UNIVERSITY OF THESSALY
SCHOOL OF ENGINEERING
DEPARTMENT OF COMPUTER AND COMMUNICATION ENGINEERING

A SEMANTIC BASED APPROACH FOR KNOWLEDGE MANAGEMENT, DISCOVERY AND SERVICE COMPOSITION APPLIED TO 3D SCIENTIFIC OBJECTS

MARIOS PITIKAKIS

A dissertation submitted in fulfillment
of the requirements for the Degree of
Doctor of Philosophy

Volos, 2010

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ, ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ ΔΙΚΤΥΩΝ

ΜΙΑ ΣΗΜΑΣΙΟΛΟΓΙΚΗ ΠΡΟΣΕΓΓΙΣΗ ΓΙΑ ΤΗΝ ΔΙΑΧΕΙΡΙΣΗ ΚΑΙ ΑΝΑΖΗΤΗΣΗ ΓΝΩΣΗΣ ΚΑΙ ΤΗΝ ΔΗΜΙΟΥΡΓΙΑ ΥΠΗΡΕΣΙΩΝ ΕΦΑΡΜΟΣΜΕΝΗ ΣΕ ΤΡΙΣΔΙΑΣΤΑΤΑ ΕΠΙΣΤΗΜΟΝΙΚΑ ΑΝΤΙΚΕΙΜΕΝΑ

ΜΑΡΙΟΣ ΠΙΤΙΚΑΚΗΣ

Μία διατριβή που εκπονήθηκε
για τις απαιτήσεις απονομής
Διδακτορικού Διπλώματος Ειδίκευσης

Βόλος, 2010

A semantic based approach for knowledge management, discovery and service composition applied to 3D scientific objects

by

Marios Pitikakis

Department of Computer and Communication Engineering
University of Thessaly

Dissertation Approved:

Catherine Houstis, Professor, University of Thessaly
_____
Dissertation Supervisor

Emmanouil Vavalis, Associate Professor, University of Thessaly
_____
Committee Member

Spyros Lalis, Associate Professor, University of Thessaly
_____
Committee Member

Elias Houstis, Professor, University of Thessaly
_____
Committee Member

Christos Nikolaou, Professor, University of Crete
_____
Committee Member

Dimitris Plexousakis, Professor, University of Crete
_____
Committee Member

Leandros Tassiulas, Professor, University of Thessaly
_____
Committee Member

Abstract

Sharing resources over the Internet has become vital for scientific or business success. People are relying on the Web for knowledge and information reuse but also as a mean to facilitate communication and support of collaborative e-Science environments. There are two distinct streams of progress that have particularly well defined philosophies, communities and manifestations: the Semantic Web and Web Services.

The Semantic Web is an effort to extend the current web and to represent knowledge in a machine processable form, while Web Services are transforming the Web into a distributed device of data and computational resources. By merging both streams of progress we come closer to the goal of fostering the next generation of semantically-enabled systems and services, and collaborative working environments. Semantically enriched information and computational models can be effectively retrieved, shared, exploited, and used to construct new knowledge, thus changing the way knowledge and services are consumed and provided over the Web.

Knowledge technologies are a key element for achieving interoperability and integration at a semantic level by sharing rich conceptualizations of specific scientific domains. In recent years, 3D content has become widespread and has massive impact in many application areas like culture heritage, education, medicine and bioinformatics, entertainment, computer-aided design etc. 3D resources hold a high knowledge value, either due to the specialized skills and expertise needed to design and manipulate them or to the scientific information carried.

This thesis presents a knowledge-based approach to 3D resource management, discovery and service composition, by making explicit and sharable the knowledge embedded in 3D resources from a representational and implementational perspective. It also provides groundwork for the technical realization of an e-Science environment and demonstrates the potential of such a platform in terms of establishing a novel,

knowledge-based and semantically enriched problem solving environment dealing with scientific content.

The main contributions of this thesis are:

- to demonstrate the usefulness of knowledge management technologies to support the formal representation, retrieval and reuse of expert knowledge by developing an ontological framework for representing shape resources;
- to design and implement an ontology-driven knowledge base for storing, querying and reasoning with such machine processable information and for supporting a semantic searching framework;
- to prove that semantically enriched Web Services can improve the efficiency of knowledge-intensive and computationally-complex application areas like 3D processing;
- to propose a prototype e-Science architecture in terms of an operational, distributed and web based software system capable of supporting the conceptual search and retrieval of scientific resources, as well as the definition and execution of scientific workflows for shape processing.

Περίληψη

Ο διαμοιρασμός πόρων στο Διαδίκτυο έχει καταστεί ζωτικής σημασίας για την επιτυχία σε επιστημονικό ή επιχειρηματικό επίπεδο. Οι άνθρωποι στηρίζονται στον Παγκόσμιο Ιστό για να επαναχρησιμοποιήσουν γνώσεις και πληροφορίες, αλλά και για να επικοινωνήσουν και να συμμετάσχουν σε συνεργατικά περιβάλλοντα e-Science. Υπάρχουν δύο διακριτά ρεύματα προόδου με καλά καθορισμένες φιλοσοφίες, κοινότητες και εκδηλώσεις: ο Σημασιολογικός Ιστός (Semantic Web) και οι Διαδικτυακές Υπηρεσίες (Web Services).

Ο Σημασιολογικός Ιστός είναι μια προσπάθεια να επεκταθεί ο Παγκόσμιος Ιστός και να αναπαρασταθεί η γνώση σε μια μορφή επεξεργάσιμη από τον υπολογιστή, ενώ οι Διαδικτυακές Υπηρεσίες μετατρέπουν τον Παγκόσμιο Ιστό σε μια κατανεμημένη συσκευή από δεδομένα και υπολογιστικούς πόρους. Με την σύγκλιση των δύο ρευμάτων προόδου, ερχόμαστε πιο κοντά στο στόχο της προώθησης σημασιολογικά ενεργοποιημένων συστημάτων και υπηρεσιών, και συνεργατικών εργαλείων νέας γενιάς. Η ανάκτηση, ο διαμοιρασμός και η εκμετάλλευση των σημασιολογικά εμπλουτισμένων δεδομένων και υπολογιστικών μοντέλων μπορούν να επιτευχθούν πιο αποτελεσματικά και να χρησιμοποιηθούν για την δημιουργία νέας γνώσης, αλλάζοντας έτσι τον τρόπο που παρέχουμε και εκμεταλλευόμαστε γνώσεις και υπηρεσίες μέσα από τον Ιστό.

Οι τεχνολογίες Γνώσης αποτελούν ένα στοιχείο-κλειδί για την επίτευξη διαλειτουργικότητας και ενοποίησης σε σημασιολογικό επίπεδο, διαμοιράζοντας μια πλούσια εννοιολογική αποτύπωση κάποιου συγκεκριμένου επιστημονικού πεδίου. Τα τελευταία χρόνια, η χρήση τρισδιάστατου περιεχομένου είναι πολύ διαδεδομένη και βρίσκει σημαντικές εφαρμογές σε πολλά επιστημονικά πεδία όπως στην πολιτιστική κληρονομιά, την εκπαίδευση, την ιατρική και βιο-πληροφορική, τη διασκέδαση, τον τεχνικό σχεδιασμό μέσω υπολογιστή κλπ. Τα τρισδιάστατα αντικείμενα και οι σχετικοί με αυτά πόροι έχουν μεγάλη γνωστική αξία, είτε λόγω των ειδικών δεξιοτήτων και

γνώσεων που χρειάζονται για το σχεδιασμό και τον χειρισμό τους είτε λόγω του όγκου των επιστημονικών πληροφοριών που εμπεριέχουν.

