables a DCC to answer queries using either approach with the same results using different formats and access methods. The chaper also discusses how to use scientific workflows to enrich the queries, resulting in the tailored pre-processing of sensor data before actually answering the query.

Chapter 7, "*Multimedia Semantic Annotation Propagation*," corresponds to [83], published on the *1st IEEE International Workshop on Data Semantics for Multimedia Systems and Applications* (DSMSA2008), which took place in conjunction with the *10th IEEE International Symposium on Multimedia* (ISM2008), and was written in collaboration with Jaudete Daltio. It describes the problem of metadata evolution, also known as annotation propagation, and presents a mechanism that is our solution to it. The chapter applies the solution to multimedia metadata, such as satellite images. It can also be applied to any kind of sensor data, or any data set for that matter, given that the metadata are defined and filled through references to ontologies. The mechanism combines the descriptions (metadata) from the input data set and the operation interface generating the descriptions to be associated to the output data set. The solution is fully extensible, allowing any kind of relationship among data sets to be considered for propagation.

Additional publications associated with work conducted within this research are:

- "*Bridging the gap between geospatial resource providers and model developers*" [90], published as a poster paper at the *16th ACM International Conference on Advances in Geographic Information Systems* (ACM-GIS2008), with an overview of the work on [89] (Chapter3), written in collaboration with Rodrigo Dias Arruda Senra;

- "*User-centered Multimedia Building Blocks*" [99], published on the **Multimedia Systems Journal**, 12(4):403-421, in 2007, and whose main author is André Santanchè, who proposed the DCCs in his doctoral thesis. The paper presents digital content authoring aspects and how DCCs were applied to bring author and user of the content together. It also discusses architectural aspects required for a functional DCC infrastructure;

- "*Providing Homogeneous Access for Sensor Data Management*" [87], a technical report at the Institute of Computing-UNICAMP, showing a preliminary overall view of our solution, written in collaboration with André Santanchè; and,

- "*An Annotation Propagation Mechanism for Multimedia Content*" [82], another technical report at the Institute of Computing-UNICAMP, showing details of our solution to the annotation propagation problem, written in collaboration with Jaudete Daltio.

Chapter 3

A standards-based framework to foster geospatial data and process interoperability

The quest for interoperability is one of the main driving forces behind international organizations such as OGC and W3C. In parallel, a trend in systems design and development is to break down GIS functionalities into modules that can be composed in an ad hoc manner. This component-driven approach increases flexibility and extensibility. For scientists whose research involves geospatial analysis, however, such initiatives mean more than interoperability and flexibility. These efforts are progressively shielding these users from having to deal with problems such as data representation formats, communication protocols or pre-processing algorithms. Once these scientists are allowed to abstract from lower level concerns, they can shift their focus to the design and implementation of the computational models they are interested in. This paper analyzes how interoperability and componentization efforts have this underestimated impact on the design and development perspective. This discussion is illustrated by the description of the design and implementation of WebMAPS, a geospatial information system to support agricultural planning and monitoring. By taking advantage of new results in the above areas, the experience with WebMAPS presents a road map to leverage system design and development by the seamless composition of distributed data sources and processing solutions.

3.1 Introduction

In geographic information science, interoperability is a key issue, given the wide diversity of available geospatial data and scientific data processing tools. There are many research initiatives to meet this challenge, from data interchange standards and service-oriented architectures (SOA) to user interface design. This paper concentrates on two kinds of interoperability aspects: processes and data [40, 69, 74].

We show that efforts towards these directions have a desirable side effect: they are progressively shielding end users from having to deal with low level data management issues. This helps bridging the semantic and operational gap between data providers and scientists whose main interest is to design and test computational models that use geospatioal data and processes, without having to concern themselves with low level implementation details. In this sense, process and data interoperability efforts are contributing to user interface interoperability. In this text, the term model refers to a computational model representing relevant aspects from natural or artificial phenomena/processes that are somehow spatially referenced – e.g. a hurricane (natural), or urban traffic (artificial).

Processes interoperability is related to how two (or more) heterogeneous systems can interact. To that end, the systems must have means of determining which operations can/ should be invoked from each other's interface to execute a task. Data interoperability concerns data representation formats and manipulation. To achieve data interoperability, data consumers must be able to interpret one data set according to the same set of concepts. Both points of view are intimately related, since processes consume and produce data.

We categorize approaches to deal with interoperability issues as: standards based, ontologies based and services based. Standards concern reaching an agreement on a domain and specifying interfaces, protocols and data representation formats. Ontologies are also related with a domain, but an *a priori* agreement is not always a requirement. Services present a generic way of encapsulating operations from a system, making them available in a uniform way. Ontologies are out of this paper's scope, being tackled elsewhere [31, 32, 82].

This paper discusses recent efforts in interoperability for geospatial processes and data that are based on standards and services. Our discussion is intertwined with the impacts of such interoperability solutions on the design and development of geospatial applications. Monolithic systems are giving place to component-based distributed architectures [107, 109]. While the former forced scientists to adapt a problem (and their solution to it) to be compatible with a system and its data formats, the latter fosters the adoption of systems and data [55] that will fit the requirements of a new problem. Sensor data applications such as environmental and urban planning [29] are pushing the need for these kinds of solution even harder. They have to rely not only on local, well known, data providers but often on distributed, discovered on-the-fly, heterogeneous systems. In order to leverage design and construction of geospatial applications, however, data must undergo complex sequence of transformations, from providers to consumers. To support the required transformations, the designers of geospatial systems are faced with a multitude of process and data interoperability solutions, which they must somehow choose and compose.

The paper presents two main contributions towards helping solve this problem, namely:

- a conceptual framework that structures those transformation steps into several layers, with clear cut interfaces and responsibilities. Each of these layers addresses one kind of interoperability problem (for instance, access mechanisms, data cleaning, data formatting). This separation helps systems designers to focus on one issue at a time, leading to modular and composable systems; and,
- a real case study of this framework showing its advantages on data and process interoperability. The framework is being adopted within WebMAPS, a multidisciplinary project involving research in computer science and agricultural and environmental sciences.

The rest of the paper is organized as follows. Section 3.2 proposes a framework for publication of geospatial data and processes, to support application development. Section 3.3 describes the WebMAPS project and how it is being built under the framework. Section 3.4 discusses approaches for interoperability, how they fit in the framework and in WebMAPS development. Section 3.5 presents related work. Section 3.6 concludes the paper and discusses ongoing work.

3.2 A framework for geospatial data management

The architecture of interoperable data management systems is often specified following a basic three-layer cycle: providers (data layer), transformers (service layer) and consumers (client layer). An example is the infrastructure provided by INSPIRE [49], an initiative for the creation of a spatial infrastructure for Europe, with a distributed network of databases, linked by common standards and protocols to ensure compatibility and interoperability of data and services. INSPIRE's architecture (client, services and data layers) includes four main groups of components: user applications, geo-processing and catalog services, catalogs and data repositories. The organization in these three layers is not unique to GIS – e.g., see [53] for data warehouse management. Though useful to understand the functionalities provided, this kind of organization is insufficient for designers of geospatial computer models to choose and compose process and data interoperability solutions.

In order to meet this challenge, we propose an extended framework which induces a methodology for geospatial data management. This framework, shown in Figure 3.1, describes a data management cycle for GIS applications – from data acquisition (at the bottom) to data publication (at the top), to be consumed by applications that embed models. This cycle can be repeatedly pipelined: the data publishers of one cycle can become the data providers of the next cycle. As will be seen, the first four layers can be compared to a *Extract-Transform-Load* (ETL) process [4, 41, 44, 103] in data warehouse environments. This organizational cycle provides the basic structure through which process interoperability problems at several levels can be dealt with.

Our full data management cycle has seven layers, which alternate between representing either data or processes. Layers 2, 4, and 6 represent data and boxes with gears (Layers 1, 3, 5, and 7) represent data manipulation operations. The flow is from bottom to top, with the operations being applied to the data on their way up. We point out that not all stages of the cycle are mandatory – e.g., a given intermediate stage may not be needed, or applications may retrieve raw data directly from providers. Furthermore, an entire cycle may be under the control of a single organization (e.g., our case study of Section 3.3), or distributed on the Web.

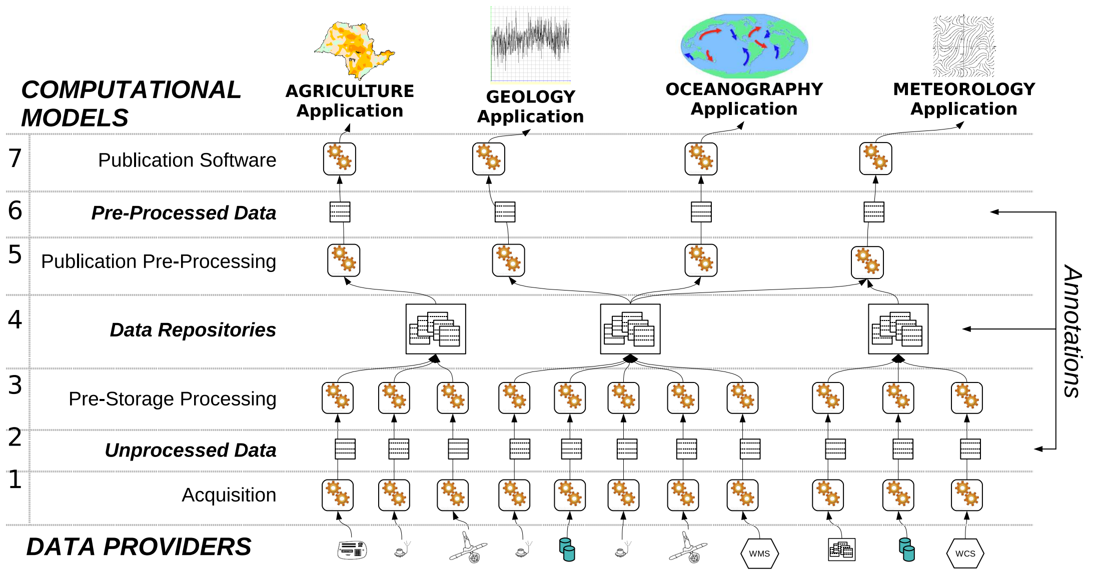


Figure 3.1: Geospatial data usage scenario.

The bottom layer houses *Data Providers* of many kinds, including sets of files, databases, sensors and data services. Sensors can range from ground-based networks to large satellite embarked multi-spectral electromagnetic sensors.

Sensor-produced data pose several new challenges to geographic applications. These data have a variety of characteristics that impact how they are stored, processed, published and accessed [45, 114]. Besides the spatio-temporal variability inherent to geospatial data, we single out the following issues particular to sensor data: (i) regularity: production of data in independent blocks or as continuous streams; (ii) transmission: manual readings, wired/wireless transmissions, error introduction, and others; (iii) position: impacts of the sensor relative position on the readings with respect to the observed phenomena; (iv) mobility: relation between sensor movement and its readings. Many other characteristics can be considered, according to the sensor capabilities and the application requirements. The broader the coverage of these aspects, the larger the number of consumers to which the data may be adequate.

Sensor data must be combined with data coming from data services. The latter deliver products provided by organizations that create or enrich a given data set and make it available. These data also have inherent characteristics that influence subsequent manipulation. Examples include issues such as which models were used to produce the data, which parameters were applied to calibrate the model, or how reliable were the data.

Next, we detail what each of the layers encompasses. Layers 1, 2 and 4 can be respectively compared to *extract*, *transform* and *load* phases of an ETL process.

Layer 1 (*Acquisition*) hosts data acquisition software, which works as a wrapper to data providers. Machine processable knowledge representation is an important issue in enabling software in this layer to access data with complex characterization from multiple data sources. Standards play an important role to deal with knowledge representation, and are further explored in Section 3.4, where we discuss how they "wrap" the data management cycles. This layer expresses the *extract* phase of an ETL process.

Layer 2 consists in *Unprocessed data*, obtained directly from data providers in a variety of formats. To be useful within a data management cycle, the data must be adapted to some representation using the characteristics and formats chosen for the cycle. This task is performed in Layer 3. Usually, unprocessed data is stored when there is a need for comparing pre-storage processing solutions, for maintaining historical raw data, or for auditing purposes.

Layer 3 (*Pre-Storage Processing*) represents the processing phase where data is transformed before its storage. Examples include signal processing functions for improving precision, data cleaning for detecting variations or errors, computing statistical parameters to detect acquisition problems, converting temporal and/or spatial resolutions, and testing data format conversions to determine accuracy. This layer corresponds to the *transform* phase of an ETL process.

Layer 4 (*Data Repositories*) corresponds to the storage facility, often a data repository of some kind, such as a database system. Two of the major issues to be dealt with in this layer are problems on what to store and how to fill in the gaps left by several types of acquisition errors. Selecting what is going to be stored is important since the amount of data acquired may be far too large to be stored in full [28, 37]. Given that geospatial systems must also cope with streamed data, this raises the additional issue of providing query mechanisms that can cope with both stored and streamed data [8]. For streamed data, the storage layer may have its role filled by a proxy service with some kind of caching mechanism. This is more natural for many kinds of applications (e.g., real time traffic monitoring). Queries may need to combine data from several data sources, even both streamed and stored data, possibly using different pre-storage processing operations. This can only be treated adequately in layers that have access to these resources, which is the case for Layer 4. The storage part of this layer is directly related to the *load* phase of an ETL process.

Layer 5 (*Publication Pre-Processing*) is responsible for transforming the data, filtering or augmenting it, in order to meet application requirements. Examples of such requirements include adjusting spatio-temporal resolution, access periodicity and specific presentation formats. Instances of operations to fulfill these requirements include composition operations (e.g., fusion of data from different data sources), scaling operations (e.g., changing temporal or spatial resolution), customization operations or more complex operation compositions. However, as most of these data are georeferenced, the more traditional GIS operations, e.g., see [94], are the most common in this phase. The execution of operations in Layer 5 are guided by application needs while operations executed in Layer 3 are oriented towards storage requirements. Thus, unless the operation was executed in Layer 3 and the result is already available in the repositories, a request from an application is executed in Layer 5.

Layer 6 (*Pre-Processed Data*) contains the pre-processed data sets, ready to be published and consumed by models. The main concern in this layer is data representation, e.g., data format, spatio-temporal resolution, and associated descriptions. An application request specifies the format, with translations applied as needed. Resolution adaptation may require interpolation algorithms for larger resolutions and summarization algorithms for smaller resolutions.

Layers 5 and 6 are not needed in non-shared data scenarios. Their appearance reveals an interesting issue: as models and algorithms become more stable and accepted within a community, they become basic and are taken for granted. This pushes the results of such models and algorithms down from the model layer to layer 5, with impact on interoperability and cooperative scientific work. An example of such migration is the georeferencing of satellite images: in the past, it was a necessary step to perform within geospatial data applications; presently it is available as a default attribute of most published images.

Layer 7 (*Publication Software*) represents the software that will make interfaces to operations and data access mechanisms available to applications. This is achieved by agreement between software providers and application developers. The need for such agreements restricts interoperability among new resources and systems. As an alternative to consensual specification, the software in this layer should provide descriptions that are sufficiently rich to allow applications to determine the suitability of software and data. Different approaches are used by the publication software depending on the requirements of the client applications, e.g., protocols with less overhead for large data sets, or richer protocols for initial stages of communication.

The publication software must also allow applications to select pre-processing operations, among the ones available, to be applied on the data before transmission. The operations are actually provided by lower layers, mainly by layer 5 (publication pre-processing) but a list of them and a selection mechanism must be present on the publication layer. Since the requirements from the applications vary, many different transformation operations may be required before the data can be used. Actually, applications can use either already pre-processed data sets (e.g. from Layer 6) or invoke operations to generate new data sets (e.g., from Layer 5). It is often undesirable or unfeasible to perform these operations within the application [114].

The upper layer (*Models*) is where the applications lie and where end users are able to interact with all the infrastructure on the layers below. Applications embed model execution, hence allowing scientists to visualize results, and to tune and interact with these models.

Annotation mechanisms are orthogonal to all layers, using metadata standards or free annotations. In other words, each data layer may be associated with annotation mechanisms that provide semantics to raw, stored or pre-processed data. Metadata have a predefined structure and expected ranges. This allows, for instance, indexing and retrieval based on the metadata, imposing, however, rigid limits. Annotations, on the other hand, have no structure and do not allow indexing, presenting a challenge for retrieval, often requiring content-based techniques [62]. Nevertheless, they allow very flexible descriptions. The proposal in [82] shows how to provide some structure to annotations without hampering flexibility by using references to ontologies. The Section 3.4 discusses how to improve metadata semantics with standards.

We point out that making these seven layers explicit is one of the keys to understand and solve the gap between resource providers and systems designers. Our decomposition of geospatial data flow and processing into identifiable layers with clear interfaces and responsibilities leverages application development. Combining solutions from previous layers enables a module on a higher layer (or even an application outside the scope of the layers) to deal with less interoperability issues. The most important aspect, however, is that our organization helps maintainability, reuse and extensibility, allowing developers to include new features at appropriate levels. This is achieved by solving one kind of interoperability issue in each layer and combining the solutions across the layers. To illustrate this, consider the problem of gathering data from two data sources, each using a different combination of access mechanism (e.g., an FTP server and a Web service) and data format (e.g., XML, CSV and binary). Once a module for each of the access mechanisms is available at the bottom layer, all modules for handling different data formats will be able to use one uniform interface to access the data from both providers.

This scheme is the same throughout the layers, with one layer adding a solution to one interoperability issue on top of the previously solved issues (by modules on the previous layers), and thus offering a uniform interface (or data format) to the next layer. Taking these stages into account helps solving several of the interoperability problems raised by the use of distributed geospatial data sources or by the invocation of external services. This will be illustrated next.

3.3 Putting the framework to use

This section discusses how the proposed framework reflects in the implementation efforts within the WebMAPS project. The project is is a multidisciplinary effort involving computer scientists and experts on agricultural and environmental sciences to develop a web-based platform for agro-environmental planning and modeling. One product from WebMAPS, vegetation index graphs, is described in Section 3.3.2 according to the layers presented in Section 3.2. Two other products are also detailed: one for automation of data acquisition (Paparazzi, see Section 3.3.3) and another for flexible data publication (see Section 3.3.4).

3.3.1 Project overview

WebMAPS is a project whose goal is to provide a platform based on Web Services to formulate, perform and evaluate policies and activities in agro-environmental planning.

The project caters to two kinds of users – farmers, and domain experts, such as agronomers or earth scientists. Farmers can enter data on their properties (e.g., production, parcels, crops). They are able to correlate data on their farms to geospatial content available on WebMAPS repositories – e.g., satellite image series or regional boundaries. Experts may want to investigate distinct kinds of data correlation and propose models to explain, monitor, or forecast crop behavior. See some of the tools at `http://www.lis.ic.unicamp.br/projects/webmaps`.

Similar to INSPIRE [49], WebMAPS can also be described using a 3-layer architecture, part of which already implemented. The Client Layer is responsible for user requests, forwarding them to be processed by the middle (Service) layer. The latter contains distinct kinds of modules, such as query processing, workflow specification, and ontology management. The Data Layer contains WebMAPS data repositories, including primary raw data (e.g., product classification from Brazilian official sources) and derived data (e.g., composite images). Geospatial data sets include satellite images, and coordinates of county boundaries. Additional data sources include information on properties, agricultural products and so on. Data are stored in the PostGreSQL database management system, with

PostGIS extension. At present, most of the services in WebMAPS are being implemented as software modules, for rapid prototyping and testing by end-users.

This kind of 3-tier architecture is useful for a high level description of the system's functionalities. However, as stressed in Section 3.2, it is not adequate from an interoperability perspective. The sections that follow discuss how we use our 7-layer framework of Section 3.2 to specify and develop some of the products offered by WebMAPS. In particular, we discuss three kinds of products: (i) the dynamic computation of NDVI graphs, starting from the acquisition of satellite images; (ii) a tool for automated image acquisition; and (iii) the interoperation with Google Maps. The first item is an example that spreads throughout most of the layers, while the last two focus on the bottom and top of the framework, respectively.

3.3.2 Illustrating the data management framework in WebMAPS

In this section, we describe one of the devised WebMAPS' products that is partially implemented and adheres to the layering described in section 3.2: computing historical NDVI profiles for a given region and period.

NDVI (*Normalized Difference Vegetation Index*) is a vegetation index. It is correlated to biomass conditions of vegetation and is widely used in distinct kinds of contexts – e.g. agriculture, biodiversity. An NDVI graph plots the average NDVI pixel value in a region through time from a temporal series of images. This can be used for crop monitoring and prediction [16, 92]. For example, in the sugar cane culture, a curve with higher values may indicate a product with better quality.

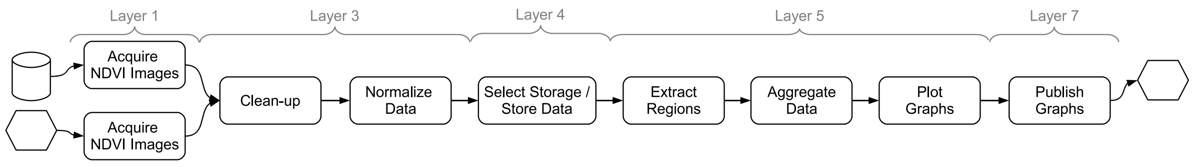


Figure 3.2: Computation and publication of NDVI series for a region.

One of the functionalities available from WebMAPS is the construction of such graphs. The user selects a region of interest R and a period T and the system computes the graph from a temporal series of satellite images, plotting the NDVI evolution for that region and period, as depicted in Figure 3.3.

NDVI graphs require two kinds of data – those acquired periodically (satellite images) and those that, once acquired, are only sporadically updated (e.g., county boundaries). This section describes the management cycle for these data within WebMAPS. We will not enter into details of acquisition periodicity nor procedures to refresh data, but such issues are embedded into constraints treated by our 7-layer framework. Figure 3.2 shows the main phases of the workflow that specifies the computation of the graph, following the layers of Figure 3.1. This workflow is shown at its more abstract level, but each step can encapsulate several processes. Moreover, though not shown in the Figure, it contains loops and cycles, which are not relevant for understanding our case study. Although this example does not have issues to be dealt with in layers 2 and 6, applications that, for instance, are heavily dependent on data representation formats, data codification or associated descriptions (metadata) would need specific solutions in these layers.

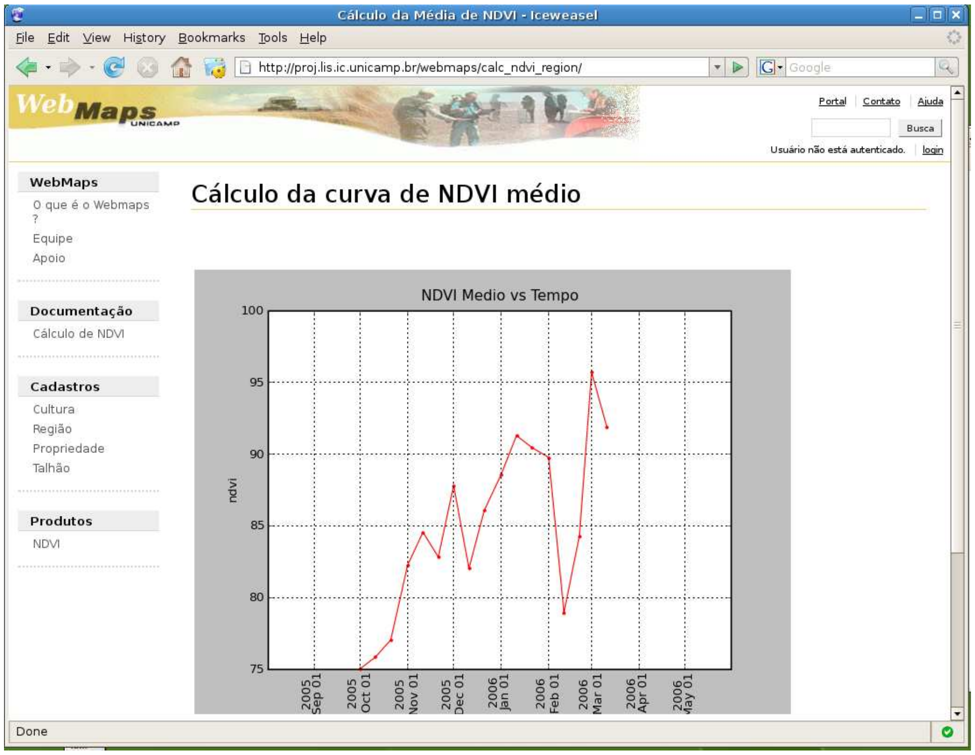


Figure 3.3: Example of an average NDVI curve for Campinas region (from WebMAPS).