Η διατριβή αυτή παρουσιάζει μια γνωσιολογική προσέγγιση για την διαχείριση, αναζήτηση και δημιουργία υπηρεσιών σχετικά με τρισδιάστατα αντικείμενα και υπολογιστικούς πόρους. Με αυτό τον τρόπο αναπαρίσταται η γνώση που εμπεριέχεται σε τρισδιάστατους πόρους και υλοποιείται ένα πλαίσιο στο οποίο η γνώση αυτή γίνεται συγκεκριμένη και εύκολα διαμοιράσιμη. Επίσης παρέχει τα θεμέλια για μια τεχνική υλοποίηση ενός περιβάλλοντος e-Science και επιδεικνύει τις δυνατότητες μιας τέτοιας πλατφόρμας θεσπίζοντας ένα καινοτόμο, βασισμένο στη γνώση και σημασιολογικά εμπλουτισμένο περιβάλλον λύσης προβλημάτων επιστημονικού περιεχομένου.

Οι κύριες συνεισφορές αυτής της διατριβής είναι:

- να επιδείξει την χρησιμότητα των τεχνολογιών διαχείρισης γνώσης για την υποστήριξη μιας τυπικής αναπαράστασης, ανάκτησης και επαναχρησιμοποίησης εξειδικευμένης γνώσης αναπτύσσοντας ένα οντολογικό πλαίσιο για την αναπαράσταση τρισδιάστατων πόρων
- να σχεδιάσει και να υλοποιήσει μια βάση γνώσης, βασισμένη σε οντολογίες, για την αποθήκευση, επερώτηση και λογική αναζήτηση πληροφοριών επεξεργάσιμων από τον υπολογιστή και να υποστηρίξει ένα σημασιολογικό πλαίσιο αναζήτησης
- να αποδείξει ότι οι σημασιολογικά εμπλουτισμένες Διαδικτυακές Υπηρεσίες μπορούν να βελτιώσουν την αποδοτικότητα των εφαρμογών που έχουν να κάνουν με γνωστικά και υπολογιστικά πολύπλοκους τρισδιάστατους επιστημονικούς υπολογισμούς
- να προτείνει μια πρωτότυπη αρχιτεκτονική e-Science από την άποψη ενός λειτουργικού, κατανεμημένου και βασισμένου στον Ιστό λογισμικού συστήματος το οποίο μπορεί να υποστηρίξει την εννοιολογική αναζήτηση και ανάκτηση επιστημονικών πόρων, όπως επίσης και τον ορισμό και εκτέλεση ροών εργασίας που σχετίζονται με επιστημονικούς υπολογισμούς τρισδιάστατων μοντέλων.

Acknowledgements

PhD theses, although presented as the work of one person, are almost always the result of close cooperation between a group of people. This particular thesis is no exception to that rule. I will use this space to thank a number of people who have helped make this thesis possible.

I would like to express my sincere thanks to my advisor Catherine Houstis and Manolis Vavalis for their full support and collaboration during my studies. I also thank Spyros Lalis for his insight and critical comments. I am very fortunate to have had the opportunity to work with them for almost ten years now. They were real teachers and their guidance and patience are greatly appreciated.

I would like also to express my gratitude to professors Elias Houstis, Christos Nikolaou, Dimitris Plexousakis and Leandros Tassiulas for serving on my thesis committee and for their helpful comments.

I am grateful to all my co-authors Bianca Falcidieno, Michela Spanguolo, Laura Papaleo, Chiara Catalano, Ricardo Albertoni, Francesco Robbiano, Alexandra Garcia-Rojas, Catherine Houstis, Manolis Vavalis, Spyros Lalis and George Vasilakis for our productive and enjoyable collaboration. We worked together on many projects and papers, of which many ideas are found in this thesis.

This work has been partially supported by the EU projects AIM@SHAPE, FOCUS K3D and ARION. I wish to thank all the project partners who have stimulated my research activities.

I owe special thanks to my officemate and friend George Vasilakis for the countless hours we spent working together, discussing research issues and other matters of life.

I also wish to express my appreciation to all of my fellow PhD students and friends who made the time I spent at Volos more fun and interesting.

Last, but not least, I would like to dedicate this thesis to my family, my wife Katerina and daughters Vasia and Stella. Their presence, at happy and hard times, and invaluable moral support were instrumental to the conduct of my research. I thank them for all their patience and understanding. I would also like to thank my parents and brother for their constant love and support.

Contents

List of Figures

List of Tables

1 INTRODUCTION

Sharing resources over the Internet has many notions and objectives: the Web is a form of resource sharing, but so are Web Services, the Grid, global computing, P2P, or social networks, to name just a few. In all these systems one important functionality is resource discovery. However, it has different meanings and requirements depending on the particular resource sharing environment, application domain or target community, and therefore different objectives and constraints.

Scientists rely on the web to share scientific resources (software tools, algorithms, papers, data resources) but also as a mean to facilitate communication. Nevertheless, current web technology is insufficient to support collaborative e-science environments and a more elaborate framework is needed.

We can identify two distinct streams of progress towards such frameworks that have particularly well defined philosophies, communities and manifestations: the Semantic Web and Web Services. The Semantic Web is an effort to extend the current web and to represent knowledge in a machine processable form which explicitly captures the meaning of the presented information and, subsequently, allows improvements of current functionalities provided on the web.

The importance of Web Services has been recognized and accepted by both the industry and academia. The Web is now evolving into a distributed device of computation from a collection of information, data and computational resources.

The next generation of the Web promises to deliver semantically enriched Web Services by annotating them with large amounts of semantic metadata *glue,* so that they can be utilized by software entities like application agents or other services with minimal or no human intervention at all.

By merging both streams of progress we come closer to the goal of fostering the next generation of semantically-enabled systems and services, and collaborative working environments as well as problem solving environments. Semantically enriched

information and computational models can be effectively retrieved, shared, exploited, and used to construct new knowledge, thus changing the way knowledge and services are consumed and provided.

This dissertation focuses on developing an open architecture for formalizing, processing, managing and sharing knowledge about scientific objects in general and 3D digital shapes and applications in particular, and supporting the semantic search and retrieval of 3D content. It also proposed the definition and execution of semantically enriched Web Services for 3D shape processing. This utilization of services that are semantically annotated enables automated discovery, composition and dynamic invocation and is aiming at changing the way knowledge and services are consumed and provided.

It is important to emphasize that although the range and scope of our approach is applied to 3D shape resources, it is generic enough and can be employed by a variety of application domains and scientific objects and services.

In the remainder of this chapter we introduce the motivation of this work, discuss the characteristics and identify the requirements of our case study (i.e. 3D shape resources), present the research context and our approaches, and give an outline of the rest of this dissertation.

1.1 Motivation

3D content is widely recognized as the upcoming wave of digital media and the success of 3D environments and applications reveal that there is a shift in the way people see and navigate the Internet. The massive impact of 3D content can be already observed in application domains like entertainment, computer-aided design and manufacture, while new application areas like culture, education, medicine and bioinformatics, as well as everyday life are emerging. Examples are provided by virtual games and consoles where 3D models are used and manipulated in order to create convincing virtual worlds or natural user interfaces (e.g. Microsoft Kinect for XBOX 360). Users are becoming more

and more actively involved in content creation and they are continuously demanding tools that are effective and can intuitively create, share, retrieve and reuse 3D content.

Rapid technological evolution and at the same time emerging needs raise new challenges, in a variety of communities, associated with the management and access of the amounts of information carried by 3D content. In addition, knowledge and information reuse are vital components in order to maintain the competitive edge and ensure scientific or business success. 3D models are being used increasingly as a basis of new activities and significant time is spent searching for the appropriate information or creating 3D models from scratch. To address the emergence of these newly created demands, the term *semantic multimedia* have been proposed as the new paradigm that encapsulates the convergence of multimedia and knowledge technologies and is also regarded as the evolution of traditional multimedia. Semantic multimedia can make it possible to efficiently use and reuse, share and access digital content in distributed and networked environments.

Thanks to technological advances, we have plenty of tools for visualizing, streaming and interacting with 3D objects, even in unspecialized web contexts (e.g. SecondLife). Conversely, tools for coding, extracting and sharing the semantic content of 3D media are still far from being satisfactory.

Scientists communicate by the use of models (e.g. physical, mathematical, organizational etc.) and models expose semantics. 3D models in particular, hold a high knowledge value, either due to the specialized skills and expertise needed to design and manipulate them or due to the scientific information carried. There are different kinds of knowledge:

- knowledge related to the geometrical and visual aspects which can be captured by a set of geometric and graphical data representing the digital object;
- knowledge related to the purpose/role of the object represented which defines its category or functionality;

- knowledge related to the application domain and the way the 3D data are represented, processed and interpreted.

In addition, creating a 3D model takes much time and effort and it would therefore be greatly advantageous to be able to reuse and adapt existing 3D models instead of creating new ones from scratch. This has motivated the development of 3D shape retrieval mechanisms that are based mainly on 3D shape matching. That is, given a user-defined shape object, a search mechanism is able to retrieve shapes that are considered to be similar enough, based on some matching criteria or algorithms. These content-based 3D shape retrieval methods usually perform much better than simple text-based solutions mainly because 3D models are poorly annotated (typically minimal metadata are provided with models, mostly for classification purposes, and in many cases not at all - even the model names are often meaningless). In some cases, where the object has a well defined meaning, a text-based search can be very effective but generally conventional text-based queries are too limited and ambiguous to be used for retrieving 3D objects.

However, even content-based search methods are characterized by relatively low precision since they are mainly based on geometrical matching and provide no consideration regarding the semantics of the object to be retrieved. Semantic Web technologies can provide the means for making the shift towards a semantically enabled representation of digital shapes. This is not only expected to improve current content-based methods for retrieving 3D objects on the Web, but also to put an entirely new perspective on the process of modeling digital shapes.

Dealing with the semantic representation of shapes is a relatively new research area. There is no concrete methodology to use in order to achieve such representations and the complexity involved for this type of scientific objects is high. Information encoded in media representing shapes is commonly implicit, based on data formats that have no relation with data interpretation and offer no grasp to their direct access and easy understanding. In contrast, through semantics, this information can be captured and be made explicit and can be used as a key to access the underlying knowledge.

Capturing the knowledge that a shape encompasses requires information concerning its meaning, its usage, its purpose, apart from its geometric and/or structural representation. It is also important that this semantic mark-up is formalized in order to not only be understood by humans but also to be machine-processable. This is crucial in realizing the vision of developing intelligent agents and programs able to interoperate and access knowledge bases, dealing with multi-dimensional objects in the same way as with any other type of information in the Semantic Web today.

Knowledge technologies are essential for achieving interoperability and integration at a semantic level: while current information technology offers plenty of tools to produce and share digital content, knowledge technologies make it also possible to share rich conceptualizations of specific scientific domains. Interoperability and semantic integration may boost new scientific communities and new approaches to the access and sharing of scientific resources. Knowledge Management (KM) has been instrumental in shifting the focus of attention to the value of knowledge in collaborative working environments. They can be utilized to

- support the representation, retrieval and reuse of expert knowledge;
- provide for the semantic interoperability of the tools used in the various stages of the lifecycle of scientific objects;
- prove that semantically enriched Web Services can improve the efficiency of knowledge-intensive and computationally-complex application areas (and the 3D digital shape processing pipeline in particular).

As pointed out in [Hendler 2003], tools are needed to support what the working scientists and researchers do in their day to day activities. Such tools must include ways that make it easier for the scientist e.g. to formulate hypotheses and design and run experiments to test them, to collect and analyze experimental data, to find possible connections amongst the data and interpret the data, to communicate with other researchers and to write scientific articles.

Such scientific activities often involve constructing a stream/chain of tasks to perform a series of computations. In the service oriented computing paradigm, this

process involves discovering appropriate and meaningful services and composing them into a scientific workflow. Scientific computing traditionally relies on software packages, each of them developed by research teams over long periods of time. It is of paramount importance to enable the sharing and the combination of these software packages and the reuse of the huge amount of work they embody. Applications dealing with shapes and shape processing workflows usually involve very complicated steps, ranging from time consuming numerical computations to the gathering of the necessary resources and/or acquiring the knowledge to perform a specific step. Resources (either shape models or software tools) may also come from different organizations and most probably with minimal documentation or described with different terminology. There are a number of issues that could address the needs of such applications:

- to be able to access and share in a simple way the available resources (i.e. models, tools, benchmarks, algorithms, services, publications etc.) and have an up-to-date documentation of them;
- to provide semantically-enriched representation, management and retrieval of scientific content and expert knowledge;
- to provide additional support in discovering services (i.e. a semantic-driven service discovery), since currently nearly all of this work is done manually which seriously obstructs scalability;
- to be able to provide the means for configuration and composition of workflows from task definitions that substantially rely on semantic driven descriptions of data, programs/tools and services, and on semantic reasoning on process knowledge (e.g. defining tasks to be performed instead of traditional querying for information);
- to be able to annotate (possibly automatically) the available resources and to capture all or a relevant subset of information throughout the processing chain and the input/output it makes use of at all stages of the process.

For shape computations the knowledge required to select available services and coordinate them into a workflow added value composite service is usually specific to the application domain. It is often the case that resource selection cannot be specified in advance of the execution of a complicated processing chain specification. The lack of machine readable semantics necessitates human intervention for service discovery and composition. Predefined service sequencing and binding is not sufficient, thus obstructing their usage in complex scientific computing contexts.

We believe that Semantic Web technologies and Web Services can successfully be combined in order to construct semantically enriched, coarse-grained components that can be integrated to form an e-Science environment for collaborating and/or problem solving services. Such combination will provide the unifying infrastructure that is necessary for building, through workflows, more complex, yet more flexible services, that are self describing, and can be used in diverse, potentially heterogeneous, application environments.

Currently, most of the semantic information and knowledge about 3D shape object and processing workflows is implicit, existing only inside the domain experts' minds. This results to poor communication and exploitation of expert knowledge and forfeits the potential for sharing and re-use of this knowledge in a much more efficient way. Capturing and formally representing expert knowledge will make it machine accessible and processable resulting in automating the process and improving efficiency and manageability through an e-Science environment. Furthermore, it will allow for more cost-effective solutions for complex shape related processes, thus, achieving higher levels of productivity.