Data Acquisition

There are many satellite imagery providers. For NDVI analysis, WebMAPS' agro-scientists have chosen to use pre- computed NDVI images provided by NASA from MODIS sensors [73, 108]. Here we faced typical problems of geospatial data acquisition. Each image depicts a geographical region much larger than the ones for which this first version of WebMAPS is being conceived (Brazil's southeast). Moreover, retrieving each image meant browsing the NASA web site to find the download link. Thus, assembling our image database became a laborious, tedious and time-consuming task. To improve on that, we have developed Paparazzi, a tool to automate the retrieval of remote data sets – see Section 3.3.3.

The second data type needed are vector-based coordinates, corresponding to the geographical regions of interest. WebMAPS offers two options: (i) ad-hoc manual definition of the region, described in *Well-Known Text Format* (a.k.a WKT) [25], a standard from the *Open Geospatial Consortium* (OGC); or, (ii) importing geospatial vector shapefiles. Brazilian county geometry shapefiles were imported from IBGE (Brazilian National Geographic Institute).

The last data type we need are textual descriptions of crops and their attributes. Here we applied screen scraping [27, 42] techniques to fetch produce code, popular name, scientific name and description from the Brazilian Ministry of Agriculture Web Portal. See Section 3.4 for details on these techniques.

Unprocessed Data

Satellite images retrieved from NASA using Paparazzi and shapefiles retrieved from the IBGE are encapsulated in temporary files, for subsequent quality checking. The rest of the data used goes directly to Layer 3 (Pre-Storage Processing). Here, we can already see the advantages of our multi-layer framework, which allows determining which stages should be followed for each kind of data.

Pre-Storage Processing

In possession of unprocessed data, we proceed to the pre-storage processing phase. There are three main concerns here: corruption detection, data normalization and assembly of the data sets. These concerns are not always present (e.g., if the data provided already have such issues solved). In particular, our NDVI graph construction example does not need to deal with the last issue, i.e., assembly of data sets.

Corruption detection is mandatory and is made explicit in our framework. First, data providers are never 100% reliable, and the acquired data may be already corrupted in its provider's domain. Second, data corruption can occur during the acquisition phase. Here, the encapsulated unprocessed files containing satellite data and geometries have their integrity automatically checked (e.g., by checksum algorithms). Corrupted or partially retrieved files are removed.

The third (textual) record type is more challenging, because information is less structured, the domain of values is open and not fully-known and we lack fail-proof tools to verify corruption. For our textual data resources (based on official government crop classification) we performed a manual check. Additional procedures are left to future work.

Data normalization is a recommended step to make data processing easier and more efficient. We automatically convert all files to a single and uniform representation format, and all measurement units to the same system. Thus, we have chosen to (a) store satellite imagery into GeoTIFF [95] files, converting into this format whenever needed, (b) convert shapefiles and textual geometries (WKT) into *Well-known Binary* representations (WKB) stored into PostGIS, and (c) represent all geographical coordinates to latitude/ longitude according to WGS84 reference ellipsoid. WKB [25] is also a standard from OGC.

Data set assembly is the last pre-storage processing step we need, and consists in putting together coherent spatio-temporal units – e.g., in our example, creating a composite NDVI image from a mosaic of acquired NDVI images. This includes issues which are sometimes called detecting the FoU (fitness of use) of a data set – see [58]. Here, the first (temporal) problem occurs when there are gaps in a time series – e.g., due to communication or device failure. The second (spatial) problem occurs when missing spatial data create "holes" in a data set (e.g., when an image mosaic has missing parts). Both problems can occur at the same time. Spatio-temporal gap problems are very common when using data from sensor networks – e.g., sensors may stop providing data for a period of time, causing problems in analyses.

There are three basic approaches to assembly problems: (i) acquire new data to fill the spatial and/or temporal gaps; (ii) apply interpolation, probabilistic or inference procedures to fill the gaps; and, (iii) mark the gaps, and forward the solution to some other layer (e.g., query processing will have to take the gaps into consideration). For satellite imagery, we have implemented the first approach, using data fetching retries and fallback data providers. For rainfall time series, we use the second approach. We have developed algorithms in which missing values are filled by combining spatial interpolation with historical data [63]. These algorithms are used by the Brazilian Ministry of Agriculture to maintain its rainfall series database for the whole country.

Data Repositories

Once the data are pre-processed, they are ready for storage. We use two types of storage: a relational database and the filesystem. Crop descriptions, geometries, textual properties, and data set descriptions are stored in PostgreSQL/PostGIS. Raster images in GeoTIFF format are stored in JFS (or XFS) filesystem partitions. We have chosen these partition types because they present good performance for large files [17]. So far, we have not used streamed sensor data. Our preliminary experiments with these data appear in [88]. Streamed sensor data for agricultural purposes are being handled within our eFarms project (`http://www.lis.ic.unicamp.br/projects/efarms`), which will act as a sensor data provider to WebMAPS.

Publication Pre-processing

This phase concerns transforming information, and ultimately preparing it for user consumption. In our example, this means (i) computing the average NDVI pixel value for the region R defined by the user; and, (ii) iterating (i) for the input time span T. These steps are performed automatically without user intervention.

The images of interest are already stored in the repositories. Steps (i) and (ii) consist in building a bitmap mask for R and applying this (cached) mask to all NDVI images within the desired time frame T, extracting the region (pixels) of interest and computing their average. The graph is constructed for these average values.

Data Publication

Data publication is the last phase in the processing workflow. In our case study, the NDVI graphs constructed in the previous phase are published as images, embedded or not in HTML pages. Other formats could also be adopted here: text files with pairs <georeferenced point, value>, tables with the values for regions, etc.

3.3.3 Automating Acquisition: Paparazzi

Paparazzi is a command-line tool we developed to automate the acquisition of satellite imagery by means of screen scraping techniques [47, 56, 59]. Paparazzi is a specialized web crawler, hand-crafted to fetch data from specific target web sites. Paparazzi is an example of a tool that was implemented within the scope of Layer 1. Paparazzi is worth using whenever the number of files to be retrieved is large, and hyperlinks to target files are not concentrated in a single page, but scattered across several pages, as is the case with NASA MODIS images. Consider the following Paparazzi command line:

```
paparazzi.py -b 2008-01-01 -e 2008-05-31 -s Brazil4 -p 250m -m 2 -r
```

It is a request for all images (`r`) from January 1, 2008 (`-b`) to May 31, 2008 (`-e`), for the geographical region named *Brazil4* (`-s`). Each image retrieved should have spatial resolution (`-p`) of 250 meters per pixel and represent NDVI measures (`-m`). *Brazil4* is a specific name used by NASA [73] to designate a given area in South America that covers SE Brasil.

This same task required downloading 152 images corresponding roughly to 7 Gb in size. If done manually, for each image, the user needs to visit three different web pages prior to starting a 50 Mb file download. In the first page the user selects the subset (Brazil4). In the next page the user selects data product (NDVI) and image resolution (250m). Finally, in the last page, the user selects the file format and starts the image download. Therefore, precious user time can be saved, if the researcher relinquishes control and responsibility of the iterative acquisition process to Paparazzi. Moreover, when adopted as a software library, Paparazzi acts in reaction to user queries over incomplete data sets, trying to fill-in gaps on demand.

3.3.4 Flexible publication

WebMAPS innovates allowing data produced in any of the framework phases to be directly accessible in many representations. Therefore, WebMAPS is not just a black-box automating GIS procedures. It is also a data gateway fostering scientific information sharing and allowing experimental results to be reproduced and validated. This is facilitated by isolating the responsibilities of each framework layer.

In particular, the three data types handled (images, geometries and text) are published by means of standard protocols and representations, explained in Section 3.4. Satellite imagery data is accessible through an OpenDAP interface; textual data and metadata are available as HTML pages annotated with microformats; and geometries are available as KML views.

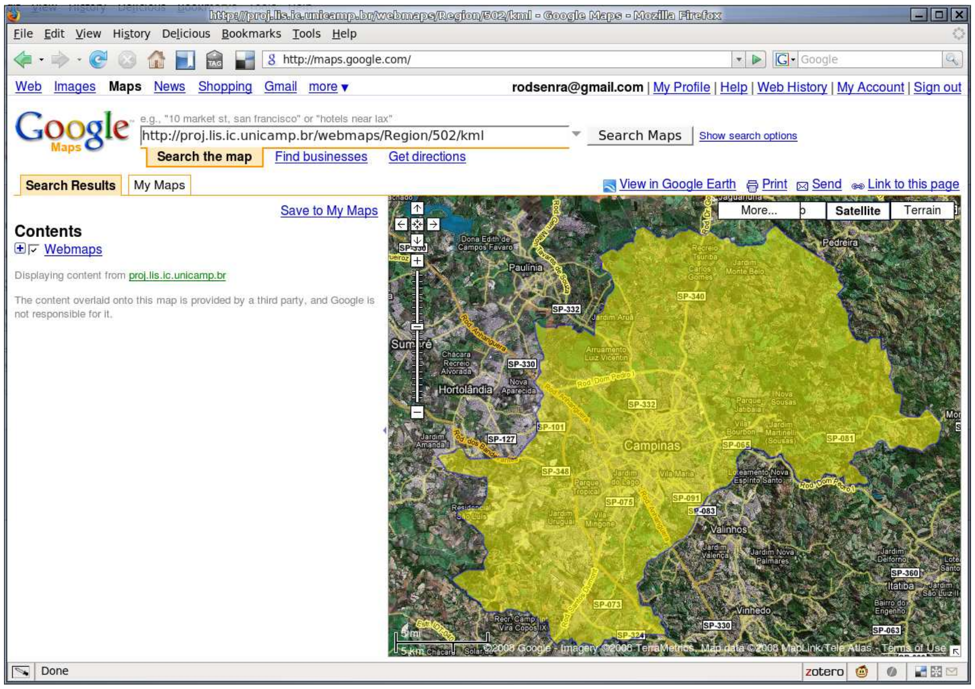
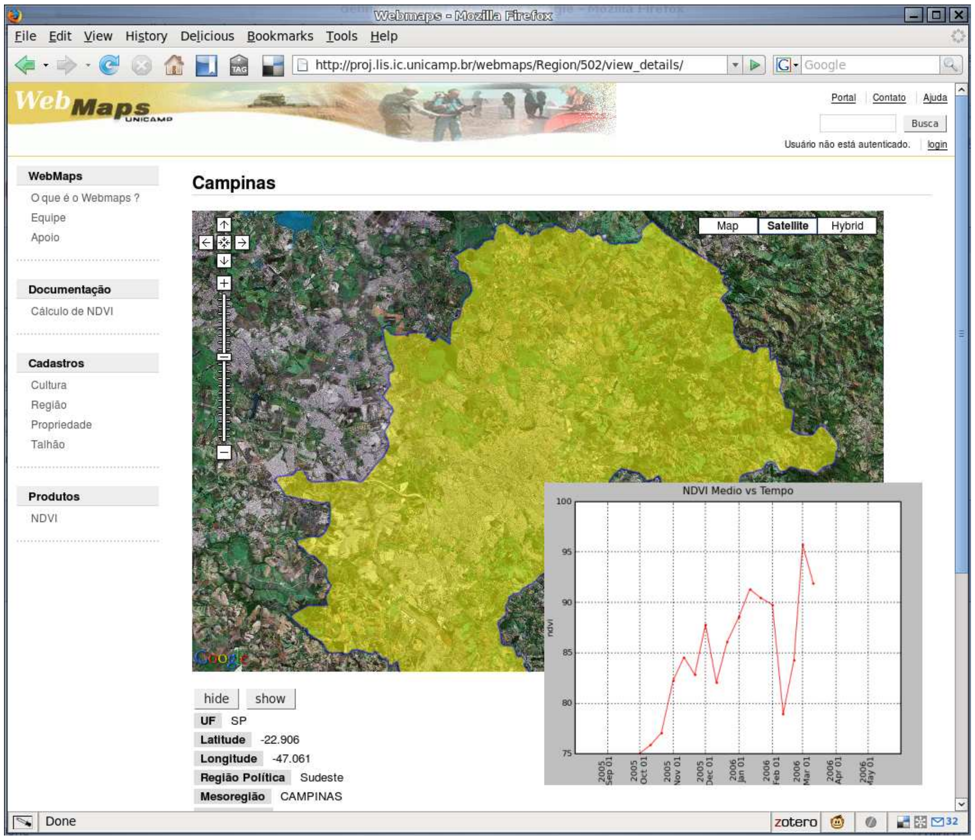

http://proj.lis.ic.unicamp.br/webmaps/Region/502/kml - Google Maps - Mozilla Firefox
File Edit View History Delicious Bookmarks Tools Help
http://maps.google.com/
Web Images Maps News Shopping Gmail more
rodsenra@gmail.com | My Profile | Help | Web History | My Account | Sign out
http://proj.lis.ic.unicamp.br/webmaps/Region/502/kml
Search Maps
Show search options
Search the map
Find businesses
Get directions
Search Results
My Maps
View in Google Earth
Print
Send
Link to this page
Save to My Maps
Contents
Webmaps
Displaying content from proj.lis.ic.unicamp.br
The content overlaid onto this map is provided by a third party, and Google is not responsible for it.
More...
Satellite
Terrain
Paulínia
Sumaré
Hortolândia
Campinas
Valinhos
Vinhedo
Done

Figure 3.4: Google accessing data from WebMAPS in KML format.

Thanks to this, WebMAPS can act as a data provider and a data client. We exemplify this by showing its interaction with Google Maps. Figure 3.4 depicts Google Maps obtaining a geometry resource served in KML format from WebMAPS. Here, WebMAPS acts as a data provider and mediator, re-distributing geometries acquired from an authoritative source (Brazilian Geographic Institute), and transformed into a suitable format to feed Google Maps. In Figure 3.5, we depict WebMAPS acting as a client of Google's map rendering service. The map rendered by Google Maps (Figure 3.4) is mashed-up with results from a user query, composing the web page shown in Figure 3.5. The query results comprehend textual metadata and a NDVI graph for the given region and time frame. For visualization sake, the chart is an overlay, not representing the original page layout.



Campinas

UF	SP
Latitude	-22.906
Longitude	-47.061
Região Política	Sudeste
Mesoregião	CAMPINAS

Figure 3.5: WebMAPS embedding map generated by Google Maps.

This interaction pattern between WebMAPS and Google Maps is a combination of resource-oriented (from WebMAPS) and service-oriented (from Google Maps) paradigms. We further discuss these approaches in Section 3.4. In this particular example, the use of KML and WKT enabled us to rapidly build a prototype for cartographic visualization, including satellite image overlays provided by Google Maps. End users are rapidly able to visually assess the quality of the data, and test the outcomes of different analyses. Hence, standards offer much more than interoperability. Their use has sped up the validation of user requirements in terms of interaction needs. More importantly, it has leveraged application development, so that users can start testing their ideas much sooner, while we work on other system issues. This does not mean that WebMAPS will necessarily always rely on Google Maps for cartographic rendering and interaction - we are also experimenting with other kinds of Web service-based solutions (see [32] for our use of GeoServer to publish GML data for biodiversity systems).

3.4 Interoperability Approaches

This section characterizes the standards and services approaches to interoperability and show how they are contemplated within our framework and in WebMAPS. We point out that ontologies are another very important approach to interoperability. Though they are part of WebMAPS future annotation facilities, they are outside the scope of this paper. In our framework, they intervene at all transversal annotation stages of Figure 3.2.

3.4.1 Selected Standards

Standards represent an agreement among research groups and are the main focus of many institutions such as OGC or W3C. From the data interoperability perspective, standards deal with representation and formatting issues. OGC's *Geographic Markup Language* GML [76] is an example of such a standard. It is an XML-based specification for geospatial data interchange. Process interoperability specifications can base their input/output formats in GML.

From the process interoperability perspective, standards are used in the specification of protocols, interfaces and descriptions of processes. Examples include OpenDap and OGC standards. OGC's main general-use standards for geospatial process interoperability are the *Web Feature Service* (WFS), the *Web Coverage Service* (WCS) and the *Web Map Service* (WMS) [76]. These standards specify the access mechanisms to, respectively, vector data, raster data and renderized maps. Vector data describes geographic features using their geometry (points, lines, polygons). Raster data represents geographic areas as arrays of cells (e.g., images). The access mechanisms are to be implemented as Web services.

The recent *Web Processing Service* (WPS) [78] specification concerns the publication of geospatial processes, a main concern in this paper. A process may be an algorithm, a calculation or a model that manipulates geospatial data. Although WPS does not describe the specific behavior of an operation, it provides general description mechanisms, such as *Profiles* and *ProcessDescriptions* [78] and support for data encoded in GML. This, however, still leaves room for semantic mismatches.

Standards must be present at least in the frontiers of our data manipulation cycle, "wrapping" it (see Section 3.2). The communication interfaces for data acquisition and publication are the two points where these solutions are most useful: as seen in Section 3.3.4, WebMAPS can be seen as a client application and a data provider to client applications. As a server, WebMAPS strives to adhere to standards, to enable interoperation with other systems. As a client, taking advantage of standard interfaces is important, but, as will be seen, being able to handle involuntary, non-standardized, access mechanisms might be equally important. As part of those efforts, its development is adopting Web services and SOAP protocols, OpenDAP, Microformats and KML, discussed next.

OpenDAP is an acronym for *Open-source Project for a Network Data Access Protocol.* It consists of a data transport architecture and HTTP-based protocol capable of encapsulating structured data, annotating the data with attributes and adding semantics that describe the data. One of its features is the ability to retrieve file subsets, and aggregate data from several files in one transfer operation. It is being increasingly adopted by earth scientists to publish their data – e.g., in oceanography [23, 26, 109]. In our framework, OpenDAP is used as a means to publish and receive data, in layers 1 and 7. For instance, images are acquired and served by WebMAPS using OpenDAP. In the first case, it is at the receiving end (Layers 1 and 2), while in the second case it is at the top of the cycle. As exemplified by [109], this allows scientists to exchange and visualize results of complex models.

Microformat is a web-based data formatting approach to re-use existing content as metadata, through standard annotations conveyed by XHTML (or HTML) classes and attributes. The intention is to allow information targeted to end-users to be also software processable. In other words, the layout and formatting markup are also used to do semantic annotations. Microformats replace more complicated methods of automated processing, such as natural language processing or screen scraping. Their use has direct impact in the representation of data in Layer 6, after being generated in Layer 5 along with other transformation processes.

In particular, Geo is a microformat used for marking up WGS84 geographical coordinates (lati,long) in XHTML. Figure 3.6 presents an example of the use of this microformat in an XHTML page. This allows parsing tools to mine for pages that contain coordinates in this format. This allows these pages to be redered using this geospatial information, e.g., in a mapping tool or loading the coordinates into a GPS device.

```
<div class="geo">          Campinas:
  <span class="latitude">  -22.906 </span>;
  <span class="longitude"> -47.061 </span>
</div>
```

Figure 3.6: Example of the Geo microformat in an XHTML page.

KML [80] is an XML-based language schema for expressing geographic annotation and visualization for 2D and 3D Earth browsers. It was initially developed for use with Google Earth. The KML 2.2 specification was accepted as a OGC standard, ensuring its status as an open standard for all geobrowsers. Our geometry files are represented in KML and accessed by Google, in which case we are acting as data providers (Layer 0 for another data management cycle). Figure 3.7 shows an example KML document where the city of Campinas is represented as a point (its centroid).