1.2 Research Context and Positioning

Automatic classification of 3D databases, 3D content annotation and content-based search and retrieval have raised many new research directions that currently represent some of the key research topics in Computer Graphics and Vision. At the same time,

knowledge technologies, such as structured metadata, ontologies, knowledge bases and reasoners, have proven to be extremely useful to support a stable and standardized approach to resource sharing, and the exploitation of these technologies for 3D content and knowledge intensive scenarios is still at its infancy.

The success of semantic 3D media largely depends on the ability of advanced systems to provide efficient and effective search capabilities, intuitive reuse and creation facilities, concerning the content, the semantics, and the context.

In this work, we address the notion of making explicit and sharable the knowledge embedded in 3D shape resources from both a representation and implementation perspective, and provide groundwork for the technical realization of a next generation e-Science environment.

The aim of this work is threefold and can be further analyzed into the following objectives:

1. Knowledge encoded in text data is explicit, being based on natural language, which can be, and effectively is, used as a key to access information. In contrast, information encoded in multi-dimensional media is implicit. The knowledge obtained by simple metadata annotation of 3D models is not sufficient and must be represented efficiently into an appropriate ontology structure suitable for the Semantic Web. Due to the complexity of the field, different domain ontologies have to be designed. The aim is to address multiple contexts and applications in a semantic-aware level of representation where the shape knowledge can be shared and exploited.

2. Resource discovery (either searching for shape resources or scientific objects in general) is a complicated process and depends on specific domain knowledge and context. Even within the area of Shape Modeling several distinct research fields exist (e.g. Computer Graphics, Computer Vision and Geometric Modeling but it is also based on a large spectrum of other fundamental domains, including differential geometry, numerical analysis, computational geometry and discrete topology) and the semantics of shapes are treated differently in each of these areas. To be able to search for shapes, different aspects of the object in question must be considered. For example, a search

could be conducted presenting as criteria the geometric aspect of the shape, its structure or its semantics. This heavily depends on the application domain, the scientific area and the usage scenario the resource is expected to take part in. This greatly complicates the representation of knowledge related to shapes but what is even more important is that this also complicates the process of searching. Searching by shape similarity (content-based retrieval) or even by using text-based methods (context-based retrieval) can be at times quite effective for 3D shapes, but the use of semantic information could further enrich the retrieval process and improve search precision and recall.

3. Embedding semantics and domain knowledge in 3D-intensive applications can highly improve the 3D processing pipeline and allow a concrete re-use of valuable resources (e.g., existing content, processing tools/services, workflows). By creating a platform that integrates, combines, adapts, and enhances existing services with semantic information, we contribute to efficient resource discovery and interoperability and reusing expert knowledge captured in the different stages of processing. These services can in turn be chained to form corresponding workflows. This can provide the unifying infrastructure that is necessary for building more complex, yet more flexible, services, that are self describing, and can be used in diverse, potentially heterogeneous, application environments.

In this context, our aim is to advance research in the direction of semantic-based shape representations and semantic-oriented services to acquire, build and process shapes with their associated knowledge. We foresee a generation of 3D resources in which knowledge is explicitly represented and, therefore, can be retrieved, processed, shared, and exploited.

The main contribution of this dissertation is the development of a technical framework for the formal representation, management, querying of, and inferencing with, knowledge about 3D shape resources and their processing pipeline (scientific computations). To fulfill the above objectives, several more specific contributions can be identified:

- Demonstrate the usefulness of knowledge management technologies to support the representation, retrieval and reuse of expert knowledge; more specifically to develop an approach to formalize shape knowledge and to contribute to the definition of an ontological framework for representing shapes.
- Provide semantic interoperability in the various stages of the knowledge lifecycle of shape models (and scientific objects in general) by developing common and domain specific ontologies and metadata.
- Design and implement an ontology-driven knowledge base for storing, querying and reasoning with such machine processable information and for supporting a semantic searching framework.
- Prove that semantically enriched Web Services can improve the discovery and composition of scientific workflows in knowledge-intensive and computationally-complex application areas. 3D resources are used as a test bed for verifying the validity of our approach and for suggesting new directions to investigate.
- Propose a flexible prototype service oriented e-Science architecture in terms of an operational, distributed and web based software system capable of supporting the conceptual discovery of shape resources (and scientific resources in general), as well as the composition and execution of scientific workflows for shape processing.

We believe that the success of the Semantic Web evolution depends to a great extent on proving its scalability from simple application scenarios to knowledge-intensive and computationally-complex ones. In this sense, our objective is to demonstrate how formalizing, sharing and re-using expert knowledge will effectively improve scientific computing in general and shape processing in particular.

Therefore, we invite the reader to note that although the above contributions are made in the context of 3D resources and shape modeling, they can be applied to different types of scientific objects and applications.

In the next section, we will give a detailed outline of the dissertation that clarifies the relation between the chosen structure, the goals and contributions mentioned above.

1.3 Outline of the Dissertation

The remainder of this thesis is organized as follows:

Chapter 2 presents the necessary background and related work required for the rest of the thesis. In particular, we introduce the fields of Knowledge Management, Semantic Web and Web Services. We also discuss about ontology languages, efforts for developing multimedia ontologies, knowledge base systems and implementations, Web Services standards and Semantic Web Services approaches and practices.

In Chapter 3, we present our ontology development methodology, propose an ontology-based conceptualization of the domain knowledge and describe the structure and motivation behind the context-dependent ontologies developed. We also present the overall architecture of the Digital Shape Workbench (DSW), a unified platform for modelling, storing, processing and reasoning about shape resources. This chapter is concluded by a detailed description of the design and implementation of the Ontology & Metadata Repository (OMR), the knowledge base back-end of the DSW, and its role in the development of the semantic search mechanism.

Chapter 4 is dedicated to our service-oriented approach towards an e-Science collaborative environment. After discussing the benefits of semantically enriched services we address the issue of dynamic service composition and execution. Finally we prove the generality of this framework by demonstrating some prototype services and workflows as a proof of concept.

In Chapter 5 we evaluate the scalability, performance and extensibility of the proposed framework by using both quantitative and qualitative methodologies. The qualitative evaluation focuses mainly on ontology development and the criteria used in this process. Next, we review the results of a questionnaire survey that reveals the current

situation about the use of 3D resources, identifies possible gaps and new areas of research, and validates our efforts in developing the DSW.

The main contributions of this work are summarized in Chapter 6 and we conclude with a discussion on future research directions.

A short list of important definitions regarding 3D Modeling terminology is given in Appendix A and a list of publications written by the author of this dissertation is provided in Appendix B.

2 BACKGROUND AND RELATED WORK

This chapter provides the necessary background to understand the rest of this thesis by introducing the fields of Knowledge Management, Semantic Web, Web Services and other related areas.

We start by discussing some aspects of knowledge representation and ontology development (Section 2.1). We present the vision of the Semantic Web, the characteristics of ontology languages and the main model and technologies proposed for allowing machines to understand the meaning of information (Section 2.2). As a part of the discussion, we briefly present some characteristic examples of the expressiveness of the OWL language.

Next, in Section 2.3, we introduce several related efforts towards semantic description of multimedia objects and multimedia ontologies and in Section 2.4 we review the most important implementations of Knowledge Base Systems. Section 2.5 presents content-based search and retrieval methods for 3D models and introduces the notion of context-based search methods.

Finally, we present several definitions of Web Services with the goal of providing a clear understanding of what a Web Service is and we discuss the current Web Service standards (Section 2.6). Certain shortcomings associated with these Web Service standards serve as a motivation for Semantic Web Services, which we introduce in Section 2.7, where we also review common emerging Semantic Web Services approaches.