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://earth.google.com/kml/2.0">
<Placemark>
  <description>City of Campinas</description>
  <name>Campinas</name>
  <Point>
    <coordinates>-22.906,-47.061</coordinates>
  </Point>
</Placemark>
</kml>
```

Figure 3.7: Example of a KML document.

The perspective of the data providers on how much effort they put in providing interoperable access mechanism can be seen as voluntary and/or involuntary. The voluntary point of view is the one we have been discussing so far in this section. It is when the data provider willingly serves its data by means of well-known standardized interfaces and protocols, fostering data interchange between systems. Voluntary access mechanism usually comply to some extent to a standard, in an effort to make the data more easily accessible. The involuntary viewpoint encompasses data providers that are only concerned with data consumption from human users, providing no facilities for external systems interested in obtaining the same information. The reasons vary from lack of resources to the deliberate wish to prevent inter-systems data sharing. Therefore, involuntary access mechanisms usually do not have standardization as a main concern, even though they may comply to standards on occasion.

In order to include involuntary data providers in our solution, we have adopted techniques from the information extraction research field. One of these techniques is called *screen scraping*, in which a computer program extracts data from the displayed output of another program. Search engines and web crawlers use web scraping techniques. Indeed, Web pages are built using text-based mark-up languages, and frequently mix content with presentation. Therefore, web screen scrapers extract machine-friendly data from XHTML and other markup formats.

We believe that automated acquisition is important to bridge the testing computational models against data from several geospatial resource providers. The Paparazzi toolset is a step towards that goal, discussed in Section 3.3.3.

3.4.2 Services Approaches

In the services category of interoperability, there are two paradigms competing in the Web: Service-oriented architectures (SOA) and Resource-oriented architectures (ROA). SOA is a direct evolution of concepts born from distributed computing theory and modular programming practices. It is an architecture where functionality is grouped around processes and packaged as interoperable RPC-style services, loosely coupled with operating systems or programming languages. SOA's goal is to facilitate the composition of distributed web services, through the standardization of interfacing, reliable messaging, transactions and security. SOA's philosophy transcends the Web medium and could be successfully applied to other contexts.

On the other hand, ROA is intimately related to the Web. It rescues the principle of Representational State Transfer (REST), defined in [35]. REST outlines how resources are defined, addressed and accessed through simple interfaces, where domain-specific data is transmitted over HTTP without any additional messaging layer or session tracking mechanisms. ROA design aims for the Web's scalability, and it is defined by five main principles: First, application state and functionality are divided into resources. Second, a resource is uniquely addressable by means of hypermedia links. Third, all resources share the same constrained, uniform and well-defined interface. Fourth, resources support the HTTP protocol operations: GET, PUT, POST, DELETE, HEAD, OPTIONS. Finally, protocols should ideally be stateless, cacheable, layered and client-server oriented. ROA is more scalable than SOA, and easier to implement due to its uniform interface and adherence to Web model and standards.

SOA and ROA are complementary paradigms, together they maximize interoperability. For this reason, we advocate the adoption of hybrid architectures, such as in WebMAPS. As mentioned in Section 3.3.4 we use SOA when we act as a client to Google, and ROA when Google plays the role of WebMAPS client.

3.5 Related Work

To the best of our knowledge, no previous work considers all the phases covered in our framework. However, there are proposals that are related with a few of the layers, some of which are presented here. They cover the entire spectrum of solutions discussed in the paper, from data and process interoperability using standards and services, to user interface aspects.

There are many studies concerning use of standards, usually restricted to just one of our layers. For instance, Aim4GDI [2] uses OGC standards for accessing distributed data sources and creating composite results. They also extract metadata from these sources, composing them in RDF for later querying (using SPARQL) and publication (in an ontology description language). However, they only consider issues at the data access and interchange level, not covering processing resources and their interoperability.

The work presented in [52] considers the use of standards for both data and process interoperability, for distributed sources. Their solution consists on a framework based on the ISO19100 series of standards. The paper materializes the framework in a travel guide system called MTGS. However, limiting the standards considered for interoperability into a single standards source hampers the construction of multi-disciplinary models and applications, preventing their evolution. This is remarked by [65], which discusses the evolution of the GML standard and the importance of integrating it with standards from other application areas.

Interoperability through services is also common, in particular taking advantage of WFS, WCS and WMS. The work of [26], for instance, describes initiatives towards combining communication and access standards, e.g., providing common grounds for OGC's WFS and WCS to work side by side with OpenDAP to access oceanographic data. Their effort concurs with ours in the sense that combining different standards into systems design is a way of leveraging interoperability. Sensor networks can also be encapsulated according to the class of service provided [20]. In such a case, services are even more appropriate.

Our main concern, however, is to provide adequate support to flexible system development. From this point of view, the motivation of GeoModeler [109] is the closest to ours, making geospatial resources more accessible to applications running models. GeoModeler is a software framework that combines software components from a GIS with modeling and simulation software, ultimately allowing various forms of analysis and visualization of oceanographic data. Their approach, however, deals with construction of centralized systems and software components interoperability in such systems. It does not consider, for instance, data acquisition and publication issues.

Our layers stimulate data and process interoperability. One concern (e.g., Layers 3 and 5) is to ensure data quality. This kind of emphasis is undertaken, for instance, by [105]. The paper presents a mediator that considers data quality as the driving force of the integration process. Their integration approach involves comparing and evaluating different data providers and keeping information on this evaluation available alongside with the data. However, they do not address process interoperability issues nor take full advantage of metadata and representation standards.

Finally, there are a variety of proposals that consider user aspects: the use of contextual information [19], or interactions in which the user is a computational system [58], or humans [14].

The influence of contextual information on the semantic descriptions of geospatial data and processes is discussed in [19]. They also evaluate how context impacts on the user interaction mechanisms in geospatial system user interfaces. The paper proposes a framework that takes advantage of contextual information and description representations in ontologies to help guide the user through the composition of distributed data and processes. Although we do not explicitly use contextual information, our goal is similar. Our solution favors the adoption of standards and services to provide this effect, with advantages on precision of terms and disadvantages in flexibility.

The focus of the framework proposed in [58] is to evaluate the suitability of geospatial time series to the requirements of a given application. Once the suitability is calculated it can be applied to assess the results produced by the application, helping determine the suitability of such results to be used as input by other applications. In our solution, application requirements are also a major concern, but we consider the impacts of interoperability solutions in meeting such requirements instead of trying to tackle them directly.

Finally, visualization of spatial and non-spatial data on the Web is the main concern of [14]. They argue that access to geospatial data aimed at visualization should be easier and more efficient than transferring whole data sets to be processed locally. They propose a browser that supports queries to geospatial services, invoking remote processes and getting the results incrementally. Their solution goes notably in the direction of leveraging the transition proposed by us (from focus on resources to focus on models), with the limitation of not considering distributed data and processing sources.

3.6 Concluding Remarks

This paper presented a framework that analyzes the management of geospatial data from a life cycle perspective. This framework is being validated in the design and development of the WebMAPS project.

By isolating each layer in the cycle, with clear interfaces and tasks, the framework induces a methodology to design and develop interoperable geographic applications. Whereas related research concentrates on providing standards or services for one given data transformation stage, we show how these efforts can be seamlessly interconnected. This allows users to shift their focus from the technology being used to the models being constructed. Besides within implementation efforts for the WebMAPS project, we also applied the framework to the development of the eFarms project, which is centered on managing data from ground-based sensing devices. The framework not only helped on understanding and implementing solutions to the problems in sensor data management, but also it made clearer the possible interactions with other solutions (such as the ones from WebMAPS) and which modules from these solutions could be reused.

Future work involves extending the WebMAPS project to comply to more access standards, both from the communication and data representation points of view. Another research issue involves the use of ontology-based techniques to speed up query processing and annotate data and processes. Again, we point out that we have not considered ontologies in this work, even though they are another important means of improving data and process interoperability. For detail on on our work in this direction, the reader is referred to [32]. Still yet another ongoing effort is to incorporate our work about diagnosing similarity of oscillation trends in time series [68]. Finally, we are investigating the possibility of storing satellite image files in PostGIS and let them be handled by the Rasdaman system (`http://www.rasdaman.com/`).

Chapter 4

Accessing and Processing Sensing Data

Scientific models are increasingly dependent on processing large volumes of streamed sensing data from a wide range of sensors, from ground based to satellite embarked infrareds. The proliferation, variety and ubiquity of those devices have added new dimensions to the problem of data handling in computational models. This raises several issues, one of which – providing means to access and process these data – is tackled by this chapter. Our solution involves the design and implementation of a framework for sensor data management, which relies on a specific component technology – Digital Content Component (DCC). DCCs homogeneously encapsulate individual sensors, sensor networks and sensor data archival files. They also implement facilities for controlling data production, integration and publication. As a result, developers need not concern themselves with sensor particularities, dealing instead with uniform interfaces to access data, regardless of the nature of the data providers.

4.1 Introduction

Advances in sensor networks have leveraged research in computational models, which can now rely on more kinds of data on real world phenomena. This, however, brings new challenges to computational science. From the data perspective, challenges include dealing with integration of heterogeneous sources, sampling rates, data redundancy, sensor data querying, fusion and summarization, processing of stream real-time data, all subject to node, sensor and communication failures. From the network point-of-view, challenges include power management, communication protocols, physical device management, or dynamic reconfiguration of nodes. This chapter is concerned with the data perspective - i.e., how to offer applications that implement the models transparently access sensor data, regardless of sensor physical characteristics, specific middleware programming environments, and data publication formats.

Middleware-like solutions, e.g. [60, 67, 115], are frequently proposed to enable applications to access the sensing data sources – usually through the offer of APIs. If, however, the models require new data processing functions, it is necessary to adapt or extend the middleware. Moreover, if extra data sources are required by the models, the applications must cope with the problem of complying with additional middleware or data formats.

To solve these problems, our approach supports uniform encapsulation of sensors and sensing data. It is based on a special kind of component paradigm – *Digital Content Components* (DCC) [99] – which provides mechanisms for uniformly encapsulating both data and/or data processing units, and for composing them into more complex elements, thereby helping solve interoperability issues. Applications that encode the models thus have homogeneous access to heterogeneous sensing devices, eliminating the need for application developers to concern themselves with whether data comes from static files or dynamic sources, as well as device-level implementation issues.

To achieve these goals, we had to create a new family of DCCs, geared towards stream and sensing data sources. As will be seen, this involved dealing with several challenges, such as how to encapsulate data sources with dynamic and/or context-sensitive behavior. Yet another obstacle concerned the need for implementing device-dependent drivers, to offer application developers uniform access to sensor generated data.

We illustrate our solution with a real case study of modeling environmental conditions for agricultural (i) planning and (ii) monitoring [36]. Planning (i) involves developing sophisticated models which run on heterogeneous files containing sensor-produced data to simulate crop growth in a given region. The main data sources are satellite-based sensors and spatially distributed networks of ground-based sensors. Once the crop is planted, monitoring (ii) concerns re-running and calibrating the models, for continuous use of the same kinds of sensing data sources, now at real time, at different sampling time frames, to capture and control crop response to changes. While data sets in phase (i) are static, phase (ii) adds the issues of real-time data capture and management. Our framework, as will be seen, provides a uniform solution to both phases.

The chapter is organized as follows. Section 4.2 presents basic concepts and an overview of our proposal. Section 4.3 shows our solution to encapsulate data sources, software and sensor data within DCCs. Section 4.4 concerns preliminary implementation results. Section 4.5 discusses related work. Section 4.6 presents conclusions and ongoing work.

4.2 Solution Basics

4.2.1 Digital Content Components

A *Digital Content Component* (DCC) is a unit of content and/or process reuse, which can be employed to design complex digital artifacts [99]. From a high level point of view, a DCC can be seen as digital content (data or software) encapsulated into a semantic description structure.

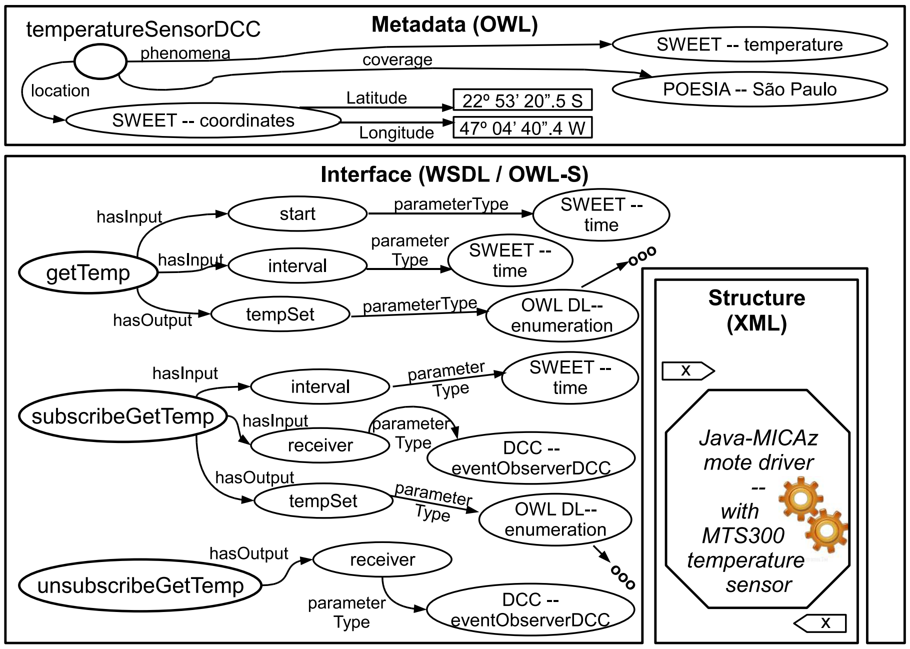


Figure 4.1: Structure of a SingleSensorDCC.

As shown in the example in Figure 4.1, it is comprised of four sections:
(i) the content itself (data or code, or another DCC), in its original format. In the example, a communication driver for a MICAz[1] sensor;
(ii) the declaration, in XML, of an organization structure that defines how DCC internal elements relate to each other (here, delimitating driver software);
(iii) specification of an interface, using adapted versions of WSDL and OWL-S – in the example, the `getTemp` and `subscribeGetTemp` operations;
(iv) metadata to describe functionality, applicability, etc., using OWL (in the example, the DCC is declared as belonging to the `temperatureSensorDCC` class).
Interface and metadata are linked to ontology terms – e.g., the `getTemp` operation has

[1]`www.xbow.com/Products/productsdetails.aspx?sid=101`

an input parameter that is a timestamp, as defined by the “Time” concept of NASA’s SWEET [93] ontology.

There are two kinds of DCC – process and passive. A *ProcessDCC* encapsulates any kind of process description that can be executed by a computer (e.g., software, sequences of instructions or plans). Their interfaces declare operations they can execute. Non-process DCCs, named *PassiveDCCs*, consist of any other kind of content (e.g., a text or video file). We refer the reader to [98, 99] for details on DCCs.

4.2.2 Overview of Our Solution

The usual approach to access sensor generated data is either to communicate with the sensor directly in its specific protocols or to use a wrapper implemented for each type of sensor. We propose instead to encapsulate all the data production particularities behind new kinds of DCCs. Besides providing data access, such DCCs can aggregate several functionalities – e.g., data delivery rate, stream data control, data annotation.

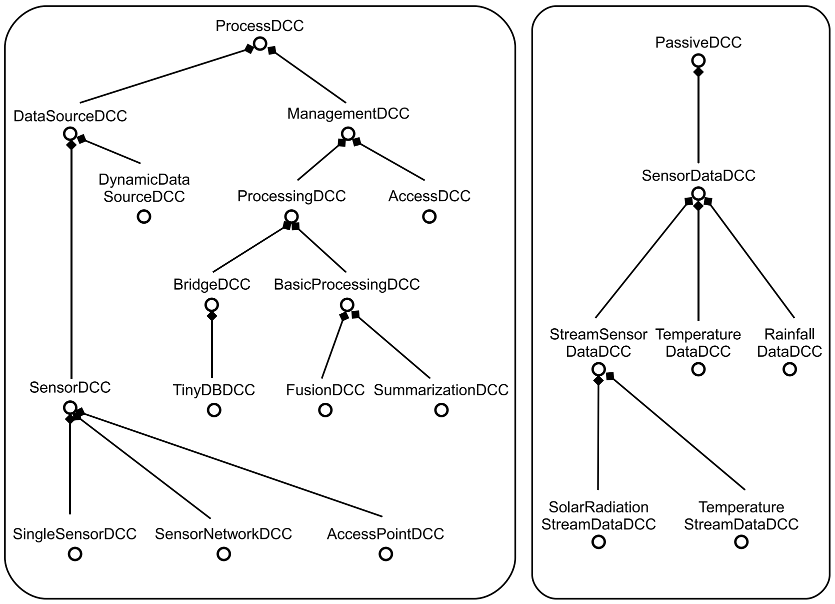


Figure 4.2: DCC taxonomy for sensor sources.

Figure 4.2, gives a functional overview, summarizing our new DCC types. On the left, there are ProcessDCCs proposed to encapsulate data sources and data manipulation software; at the right side are the PassiveDCCs proposed to encapsulate data itself.

The diamond ended lines represent the subclass relationship, i.e., a SensorDataDCC *is a* PassiveDCC. From the process point of view, there are two main branches: *Data-*

SourceDCC and *ManagementDCC*. The former is used for data source encapsulation – see Section 4.3.1; and the latter is the basis for encapsulating data manipulation functions, discussed in Section 4.3.2. As will be seen, data sources can be dynamic or stable. A *SensorDataDCC* (Section 4.3.3) encapsulates sensor generated data. Data encapsulation consists of wrapping data with accessibility rules, descriptive metadata and structure. Data, devices and software are accessed through the same interface scheme, the major advantage of adopting a DCC framework.

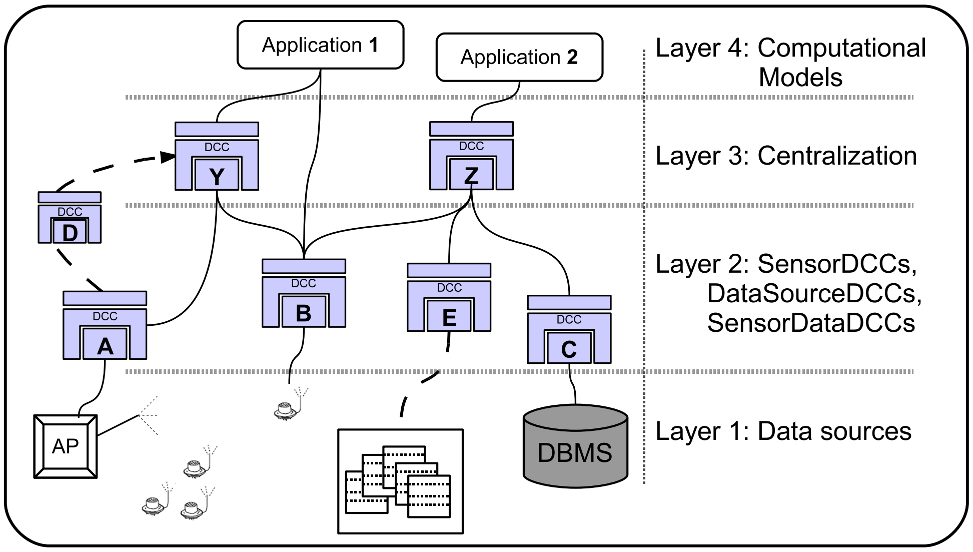


Figure 4.3: Management Layers.

Figure 4.3 gives an architectural overview of our solution, using the DCC types from Figure 4.2. It shows the encapsulation of data in PassiveDCCs and of data sources and data management functions in ProcessDCCs. Continuous lines indicate data flow between the elements (e.g., from **B** to **Y**), whereas dotted lines indicate reference to data sources (e.g., from **A** to **D** to **Y**). The bottom Layer contains the data sources: sensors (and their auxiliary devices and communication features), DBMS and other kinds of data sources (e.g., repositories of text, satellite images, historical time series). The second Layer contains the DCCs that provide access to data (e.g., SensorDCCs **A** and **B**, DynamicDataSourceDCC **C**, and SensorDataDCCs **D** and **E**), which play a role comparable to that of a mediator to access the data. The third Layer has data organization and centralization features (pre-processing, summarization, fusion). Applications, which implement the computational models, are in Layer four and access raw data from Layer two or pre-processed data from Layer three.

4.3 Encapsulation of Resources

This section explains how we met the challenges involved in the encapsulation of three kinds of resources: data sources (Section 4.3.1), manipulation software (Section 4.3.2), and sensor data (Section 4.3.3).

4.3.1 Data Sources

We consider two kinds of data sources: stable and dynamic. Stable data sources are characterized by eventual updates, being used here to supply computational models with context data, e.g., the spatial location of sensors' readings (including geographic coordinates or region names), data quality parameters, etc. These sources can also supply metadata, depending on the application. There are many implementations of these sources, including DBMS, XML files, or even web services.

Encapsulation of stable sensor-related data sources into ProcessDCCs is a straightforward application of DCC techniques – see [98, 99]. Sensor-related challenges appear when dynamic access is considered.

Dynamic data sources go through systematic updates. Two kinds of dynamic data sources are considered here: (i) sensing devices and (ii) subscribable services. Sensing devices are encapsulated within a *SensorDCC*, a specialization of a ProcessDCC. Sensor encapsulation is further explored in below.

Subscribable services provide data under some sort of agreement (e.g., upon request). An example is a weather forecasting service which produces updates on the rainfall forecast every hour, or when some threshold is crossed. These services are encapsulated into *DynamicDataSourceDCCs*, which offer access to the same functions of the service plus simulation of stable data source functions. Following the previous example, a DynamicDataSourceDCC encapsulating a rainfall forecast service can generate a notification adapted to each forecast event received. It can also offer, for instance, a summary of the forecasts for a period.

When a sensor is encapsulated within a DCC, its features are exposed through the uniform interface provided by the DCC. The major advantage of this approach is the separation of concerns it provides. On the one hand, there is the problem of developing sensor device drivers, including here other sources of sensing data, such as other middlewares (see Section 4.3.2). On the other hand, using DCCs, application changes do not require driver modification, and sensor (or middleware) changes do not affect applications.

Since each kind of sensing device has a specialized format for outputting data, different drivers must be implemented for distinct platforms, resulting in different SensorDCCs.

As an example, a specific SensorDCC implementation had to be built to access TelosB[2] devices (sensor network enabled device). SensorDCCs are similar to proxies to access the data. They allow creating a network of heterogeneous sensors as if they were homogeneous – e.g., in a crop monitoring model, a network with both rainfall and temperature sensors.

If one single sensor is encapsulated, we call it a *SingleSensorDCC*; if a sensor network is encapsulated, we have a *SensorNetworkDCC*. Both are SensorDCCs. A SensorNetworkDCC is responsible for all the sensors it encapsulates. Any message sent to this DCC is relayed to the encapsulated sensors, and it controls the forwarding of the data generated by its sensors. The sensors are not aware of the existence of the DCCs, thus suffering no interference in their functioning.

Figure 4.1 shows an example of a SingleSensorDCC we implemented, with details omitted. The structure section describes the organization of a software module that communicates with the sensor to access its data, i.e., it is the driver that establishes the communication between the SensorDCC and the sensor. Here, the driver implements a Java communication interface with a MICAz mote coupled with a temperature sensor. The `getTemp` operation receives a time interval in which the readings (floating-point numbers representing Celsius temperatures) will be returned to the caller. The second operation (`subscribeGetTemp`) receives the frequency in which it should pack and send the polled sensor data, until the (`unsubscribeGetTemp`) operation is invoked. The metadata section describes the SensorDCC: `sensorType` indicates the type of sensing device that is encapsulated within the DCC, in this case a TemperatureSensorDCC; `phenomena` indicates which kind of measure the produced data represents; `coverage` shows in which region the sensor is acting; finally, `location` specifies where the sensor is located.

SensorNetworkDCC allow representing an entire network within one DCC. The schematics are similar to the SingleSensorDCC, adapting the structure part to take care of multiple sensors. An external request sent to the DCC (e.g., `getTemp`) is translated into a request that is retransmitted to the sensors (through the wireless network) by the drivers. The query can be answered by every sensor individually or with data condensed within the network. The results make the inverse flow path. A SensorNetworkDCC is particularly valuable in networks that do not univocally identify each sensor, or in situations where it is not interesting or feasible to control each sensor node individually.

A sensor network can actually be encapsulated through its access point or base station. A DCC that encapsulates an access point (*AccessPointDCC*) creates an interface to the entire network. Through these interfaces, the applications can query the sensor network.

[2] `www.xbow.com/Products/productdetails.aspx?sid=252`

4.3.2 Encapsulation of Data Manipulation Functions

Each SensorDCC can offer individual management methods in its interface – e.g., setting and reconfiguring data generation parameters. However, controlling each SensorDCC on an individual basis may not be feasible. Higher level management layers can be created in order to further facilitate the management of data production and annotation. We thus introduce a specialized DCC, called *ManagementDCC*, which aggregates operations whose implementation is based on the individual management operations of each SensorDCC.

Processing Data

The processing of sensor data requires several specialized functions – e.g., summarization. We encapsulate such functionalities within *ProcessingDCCs*, a specialization of ManagementDCC – located in Layer 3 (centralization) of Figure 4.3. There are two kinds of ProcessingDCCs: *BasicProcessingDCCs*, which are implemented to directly compute functions on sensor-produced data, and *BridgeDCCs*, which serve as a connection point to sensor middlewares.

The former include functionalities such as data filtering, clustering and classification, application of association rules, multi-source data fusion, data summarization, among others. They, can be combined to obtain more complex processes.

BridgeDCC exist in order to take advantage of the many solutions already implemented in other frameworks. Consider, for instance, TinyDB [67], a popular middleware solution for accessing sensor data. Its SQL-like queries can be offered by operations on a BridgeDCC interface. Instead of having to become familiar with TinyDB, the application sends a request to the corresponding BridgeDCC, which translates it into the appropriate syntax, forwards the request and returns the result. The query proxy system from Cougar [115], or application adaptation update mechanisms from Impala [64], can also be made available through a BridgeDCC's high level interface. The same interface specification can thus serve to access these distinct systems.

Accessing Data

The *AccessDCC* is a ManagementDCC that offers operations for higher level data access. These operations reflect all the features offered by DCCs in lower layers. Consider, the following three examples. A BridgeDCC that implements access to the TinyDB middleware will transform requests from an AccessDCC into TinyDB's SQL-like queries. A SensorDCC that offers access to reprogramming sensor nodes may receive a parameter that contains the compiled software to reprogram the node. A ProcessingDCC that classifies data may receive as a parameter the maximum number of categories desired.

4.3.3 Encapsulation of Data

This section discusses the *SensorDataDCC*, a specialization of a PassiveDCC that encapsulates sensor generated data (Section 4.3.3), and its specialization, *StreamDataDCC* (Section 4.3.3), which encapsulates streamed data from sensor data sources.

Sensor Data

An infrared satellite image, or a file containing a temporal series of rain data, are typical examples of environmental data to be encapsulated into a SensorDataDCC. Like any DCC, a SensorDataDCC is annotated using ontology terms, e.g., data type, the physical phenomenon being measured (temperature, level of moisture, etc), the geographical location of the reading.

Any sensor generated data can be encapsulated in a SensorDataDCC. One option is to simply encapsulate existing files. Another is to apply filters to these files. Here we consider three factors that limits sensor data encapsulation: size, granularity and time.

From the granularity (number of readings) viewpoint, the options considered by us for creating a SensorDataDCC are: ignore readings (store only metadata from the sensor, for verification of the sensor's capabilities); one reading (a single reading is stored, for instance, the temperature at a given timestamp); a pre-defined number of readings (e.g., the last five readings of the sensor); and an unbounded number of readings, which requires a signal to stop transmission. From the size viewpoint (storage occupied), the options are: a pre-defined limit on the size of the DCC (e.g., using a memory window); and an unbounded size, also requiring a stop signal. From the time viewpoint, the options are: a pre-defined start/stop time limit, time interval or no time limit (any reading can be considered);

We can also combine the dependency among pairs of factors, such as (i) size dependent granularity (e.g., a pre-defined number of messages limited by their total size); (ii) size dependent timing (e.g., a start time for a pre-defined size); or (iii) time dependent granularity (e.g., all the messages within a time frame). Combinations of the three factors are also possible (e.g., all the messages within a pre-defined time and memory windows). Dealing with a stream source includes a fourth dimension to the problem, discussed next.

Streamed Sensor Data

DCCs have, by nature, a closed scope specification, i.e., they are clearly defined and can only be changed by an authorized user. Thus, a *StreamSensorDataDCC* does not directly support data stream encapsulation, rather it uses different strategies to mimic the behavior of stream data. This is a problem for a computational model that needs to handle stream data – e.g., for crop monitoring. Frequently, not all data can be stored, which either requires immediate consumption by applications, or some form of caching, or selective storage. Each of these approaches are implemented with DCCs for dealing with this incompatibility.

Since StreamSensorDataDCCs are passive, streams require a ProcessDCC – say, `P` – to pre-process the data to be encapsulated. In the first approach (forwarding the data for immediate consumption), `P` accesses the stream source only when needed, i.e., when an operation is posted to `P`. In this case, `P` ignores the data stream production until data is requested.

In the caching approach, `P` keeps the data available (e.g., in main memory) using a window of limited size combined with a discarding policy, e.g., first-in-first-out, least-used, or least-recently-used.

In selective storage, `P` polls the stream source in order to acquire the data at some constant rate, and stores it according to one (or more) of these three strategies: (i) directly into a database management system; (ii) in files, using some kind of structure for the data; (iii) `P` sends on the data to one or more additional ProcessDCCs that will take care of the data from that point on. In strategies (i) and (ii) the implementation of the storage must use a selection policy on the datasets to be stored. Examples of such policies include those for caching plus summarization (e.g., only means are stored), sampling methods (only a few values are stored), outlier detection methods (only out-of-range values are stored), and so on. In strategies (ii) and (iii), files can be encapsulated by SensorDataDCCs.

4.4 Implementation Issues

This section gives an overview of DCC implementation aspects. The presentation concentrates on specific components, thereby exemplifying some of the problems encountered. Code was implemented in Java. Annotations follow the OWL vocabulary and refer mainly to three ontologies: NASA's SWEET [93], POESIA [36] and the DCC ontology [98]. Performance issues are yet to be evaluated with a larger number of sensing devices, data sources and applications.

4.4.1 Data Encapsulation

This section describes three of the implemented SensorDataDCCs (Section 4.3.3), namely, *TemperatureSensorDataDCC*, *RainfallMapSensorDataDCC* and *RainfallMapSetSensorDataDCC*.

The TemperatureSensorDataDCC encapsulates a temperature time series stored in a plain text file, whose records are pairs <temperature, timestamp> – both measures as defined by SWEET – captured by a sensor in a given geographic location. There is one

TemperatureSensorDataDCC per sensor/location. Operations available include retrieval of one pair, or a sequence thereof, and some summarization computations (e.g., average). For instance, `getTemp(Date begin, Date end)` returns the set of all temperature readings within the time frame from *begin* to *end*. DCC metadata contain location coordinates, the measurement unit (e.g., Celsius degrees), and information on originating sensor.

The RainfallMapSensorDataDCC encapsulates one GeoTIFF file (standard where each pixel corresponds to a given geographical coordinate and contains one rainfall measure for a particular month of a particular year). Its operations concern: getting the entire GeoTIFF file, getting values from individual pixels (given either geographical coordinates or pixel relative position in the image). Metadata are of a similar nature to those provided for the temperature sensor.

We recall that SensorDataDCCs are passive components; thus, the implementation of these DCCs had to be followed by the implementation of ProcessDCCs, which contains the code of all interface operations. The main challenges faced here were determining the appropriate operations and their parameters, and the actual implementation of the operations.

4.4.2 Data Sources Encapsulation

Here we discuss issues of implementing DCCs described in Section 4.3.1. Implementation was divided in two parts – creating communication drivers to access data through sensor-specific protocols, and creating the DCCs themselves. The first part required familiarity with the characteristics of each sensor, e.g., to interpret the data packets emitted. This kind of effort is needed for every new kind of sensor device encountered – not only in our approach, but for any environment that wishes to support access to sensor data. The difference to other solutions appears in the second part. Whereas they require extensive communication-oriented coding to forward data delivered by the drivers, DCCs directly map driver operations to interface operations. Thus, coding effort is much smaller – the DCC approach not only supports homogeneous access from external applications, but also simplifies a programmer's work. Challenges in developing these DCCs were mostly associated with semantic annotations (e.g., finding appropriate ontology terms).

We implemented the communication driver using TinyOS [61] interfaces. Sensors were programmed with TinyOS and software developed in the NesC language, a C-derived component programming language.

We implemented SensorDCCs for the MICAz mote (MicazSensorDCC) and for the TelosB mote (TelosbSensorDCC). These SensorDCCs have similar interfaces and were tested with temperature and light readings. The MICAz mote requires an auxiliary MTS300 sensorboard, whereas TelosB came with integrated sensors. The MicazSensorDCC and the TelosbSensorDCC are generating real-time data and making them available in four ways (operations implemented): (i) on-demand access (application receives the data as needed in real time); (ii) generating text file data outputs; (iii) generating new SensorDataDCCs (TemperatureSensorDataDCCs); and (iv) updating existing SensorDataDCCs.

4.4.3 Examples with sensors

Models in agricultural planning require analysis of several kinds of sensing data, and their comparison along time, to detect trends in climatological values and their relationship with productivity of a given crop (e.g., [81]).

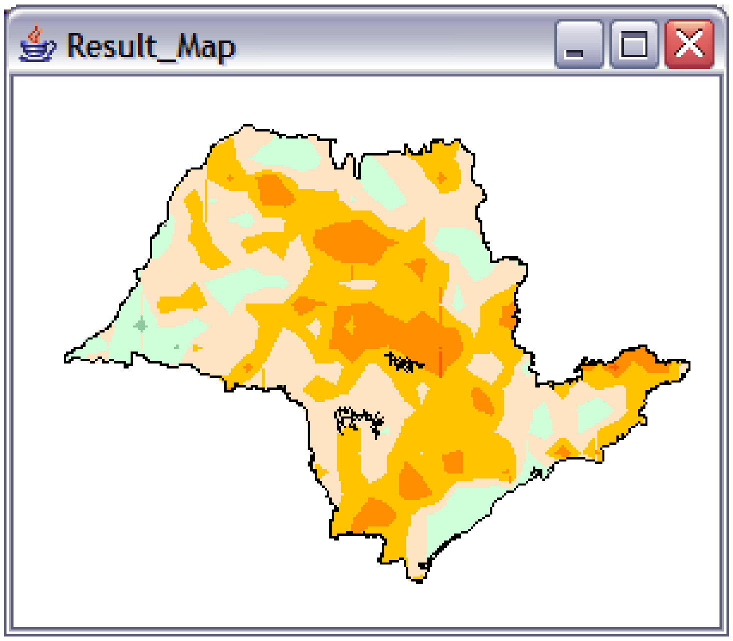


Figure 4.4: Screenshot comparing rainfall data.

Consider a model that needs to compare rainfall data from two different periods, for São Paulo State, producing a map that shows the result of this comparison. Figure 4.4 shows a screen capture of this application. It invokes `getMap` operations on two RainFallMapSetSensorDataDCC containing data on São Paulo State. One of these DCC encapsulates one GeoTIFF file, created from historical rainfall data, whereas the other encapsulates a service that provides rainfall data expressed in the GeoTIFF standard. Once these two datasets are obtained, the model calculates their "difference"; this is obtained by comparing the values of pixels that correspond to the same geographical positions. This may also requires pre-processing the data, to ensure that both datasets are scale and coordinate compatible (information provided by associated DCC metadata).

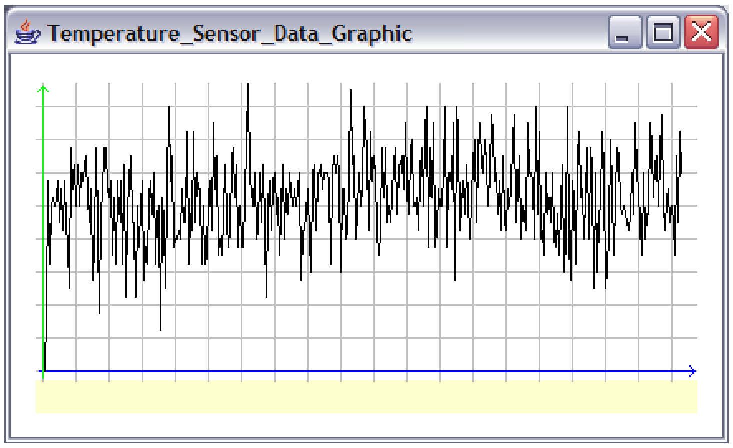


Figure 4.5: Polling a TemperatureDCC.

Monitoring models, on the other hand, process sensing data at real time. Figure 4.5 shows the result of a loop that polls a MicazSensorDCC, and plots a real time temperature curve, to be interpreted by experts. Polled data is also fed to another set of modules. In agriculture, experts typically compute aggregate values (e.g., average or maximum for a given time window) that are then used in a broader context – e.g., to create a dynamic average temperature map of an area, or to detect deviations from the norm. Such real time data is also stored in temporary files, to re-run the original planning models (e.g., as in the previous example).

We point out that, from an application point of view, all was achieved by invoking DCC operations, regardless of the characteristics of the underlying sensing sources. A planning model will process a sequence of invocations, whereas monitoring will perform a loop that will invoke a given sequence over and over, at specific time intervals.

4.5 Alternative Solutions

We have commented on related work throughout the chapter. There remains to compare our proposal with alternative approaches to data management solutions, in particular to homogeneous access to resources, which we achieve by encapsulating them within DCCs.

The solutions considered are: (i) Specialized (and language specific) implementations; (ii) Standards for data access such as Service Data Objects (SDO)[3] (iii) Software components and communication middlewares, such as CORBA, COM (DCOM and COM+) and .NET , EJB, and others; and, (iv) WSN middleware as defined by [43]. The first approach has the classic overhead of unnecessary repetition of work, hard maintenance, lack

[3]`www.jcp.org/en/jsr/detail?id=235`

of standardization and interoperability. SDO or similar initiatives only take into account data representation; furthermore a DCC can offer access to data using SDO. Components and general middleware lack stream manipulation flexibility, semantic descriptions, and, more importantly, homogeneous treatment of data, devices and software.

Exploring further the WSN middlewares, four approaches are close to ours. *Global Sensor Network* (GSN) [1, 54], has similar goals. However, data management is restricted to homogeneously accessing the network using a declarative language, while our proposal considers including pieces of software in new DCCs. The *Sensor Network Services Platform* (SNSP) [101] proposal considers the possibility of including processing software through the concept of "auxiliary service". However, it is centered around a formal specification of levels and services, leaving aside implementation issues, both for data publishers and data consumers. *Hourglass* [51] concisely considers processing solutions, but requires the use of a specific definition language and uses low-level TCP socket communication schemes, while DCCs use Semantic Web standards for both specification and communication. *IrisNet* [39] limits the use of sensor data to a hierarchical XML database, using XPath queries. DCC interfaces, on the other hand, can support a wide range of access mechanisms.

Other proposals act in more specialized branches. *TinyDB* [67] provides access to an entire network in a single entry point, which uses an SQL-like query system. It also aggregates data readings, decreasing the transmission costs, but at the cost of maintaining information on the network structure, compromising scalability. *Cougar* [115] works well on large sensor sets and makes data access easy with its query system; however, its focus is centered in efficient sensor programming, while our proposal is concerned with the management of data from the sensor outwards. *Impala* [64] is limited to a specific handheld hardware, but supports protocol and operation mode updates. *Maté* [60] addresses issues such as protocol updates and node heterogeneity (using a virtual machine approach), but lacks effective and easy communication with applications. *Magnet* [10] delivers a Java virtual machine on top of the network, facilitating the development of Java applications, but is unfit for nodes with limited capacity. These solutions can be encapsulated in our BridgeDCC.

4.6 Concluding Remarks

This chapter presented a framework to support flexible management and publication of sensor-produced data, an ever-growing need of computational models. Part of this framework has been implemented and validated. It is based on two main aspects: the use of DCCs to provide uniform access to sensor data, sensing devices and software to process the data; and the construction of ManagementDCCs to coordinate sensor data integration and publication.

Through this solution, applications that implement scientific computational models do not need to concern themselves with device dependent issues or with whether they are handling historical data files, or real time data streams. They just need to invoke the appropriate sensor and management DCCs to access the desired data, simplifying the problem of model construction and tuning.

Ongoing work involves several issues. As mentioned in Section 4.4, we are constructing more SensorDCCs, and extending our implementation to process data from large sensor networks. We are also working on the actual construction of BridgeDCCs to expand the compatibility base for experiments.

Chapter 5

Applying Scientific Workflows to Manage Sensor Data

There is a world wide effort to create infrastructures that support multidisciplinary, collaborative and distributed work in scientific research, giving birth to the so-called e-Science environments. At the same time, the proliferation, variety and ubiquity of sensing devices, from satellites to tiny sensors are making huge amounts of data available to scientists. This chapter presents a framework with a twofold solution: (i) using a specific kind of component – DCC – for homogeneous sensor data acquisition; and (ii) using scientific workflows for flexible composition of sensor data and manipulation software. We present a solution for publishing sensor data tailored to distributed scientific applications.

5.1 Introduction

In the recent past, e-science initiatives focused in research areas that had need for high performance computing, e.g., meteorology, genomics, particle physics. This has fostered research in Computer Science mainly in computer clusters and grids, and in data-intensive support systems. Currently we see a movement to offer computational support to a number of other research areas, including environmental planning and monitoring, agriculture, biodiversity, social sciences, and arts [5, 46, 96]. Supporting these new areas involves research in other Computer Science fields, such as databases (storage, retrieval and integration of multimodal and heterogeneous sources), compilers, human-computer interfaces. In particular, data generated by sensing devices are increasingly important to scientific research and applications. This chapter concerns supporting sensor data acquisition, manipulation and publication.

We are facing the proliferation of several kinds of sensing devices, from satellites to tiny sensors. This has opened up new possibilities for us to understand, manage and monitor a given environment, from the small – e.g., a room – to the large – e.g., the planet. In particular, wireless sensor networks (WSN), i.e., networks of communicating small sensing devices, powered by batteries and with limited storage and processing facilities, are subject to intensive research. Network nodes are frequently heterogeneous, generating distinct kinds of data at different time intervals. From the data perspective, challenges include dealing with integration of heterogeneous sources, data redundancy, data streams and real-time data, data fusion and summarization, all subject to node, sensor and communication failures. From the network point-of-view, challenges comprise physical device management, event detection and notification, power management, dynamic reconfiguration of nodes and network, and support to different simultaneous applications, among others. However, the proliferation, variety and ubiquity of these devices add new dimensions to the problem of heterogeneous data management.

Figure 5.1 outlines our proposal to deal with these issues, where layers denote different data access and manipulation levels, from data sources to applications. The continuous lines denote a data flow path and the dotted lines denote a reference to data sources. The first layer contains the data sources such as services, DBMS, files (e.g., text, imagery), and, in particular, sensor and their auxiliary devices. The second layer, detailed in section 5.2, contains a specific kind of component: *Digital Content Components* (DCC) [99], which have Semantic Web conformant annotations. Both data and data sources[1] are encapsulated within DCCs, so uniform interfaces are available to the applications, for dynamic (e.g., sensors) and stable (e.g., time series files) sensor data sources; and for dynamic (e.g., stream) and stable (e.g., on image) sensor generated data.

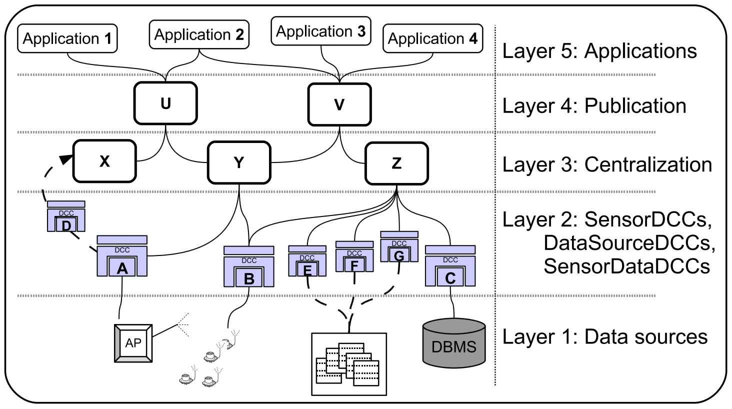


Figure 5.1: Management layers.

The third layer contains elements that centralize and manipulate the encapsulated

[1]We distinguish between data – the bits and bytes – and data sources – device or resources that provide data. This distinction can become blurred at times – e.g., a file can be treated either as data or as a data source.

sensor data from layer 2, offering functions such as data fusion, summarization, classification and sampling. In this layer, the processing elements can be single DCCs and/or scientific workflows. Workflows are applied to control and compose the basic functions so that the data are tailored to fit applications' needs. The fourth layer has the publication and data access mechanisms, offering high abstraction level interfaces. Applications in Layer 5 are regarded as clients of Layers 3 and 4.

Sensor data have particular requirements to be dealt with, specially concerning data streams manipulation and data fusion schemes. Sensor networks present an even bigger challenge as some of these solutions can be implemented within the network. Our DCC implementations provide solutions to these aspects, including the possibility of using intra and/or extra-network algorithm implementations. In this chapter we explore the composition of these solutions using workflow activities, which are transparently executed by invoking DCC operations.

This proposal has the following advantages:
(1) it provides homogeneous access to heterogeneous sensing devices;
(2) it enables applications to have multiple views of sensing data, by taking advantage of scientific workflows to mediate sensor data access. This fosters reuse of solutions for managing these data;
(3) it eliminates the need for an application to concern itself with whether a data source is static or dynamic. Items (1) and (3) are presented in detail in [87]. This chapter's main contribution lies on issue (2), presenting workflow specifications and categories for sensor data manipulation.

5.2 Revision on DCC and Encapsulation of Resources

A *Digital Content Component* (DCC) is a unit of content and/or process reuse, which can be employed to design complex digital artifacts [97, 99]. From a high level point of view, a DCC can be seen as digital content (data or software) encapsulated into a semantic description structure. As shown in Figure 5.2, it is comprised of 4 sections:
(i) the content itself (data or code, or another DCC), in its original format. In the example, the content is a driver for communicating and gathering data from a MICAz sensor (`www.xbow.com/Products/productsdetails.aspx?sid=101`);
(ii) the declaration, in XML, of a structure that defines how DCC internal elements relate to each other (here, delimitating the object code of the driver);
(iii) specification of an interface, using adapted versions of WSDL and OWL-S – e.g., the `getTemp` and `subscribeGetTemp` operations, in the example;
(iv) metadata to describe functionality, applicability, etc., using OWL (in the example, the DCC is declared as belonging to the `TemperatureSensorDCC` class and being located

at the `longitude` and `latitude` specified.
Interface and metadata are linked to ontology terms – e.g., the `getTemp` operation has as input parameter a timestamp as defined by the "Time" concept of NASA's SWEET [93] ontology.

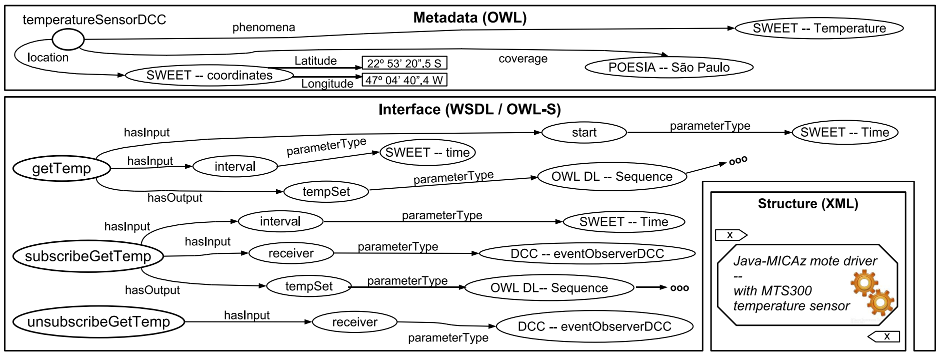


Figure 5.2: DCC schematic structure (from [87]).

There are two main kinds of DCC – process and passive. A *ProcessDCC* encapsulates any kind of process description that can be executed by a computer (e.g., software, sequences of instructions or plans). Their interfaces declare operations they can execute. Non-process DCCs, named *PassiveDCCs*, consist of any other kind of content (e.g., a text or video file); their interfaces declare their *potential functionality*, in the sense that the operations can be requested from them, but their execution code is not part of the content (e.g., a video player software is not part of a video). Since passive components by definition cannot embed executable code, operation implementations are encapsulated into special ProcessDCCs called *CompanionDCC* (e.g., a piece of music `M` stored in a PassiveDCC can be played by attaching `M` to a suitable music player in a CompanionDCC). We refer the reader to [99] for details the DCC infrastructure.