The development of the infrastructure and the services considered in Chapter 3 and Chapter 4, are mainly based on the topics discussed in this chapter.

2.1 Ontology-based Knowledge Representation

Knowledge representation and management is concerned with acquiring, accessing and maintaining knowledge within an organization. Knowledge representation within the Semantic Web vision deals with the modelling and sharing of the meaning of terms in a given domain. Achieving such a common understanding is accomplished by agreeing on an appropriate way to conceptualize the domain. Ontologies represent a key ingredient in knowledge management and content-based systems. Designing an ontology actually means to determine the set of semantic categories which properly reflect the particular conceptual organization of the domain of information on which the system must operate, thus optimising the quantity and quality of information retrieval.

The most commonly cited definition of ontology is: "*An ontology is a formal, explicit specification of a shared conceptualisation*" [Gruber 1993]. In this usage, an ontology is a set of concepts - such as things, events, and relations - that are specified in some way in order to create an agreed-upon vocabulary for exchanging information.

An ontology is therefore more than a taxonomy or classification of terms. Although a taxonomy contributes to the semantics of a term, ontologies include richer relationships between terms. It is these rich relationships that enable the expression of domain-specific knowledge.

The design of a proper ontology appears as a crucial factor in many tasks of knowledge organization and structuring. Formally speaking, an ontology is a *structured system of categories* or *semantic types*, so that knowledge about a certain domain can be organized through the categorization of the entities of the domain in terms of the types in the ontology. Therefore, the representational power of the ontology strongly depends on whether the architecture of the type system is able to express the organizational structure of the target domain knowledge.

The motivation behind the development of an ontology for a particular domain of interest falls in the following areas:

- Sharing a common understanding of the information in a knowledge domain;

- Improving interoperability among applications that use the domain knowledge;
- Making domain assumptions explicit so that applying changes as these assumptions evolve becomes easier;
- Enabling re-use of the domain knowledge.

An ontology is designed to define unambiguously the meaning of terms in a specific context by breaking them down into formal concepts with explicit relationships. Through the use of ontologies we aim to establish an agreed model, to be consistent and to integrate a range of different perspectives. Although still an evolving discipline, ontology engineering has widely and rapidly been adopted by computer science communities for different application contexts. Ontologies play a central role in the Semantic Web vision as they promote interoperability between applications and systems.

2.2 Ontology Languages and the Semantic Web

The term Semantic Web was introduced by Tim Berners-Lee [Berners-Lee, 1998] to capture the vision of a World Wide Web where information is shared, not just between human end users, but between machines as well. In other words, the Semantic Web is about enriching the current Web with machine processable data, to enable machines to share information and thus better help humans navigate, combine and retrieve information from the vast repository of knowledge that is today's Web.

Ontologies are considered key enabling constructs for the Semantic Web as they interweave human understanding of terms with machine processability. The use of ontologies and supporting tools offer an opportunity to improve significantly knowledge management capabilities.

An ontology language is a formalism used for representing ontologies and modelling information in general. It is perhaps the single most important design element that has to be considered in designing a knowledge base. This is because it solely determines how

information is to be modelled and what storage and management back-end solutions can be used.

The different ontology languages that exist today mainly differ in the expressive power they offer. The expressiveness of the language imposes restrictions on its decidability and complexity. We can identify three main features that characterise an ontology language:

- **Expressiveness.** Ideally we would require an algorithm to be able to process all of the expressions within the representation in its native format. In the case of an ontology language like OWL [Dean and Schreijber 2004] for example, this means accounting for all of the explicit classes, relationships and properties represented in the document.
- **Decidability.** In order to be useful within a machine-to-machine system, algorithms must be able to reach conclusions or calculate inferences in a predictable manner. For example, if an implicit relationship logically exists, we would expect the algorithm to be able to discover it. On the other hand, if one does not exist, or if the system is posed a logical fallacy, we would expect the algorithm to similarly respond with a rejection of the arrangement.
- **Performance.** Logical processing algorithms can be quite complex, and thus require extensive processing time or resources in order to execute. However, in order to be truly useful, an algorithm must process within the time constraints of the other system elements – which in today's computing environments very often means the ability to provide logical conclusions within milliseconds.

The first two features – expressiveness and decidability – are directly associated with the ontology language. On the other hand, performance is more general and can be associated with the implementing algorithm as well.

The Semantic Web vision, actively promoted by the World Wide Web Consortium, requires a generic mechanism for expressing machine readable semantics of data. The Resource Description Framework (RDF) [Beckett 2004] is the foundation for processing metadata, providing a simple data model for describing resources and a standardized

syntax. RDF provided very simple semantics and this, in addition to the fact that it is XML based, was the main reason for its success.

RDF Schema [Brickley and Guha, 2004] provides a richer vocabulary for describing properties and classes of resources, also offering semantics for generalization hierarchies. RDF Schema is perhaps the simplest ontology language today. The expressive capabilities of RDF Schema have been surpassed by more recent ontology languages like DAML+OIL [Horrocks et al. 2001] and OWL [Dean and Schreijber 2004].

DAML+OIL is the combination of DAML [DARPA Agent Markup Language], and OIL (Ontology Inference Layer) [Fensel et al. 2001]. It has an RDF/XML syntax and describes the structure of a domain (Schema) in an object-oriented manner. DAML+OIL consists of a set of axioms asserting the relationships between classes and properties.

OWL is developed on top of RDF and DAML+OIL and facilitates greater machine readability of Web content than that supported by XML, RDF, and RDF Schema by providing additional vocabulary along with formal semantics. OWL has three increasingly-expressive sublanguages: OWL Lite, OWL DL, and OWL Full. OWL Full is undecidable and is unlikely that any reasoning software will be able to support complete reasoning for every feature of OWL Full. OWL provides some new features that distinguish it from older ontology languages and facilitate reasoning both at schema and instance level. Some notable examples of its expressiveness will be given in the following paragraphs.

Generally, in ontology development, *classes* are created to encapsulate the meaning of concepts in a specific target domain. A class describes all those features that define the concept in the knowledge domain. An *individual*, or an *instance*, of that class in the ontology represents an object whose type is characterized by the concept in the domain. Relations between classes are defined by *object properties*. Object properties describe the kind of associations that are possible between classes. On the other hand, *datatype properties* describe features of a class and represent attributes that have a specific type such as *boolean*, *integer*, *string*, etc.

Restrictions can be imposed both on classes and on properties in order to refine their semantics. Suppose for example that we wish to declare, that a class *C* satisfies certain conditions, that is, all instances of *C* satisfy the necessary conditions. Using an OWL *Restriction* in an ontology describing a University, for example, we can easily describe the axiom that says that *all academic staff members must teach at least one undergraduate course every semester*. This imposes a restriction on the class *Academic Staff*, thus making it more specific. In this example we state that every professor must teach an undergraduate course every semester. In terms of logic we have a *universal quantification*.

Properties can also be defined to have special characteristics. For example, sometimes it is useful to say that a property is *transitive* (like "greater than"), *unique* (like "is mother of"), *symmetric* (like "is sibling of"), or the *inverse of* another property (like "eats" and "is eaten by").

In OWL it is also possible to define boolean combinations (*union*, *intersection*, *complement*) of classes. In the aforementioned University example, we can say that courses and staff members are *disjoint* and we can find all the people at the University by finding the union of the *Staff Member* and the *Student* classes.