Recalling Figure 5.1, the second layer contains Passive and ProcessDCCs respectively encapsulating data and data sources. DCCs that provide access to sensor data, called *DataSourceDCCs*, play a role comparable to that of a mediator to access the data produced by a sensor. A specialization of a DataSourceDCC is the *SensorDCC*, which takes care of accessing sensors and collecting their data. One SensorDCC can encapsulate one (DCC **B**) or more sensors (**A**) by, for instance, encapsulating an access point of a wireless sensor network. Data can be delivered to the next layer in its original format (**C**) or encapsulated in another DCC, as done by **A**, delivering **D**. **D**, **E**, **F** and **G** are examples of *SensorDataDCC*, a PassiveDCC used for sensor data encapsulation and annotation. For more details on encapsulation of resources the reader is referred to [87]. Layer 3 (**Y** and **Z**) has data organization and centralization features (pre-processing, summarization and fusion). Finally, the fourth layer has the publication and access control features, and offers raw and processed data to applications in Layer 5. Features of layers 3 and 4 can be implemented either as software, event-based composition (using publish/subscribe techniques), or flow-oriented compositions (using workflows). The third kind is our focus in this chapter and is explored next.

5.3 Scientific Workflows

5.3.1 Basic Concepts

A *workflow* is a specification (or model) of a *process*, which is a set of inter-dependent steps needed to complete a certain task. A *Workflow Management System* (WFMS) is a computer system for specification, execution and management of workflows [113]. A *scientific workflow* is a specification of a process that describes a scientific experiment [111]. While most business workflows are first specified and then executed several times, a scientific workflow is often specified while an experiment is conducted, focusing on process documentation. Other singular characteristics include: (i) emphasis on data centric processes; (ii) a high degree of flexibility on specification and adaptation; (iii) support to uncertainty at specification and execution time, as well as a great number of exceptions during execution; (iv) possibility of modifying a workflow during its execution.

These characteristics pose problems to workflow specification, for which there are several proposals (e.g., WS-BPEL). Our workflow data model [70, 85] is compliant with international standards and induces a methodology for scientific workflow specification. Our model has been implemented in the WOODSS [70] framework for scientific workflow management. The several abstraction levels of a workflow are stored in a relational database and can be reused as workflow building blocks. Moreover, the specification of types and of abstract workflows capture the notion of *workflow design* independent of execution aspects, allowing design reuse. This supports the need scientists have for sharing and reusing not only executable procedures, but also their specification. The model requires that the users first specify types of workflow building blocks, next combine them into abstract specifications (*abstract workflows*), and finally refine these specifications into executable workflows (*concrete workflows*).

5.3.2 Management Workflows

Our solution adopts scientific workflows as a flexible way to coordinate the management of sensing data. Here, rather than directly accessing sensor data files, or interacting with sensor middleware, the idea is to take advantage of DCC interfaces and composition mechanisms. Thus, a solution constructed for one scenario or one set of sensors can be easily reused in different environments, adapting the DCCs that are available in the repository. Applications can select or post adequate workflow specifications in order to obtain the desired data. This solution adds reusability and flexibility for sensor data management.

The goal of a *management workflow* is to manage the sensor data in order to support complex application requests. More specifically, it controls SensorDCCs and Process-DCCs (embedding data manipulation software), coordinating the tasks of data extraction, processing and interpretation. Furthermore, a management workflow can access SensorDataDCCs as additional data sources.

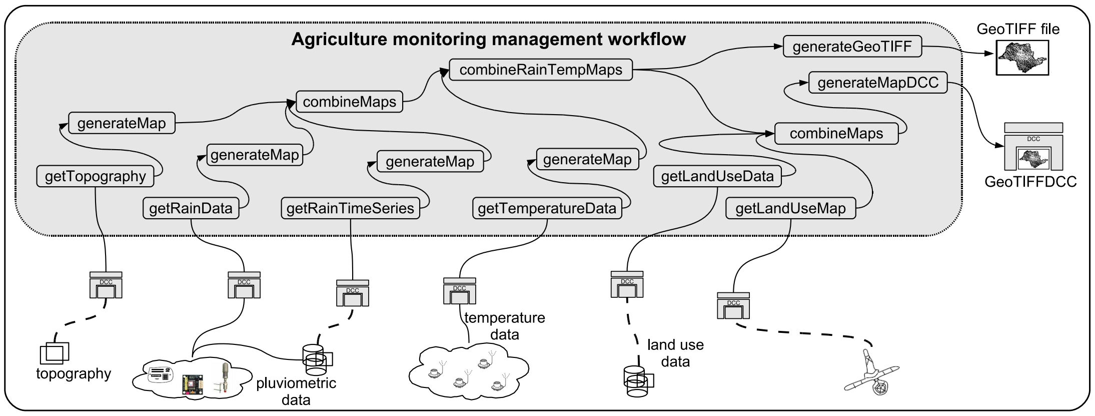


Figure 5.3: An example of a management workflow for crop monitoring.

Figure 5.3 shows an example of a management workflow for agricultural monitoring – i.e., it controls conditions in a given region to help crop management. It is used to acquire data from distinct kinds of sensors and sensor-derived data files. It periodically produces derived data, here a file of a map in GeoTIFF (a bitmap format that associates geographic coordinates to pixels). For instance, activity `getRainData` accesses a SensorDCC for real time rainfall readings, while `getRainTimeSeries` accesses a SensorDataDCC for historic rainfall data. Both generate maps that are combined to detect rainfall evolution patterns, taking into account a region's topography (map generated from topographic data).

Workflow activities access DCCs for data input, and can generate other DCCs, e.g., GeoTIFFDCC. The main mechanism to determine which DCCs can be used for a given activity is based on a combination of ontology annotations and type matching (of DCC operation interfaces and metadata, and activity specification). This is supported by the DCC management infrastructure – see section 5.4. We point out a few aspects that characterize our solution. The monitoring workflow is only concerned with sending data requests to DCCs, *regardless of the nature* of the data sources. Moreover, the workflow *manages and publishes* sensor data, and publication results can themselves be encapsulated in a DCC.

Our workflows are specified favoring our reuse methodology [70]. The first step consists in the definition of the data and activity types. Next, activities are created from types: after choosing an activity type, one can choose the data that is going to be used by that activity, based on the data types of the activity's parameters. Activities may be created only at an abstract specification level – thus defining an *abstract workflow*. This workflow can be then customized to specific situations, and subsequently instantiated for execution. Management workflows can be reused and adapted to solve new data needs. To make a workflow executable, pieces of software must be associated to each of its activities. This is where the DCCs come into action. Workflow execution is carried out transparently by a WFMS, which uses DCC operations, coordinating the data flow.

5.3.3 Validation Workflows

We can use several validation techniques to make sure that data follow given quality or validity parameters. This evaluation can be specified using the workflow and DCC based infrastructure. An interesting aspect of these mechanisms is the possibility of offering data quality assurance to the applications.

We propose five methods for data validation, which can be used individually or in a sequential combination inside a workflow. The methods are: sampling, summarization, pre-processing, and pre- and post-condition verification. Sampling, summarization and pre-processing are basic features frequently used to extract information from data sets. In the validation context, the idea is to extract representative sets of values from sensor-produced data, perform additional processing and evaluate the results against some benchmark. Pre- and post-conditions consist in evaluating logical expressions using the sensing data as variables.

All the validation schemes can be stored and published along with the data. Thus, data production can be traced back to its source, so that all the manipulations are available to be analyzed. Pre- and post-conditions can be stored as text. Pre-processing, summarization and sampling can be stored by references to the respective DCCs used (including DCCs that encapsulate workflows).

5.3.4 Publication Workflows

Publication is the final aspect of data management in our work. Management workflows are geared towards supplying data for specific application needs (e.g., generating erosion maps or controlling the temperature in a factory environment). Publication workflows, on the other hand, are general purpose data providers.

Functionalities to publish sensor data include: (i) data fusion schemes, even exerting device reconfiguration control; (ii) data summarization and sampling, with configuration parameters; (iii) application of statistical analysis over the readings; (iv) data classification, filtering, clustering and many other mining related techniques. These functionalities are accessed and composed into a publication workflow via management operations offered by individual and aggregated operations on SensorDCCs.

5.4 Implementation Issues

Figure 5.4 illustrates the main modules of the architecture. Scientists (the main users) interact with it via the GUI (Graphical User Interface) to design workflows and DCCs, and monitor workflow and/or DCC execution. The architecture relies on the following subsystems: (i) WOODSS [70] – a scientific workflow specification and documentation environment developed at UNICAMP; (ii) ANIMA [99], an infrastructure developed to support DCC execution and management; (iii) a set of modules to design DCCs, eventually reusing and modifying DCCs from a repository [97]; and (iv) a basic infrastructure needed to monitor and execute workflows, including the WFMS.

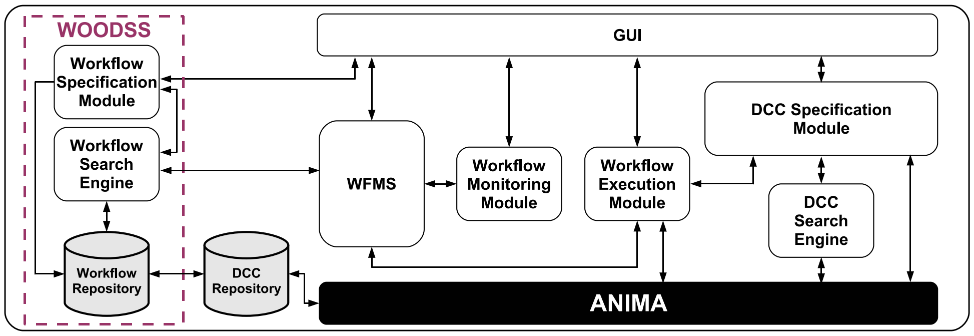


Figure 5.4: The main modules of the architecture.

In more detail, WOODSS allows users to specify scientific workflows, using our methodology, and to annotate both workflows and data manipulated by them. Workflow specifications (at all abstraction levels), their components, and annotations are stored in the Workflow Repository. Scientists can construct a workflow from scratch, or reuse and adapt stored workflows, retrieved with help from the Workflow Search Engine, based on workflow metadata – e.g., are there any workflows that create erosion maps? Repository records point to external elements they invoke (e.g., a Web Service) or manipulate (e.g., data sets) – [70].

DCC specification and management are handled by the modules on the right of the figure. Users interact with the GUI either to specify a new DCC from scratch, or to construct a DCC reusing and adapting existing stored components. The goal of the DCC Search Engine is to help DCC design and composition, supporting the user in finding the most appropriate DCC to reuse for a given application purpose – e.g., is there any SensorDataDCC that encapsulates rainfall time series for São Paulo State? This engine's search mechanisms are based on a set of algorithms that combine type annotations, metadata and interface matching [97].

We implemented SensorDCCs for the MICAz mote and for the TelosB mote (`www.xbow.com/Products/productsdetails.aspx?sid=126`). These SensorDCCs have the same interfaces and were tested with temperature and light readings. SensorDataDCCs implemented include temperature, light, rainfall map and rainfall map set.

To select a workflow for execution, a user (or an application) sends a request to the WFMS, which invokes the Workflow Search Engine to retrieve the desired workflow from the Workflow Repository. All data and DCCs needed to execute the workflow are all retrieved in this search step (e.g., our SensorDCCs).

Subsystems (i) through (iii) are already implemented, and we do not intend to implement a workflow engine. Rather, we will use some available system. Meanwhile, subsystem (iv) has been replaced by a simple coordination mechanism, extracted from Anima's synchronization modules. This allows simulating reasonably complex workflows, including those with parallel branches, synchronization and loops. The GUI has distinct modules that support graphical workflow design, DCC specification and construction of composite DCCs [99]. Present implementation efforts are concerned with two issues: first, we are concentrating on the development of more DCCs for sensors. Next, we will support internal interactions among WOODSS, WFMS and DCC modules for their integrated execution. Present integration occurs only at the repository level (the Workflow Repository points at DCCs stored in the DCC Repository). Moreover, the DCC Search Engine is not yet integrated into the system. We also intend working on this.

5.5 Related Efforts

We have commented on related work throughout the chapter. There remains to compare our proposal with alternative approaches, in particular regarding two aspects. One aspect concerns the (general-purpose) data management solutions, which we propose solving through scientific workflows. Another issue is homogeneous access to resources, achieved by encapsulating the sensors within DCCs.

Scientific workflows are extensively used in e-Science, e.g., orchestrating Web services, or specifying execution of experiments in a grid environment [71, 116]. Other efforts include applying workflow technologies [15, 22, 66, 117], distributed scientific data [21] and multi-modal scientific data management [106]. The major difference of these approaches to ours is the homogeneous treatment of data, data sources, and software. To the best of our knowledge, no proposals exist to use such workflows to manage access to heterogeneous sensing devices. Other alternatives for data management solutions include: (i) Specialized implementation, e.g., an entire software system for one specific application, which has the classic overhead of unnecessary repetition of work, hard maintenance, lack of standardization and interoperability. (ii) Other composition techniques, such as a publish-subscribe scheme [34], which are also promising, but are not suited for flow (or process) oriented executions.

From the point-of-view of homogeneously accessing the data, one approach alternative to ours is to directly use Web services to publish data, and to have access to sensors. However, this has two drawbacks: (1) unlike PassiveDCCs, Web services do not actually encapsulate data, and thus are always associated with some specific implementation; (2) a Web environment is mandatory for Web services, while DCCs can also be used in a standard programming environment, regardless of the Web (and its overhead). Other accessing solutions considered are: (i) Specialized (and language specific) implementations; (ii) Software components and communication middlewares, such as CORBA, DCOM and .NET, EJB, and others; (iii) WSN middleware as defined by [43]. The first approach has the same drawbacks of the specialized implementation for data management. Components and general middleware lack flexibility, semantic descriptions, and, more importantly, homogeneous treatment of data, devices and software. WSN middleware [39, 67, 115] are centered either on specific platforms or specific applications, none consider accessing data and software through homogeneous interfaces.

5.6 Concluding Remarks

With the possibility huge amounts of sensor data production, efficient management of these data is mandatory. In scientific research this is an even more sensible problem, since both the data sources and the applications accessing the data are typically heterogeneous and distributed. We described a solution to provide means to access and process sensor data in this scenario, aiming at flexibility and reusability of solutions. The main contribution of this work is the specification and implementation of a framework that: (i) provides distributed access and processing features to sensor data using DCCs, and (ii) flexible composition mechanisms using workflows for managing these DCCs. Ongoing work involves inclusion and evaluation of more sensor data sources and improving the execution mechanisms.

Chapter 6

Sensor Data Publication on the Web for Scientific Applications

This chapter considers the problems of sensor data publication, taking advantage of research on components and Web service standards. Sensor data is widely used in scientific experiments – e.g., for model validation, environment monitoring, and calibrating running applications. Heterogeneity in sensing devices hamper effective use of their data, requiring new solutions for publication mechanisms. Our solution is based on applying a specific component technology, *Digital Content Component* (DCC), which is capable of uniformly encapsulating data and software. Sensor data publication is tackled by extending DCCs to comply with geospatial standards for Web services from OGC (*Open Geospatial Consortium*). Using this approach, Web services can be implemented by DCCs, with publication of sensor data following standards. Furthermore, this solution allows client applications to request the execution of pre-processing functions before data is published. The approach enables scientists to share, find, process and access geospatial sensor data in a flexible and homogeneous manner.

6.1 Introduction

Sensors are fast becoming one of the main data providers for scientific applications. A given set of sensors may provide data to meet the needs of distinct applications – e.g., rainfall and temperature sensors may be used by researchers in environmental planning, habitat monitoring or epidemiology. However, each application domain – and each application within a domain – will require distinct kinds of data granularity and sampling. So, the question we answer is the following: how to devise a solution to the problem of sensor data publication, to support homogeneous access mechanisms to geospatial sensor based data from heterogeneous sources.

Our solution is based on a specific component technology, *Digital Content Components* (DCCs), which we extended to comply with geospatial Web service standards from OGC (*Open Geospatial Consortium*). DCCs are capable of uniformly encapsulating data and software, and are annotated with metadata and references to ontologies, following Semantic Web standards. We use them to encapsulate access to sensing data, homogeneously integrating them into a single framework. DCCs provide basic data manipulation functions, and can be composed into arbitrarily complex procedures. We use these functions to propose an extension to the OGC standards and proceed to show how to use DCCs to implement these extended standards.

We use a running example based on epidemics monitoring of dengue fever. It is caused by a virus and is transmitted to humans by the *Aedes aegypti* mosquito. Efforts to monitor dengue epidemics require combining geospatial data such as registered disease cases, geographical and demographic characteristics of the population, environmental data that is known to affect disease spread (e.g., rainfall or temperature), and locations where the mosquito is found.

The acquisition and access to such environmental readings for experimental research is an open issue. Our work contributes towards solving these problems.

The rest of the chapter is organized as follows. Section 6.2 presents the basics of the DCC technology and explains the encapsulation of resources into DCCs. Section 6.3 describes our proposal for multi-level integration of sensor data Web publication. Section 6.4 discusses implementation issues. Section 6.5 considers related efforts. Section 6.6 presents concluding remarks and ongoing work.

6.2 DCCs and Resource Encapsulation

This section briefly presents DCCs (section 6.2.1) and explains how we apply them to homogeneously encapsulate three kinds of resources: data, data sources (e.g., databases and sensors), and software.

6.2.1 DCC Basics

A *Digital Content Component* (DCC) is a unit of content and/or process reuse, which can be employed to design complex digital artifacts. It can be seen as digital content (data or software) encapsulated into a semantic description structure. It is comprised of four sections (Figure 6.1):
(1) the content itself (data or code, or another DCC), in its original format. In the example, the content is a driver for communicating and gathering data from a MICAz[1] sensor;

[1] `www.xbow.com/Products/productdetails.aspx?sid=101`

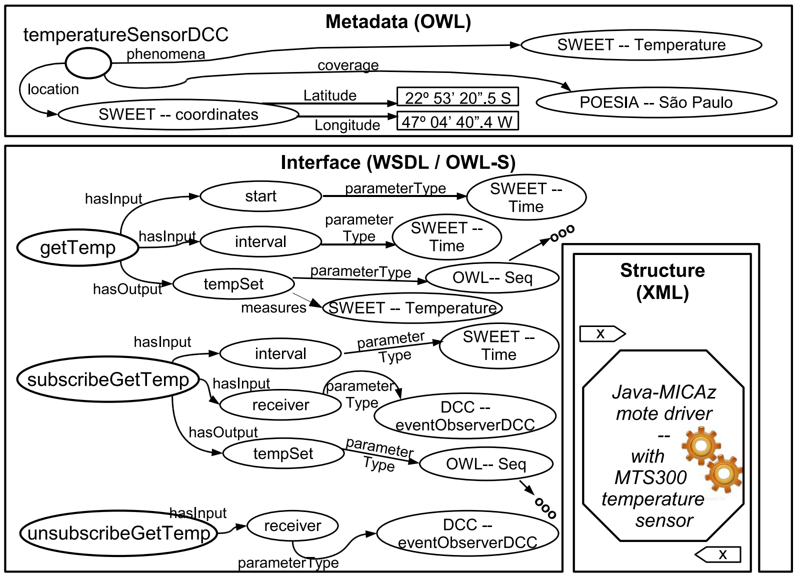


Figure 6.1: A DCC for sensor access.

(2) the declaration, in XML, of a structure that defines how DCC internal elements relate to each other (here, delimitating the object code of the sensor's driver);
(3) specification of an interface, using adapted versions of WSDL and OWL-S – e.g., `getTemp` and `subscribeGetTemp` operations;
(4) metadata to describe functionality, applicability, etc., using OWL (the DCC is declared as belonging to class `TemperatureSensorDCC` and located at `longitude` and `latitude` specified).
Interface and metadata are linked to ontology terms – e.g., input parameters of the `getTemp` operation are timestamps defined by the "Time" concept of NASA's SWEET [93] ontology.

There are two main kinds of DCC – process and passive. The first encapsulates any kind of process description, and Passive DCCs consist of any other kind of content (e.g., a text or video file). See [99] for details.

6.2.2 Encapsulation of Data (Access and Manipulation)

We encapsulate data withing PassiveDCCs and sensing sources in ProcessDCCs. DCCs can be used to homogeneously publish any kind of sensor generated data, be it static and/or streamed data [86] . A satellite image, or a file containing a temporal series of temperature data, are typical examples of static data, while continuous temperature reading transmissions are an example of streamed data. These DCCs are annotated using ontology terms such as data type, the physical phenomenon being measured (temperature, solar radiation, etc), the geographical location of the reading (e.g., GPS-provided). See [86] for more details on data encapsulation.

The number of sensors encapsulated within a ProcessDCC depends only upon the implementation of the sensor driver (as the MICAz driver in Figure 6.1). A sensor network, for instance, can be encapsulated by a DCC with a driver that communicates with the network's access point.

Manipulation of sensor data can occur in two levels: within a sensor or a network (signal processing, in-network fusion, etc) or externally (filtering data by region, fusion of heterogeneous water temperature sensors, etc). External processing, sometimes called post-processing, is usually application oriented – e.g., summarization of temperature readings per region and time period for a dengue spread simulation.

6.3 Sensor Data Publication

In most cases, sensor data is georeferenced. This makes it possible to employ general-use geospatial standards and services for sensor data publication (section 6.3.1). Section 6.3.2 shows how to use DCCs to publish sensor data in different scenarios. DCC annotations are translated into Web standards-compliant metadata, and DCCs are used to implement Web services (section 6.3.3).

6.3.1 OGC Standards

Geospatial Data Publication

The *Open Geospatial Consortium* (OGC) is an international organization that leads the development of standards for interoperability among geospatial applications. OGC's main general-use standards for geospatial data interoperability are the *Web Feature Service* (WFS), the *Web Coverage Service* (WCS) and the *Web Map Service* (WMS) [76]. A central notion is that of *feature*, i.e., a geospatial object. These standards specify the access mechanisms to, respectively, vector data (point-line-polygon), raster data (image-based) and rendered maps. These standards are specified to be implemented as Web services.

The WFS specification provides a standardized means to access geospatial data encoded in GML (*Geographic Markup Language*) [76] for the transport and storage of georeferenced data. A WFS-compliant service implements operations that allow retrieval of data and metadata, using several kinds of filters. The WCS specification allows interactions similar to these of WFS, but for raster data. Finally, the WMS specification allows clients to pose queries to retrieve rendered maps. Queries can specify a map's geographic extent, output format and the style – which is defined as *Style Layer Descriptor* (SLD) [76] files.

Roughly speaking, a query to retrieve Features (WFS), Coverages (WCS) or Maps (WMS) can be expressed by a tuple <`query,filter,style`>. The `query` is subject to `filters`, and `style` is the SLD specification for maps. Section 6.3.2 describes our extension to this approach.

Sensor Data Publication

OGC is now working on standards for sensor interoperability and sensor data access and publication. Its *Sensor Web Enablement Working Group* (SWE – `http://www.opengeospatial.org/projects/groups/sensorweb`) is proposing standards for data encoding and common Web service interfaces for data access. The encoding proposals are *Observation & Measurements Schema* (O&M), *Sensor Model Language* (SensorML or SML), and *Transducer Markup Language* (TransducerML or TML) [77]. As most of OGC's standards, the languages are defined by means of XML Schemas. The service interface proposals are *Sensor Observation Services* (SOS), *Sensor Planning Service* (SPS), *Sensor Alert Service* (SAS), and, *Web Notification Services* (WNS) [77]. The SWE Common [77] initiative aims at a common vocabulary to be used within the SWE framework.

The most basic encoding is TML, which deals directly with transducer (sensor or actuator) data. Higher level encoding is covered by SML, which can represent the processes sensor data went through. The last encoding level is O&M, which represents sensor originated data independently from the level of data processing.

SML is used for modeling and representing processes that generate sensor data. SML data sources are not restricted to sensors alone, and can also be a sensor network, a sensor wrapper or database with sensor data, etc. In SML a *ProcessModel* is an atomic processing block that defines its own inputs, outputs and parameters. It is also related to a *ProcessMethod*, which defines the interface and behavior for a process as well as metadata about the data it can provide. A *ProcessChain* is a composite processing block, built upon ProcessModels or other ProcessChains.

From the service interfaces point of view, SOS is intended to provide access to sensor data represented in any of the three encoding proposals (O&M, SML and TML). SPS focus on providing access to data acquisition and manipulation capabilities from resources (e.g., processing systems, archiving systems, sensors and/or auxiliary systems). SAS and WNS are intended to provide means of subscribing to a service (SAS) for update notifications (WNS).

6.3.2 Accessing Sensor Data

Many interoperability issues can be solved by publication of sensor data using the analyzed OGC standards. Sensor data can be accessed by using WFS (data as a feature) or SOS (with specific mechanisms for sensor data access). In either case, access is carried out by posting a query to a standard-compliant Web service. The query and the result format standards provide means to uniformly describe, publish and access data produced by sensor devices. If WFS is used, an application domain schema must be previously agreed upon by the participants.

However, in a research scenario using sensor data, access via query posting does not always suffice. Two unresolved issues are the following:

- Scientists need to be able to request data pre-processing before executing a query;
- Good pre-processing functions used in models need to be made available to other scientists, for reuse and validation of each other's work;

OGC is trying to solve these issues by enhancing query filter mechanisms. Nonetheless, more flexible support is also desirable. Besides the filtering option offered by OGC, we propose two novel solutions: (a) the producer should publish a list of pre-processing functions to be chosen by the consumer, or (b) the producer should allow the consumer to post the entire processing operation within a request. The latter can be achieved by posting a procedure schematics (such as a workflow).

Option (a) is easy to implement. The publication interface can provide these functions as different service operations. Nevertheless, this requires modifying the service every time a new function is to be made available. Option (b) is particularly interesting, given available standard ways to represent Web service compositions (viz., BPEL – `www.oasis-open.org/committees/wsbpel/`) – e.g., specifying a given sequence of operations on temperature readings before publishing the data.

Therefore, we propose to extend OGC's combination <`query,filter,style`> to <`query,schematics,style`> for data access. Section 6.3.3 details this solution.

6.3.3 DCCs and the Standards

Our publication proposal adopts DCCs and OGC standards in complementary roles. On the one hand, OGC has a well established XML-based standard to represent geospatial metadata. On the other hand, DCCs adopt OWL, which opens plenty of integration possibilities.

We start by uniformly encapsulating sensor data within DCCs, which have associated annotations. These annotations can be used translating data within the DCC to any of the OGC sensor data encoding standards (detailed below).

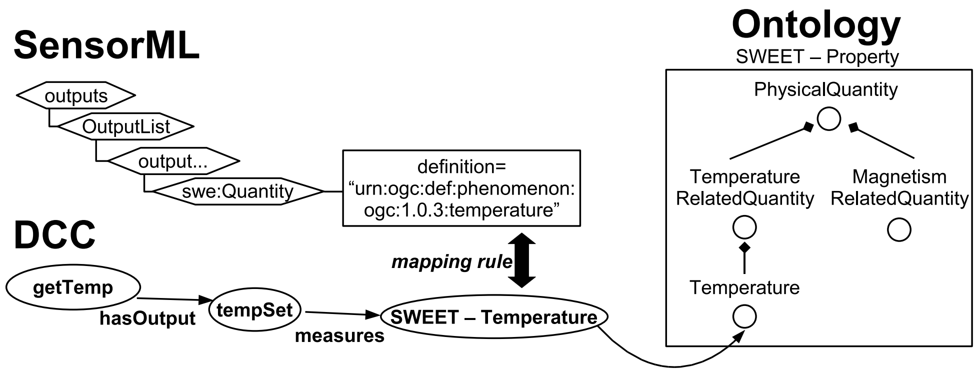


Figure 6.2: Mapping SML to DCC Metadata.

Consider, in our dengue example, a temperature sensor annotated with SML metadata, handled by the DCC presented in Figure 6.1. SML metadata can be mapped to DCC metadata by establishing a correspondence between key SML elements and ontology concepts in OWL. An example is illustrated in Figure 6.2, which maps a SML sensor output to a DCC output. On the upper left of the figure, the SML XML annotation defines that the sensor outputs temperature readings (`urn:ogc:def:phenomenon:ogc:1.0.3:temperature`). On the lower left of the figure, this output is mapped to a DCC operation output: OGC XML temperature is mapped to the "temperature" concept, part of SWEET ontology (right of the figure).

Published sensor data can be combined with other kinds of data. Consider, now, combining sensor data on rainfall and temperature with data on mosquitoes and diseases. Through DCCs we create bridges between OGC temperature concepts and other ontologies for mosquitoes and diseases. Our TemperatureDCC (Figure 6.2) can be composed with other DCCs – e.g., its OWL metadata can be further connected with OWL ontologies for insects (mosquito) and diseases (Dengue).

Sensor data may need to be published at distinct granularities. A user may need a high level summary of mosquito locations, and also a detailed view on the rainfall data. This issue is considered by efforts in SWE. Using SWE alone, however, is not enough. SWE considers only process representation and not access to process execution and customization. With our extension, these processes become available for invocation through DCC interfaces.

DCC interfaces are described in WSDL and OWL-S and thus are compatible with Web service interfaces. This way, we achieve adherence to OGC's standards for data representation and data access. Moreover, since DCCs were originally conceived as reuse units, they support flexibility in adding new pre-processing functions. Finally, DCCs can offer access not only via SOAP messages (in a Web environment), but also be used in a standard programming environment.

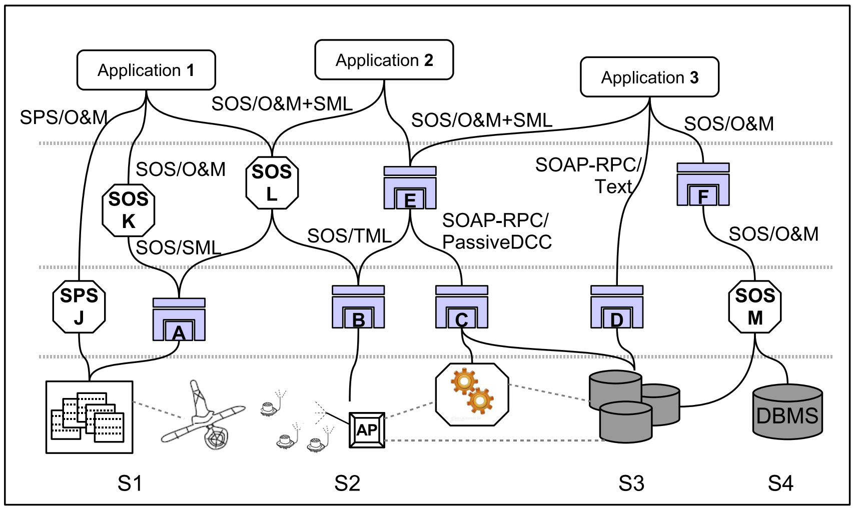


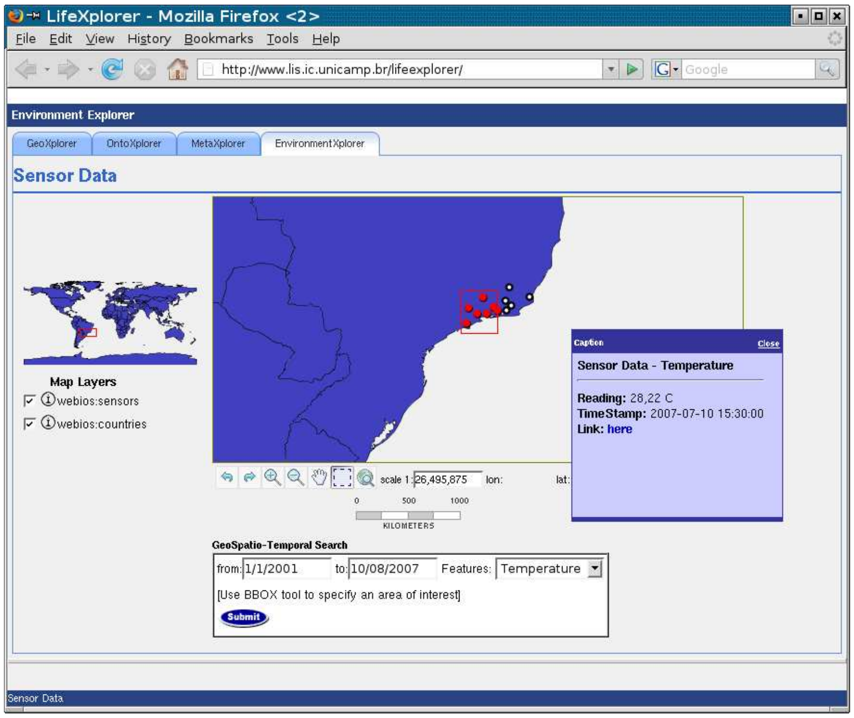
Figure 6.3: A multi-level integration example scenario combining DCC and SWE.

Figure 6.3 shows an example scenario combining data access using (i) specific implementations of OGC's publication standards (octahedrons) – e.g., SOS/SML, SPS/O&M; and (ii) DCCs (squares), using DCC communication mechanisms – e.g., RPC/PassiveDCC – and OGC publication standards – e.g., SOS/O&M, for Web service implementations. The labels in the lines show which communication mechanism and which sensor data encapsulation strategy are employed. Each level passes data through a processing function – from raw production to complex manipulations. For instance, specific pre-processing can be invoked to filter out outliers, summarize data or merge data from several sources. Basically, one can design distinct DCCs dedicated to each such pre-processing step, and application designers can dynamically choose which implementation to adopt.

Applications and other processing services may access data available in any level. Consider *Application 3*, for instance. Let `S1` through `S4` be data sources, where `S1` are satellite images from which vegetation can be derived, `S2` is a network of temperature sensors, `S3` is a database of current infection case data, and `S4` is historical infection data. The application corresponds to a scenario of map generation using: sensor location details (from **B** and **E**), sensor fused temperature data (from **C** and **E**), infection case data (from **D**), and time-series data on monthly distribution of past cases (from **M** and **F**). Maps generated by *Application 3* can be a final result – the spatial distribution simulating new case occurrences for the next month. They also can be used to feed another application.

6.4 Implementation

A few components were implemented to illustrate the main ideas. Two sensor platforms, TelosB[2] and MICAz, and a database system, PostgreSQL, were used as data sources. One sensor data processing unit was implemented, using a simple summarization algorithm. The GeoServer (`http://www.geoserver.org/`) implementation of the Feature-Map-Coverage (FMC) services, i.e., a WFS/WMS/WCS server, was used to publish data as a service. The MapBuilder (`http://communitymapbuilder.osgeo.org/`) tool was combined with the DOJO Javascript toolkit (`http://dojotoolkit.org/`) to enable Web browser visualization.



Sensor Data - Temperature

Reading: 28,22 C
TimeStamp: 2007-07-10 15:30:00
Link: here

GeoSpatio-Temporal Search

from: 1/1/2001 to: 10/08/2007 Features: Temperature

[Use BBOX tool to specify an area of interest]

Figure 6.4: A screen capture of the application.

The code running on the sensors was implemented in NesC [38], a C-derived component oriented programming language, using the TinyOS [61] interfaces. Data access on the sensors was carried out by ProcessDCCs with encapsulated drivers, implemented in Java.

A simple application was developed using these components. Temperature data is collected by the sensors and acquired by the respective ProcessDCCs. Operations of these DCCs allow access to the data in real-time. Other ProcessDCCs access these data for making them available as raw real-time data, summarized data, and time-series (stored

[2] `www.xbow.com/Products/productdetails.aspx?sid=252`

on the database system). The stored time-series are available to the FMC server through the database and real-time data are available directly.

Figure 6.4 shows a screen capture of the client application. Air temperature data is available in real-time and as time-series, both with the option of using the summarization feature.

6.5 Related Efforts

Efforts related to our work include scientific and sensor data manipulation and publication.

General approaches are also possible. The ones considered are: (i) Specialized implementations; (ii) Software components and communication middlewares, such as CORBA, COM+ and .NET, EJB, and others. The first approach has the classic overhead of unnecessary repetition of work, hard maintenance, poor standardization and interoperability. Components and middleware lack flexibility, semantic descriptions, and, more importantly, homogeneous treatment of data, sources and software.

[48] address collaboration in a peer-to-peer scientific data sharing scenario. They claim that emerging patterns typical of scientific collaboration can be exploited to improve data sharing and search mechanisms. Although we are not concerned with network organization issues, DCC's caching mechanisms [99] have a similar effect, reducing latency time for frequently accessed resources.

[6] propose a grid architecture to integrate sensor networks. The architecture regards sensors as grid resources and employs SML to describe them. The proposal does not consider publication of sensor data outside their grid infrastructure, hampering data reuse. [24] follow a similar approach, employing grid technology. Their architecture is strongly based on OGC standards for sensor data, which is in line with our approach.

6.6 Concluding Remarks

This chapter presented a solution for homogeneous access on the Web to sensor generated data, for scientific research. By extending OGC service standards and implementing them as DCCs, our solution fosters interoperability in situations where geospatial sensor data need to be combined with other kinds of data sources, a common scenario in scientific research. Moreover, thanks to DCC construction principles, the solution supports posting of schematics (e.g., a workflow) to pre-process data before publication.

Ongoing work includes developing new kinds of DCC for implementing new mappings. We are also extending the DCC design and combination framework of [99] to support automated publication of DCCs as Web services.

Chapter 7

Multimedia Semantic Annotation Propagation

Scientific research is producing and consuming large volumes of multimedia data at an ever growing rate. Annotations to the data helps associating context and enhances content management, making it easier to interpret and share data. However, raw data often needs to go through complex processing steps before it can be consumed. During these transformation processes, original annotations from the production phase are often discarded or ignored, since their usefulness is usually limited to the first transformation step. New annotations must be associated with the final product, a time consuming task often carried out manually. Systematically associating new annotations to the result of each data transformation step is known as *annotation propagation*. This chapter introduces techniques for structuring annotations by applying references to ontologies and automatically transforming these annotations along with data transformation processes. This helps the construction of new annotated multimedia data sets, preserving contextual information. The solution is based on: (i) the notion of semantic annotations; and (ii) a set of transformations rules, based on ontological relations.

7.1 Introduction

Scientific applications are producing and consuming ever growing volumes of multimedia data, which may vary from sensor based data (e.g., aboard satellites or ground-based) to video and sound recordings. In this scenario, scientists constantly need to share and reuse their data sets, being hampered by the wide spectrum of data production devices, actors and contexts that are involved in a lifecycle that *produces*, *transforms* and *consumes* data.

Metadata and annotations have been used as the primary means of describing data sets. Metadata are, often, text fields associated to data (e.g., in a file header) to be directly applied in automated data management tasks, such as indexing, searching, or context integration. Annotations, on the other hand, are more flexible, often representing personal remarks created by data producers or consumers, but are also more limited when considering management tasks.

Albeit helpful in improving data interpretation, metadata/annotations become less useful, or even useless, as soon as the data set goes through some sort of processing function. The resulting data set requires new metadata/annotations. Roughly speaking, this characterizes the scenario for *metadata evolution* or *annotation propagation* [12, 13]. This poses the following problems: (1) how to propagate relevant metadata/annotations that would otherwise be discarded during a transformation? and, (2) how to support automatic creation of metadata/annotations for the transformed data, taking context into account? Our work contributes towards solving these two questions.

In more detail, the traditional life-cycle for data sets is *(a) production – (b) transformation – (c) consumption*, where stage (b) may involve several steps. Most data interpretation tasks occur in the last stage. Adding metadata/annotations to the process improves the interpretation, and the cycle becomes *(a) production – (a') annotation – (b) transformation – (b') (re-)annotation – (c) consumption*. The main interest in this chapter is on how to (partially or totally) automate stage (b'). In particular, we are concerned with combining the notions of metadata, annotations and ontologies producing what we call *semantic annotations*, in which annotations are structured and are defined in terms of references to ontology concepts and/or relationships. Terms from ontologies help provide contextual information.

Our approach uses semantic annotations both from data and from operations that transform the data, i.e., we assume these annotations are available. We then propose a mechanism through which annotations are generated and attached to data sets produced by a data transformation operation. These new annotations are derived from the annotations made on the input data and the operation's interface, using a set of propagation rules that are based on ontology terms and relationships.

The main contributions of this chapter are therefore: (i) a general definition for the annotation propagation problem, applicable to several different environments of data transformation; and (ii) an extensible ontology-guided technique for solving the annotation propagation problem using semantic annotations. Though placed in the multimedia data management context, our solution can be extended to any environment where digital content is acquired, transformed and shared.

The remainder of the chapter is organized as follows. Section 7.2 shows an example that we use to illustrate our proposal. Section 7.3 defines the annotation propagation problem. Section 7.4 presents our solution to the problem using semantic annotations. Section 7.5 discusses related work. Section 7.6 presents conclusions and future work.

7.2 Motivating Example

Figure 7.1 illustrates a multimedia data transformation process in environmental modeling that combines satellite images with temperatures readings (from ground sensors) for a given region and period and generates JPEG images showing how temperature influences vegetation growth in the region. In a high level, the steps are the following:
(1a) acquire temperature data (data streams) and (1b) satellite images for a given region (in a format called GeoTIFF[1]);
(2a) convert the temperature readings and (2b) compute the greenness of vegetation (using the so-called NDVI method[2]) on the satellite image generating a NDVI image;
(3a) generate a temperature map interpolating readings (e.g., using Thiessen polygons) and (3b) classify the regions in the NDVI image according to the greenness range – both maps generated at step 3 are GeoTIFF images;
(4) combine the two images into one; and,
(5) convert the resulting map into a JPEG image.

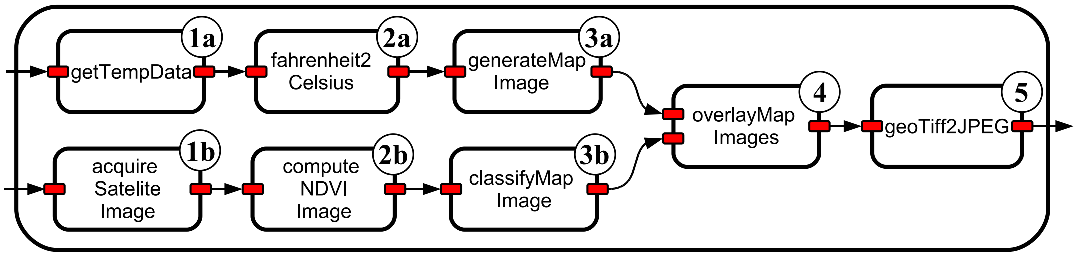


Figure 7.1: Data transformation process.

The resulting images illustrate the correlation between temperature and vegetation conditions in the area and is sufficient for a high level view of this problem. However, it is unsuitable for any kind of scientific study on environmental conditions. First, each JPEG image should be annotated indicating that it is the result of combining satellite and sensor data. This may still not be enough – sensor type and calibration, satellite type and spectral band used must be informed. This contextual information is lost during the transformation process, unless all multimedia data are manually annotated. This is difficult for large processes and impossible if parts of the process are done by different people or organizations.

The more complex the data and the transformations performed, the greater the need for contextual information. The annotations for the input data sets (satellite image and

[1]An image format where each pixel corresponds to a given geographical coordinate.
[2]Normalized Difference Vegetation Index (NDVI) indicates the levels of live green vegetation, usually over satellite images.

temperature readings) are provided by their source. However, at the end of the process, the output images have no associated annotations.

7.3 The Annotation Propagation Problem

7.3.1 Semantic Annotations

We combine characteristics of metadata and annotations into *semantic annotations*: using the structure from the first, filling its contents with references to ontologies, which provide the flexibility of the latter. Based on RDF structuring, we define semantic annotations as follows.

> **Annotation Units**. An *annotation unit a* is a triple `<s,p,o>`, where `s` represents the subject being described, `p` represents a property that describes it, and, `o` represents a describing object or value.

> **Semantic Annotation**. A *semantic annotation* M is a set of one or more annotation units, with at least one unit having as its subject the entity being described.

A semantic annotation is materialized as an RDF graph, which is represented as a set of RDF triples (*subject – predicate – object*); subject and predicate are identified by an URI[3] while the object may be an URI or a literal. Note that an object on one annotation unit may be itself a subject on another unit. This is the basic structuring element for semantic annotations. For space saving we omit the namespaces for terms in the text and figures.

We assume that a data transformation operation is a black-box that can be invoked providing data as input and producing output data. Semantic annotations are basically used to describe two entities in our solution: the data sets used and the interfaces of transformation operations. Figure 7.2 shows a transformation operation. The input data set D is annotated with M and the output data set $\mathsf{D'}$ is annotated with $\mathsf{M'}$. The operation has its input interface I described by semantic annotation $\mathsf{M_i}$ and its output interface O described by $\mathsf{M_o}$.

7.3.2 Annotation Propagation

Let $(\mathsf{T}, \mathsf{I}, \mathsf{O}, \mathsf{D}, \mathsf{D'})$ denote an application of a data transformation operation T which has an input interface I and an output interface O, and is applied on a data set D, resulting in derived data $\mathsf{D'}$. The definition for T was adapted from [30]. Also, let $(\tau, \mathsf{M_i}, \mathsf{M_o}, \mathsf{M}, \mathsf{M'})$

[3]Or, to be more precise, a URIref, which is a URI that may have a fragment identifier (the symbol "#") at the end, for referencing parts of the URI.

denote an application of an annotation transformation τ that manipulates $\mathsf{M_i}$ (the annotation of I), $\mathsf{M_o}$ (the annotation of O) and M (the annotation of D) to achieve $\mathsf{M'}$ (the derived annotation of $\mathsf{D'}$). The annotation propagation problem is defined as follows.

> **The Annotation Propagation Problem**. Consider a transformation T, with an input interface I and an output interface O, applied to a data set D, transforming it into another data set $\mathsf{D'}$. Which transformation τ can generate the new annotations $\mathsf{M'}$, given the previous annotation M on the data, the annotation of the input interface $\mathsf{M_i}$ and the annotation of the output interface $\mathsf{M_o}$?

If we now extend this definition to consider semantic annotations, the problem becomes: how to combine the sets of annotation units from the semantic annotation of the data set, the input interface and the output interface, to generate a new set of annotation units that will constitute the new semantic annotation. The mechanism to do that should ensure the consistency of the new set as well as its completeness regarding the available annotations.

For the remainder of the text the term annotation refers to semantic annotation.

As the operations considered are black-box operations, the annotation propagation in each step must be carried out outside the scope of the operation, i.e., by an external application. Thus, data transformation and annotation propagation do not interfere with each other.

Let us go back to our running example, and single out the classifyMapImage transformation operation that classifies an NDVI image generating a classified image. Figure 7.2 shows the association of annotations: M to the input data set (D), $\mathsf{M'}$ to the output data set ($\mathsf{D'}$), $\mathsf{M_i}$ to the input interface of the operation (I) and $\mathsf{M_o}$ to the output interface of the operation (O). The bottom of Figure 7.2 portrays the transformation. The input data set (D) is an NDVI GeoTIFF image. The operation has one input interface (I), which takes one parameter (p1), and one output interface (O), which produces one parameter (p2). The output data set ($\mathsf{D'}$) is the classified GeoTIFF image. These entities (data sets and interfaces) are described with semantic annotations, e.g., the pair $(\mathsf{O}, \mathsf{M_o})$ denotes that the semantic annotation $\mathsf{M_o}$ is associated with output interface O, similarly to (D, M), $(\mathsf{I}, \mathsf{M_i})$ and $(\mathsf{D'}, \mathsf{M'})$.

The top of the figure illustrates an ontology repository [31], a data space containing domain ontologies with the contextual and semantic information. The graphs in the boxes in the middle of the figure show how annotations are structured. M (at the left) is the annotation for the data set D: it is a graph rooted at idD (the ID of the entity being described) and each edge defines the scope of one annotation *unit*. Units are stored as RDF triples. Thus, the annotation for D is {(idD, RDFType, NDVI), (idD, hasEncoding, GeoTIFF-bitmap), (idD, capturedBy, Terra-MODIS) (idD, usedBand, band4),

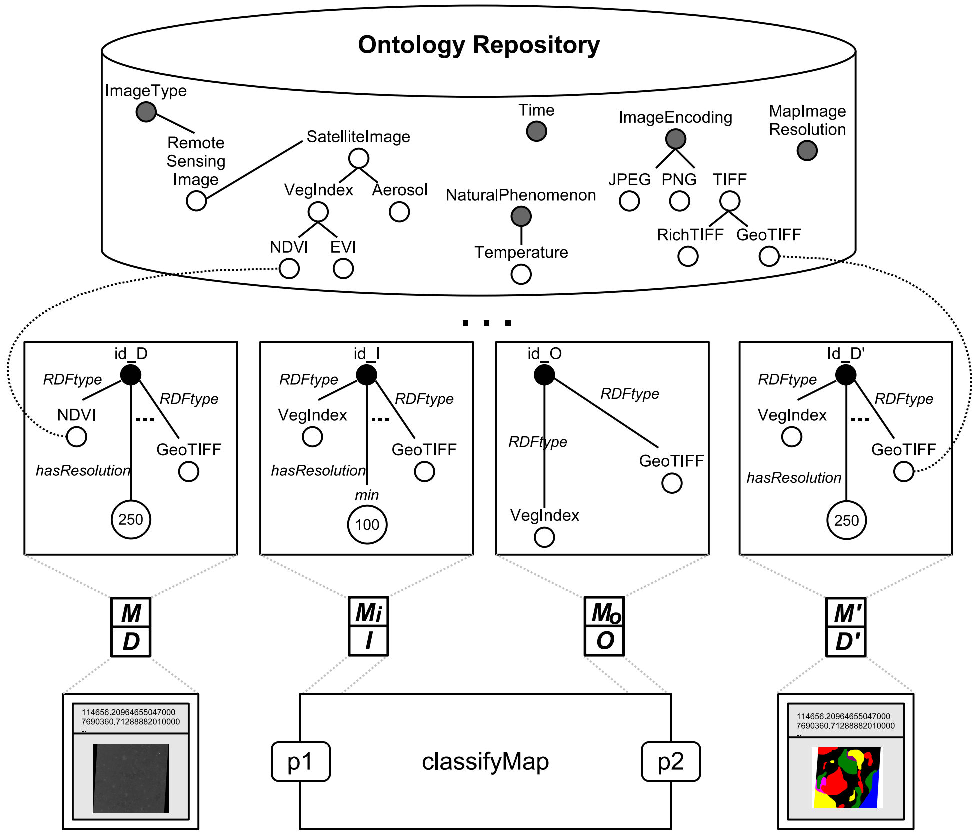


Figure 7.2: Data and interface annotations.

(idD, usedBand, band5), (idD, hasOrbit/Point, 220/075F), (idD, hasDatum, WGS84), (idD, hasProjection, UTM_ELLIPSOID), (idD, capturedOn, 20010323), ...}, indicating that it is an *NDVI image*, encoded in a *bitmap GeoTIFF*, captured by the *Terra-MODIS* satellite sensor, used *spectral bands 4* and *5*, and so on. By the same token, M_i says that I takes a parameter that should be a *Vegetation Index (VI) image*, encoded in a *bitmap GeoTIFF*, and so on; M_o indicates that O has as its output a *VI image*, encoded in a *bitmap GeoTiff*, and so on. A consistent combination of these annotations (M, M_i and M_o) should be used to generate the resuting propagated annotation (M').

7.4 Semantic Annotation Propagation

This section presents our solution for the annotation propagation problem. In this solution, any data transformation operation performed on a data set must be accompanied by transformations on the associated annotations. No assumptions are made about the format or granularity of the data. The annotation propagation mechanism should work equally for any kind of multimedia data. The presented mechanism for annotation propagation considers a basic data transformation process: a single operation with one data input and one data output.

7.4.1 Propagation Rules

Consider a transformation (T, I, O, D, D') with interface annotations $\mathsf{M_i}$ and $\mathsf{M_o}$ and let (D,M) and (D',M') be, respectively, its input and output data and annotations. The propagation problem can be simplified into the following issue: which mappings can be done between M and $\mathsf{M_o}$, meaning what were the effects of applying T to D with respect to its annotations?

We divided our solution this problem in two parts: (i) a set of abstract propagation rules to select pairs of annotations units for comparison; and, (ii) a set of ontological relations, each specifying how to compare a pair of annotation units and which annotation should be derived from the comparison. Our solution to the first part is a set of annotation propagation rules, presented next. Our solution to the second part is devised in Section 7.4.2. In this work, the annotation units are defined as RDF triples; however, other annotation units could also be used for the comparison, such as sub-paths, sub-trees or sub-graphs of the RDF graphs.

Figure 7.3 shows a high level view of our annotation propagation algorithm. Its input includes the semantic annotations M and $\mathsf{M_o}$, a set ($\Re$) of ontological relations ($\mathcal{R}_k$), such as the ones presented in Section 7.4.2. Its output is the resulting propagated semantic annotation (Δ).