These are certain features that make OWL a good choice for specifying ontologies. Its advanced semantics make automated reasoning support possible and also provide the necessary expressiveness needed both in specifying complex schema restrictions and in making interesting queries.

Another initiative in ontology languages is the Semantic Web Rule Language (SWRL) [Horrocks et al. 2003]. It is based on a combination of the OWL DL and OWL Lite including a high-level abstract syntax for Horn-like rules. SWRL is a W3C candidate and has currently limited support. It is however expected to be quickly standardized and adopted and to provide the next layer in the Semantic Web stack of languages.

2.3 Towards Multimedia Ontologies

The advent of Semantic Web technologies represents an important advance in creating networks of knowledge, based on various on-line available resources. In practice, different methods have been adopted for documenting 3D content. There is a desperate need and a deep awareness of the necessity of harmonizing standards. As yet, interoperability is obtained by reducing the interoperable information to the Dublin Core (DC) metadata [Dublin Core 2006]. The problem, however, is that the DC has been created based on different objectives, and does not preserve the richness of existing repositories, and this is even more evident in 3D digital content.

Semantic descriptions are defined in order to be able to categorize, retrieve and reuse multimedia elements. Examples of domain-specific ontologies and metadata have been developed for a wide set of applications, from Cultural Heritage [Doulaverakis et al. 2005] to Biomedicine [Catton et al. 2005].

In the present context, metadata about 3D models exist in a number of different levels. Some metadata are concerned about the origins of an individual model, including the producer, the methods of capture, the conditions at the time of capture, information on the settings used for equipment, perhaps on the algorithms used, etc. There may also be items of legal interest connected to the artefact: owner, copyright status, watermarks, fee for re-use, etc. Other metadata contain links and relationships to data, which are part of the same collection, recorded at the same time, in the same season, by the same person, etc. There surely exist several approaches; however none of them deal with the difficulties that inherently exist in capturing 3D shape knowledge, at least to extent that were addressed during AIM@SHAPE.

All standardization efforts are classified along two principal dimensions: content type (i.e., the standard's subject domain), and semantic depth. A 3D digital object is a collection of all the geometric, structural and visual data (model representation) as well as semantic features and properties.

The management of collections of 3D objects is at an early stage and there is significant debate whether the approaches adopted for digital libraries are suitable also for this kind of digital objects. There have been significant efforts to define metadata formats and introduce semantic information. For example, from the perspective of documenting historic and cultural heritage artefacts in museums and archives, the CIDOC-CRM [International Council of Museums 2009] initiative has reached ISO standard. Specifically the [ISO 21127:2006] establishes guidelines for the exchange of information between cultural heritage institutions and is based on input from the International Council of Museums (ICOM). This approach has been adopted by a number of other projects and was the basis of the ontological work included in the EPOCH Common Infrastructure [EPOCH 2008] as well as in the ongoing project 3D COFORM [3D COFORM 2010].

It is worth to mention that the CIDOC-CRM is an object-oriented extensible data model and can be regarded on one side as an ontology to describe the semantics behind data structures of cultural repositories, and on the other side it can be seen as a top-level ontology for integrating the various terminologies. It provides definitions and a formal structure for describing the implicit and explicit concepts and relationships used in cultural heritage documentation. In [Theodoridou et al. 2010] an extension of the CIDOC CRM ontology was introduced, able to capture the modeling and the query requirements regarding the provenance of digital objects.

Semantic description of multimedia items has been mainly developed for audio, video and images. Domain-specific ontologies are focused on describing the content and the parts of a multimedia scenario, such as elements in a scene, colors, motion duration, etc. Most of these kinds of ontologies, which deal with content description, make complete or partial use of the MPEG-7 [ISO-IEC Moving Picture Experts Group working group 2004]. MPEG-7 is a standard for describing multimedia content that addresses a wide variety of media types including: still pictures, graphics, 3D models, audio, speech, video, and combinations of these. Furthermore, MPEG-7 also provides an ontology [Hunter 2001] which embodies a general and large representation of metadata. The

Visual Descriptors Ontology [Bloehdorn et al. 2004], written in RDFS, aims to offer a more extensive description of the visual part of MPEG-7. The Core Ontology for Multimedia (COMM) [Arndt et al. 2007] is another ontology that extends MPEG-7 to provide richer multimedia semantics by using generic software patterns which create a layer between MPEG-7 concepts and domain-specific interpretations.

There are efforts towards a generalized multimedia ontology [AceMedia 2005], which represent the challenge of unifying concepts among domain specific and top-level ontologies. Attempts of combined ontologies also exist. For example, as shown in [Hunter et al. 2003], an ontology was developed based on the ABC ontology and model [Lagoze & Hunter 2001], which imported the MPEG-7 ontology, the MPEG-21 Rights ontology [Pereira 2001] and the CIDOC-CRM ontology. The metadata input was a combination of attributes coming from all the above ontologies, providing semantic interoperability across communities. Physical artefacts and digital objects were described from a museum/library/archive record point of view, with rights/access constraints, and were annotated with spoken/textual descriptions.

Another kind of standard deals with data exchange formats. By standardizing data exchange formats for 3D objects, data files can be interpreted to build data structures for use in a variety of applications. Currently, X3D [X3D 2009] (succeeding the earlier VRML) and COLLADA [COLLADA 2009] are two popular standards for encoding 3D objects into common formats for sharing between applications.

For modelling and visualization purposes, X3D offers many types of geometries, appearances and rigid transformations (i.e. rotation, translation and scaling). The geometries include 2D primitives (such as points, arcs, circles, ellipses, lines and polygons) and 3D primitives (such as boxes, cones, cylinders and spheres). If complex 3D geometries are desired, polygon meshes can be employed. Besides primitives for general modelling purposes, specialized components, such as [Humanoid Animation 2009] and NURBS for CAD applications, exist in X3D as well.

Whilst X3D has particular legacy value and familiarity in the field, COLLADA is increasingly used in 3D content creation. It defines an XML-based schema to make it

easy to transport 3D data between applications - enabling diverse 3D authoring and content processing tools to be combined into a production pipeline. It is also an extensible format that is foreseen to grow to support increasingly sophisticated 3D features, as they evolve [Arnaud & Barnes 2006].

The Common Shape Ontology (CSO), presented in section 3.1, [AIM@SHAPE 2005] targets different kinds of multimedia content, ranging from 2D/3D images to videos, 3D models and 3D animations, and maintains top-level information that is sharable and usable in different domains. Nevertheless, unlike most of the aforementioned ontologies, the CSO deals with 3D models as a key resource type, focusing on their specificities (i.e. it has been designed and used for a full characterization of digital shapes).

2.4 Knowledge Base Systems

Knowledge Management Systems (KMS) are designed to allow users to access and utilize rich sources of data, information and knowledge stored in different forms, but also to support knowledge creation, knowledge transfer and continuous learning for the knowledge workers. Recently KMSs, unlike databases, have aimed beyond the mere administration of electronic information; they now aim at fostering learning processes, knowledge sharing, collaboration between knowledge workers irrespective of their location, etc.

The support of querying facilities has always been a primary requirement for repositories of any kind. The proliferation of knowledge caused by the widespread use of the Web as a knowledge communication platform has posed the same and even more imperative requirements for performing queries and locating resources into the vast information space. However, the data models used to represent and encode knowledge on the Web differ from the traditional data structures. RDF, RDFS and OWL are the W3C standards used to encode web-based data. Thus, the functionality that a querying language should support is according to the structure and peculiarities of the new

paradigms. Practical Description Logic (DL) reasoners such as Racer [Galinski et al. 2005] and Pellet [Sirin et al. 2007] offer a functional API for querying a knowledge base.