```
1.  $\Delta \leftarrow \emptyset$
2.  foreach $\mathcal{R}_k$ in $\Re$ do
3.      $\mathcal{P}_k \leftarrow \emptyset$
4.      $\Omega \leftarrow \mathcal{F}_k(M)$
5.      $\Theta \leftarrow \mathcal{F}_k(M_o)$
6.      foreach $a$ in $\Omega$ do
7.          foreach $b$ in $\Theta$ do
8.              $\mathcal{P}_k \leftarrow \mathcal{P}_k \cup \mathcal{R}_k(a,b)$
9.      $\Delta \leftarrow \Delta \cup \mathcal{F}_k^{-1}(\mathcal{P}_k)$
10. return $\Delta$
```

Figure 7.3: Propagation Algorithm.

Before the propagation rules can be applied, it is necessary to retrieve the annotation units from the RDF graph. To be general, lets define a deconstruction function $\mathcal{F}$ to accomplish this. The result is a set of comparable units, which is represented by Ω (for M) and Θ (for $\mathsf{M_o}$), in lines 4 and 5, respectively. In this work this function simply singles out the RDF triples from the RDF graph, which are our basic comparison unit. Conversely, we also define a reconstruction function $\mathcal{F}^{-1}$ to recreate the RDF graph from the resulting unit(s) of the comparison of a pair of annotation units. In the algorithm, this is done in line 9, adding to the resulting semantic annotation Δ. The deconstruction and reconstruction steps are carried out for each relation, since each relation may be based on different comparison units. In this chapter, all relations are based on annotation units as defined in Section 7.3.1.

The propagation rules simply enforce a systematic selection of a pair of annotation units and the application of a comparison under a given ontological relation. This is done in lines 6-8 of the algorithm. These steps are repeated for all the selected ontological relations (i.e., all $\mathcal{R}_k$ in $\Re$). The results for all possible pairs is then returned as the result for that relation (the variable $\mathcal{P}_k$ on the algorithm). If the annotation units do not match under the ontological relation (line 8), the result of $\mathcal{R}_k(a, b)$ is empty. Intuitively, the rules recursively consider single comparable annotations units, checking the compatibility between all pairs of units and generating new units for the resulting set.

Therefore, to use the our propagation mechanism one must specify: (i) a set ($\Re$) of ontological relations ($\mathcal{R}_k$), each with a template result ($\mathcal{P}_k$) specifying the outcome of the comparison of two annotation units under this relation; (ii) a set of functions $\mathcal{F}_k$ to translate M into Ω and $\mathsf{M_o}$ into Θ; (iii) a respective inverse $\mathcal{F}_k^{-1}$ to translate $\mathcal{P}_k$ into Δ.

7.4.2 Ontological Relations

The ontological relations used in this chapter manipulate the basic elements from an OWL ontology, i.e., the classes, instances and properties. These elements are to be compared to determine which among them will be used as the derived annotation.

The choice of which ontological relations to use in a propagation should be guided by which aspects are useful in a given transformation process. For instance, if class hierarchy relationships are useful, generalization and specialization ontological relations should be used – e.g., considering descendant full compatibility or restricting to only direct subclass compatibility.

We categorize the ontological relations according to which element from the annotation units are the focus of the comparison. Three categories are defined: classes, instances and properties. Because of space limitation, only four ontological relations are listed here – see [82] for specifications of more ontological relations. The first three are on the classes category and the last one are on the instances category. It is possible to specify many other ontological relations to be used with our mechanism, specially when considering properties. For all ontological relations presented, the annotations units being compared (a and b) are RDF triples <*Subject,Predicate,Object*> and have the form $a =$ `<sa,pa,oa>` and $b =$ `<sb,pb,ob>`.

Class generalization. Relation based on the rdfs:subClassOf construct (defined as part of RDF Schema), which allows replacing general annotation units with more specific ones.

$$\mathcal{R}(a,b) = \begin{cases} \texttt{<sa,pb,ob>} \\ \qquad \text{if } \texttt{sa} \textit{ subClassOf } \texttt{sb} \\ \texttt{<sb,pb,sa>} \\ \qquad \text{if } \texttt{sa} \textit{ subClassOf } \texttt{ob} \end{cases}$$

This relation should be read as: given two annotation units $a =$ `<sa,pa,oa>` and $b =$ `<sb,pb,ob>`, generated by a deconstruction function for comparison under *subClassOf*, then the propagated annotation unit is the set composed by `<sa,pb,ob>`, if `sa` is a *subClassOf* `sb`. All other relations are to be read the same way.

Class specialization. This relation is also based on the rdfs:subClassOf construct, since generalization and specialization are both described by this ontological relation.

$$\mathcal{R}(a,b) = \begin{cases} \texttt{<sb,pa,oa>} \\ \qquad \text{if } \texttt{sb} \textit{ subClassOf } \texttt{sa} \\ \texttt{<ob,pa,oa>} \\ \qquad \text{if } \texttt{ob} \textit{ subClassOf } \texttt{sa} \end{cases}$$

Class complement. Relation based on the owl:complementOf construct, which represents a class with all properties that are not part of the original class. When two or more units are returned, the result is the conjunction of them.

$$\mathcal{R}(a,b) = \begin{cases} \texttt{<sa,pa,oa>,<sb,pb,ob>} \\ \qquad \text{if } \texttt{sa} \textit{ complementOf } \texttt{sb} \\ \texttt{<s,pa,oa>,<sb,pb,s>} \\ \qquad \text{if } \texttt{sa} \textit{ complementOf } \texttt{ob}\text{, where } s = sa \cup ob. \\ \texttt{<sa,pa,s>,<s,pb,ob>} \\ \qquad \text{if } \texttt{oa} \textit{ complementOf } \texttt{sb}\text{, where } s = sb \cup oa. \\ \texttt{<sa,pa,oa>,<sb,pb,ob>} \\ \qquad \text{if } \texttt{oa} \textit{ complementOf } \texttt{ob} \end{cases}$$

Instance equivalence. Relation based on the owl:sameAs, which states that two instances (represented by different URIs) are, in fact, the same.

$$\mathcal{R}(a,b) = \begin{cases} \texttt{<sa,pa,oa>} & \text{if } \texttt{sa} \textit{ sameAs } \texttt{sb} \\ \texttt{<sa,pa,oa>,<sb,pb,ob>} & \text{if } \texttt{sa} \textit{ sameAs } \texttt{ob} \\ \texttt{<sa,pa,oa>,<sb,pb,ob>} & \text{if } \texttt{oa} \textit{ sameAs } \texttt{sb} \\ \texttt{<sa,pa,oa>} & \text{if } \texttt{oa} \textit{ sameAs } \texttt{ob} \end{cases}$$

7.5 Related Work

This chapter is motivated by the increasing need in scientific applications to effectively manage multimedia data, which led to the proposal of a mechanism for annotation propagation. Related work involves therefore examples of multimedia data annotation and propagation proposals.

Metadata have been used in several contexts [3] including multimedia [57, 72]. Particularly, they have proven useful in establishing means of assigning descriptions to multimedia content, its execution and user interaction environments. The works of [11, 91] argue that descriptions of (i) multimedia content and (ii) users' preferences related to multimedia content are essential to achieve what they call *universal multimedia access* (UMA). In UMA, the multimedia content should be available anywhere and anytime, possibly using content adaptation to achieve this. This view can be generalised to context description, using metadata to describe the whole environment along with the content manipulated. However, there have been some difficulties in using metadata. The work of [102] analyses the trade-offs of using metadata in several scenarios, including appropriate uses where the environment is data-driven and/or requires analytical decision making; and less appropriate uses when considering more intuitive or politically charged environments. It also points out the importance of metadata quality and completeness in the successful use of metadata. The work of [18] discusses lack of motivation to go through the tedious process of creating metadata, discussing the case for MPEG-7 and its commercial motivation as a reason for its relative success. These works evidence the need for more automation on the generation of metadata, which is the main concern of our solution.

Other papers concern the adaptation of metadata to be used on the Web, often following standards of the Semantic Web [7, 104, 112]. The application of Semantic Web standards to describe multimedia content involves the production of semantic annotations and thus could profit from our annotation propagation solution. On another front, work with Semantic Web Services[4] provide the possibility of semantic annotations on interfaces of operations, thereby meeting a requirement of our solution (i.e., describing the

[4]e.g., SA-WSDL (`http://www.w3.org/2002/ws/sawsdl`).

interfaces of operations). The Multimedia Annotation Interoperability Framework [110] also considers the problem of describing transformation operations focusing on multimedia content.

One use of the term “annotation propagation” regard the automated update of annotations in a hierarchy (e.g., if a group receives an annotation, all entities from that group receive it as well; or if an annotation is corrected, all related annotations gets corrected as well). Our use of the term, however, regard data transformations and their impact on the annotations. To the best of our knowledge, there are two approaches in which the notion of “annotation propagation” is the same as ours. The first solution is presented in [12]. Their solution is placed in a data warehouse environment, with the annotations stored in extra (data) columns. The paper presents rules for propagating these annotations fields to answer queries. Rule implementation is based on pSQL, an extension of the SQL query language that supports a *propagate* clause to enable the application of propagation schemes. The solution of [12] is restricted to be used within databases and data warehousing environments, being limited by the query language. It only considers fine-grained annotations, in an item by item basis. Our solution, on the other hand can be applied to any kind of transformation and allows any granularity on the annotations, provided that both the transformations and the data are described with semantic annotations.

The second proposal [13] solves the annotation propagation problem taking advantage of schema mapping compositions. Their solution is restricted to relational algebra operations and is applicable to sequences of transformations modelled as workflows. Similar to [13], we also annotate content using ontologies, without restriction to database systems environments.

As far as we know, ours is the only solution that considers general data transformation operations. It is also the only one to take into account both previous data descriptions and operation interface description.

7.6 Concluding Remarks

This chapter presented a new approach to annotation propagation on multimedia data sets. A solution to the annotation propagation problem offers several advantages, such as: (i) lessening annotation efforts, (ii) decreasing the loss of information along the transformation process, (iii) documenting data origins (for traceability), and (iv) providing quality information. This work focused on the first two issues.

Our approach allows combining content-based metadata with contextual information provided by ontologies. Solutions to the annotation propagation problem have so far been restricted to database operations. Our definition showed how to extrapolate a solution to the problem that encompasses general transformation operations. Rather than having to consider the operations themselves, our propagation mechanism just needs to know the input/output interfaces of the operation. Our mechanism can also be extended by increasing the number of ontological relations. This allows tailoring annotation propagation behaviours, permitting new specific relationships to be considered.

As future work we intend to investigate less restrictive extensions to our propagation mechanism, also allowing, for instance, annotations that are not matched to be propagated. This can lead to richer annotations at the end of the propagation process. Dealing with multiple inputs/outputs, considering the annotation to an interface input and composition of operations are also aspects that we intend to explore further. Another possibility would be using RDF(S)/OWL inference rules or a reasoner instead of the propagation rules.

Capítulo 8

Conclusões

Esta tese combina diversos aspectos de pesquisa, incluindo interoperabilidade de dados e processos, descrição de dados, Componentes de Conteúdo Digital, workflows científicos, padrões Web, ontologias, e evolução de metadados, todos com foco em gerenciamento de dados de sensores. O resultado é um *framework* que permite a aquisição de dados a partir de fontes heterogêneas de dados de sensores, pré-processamento desses dados de uma forma flexível e orientada a aplicações, e mecanismos específicos de publicação, sempre mantendo a validade e corretude dos metadados associados.

8.1 Contribuições

Esta tese tem cinco contribuições principais:

1. A caracterização do cenário atual do gerenciamento de dados de sensores em, camadas de processamento e representação de dados, explicitando claramente as fronteiras para problemas e soluções nessa área (Capítulo 3);

2. O provimento de interfaces uniformes para acesso a dados de sensores, fontes de dados de sensores e funções de processamento aplicadas a sensores, através da extensão de uma tecnologia de componentes (DCCs). Essa tecnologia é usada como um *middleware* para gerenciamento de dados de sensores, e teve seus campos de contexto adaptados para descrever os conjuntos de dados de sensores encapsulados, permitindo maior precisão na seleção e transformação desses conjuntos de dados (Capítulo 4);

3. O uso e extensão de workflows científicos no gerenciamento de operações de transformação aplicadas a dados de sensores, considerando diferentes requisitos e classes desses workflows para tarefas específicas (Capítulo 5); e o uso desses workflows para

permitir às aplicações transformações mais precisas dos conjuntos de dados em uma consulta (Capítulo 6);

4. A combinação de padrões Web e metadados baseados em ontologias em um mesmo serviço de consulta, permitindo que aplicações que usem qualquer um dos dois mecanismos possam consultar conjuntos de dados de sensores com os mesmos resultados usando diferentes representações (Capítulo 6);

5. A proposta de um mecanismo para a atualização sistemática de metadados juntamente com as transformações nos dados, durante todo o seu ciclo de vida, do produtor (aquisição) até o consumidor (publicação) – Capítulo 7.

Estas contribuições abordam os seguintes aspectos: Cenário geral (Capítulo 3); Acesso (Capítulo 4); Gerenciamento (Capítulo 5); Publicação (Capítulo 6); e, Anotações (Capítluo 7).

8.2 Extensões

Há muitas extensões possíveis para essa tese, tanto ao *framework* quanto de implementação. Dentre elas, destacamos:

- **Programação/Calibração de Sensores.** O *framework* usando DCCs e seus metadados pode ser estendido para lidar não apenas com dados de sensores, mas também com o processo de programação e/ou calibração desses sensores, i.e., código e configuração, respectivamente. Para alcançar isso, duas abordagens podem ser adotadas: (a) uma representação comum para o código/configuração e componentes específicos transferência desse código/configuração para cada tipo de sensor; ou, (b) representações específicas e componentes comuns. Duas funcionalidades podem ser diretamente derivadas dessa extensão: (i) interfaces de alto nível para reprogramar/recalibrar sensores (ou redes de sensores); e, (ii) reprogramação/recalibração automatizada de sensores baseadas nos resultados de processos de monitoramento – e.g., um evento específico aciona um processo que substitui a programação ou configurações de um conjunto de sensores para adaptá-los novas condições do ambiente.

- **Combinação de Dados de Sensores e Dados Científicos.** Dados de sensores são frequentemente usados em pesquisa científica. Como mencionado brevemente no Capítulo 4, um passo lógico de extensão desta tese envolveria incluir suporte no *framework* para considerar fontes de dados científicos em domínios diferentes, combinando-os com os dados de sensores. Isso envolveria a identificação de características dessas fontes de dados e a construção de novas hierarquias de componentes especificamente para essas fontes.

- **Workflows e Linguagens de Consulta.** O Capítulo 6 apresentou a idéia de se incluir workflows científicos em serviços de consulta propostos pela OGC. Uma extensão poderia tratar de integrar especificações de workflows em uma linguagem de consulta (e.g., SQL ou SPARQL). Por exemplo, considere consultas SQL com a especificação de um workflow embutida, a serem executadas em um Sistema Gerenciador de Bancos de Dados (SGBD). Essas especificações poderiam ser tratadas por bibliotecas específicas que as processariam fora do SGBD (antes e/ou depois da execução da consulta). Alternativamente o SGBD poderia processar as consultas usando *stored procedures* com procedimentos para tratar a especificação de workflow e outros para tratar cada atividade individualmente.

- **Workflows dinâmicos.** Dada a natureza dinâmica de sensores e dados de sensores sobre fenômenos medidos, os próprios workflows gerenciando o processamento desses dados deveriam ser dinâmicos. Assim, uma direção interessante seria estudar mudanças dinâmicas de workflows que sejam sensíveis a mudanças de contexto, quer por novas necessidades de aplicações, quer por mudanças nas redes de sensores.

- **Mapeamentos entre Padrões e Ontologias.** A criação de mais mapeamentos entre padrões e ontologias (a exemplo do que foi proposto no Capítulo 6) possibilita a criação de componentes capazes de responder consultas que usem padrões ou ontologias para acesso a dados em vários outros domínios de aplicação. Isso pode levar à especificação de uma metodologia para a criação de tais mapeamentos, por exemplo, através da aplicação de técnicas de *matching* de esquemas e ontologias, favorecendo esforços de integração de dados científicos e de sensores.

- **Workflows e Propagação de Anotações.** O mecanismo de propagação de anotações proposto no Capítulo 7 está limitado a apenas uma operação de transformação por vez. Sequências de operações podem ter um impacto diferente nas anotações geradas no final do processo, i.e., a estrutura do processo pode influenciar as anotações a serem geradas. Situações nas quais isso pode ocorrer incluem repetição de um conjunto de operações sobre um conjunto de dados, e a utilização de um mesmo conjunto de dados em duas sequências diferentes e paralelas que se juntam em um passo posterior no final do processo. Como o mecanismo de propagação deveria se comportar nessas situações não está completamente claro. Algum avanço foi conseguido em [82], mas ainda existem problemas em aberto a serem considerados. Acreditamos que modelos de workflows podem ajudar no entendimento desses tipos de impacto.

- **Propagação de Anotações Sensível ao Conteúdo** O mecanismo de propagação de anotações proposto no Capítulo 7, considera apenas as anotações dos dados de entrada e das interfaces das operações para gerar novas anotações para os dados de saída. Entretanto, levar em consideração o conteúdo dos dados de saída poderia levar à geração de anotações diferentes, provavelmente mais detalhadas. Como exemplo, tamanhos ou esparsidades diferentes nos dados de saída poderia implicar em anotações diferentes. Mais que isso, funções específicas para avaliar cada tipo de dado de saída deve originar anotações bastante mais detalhadas.

- **Implementação e Integração.** A maioria dos trabalhos realizados necessita de um maior esforço de implementação. Em particular é preciso: (i) estudar formas mais eficientes de criar DCCs, já que seu extenso conjunto de metadados ainda deve ser criado manualmente; (ii) integrar um motor de execução de workflows completo para suportar especificações mais complexas de workflows científicos; (iii) integrar os módulos implementados para validar cada parte da proposta, pois apenas protótipos limitados foram implementados para testes. Além disso, o mecanismo de propagação de anotações precisa ser implementado e testado.

Referências Bibliográficas

[1] K. Aberer, M. Hauswirth, and A. Salehi. A Middleware for Fast and Flexible Sensor Network Deployment. In *Proceedings of the 32nd VLDB Conference (Demo Session)*, 2006.

[2] T. Aditya and M.-J. Kraak. Aim4GDI: Facilitating the Synthesis of GDI Resources through Mapping and Superimpositions of Metadata Summaries. *Geoinformatica*, 11(4):459–478, 2007.

[3] M. Agosti and N. Ferro. A Formal Model of Annotations of Digital Content. *ACM Transactions on Information Systems*, 26(1):3, 2007.

[4] H. Agrawal, G. Chafle, S. Goyal, S. Mittal, and S. Mukherjea. An Enhanced Extract-Transform-Load System for Migrating Data in Telecom Billing. In *IEEE 24th Int. Conf. on Data Engineering*, pages 1277–1286, 2008.

[5] G. Almes, J. Cummings, J. P. Birnholtz, I. Foster, T. Hey, and B. Spencer. CSCW and cyberinfrastructure: opportunities and challenges. In *Proc. of the 2004 ACM Conf. on Computer Supported Cooperative Work*, pages 270–273, 2004.

[6] G. Aloisio, D. Conte, C. Elefante, G. P. Marra, G. Mastrantonio, and G. Quarta. Globus Monitoring and Discovery Service and SensorML for Grid Sensor Networks. In *Proc. 15th IEEE WETICE'06*, 2006.

[7] R. Arndt, R. Troncy, S. Staab, L. Hardman, and M. Vacura. COMM: Designing a Well-Founded Multimedia Ontology for the Web. In *Proc. 6th Int. Semantic Web Conf. and 2nd Asian Semantic Web Conf.*, pages 30–43, 2007.

[8] H. Balakrishnan, M. Balazinska, D. Carney, U. Çetintemel, M. Cherniack, C. Convey, E. F. Galvez, J. Salz, M. Stonebraker, N. Tatbul, R. Tibbetts, and S. B. Zdonik. Retrospective on Aurora. *The VLDB Journal*, 13(4):370–383, 2004.

[9] M. Balazinska, A. Deshpande, M. J. Franklin, P. B. Gibbons, J. Gray, M. Hansen, M. Liebhold, S. Nath, A. Szalay, and V. Tao. Data Management in the Worldwide Sensor Web. *IEEE Pervasive Computing*, 6(2):30–40, 2007.

[10] R. Barr, J. C. Bicket, D. S. Dantas, B. Du, T. W. D. Kim, B. Zhou, and E. G. Sirer. On the need for system-level support for ad hoc and sensor networks. *ACM SIGOPS Operating Systems Review*, 36(2):1–5, 2002.

[11] P. Beek, J. R. Smith, T. Ebrahimi, T. Suzuki, and J. Askelof. Metadata-driven Multimedia Access. *IEEE Signal Processing Magazine*, 20(2):40–52, 2003.

[12] D. Bhagwat, L. Chiticariu, W. Tan, and G. Vijayvargiya. An annotation management system for relational databases. *The VLDB Journal*, 14(4):373–396, 2005.

[13] S. Bowers and B. Ludäscher. A Calculus for Propagating Semantic Annotations Through Scientific Workflow Queries. In *EDBT Workshops*, pages 712–723, 2006.

[14] F. Brabec and H. Samet. Client-Based Spatial Browsing on the World Wide Web. *IEEE Internet Computing*, 11(1):52–59, 2007.

[15] K. R. Braghetto, J. E. Ferreira, and C. Pu. Using control-flow patterns for specifying business processes in cooperative environments. In *SAC '07: Proceedings of the 2007 ACM symposium on Applied computing*, pages 1234–1241, 2007.

[16] M. E. Brown, J. E. Pinzon, K. Didan, J. T. Morisette, and C. J. Tucker. Evaluation of the consistency of long-term NDVI time series derived from AVHRR, SPOT-vegetation, SeaWiFS, MODIS, and Landsat ETM+ sensors. *IEEE Transactions on Geoscience and Remote Sensing*, 44(7):1787–1793, 2006.

[17] R. Bryant, R. Forester, and J. Hawkes. Filesystem Performance and Scalability in Linux 2.4.17. In *Proc. of the USENIX Annual Technical Conference (FREENIX Track)*, pages 259–274, 2002.

[18] D. C. A. Bulterman. Is It Time for a Moratorium on Metadata? *IEEE Multimedia*, 11(4):10–17, 2004.

[19] G. Cai. Contextualization of Geospatial Database Semantics for Human—GIS Interaction. *Geoinformatica*, 11(2):217–237, 2007.

[20] A. T. Campbell, S. B. Eisenman, N. D. Lane, E. Miluzzo, R. A. Peterson, H. Lu, X. Zheng, M. Musolesi, K. Fodor, and G.-S. Ahn. The Rise of People-Centric Sensing. *Internet Computing, IEEE*, 12(4):12–21, 2008.

[21] M. C. Cavalcanti, M. Mattoso, M. L. Campos, E. Simon, and F. Llirbat. An Architecture for Managing Distributed Scientific Resources. In *Proc. of the 14th Int. Conf. on Scientific and Statistical Database Management*, 2002.

[22] M. C. Cavalcanti, R. Targino, F. Baião, S. C. Rössle, P. M. Bisch, P. F. Pires, M. L. M. Campos, and M. Mattoso. Managing structural genomic workflows using Web services. *Data & Knowledge Engineering*, 53(1):45–74, 2005.

[23] L. H. Chambers, E. J. Alston, D. D. Diones, S. W. Moore, P. C. Oots, and C. S. Phelps. The MY NASA DATA Project: Tools for Knowledge Sharing and Discovery. In *Proc. of the 6th Annual NASA Earth Science Technology Conference*, 2006.

[24] X. Chu, T. Kobialka, B. Durnota, and R. Buyya. Open Sensor Web Architecture: Core Services. In *4th Int. Conf. on Intelligent Sensing and Information Processing*, pages 98–103, 2006.

[25] Open Geospatial Consortium. OpenGIS Implementation Specification for Geographic Information – Simple Feature Access. `http://www.opengeospatial.org/standards/sfa` (as of September 2008).

[26] P. Cornillon, J. Caron, T. Burk, and D. Holloway. Data access interoperability within IOOS. In *Proc. of MTS/IEEE OCEANS*, pages 1790–1792 Vol. 2, 2005.

[27] V. Crescenzi and G. Mecca. Automatic Information Extraction from Large Websites. *Journal of the ACM*, 51(5):731–779, 2004.

[28] T. Critchlow, K. Fidelis, M. Ganesh, R. Musick, and T. Slezak. DataFoundry: information management for scientific data. *IEEE Transactions on Information Technology in Biomedicine*, 4(1):52–57, 2000.

[29] D. Cuff, M. Hansen, and J. Kang. Urban sensing: out of the woods. *Communications of the ACM*, 51(3):24–33, 2008.

[30] Y. Cui and J. Widom. Lineage tracing for general data warehouse transformations. *The VLDB Journal*, 12(1):41–58, 2003.

[31] J. Daltio and C. B. Medeiros. Aondê: An Ontology Web Service for Interoperability across Biodiversity Applications. *Information Systems*, 33(7-8):724–753, 2008.

[32] J. Daltio, C. B. Medeiros, L. Gomes Jr, and T. M. Lewinsohn. A framework to process complex biodiversity queries. In *Proc. of the ACM Symposium on Applied Computing*, pages 2293–2297, 2008.

[33] DarwinCore (DwC) Biodiversity Information Standards (TDWG) working group. The Darwin Core Standard. `http://www.tdwg.org/activities/darwincore/` (as of Apr 2008).

[34] P. Eugster, P. A. Felber, R. Guerraoui, and A. Kermarrec. The many faces of publish/subscribe. *ACM Computing Surveys*, 35(2):114–131, 2003.

[35] R. T. Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine, 2000.

[36] R. Fileto, L. Liu, C. Pu, E. D. Assad, and C. B. Medeiros. POESIA: An Ontological Workflow Approach for Composing Web Services in Agriculture. *The VLDB Journal*, 12(4):352–367, 2003.

[37] D. Ganesan, D. Estrin, and J. Heidemann. Dimensions: why do we need a new data handling architecture for sensor networks? *ACM SIGCOMM Computer Communication Review*, 33(1):143–148, 2003.

[38] D. Gay, P. Levis, R. Behren, M. Welsh, E. Brewer, and D. Culler. The nesC language: A holistic approach to networked embedded systems. In *PLDI '03: Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation*, 2003.

[39] P. B. Gibbons, B. Karp, Y. Ke, S. Nath, and S. Seshan. IrisNet: An Architecture for a World-Wide Sensor Web. *IEEE Pervasive Computing*, 2(4), 2003.

[40] M. F. Goodchild, M. J. Egenhofer, R. Fegeas, and C. A. Kottman (editors). *Interoperating Geographic Information Systems*. Kluwer, 1999.

[41] L. Haas. Beauty and the beast: The theory and practice of information integration. In *Proc. of the 11th Int. Conf. on Database Theory*, pages 28–43, 2007.

[42] B. Habegger and M. Quafafou. Building Web Information Extraction Tasks. In *Proc. of the IEEE/WIC/ACM Int. Conf. on Web Intelligence*, pages 349–355, 2004.

[43] S. Hadim and N. Mohamed. Middleware Challenges and Approaches for Wireless Sensor Networks. *IEEE Distributed Systems Online*, 7(3), 2006.

[44] A. Y. Halevy, N. Ashish, D. Bitton, M. Carey, D. Draper, J. Pollock, A. Rosenthal, and V. Sikka. Enterprise information integration: successes, challenges and controversies. In *Proc. of the 24th ACM SIGMOD Int. Conf. on Management of Data*, pages 778–787, 2005.

[45] J. M. Hellerstein, W. Hong, and S. Madden. The Sensor Spectrum: Technology, Trends and Requirements. *SIGMOD Record*, 32(4):22–27, 2003.

[46] T. Hey and A. E. Trefethen. Cyberinfrastructure for e-Science. *Science*, 308:817–821, May 2005.

[47] D. Huynh, S. Mazzocchi, and D. Karger. Piggy Bank: Experience the Semantic Web inside your web browser. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(1):16–27, 2007.

[48] Iamnitchi, Ripeanu, and Foster. Locating Data in (Small-World?) Peer-to-Peer Scientific Collaborations. In *Proc. International Workshop on Peer-to-Peer Systems (IPTPS), LNCS*, 2002.

[49] INSPIRE. Infrastructure for spatial information in the europe. `http://www.ec-gis.org/inspire/` (as of September 2008).

[50] Institute of Computing, Agriculture Engineering Faculty, and Center for Research and Education in Agriculture – UNICAMP. The WebMAPS Project. `http://www.lis.ic.unicamp.br/projects/webmaps/` (as of Oct 2008).

[51] J. Shneidman and P. Pietzuch and J. Ledlie and M. Roussopoulos and M. Seltzer and M. Welsh. Hourglass: An Infrastructure for Connecting Sensor Networks and Applications. Technical report, Harvard University, 2004. TR-21-04.

[52] S.-G. Jang and T. J. Kim. Modeling an Interoperable Multimodal Travel Guide System using the ISO 19100 Series of International Standards. In *Proc. of the 14th Annual ACM Int. Symposium on Advances in Geographic Information Systems*, pages 115–122, 2006.

[53] M. Jarke, M. A. Jeusfeld, C. Quix, and P. Vassiliadis. Architecture and quality in data warehouses: An extended repository approach. *Information Sytems*, 24(3):229–253, 1999.

[54] K. Aberer and M. Hauswirth and A. Salehi. The Global Sensor Networks middleware for efficient and flexible deployment and interconnection of sensor networks. Technical report, Ecole Polytechnique Fédérale de Lausanne, 2006. LSIR-REPORT-2006-006.

[55] V. Kantere and T. Sellis. Handling spatial data in distributed environments. In *Proc. 15th Annual ACM Int. Symp. on Advances in Geographic Information Systems*, 2007.

[56] R. Kosala and H. Blockeel. Web mining research: a survey. *ACM SIGKDD Explorations Newsletter*, 2(1):1–15, 2000.

[57] H. Kosch, L. Boszormenyi, M. Doller, M. Libsie, P. Schojer, and A. Kofler. The Life Cycle of Multimedia Metadata. *IEEE Multimedia*, 12(1):80–86, 2005.

[58] L. Fu and L.-K. Soh and A. Samal. Techniques for Computing Fitness of Use (FoU) for Time Series Datasets with Applications in the Geospatial Domain. *Geoinformatica*, 12(1):91–115, 2008.

[59] A. H. F. Laender, B. A. Ribeiro-Neto, A. S. da Silva, and J. S. Teixeira. A brief survey of web data extraction tools. *SIGMOD Record*, 31(2):84–93, 2002.

[60] P. Levis and D. Culler. Maté: A Tiny Virtual Machine for Sensor Networks. In *Proc. 10th Int. Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS-X)*, pages 85–95, 2002.

[61] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. *Ambient Intelligence*, chapter TinyOS: An Operating System for Wireless Sensor Networks. Springer Verlag, 2004.

[62] M. S. Lew, N. Sebe, C. Djeraba, and R. Jain. Content-based multimedia information retrieval: State of the art and challenges. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 2(1):1–19, 2006.

[63] J. G. S. Lima, C. B. Medeiros, and E. D. Assad. Integration of heterogeneous pluviometric data for crop forecasts. In *Proc. of the 5th Brazilian Symposium on GeoInformatics*, 2003.

[64] T. Liu and M. Martonosi. Impala: A Middleware System for Managing Autonomic, Parallel Sensor Systems. In *Proc. ACM SIGPLAN Symp. Principles and Practice of Parallel Programming (PPoPP 03)*, 2003.

[65] C.-T. Lu, R. F. Santos Jr, L. N. Sripada, and Y. Kou. Advances in GML for Geospatial Applications. *Geoinformatica*, 11(1):131–157, 2007.

[66] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. A. Lee, J. Tao, and Y. Zhao. Scientific workflow management and the Kepler system. *Concurrency and Computation: Practice & Experience*, 18(10):1039–1065, 2006.

[67] S. R. Madden, M. J. Franklin, and J. M. Hellerstein. TinyDB: An Acquisitional Query Processing System for Sensor Networks. *ACM Transactions on Database Systems*, 30(1):122–173, 2005.

[68] L. E. Mariote, C. B. Medeiros, and I. Lee. Diagnosing Similarity of Oscillation Trends in Time Series. In *Proc. of the 7th IEEE Int. Conf. on Data Mining Workshops*, pages 643–648, 2007.

[69] C. B. Medeiros and A. C. Alencar. Data quality and interoperability in GIS (In Potuguese). In *Proc. of the 1st Brazilian Symposium on GeoInformatics*, 1999.

[70] C. B. Medeiros, J. Perez-Alcazar, L. Digiampietri, G. Z. Pastorello Jr, A. Santanchè, R. S. Torres, E. Madeira, and E. Bacarin. WOODSS and the Web: Annotating and Reusing Scientific Workflows. *SIGMOD Record*, 34(3):18–23, 2005.

[71] L. Meyer, J. Annis, M. Wilde, M. Mattoso, and I. Foster. Planning spatial workflows to optimize grid performance. In *SAC '06: Proceedings of the 2006 ACM symposium on Applied computing*, pages 786–790, 2006.

[72] Moving Picture Experts Group (MPEG) – ISO/IEC-JTC1. ISO/IEC15938 (parts 1–10) Multimedia content description interface. `http://www.iso.org/iso/search.htm?qt=15938&sort=rel&type=simple&published=true` (as of Sep 2008).

[73] NASA. MODIS Rapid Response System. `http://rapidfire.sci.gsfc.nasa.gov/` (as of September 2008).

[74] NSF/NCGIA/OpenGIS, editor. *Int. Conf. and Workshop on Interoperating Geographic Information Systems*, 1997. `http://www.ncgia.ucsb.edu/conf/interop97` (as of June 2008).

[75] OGC. Open Geospatial Consortium, Inc. `http://www.opengeospatial.org` (as of June 2008).

[76] OGC. OpenGIS Reference Model (ORM). `http://portal.opengeospatial.org/files/?artifact_id=3836` (as of September 2008).

[77] OGC. OpenGIS Sensor Web Enablement Architecture Document. `http://www.opengeospatial.org/pt/14140` (as of June 2008).

[78] OGC. OpenGIS Web Processing Service. `http://www.opengeospatial.org/standards/wps` (as of September 2008).

[79] OGC. Sensor Web Enablement Working Group. `http://www.opengeospatial.org/projects/groups/sensorweb` (as of Oct 2007), 2007.

[80] Open Geospatial Consortium, Inc. KML Implementation Standard. `http://www.opengeospatial.org/standards/kml/` (as of September 2008).

[81] S. Park, J.J. Feddema, and S.L. Egbert. Hydroclimatological parameters and their relationship with land surface temperatures (LST) and NDVI anomalies. In *Proc. 2002 ASPRS-ACSM Annual Conference*, 2002.

[82] G. Z. Pastorello Jr, J. Daltio, and C. B. Medeiros. An Annotation Propagation Mechanism for Multimedia Content. Technical Report IC-08-017, Institute of Computing, University of Campinas, August 2008.

[83] G. Z. Pastorello Jr, J. Daltio, and C. B. Medeiros. Multimedia Semantic Annotation Propagation. In *Proc. 1st IEEE Int. Workshop on Data Semantics for Multimedia Systems and Applications (DSMSA) – 10th IEEE Int. Symposium on Multimedia (ISM)*, pages 509–514, 2008.

[84] G. Z. Pastorello Jr, L. C. Gomes Jr, C. B. Medeiros, and A. Santanchè. Sensor Data Publication on the Web for Scientific Applications. In *Proc. 4th Int. Conf. on Web Information Systems and Technologies*, pages 137–142, 2008.

[85] G. Z. Pastorello Jr, C. B. Medeiros, S. M. Resende, and H. A. Rocha. Interoperability for GIS Document Management in Environmental Planning. *Journal on Data Semantics*, 3(LNCS 3534):100–124, 2005.

[86] G. Z. Pastorello Jr, C. B. Medeiros, and A. Santanchè. Applying Scientific Workflows to Manage Sensor Data. In *Proc. 1st e-Science Workshop – 22nd Brazilian Symposium on Databases*, pages 09–18, 2007.

[87] G. Z. Pastorello Jr, C. B. Medeiros, and A. Santanchè. Providing Homogeneous Access for Sensor Data Management. Technical Report IC-07-012, Institute of Computing, State University of Campinas, May 2007.

[88] G. Z. Pastorello Jr, C. B. Medeiros, and A. Santanchè. Accessing and Processing Sensing Data. In *Proc. 11th IEEE Int. Conf. on Computational Science and Engineering*, pages 353–360, 2008.

[89] 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*, 2008. Submitted for review.

[90] G. Z. Pastorello Jr, R. D. A. Senra, and C. B. Medeiros. Bridging the gap between geospatial resource providers and model developers. In *Proc. 16th ACM Int. Conf. on Advances in Geographic Information Systems*, pages 379–382, 2008.

[91] F. Pereira and I. Burnett. Universal Multimedia Experiences for Tomorrow. *IEEE Signal Processing Magazine*, 20(2):63–73, 2003.

[92] P.J. Pinter Jr, J.L. Hatfield, J.S. Schepers, E.M. Barnes, M.S. Moran, C.S.T. Daughtry, and D.R. Upchurch. Remote sensing for crop management. *Photogrammetric Engineering and Remote Sensing*, 69(6):647–664, 2003.

[93] R. Raskin and M. Pan. Semantic Web for Earth and Environmental Terminology (SWEET). In *Proc. of the Workshop on Semantic Web Technologies for Searching and Retrieving Scientific Data*, 2003.

[94] P. Rigaux, M. Scholl, and A. Voisard. *Spatial Databases – With Application to GIS*. Morgan Kaufmann/Elsevier, 2001.

[95] N. Ritter and M. Ruth. The GeoTiff data interchange standard for raster geographic images. *International Journal of Remote Sensing*, 18(7):1637–1647, 1997.

[96] P. A. Roberto, R. L. T. Santos, M. A. Gonçalves, and A. H. F. Laender. On RDBMS and Workflow Support for Componentized Digital Libraries. In *Proc. of the XXI SBBD*, pages 87–101, 2006.

[97] A. Santanchè and C. B. Medeiros. Self Describing Components: Searching for Digital Artifacts on the Web. In *Proc. of XX Brazilian Symposium on Databases*, pages 10–24, 2005.

[98] A. Santanchè and C. B. Medeiros. A Component Model and an Infrastructure for the Fluid Web. *IEEE Transactions on Knowledge and Data Engineering*, 19(2):324–341, 2007.

[99] A. Santanchè, C. B. Medeiros, and G. Z. Pastorello Jr. User-centered Multimedia Building Blocks. *Multimedia Systems Journal*, 12(4):403–421, 2007.

[100] ANSI Health Level Seven. Hl7 standards. `http://www.hl7.org/Library/standards.cfm` (as of Oct 2008).

[101] M. Sgroi, A. Wolisz, A. Sangiovanni-Vincentelli, and J. M. Rabaey. *Ambient Intelligence*, chapter A service-based universal application interface for ad hoc wireless sensor and actuator networks. Springer, 2005.

[102] G. Shankaranarayanan and A. Even. The Metadata Enigma. *Communications of the ACM*, 49(2):88–94, 2006.

[103] A. Simitsis, P. Vassiliadis, and T. Sellis. Optimizing ETL Processes in Data Warehouses. In *Proc. of the 21st Int. Conf. on Data Engineering*, pages 564–575, 2005.

[104] G. Stamou, J. Ossenbruggen, J. Z. Pan, G. Schreiber, and J. R. Smith. Multimedia Annotations on the Semantic Web. *IEEE Multimedia*, 13(1):86–90, 2006.

[105] S. Thakkar, C. A. Knoblock, and J. L. Ambite. Quality-Driven Geospatial Data Integration. In *Proc. of the 15th Annual ACM Int. Symposium on Advances in Geographic Information Systems*, 2007.

[106] R. S. Torres, C. B. Medeiros, M. A. Gonçalves, and E. A. Fox. A digital library framework for biodiversity information systems. *International Journal on Digital Libraries*, 6(1):3–17, 2006.

[107] M. J. Ungerer and M. F. Goodchild. Integrating spatial data analysis and GIS: a new implementation using the Component Object Model (COM). *International Journal of Geographical Information Science*, 16(1):41–53, 2002.

[108] W. J. D. van Leeuwen, A. R. Huete, and T. W. Laing. MODIS Vegetation Index Compositing Approach: A Prototype with AVHRR Data. *Remote Sensing of Environment*, 69(3):264–280, 1999.

[109] T. C. Vance, N. Merati, S. M. Mesick, C. W. Moore, and D. J. Wright. GeoModeler: tightly linking spatially-explicit models and data with a GIS for analysis and geovisualization. In *Proc. of the 15th Annual ACM Int. Symposium on Advances in Geographic Information Systems*, 2007.

[110] W3C Multimedia Semantics Incubator Group. Multimedia Annotation Interoperability Framework. `http://www.w3.org/2005/Incubator/mmsem/XGR-interoperability/` (as of September 2008).

[111] J. Wainer, M. Weske, G. Vossen, and C. B. Medeiros. Scientific Workflow Systems. In *Proc. of the NSF Workshop on Workflow and Process Automation Information Systems*, 1996.

[112] H. Wang, S. Liu, and L.-T. Chia. Image retrieval with a multi-modality ontology. *Multimedia Systems*, 13(5-6):379–390, 2008.

[113] WfMC – Workflow Management Coalition. The Workflow Reference Model. Technical report, Workflow Management Coalition, 1995. TC-1003.

[114] Y. Xue, W. Wan, Y. Li, J. Guang, L. Bai, Y. Wang, and J. Ai. Quantitative Retrieval of Geophysical Parameters Using Satellite Data. *Computer*, 41(4):33–40, 2008.

[115] Y. Yao and J. Gehrke. The Cougar Approach to In-Network Query Processing in Sensor Networks. *SIGMOD Record*, 31(3):9–18, 2002.

[116] J. Yu and R. Buyya. A taxonomy of scientific workflow systems for grid computing. *SIGMOD Record*, 34(3):44–49, 2005.

[117] Y. Zhao, M. Wilde, I. Foster, J. Vöeckler, J. Dobson, E. Gilbert, T. Jordan, and E. Quigg. Virtual data grid middleware services for data-intensive science. *Concurrency and Computation: Practice & Experience*, 18(6):595–608, 2006.