Various Knowledge Base Systems (KBS) have been developed for managing, storing, reasoning and querying Semantic Web information. They differ in a number of important ways. For instance, many KBSs are memory-based while others use secondary storage in order to provide persistence. Another key difference is the degree of reasoning provided by the KBS. Some KBSs only support RDF/RDFS inference while others aim at providing OWL reasoning. This section focuses on existing ontology storage and management systems built specifically for the Semantic Web. Within the scope of this thesis a plethora of systems has been examined. The most important of these systems are briefly described below focusing on their intended use, underlying technology and additional features.

HP Labs has developed Jena, a Java framework for building Semantic Web applications [McBride 2002] (available as open source software under a BSD license). Jena implements APIs for dealing with Semantic Web building blocks such as RDF and OWL. It contains a rich set of features for dealing with RDF including an inference engine, the ability to reason using custom rules, and OWL-based ontology processing. It provides a semantic layer which abstracts a lot of the underlying knowledge that is required to work with the RDF or OWL syntax. The Jena Ontology API is language-neutral and as such it supports the common elements found in ontology languages. The Inference API allows access to the available reasoners for OWL but to utilize all available functionality one should communicate directly with the reasoner.

HP Labs also maintains Joseki [W3C RDF Data Access Working Group 2009], a web application for publishing and querying RDF models on the web which is also available under a BSD license. It is built on Jena and, via its flexible configuration, allows a Model, represented as a set of files or within a database, to be made available on a specified URL and queried using the SPARQL language [Prud'hommeaux and Seaborne 2008].

3store [ATK project 2008] is maintained by a development team at the University of Southampton. 3store is a core C library that uses MySQL to store its raw RDF data and caches. 3store is available under the GPL, and Redland under the LPGL or MPL licenses. It provides access to the RDF data via RDQL [Seaborne 2004] or SPARQL over HTTP, on the command line or via a C API. 3store can infer RDF and RDFS entailments and can also communicate through an Apache module. The results are cached in an in-memory store for browsing.

Sesame [Aduna Open Source project 2009] is an open source Java framework for storing, querying and reasoning with RDF and RDF Schema and is distributed under the LGPL license. It can be used as a database for RDF and RDF Schema, or as a Java library for applications that need to work with RDF internally. Sesame provides application developers a toolbox that contains the necessary tools to parse, interpret, query and store ontologies and metadata. Sesame can be used as a Server with which client applications or human users can communicate over HTTP. It supports RDF Schema inferencing and can support OWL inferencing and querying through the OWLIM Semantic Repository [Kiryakov et al. 2005]. OWLIM is a high-performance semantic repository, packaged as a Storage and Inference Layer (SAIL) for the Sesame RDF database. Much like Jena, Sesame can use open source RDBMS's in addition to its in-memory database.

IBM's Integrated Ontology Development Toolkit (IODT) [IBM Corporation 2004] is a toolkit for ontology-driven development. It includes an Ontology Definition Metamodel (EODM) implemented in Eclipse Modeling Framework (EMF) [Steinberg et al. 2008] and an OWL ontology repository named Scalable Ontology Repository (SOR). EODM is the run-time library that allows the application to read and serialize an RDFS/OWL ontology in RDF/XML format, manipulate an ontology using Java objects, call an inference engine and access inference results, and transform between ontology and other models. SOR is an OWL storage, inference, and query system based on RDBMS. It supports DLP (Description Logic Program), a subset of OWL DL, and the SPARQL language.

More details about existing publicly available data store systems built specifically for the Semantic Web together with their performance evaluation can be found in [ATK project 2008, Lee 2004].

Commercial solutions for semantic data stores are also available. One of the more advanced is provided by the Enterprise Edition of ORACLE Database 11g. It provides management capabilities with native support for RDF/RDFS/OWL and inference using OWL and RDFS semantics, querying of RDF/OWL data and ontologies using SPARQL-like graph patterns embedded in SQL, and ontology-assisted querying of relational data. A Java API and SPARQL support via an Oracle Jena adaptor is also available.

Another business oriented solution is ITM e-Knowledge from MONDECA, which is a knowledge representation management application based on Semantic Web technology and ontologies. It supports XML, RDF, SKOS, OWL and an optional reasoning module using the data in the database and a reasoning and inference rules editor.

Due to their powerful knowledge representation formalism and associated inference mechanisms, ontology-based systems are emerging as a natural choice for the next generation of KMSs operating in organizational, inter-organizational as well as community contexts.

2.5 3D Search and Retrieval

As 3D objects are steadily growing in number and their popularity is rapidly increasing, the (mostly academic) community has started focusing its attention on search and retrieval tools. The majority of the existing 3D repositories provide rather poor facilities for searching, browsing and downloading scientific resources, and lack the knowledge-based support for sharing, documenting and searching for heterogeneous resources (like data, tools, publications etc.).

For example, the Stanford 3D Scanning repository[1] is simply a web page with links to datasets, models, and software tools that can be used to visualize or convert the format of the models. The models are described with textual information without any organized metadata. Similar repositories are the Digital Michelangelo Project archive of 3D models[2] and the Large Geometric Models Archive[3]. The Princeton Shape Benchmark (PSB)[4] has some structured organization and basic services, and provides 3D models and software tools for evaluating shape-based retrieval and analysis algorithms. The McGill 3D Shape Benchmark[5] offers a repository for testing 3D shape retrieval algorithms. The models are organized in two databases organized in a number of basic level categories but no metadata are associated with the shapes. The National Taiwan University provides several services that can be accessed from a single web site[6]. Among these services are a benchmark and a database for 3D models, a retrieval system, and a tool implementing the shape comparison mechanisms used in the retrieval system. The models have been clustered into classes and the retrieval system allows the user to query the database by using specific geometric criteria. The CCCC 3D repository[7] consists of a couple of thousands of models classified into several categories. The CCCC is not just a database since it has been designed to be a search engine for 3D models. Its main feature is a user interface that allows the user to search for models by using different geometric similarity criteria. Only the PSB and the CCCC repositories provide a small set of metadata with their models, describing mainly geometric characteristics (like the number of vertices and triangles or the normalized average distances along axes).

Traditional text based document search schemes are not effective for the classification, search, and reuse of 3D data [Kazhdan et al. 2004]. The only effective means to perform context-based retrieval rely on: a) textual annotations of media (e.g. keywords), which are commonly inserted manually and constitute only a negligible

[1] Stanford 3D Scanning Repository, Stanford University, http://graphics.stanford.edu/data/3Dscanrep/
[2] The Digital Michelangelo Project, Stanford University, http://graphics.stanford.edu/projects/mich/
[3] Large Geometric Models Archive, Georgia Tech, http://www.cc.gatech.edu/projects/large_models/
[4] The Princeton Shape benchmark, http://shape.cs.princeton.edu/benchmark/
[5] McGill 3D Shape Benchmark, http://www.cim.mcgill.ca/~shape/benchMark/
[6] http://3d.csie.ntu.edu.tw/~dynamic/index.html
[7] http://infovis.uni-konstanz.de/research/projects/SimSearch3D/

portion of the information stored and b) (semi-) automatically extracted low-level features, which extend the techniques used in pattern recognition and in image processing.

Numerous research efforts focus on content-based retrieval, based on methods that measure shape similarity between 3D models [Veltkamp 2001, Daras et al. 2004, Funkhouser et al. 2005]. This has led to the development of the first experimental search engines, such as the 3D model search engine at Princeton University, the 3D model retrieval system at the National Taiwan University, the Ogden IV system at the National Institute of Multimedia Education, Japan [Chen et al. 2003], the 3D model similarity search engine at University of Konstanz, the 3D retrieval engine at Utrecht University, the GEORGLE system at Technion, Israel Institute of Technology [Leifman et al. 2005], the search engine for CAD models at Purdue University [Jiantao & Ramani 2007], and the AIM@SHAPE geometric search engine. Finally, we mention the 3D search engine developed by the project [VICTORY 2006]. VICTORY aims to develop a distributed 3D search engine that will enable searching, accessing and retrieving 3D and other types of digital content in a distributed object repository, through Peer-to-Peer networks. The 3D search demonstrated in the project web site makes use of five different global descriptors, which can be used as different alternatives in the search. The search, however, does not allow for combination, weighting or other kinds of user interactions with the search engine.

Very few commercial 3D search engines are currently available. As an example we mention the Geolus Search engine, commercialized by the Siemens PLM Software, which represents the current state-of-the-art in terms of commercial search engines for PLM (Product Lifecycle Management).

Geometry-based retrieval is just one of the issues that have to be addressed, while the peculiarities of the domain of usage make the general problem inherently complex. The knowledge is not solely carried by digital media, but also by the means used to acquire and transform them. Every context adds in principle another semantic layer of information to 3D resources. Current 3D media repositories are focusing on the

geometric aspect of models, and not on the knowledge they represent. While on the geometric or structural representations of resources there are numerous approaches, at the semantic level very little work has been done until now. We believe that the addition of explicit semantics can further improve search results.

In the last few years there has been a considerable increase of interest for techniques to extract and stream knowledge embedded into multimedia content, ranging from basic research efforts to projects seeking an integrated approach at European level (e.g [SCHEMA 2002], [AceMedia 2005] and [FOCUS K3D 2008]).

2.6 Web Services

Various definitions of a Web Services (WS) exist. According to the most wide and basic definition, almost any system accessible over the Internet (including e.g. web pages with CGI scripts) could be considered a Web Service. A somewhat more restricted view [UDDI Consortium 2001] a Web Service is a "*self-contained, modular business applications that have open, internet-oriented, standards-based interfaces*". A strong focus on open standards is what distinguishes Web Services from previous middleware and workflow systems which were most often vendor specific. An Internet orientation is the outcome of a general Internet prevalence, with information systems crossing corporate boundaries and with trends such as cloud computing having a strong impact on corporate information systems.

W3C in its Web Services Architecture Requirements document [Austin et al. 2004] gives an even more specific definition: "*A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. A Web service is identified by a URI, whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols.*" This W3C definition mentions most of the key characteristics of Web Services.

Interoperability of machine-to-machine interactions is a corner-stone motivation of Web Services. Web Services primarily address interactions between systems rather than human-to-machine interactions. The need for interoperability of systems is not new at all and the goal of Web Services is in line with traditional middleware systems.

The W3C definition focuses on WS interfaces and bindings, with

- an interface being an abstract boundary that a service exposes, typically in the form of logical grouping of operations supported by a Web Service, and
- a binding specifying an association between an interface, a concrete protocol and a data format. The interfaces and bindings are required to be described using XML.

This means that WS interfaces and bindings need to be explicitly described, i.e., an explicit specification of WS interfaces and bindings must exist (and be available to other systems so that they can interact with a Web Service). A requirement of using XML for WS descriptions is quite a natural choice given XML being an accepted standard and given XML characteristics such as application, technology and vendor neutrality. Also Web Services can be seen as an extension of the stack of W3C standards and thus benefit from its high acceptance and availability of tools.

As Web Services are distributed systems by their nature and since they operate in (presumably open) network environments, a natural requirement is that their descriptions might need to be discovered. In other words, service descriptions should allow other software systems to discover existing services which comply with some specified requirements. The interaction of Web Services is based on the publish-find-bind paradigm, which specifies three distinct roles: service requestor, service registry and service provider. The service provider publishes the service description to the service registry, while the service requestor finds the desired service in the registry, binds to it and retrieves the required functionality. An architecture that supports this paradigm is called a Service-Oriented Architecture (SOA) [Papazoglou 2003].

A Web Service provides functionality including access to data sources through a web-accessible interface in a programming language-independent and platform-

independent manner [Vaughan-Nichols 2002]. The Web Service interface is described in a machine understandable way, which is a fundamental requirement for interoperability. Based on these interfaces, Web Services connect readily available software components on the Web in a loosely coupled way. Loosely coupled means that during design time a specific service can be substituted by an abstract specification describing requirements for a requested service. A specific service matching the requirements can be discovered later, e.g., during runtime, thus supporting dynamic binding, which is another important characteristic of WS systems and SOA. This enables the reuse of software components in different applications. Moreover, as Web Services communicate based on platform-independent protocols (e.g. HTTP) and exchange formats (e.g. XML), they can be reused by any application written in any programming language and/or running on any operating system.

Web Service standards provide a specific realization complying with the above described characteristics and a good basis for achieving some level of interoperability. The Web Services Description Language (WSDL) [Chinnici et al. 2007] allows to declaratively describe interfaces (i.e. a set of operations), the format of messages, and the data structures that are used to communicate with a Web Service. The SOAP [SOAP 2007] protocol addresses messages exchange between distrusted applications. UDDI [Clement et al. 2004] presents a platform-independent specification of describing and discovering Web services and Web service providers. WS-BPEL [Alves et al. 2007] is a language that describes WS-based business process and allows individual Web Services to be composed into complex business processes. A business process is basically a workflow in which interactions with Web Services are the basic building blocks. By chaining different web services clients can generate new composited Web Services. All these standards and specifications are XML-based and they are all designed to integrate easily with each other. Thus, WSDL provides mechanisms for bindings in SOAP, WS-BPEL processes are tailored to use WSDL interfaces, and UDDI integrates very well with WSDL services. However, WSDL and SOAP based Web Services do not present the only solution. A prominent lightweight alternative popular mainly in publicly available

Web Services is offered by the REST (Representational State Transfer) [Fielding 2000] style services.

Service discovery and service composition are two types of problems commonly addressed in Web Service based systems. Web services discovery addresses a problem of finding a suitable service or a set of services which are able to match specified requirements or objectives. Service discovery and selection can be performed manually, in which case the user is responsible for selecting a matching service. In case of automated discovery, matchmaking algorithms need to be applied to decide about a degree of match between a service request and a service advertisement. An automated matching can be based on various criteria and parameters related to Web services. Typically, the so called functional properties of WS are taken into account, such as types of messages supported by the service, datatypes of service inputs and outputs, etc. Often also non-functional properties, such as quality of services (QoS) parameters, play a significant role during matchmaking. Most of the current research focuses on discovery based on information specified in service interfaces.

The terms “orchestration” and “choreography” are widely used to describe business interaction protocols that coordinate and control collaborating services [Papazoglou et al. 2007, Pelz 2003, Barros et al. 2006]. Choreography is concerned with describing the external visible behavior of services, as a set of message exchanges, rules of interaction and agreements, from the functionality consumer point of view. Service choreography can be achieved via the Web Services Choreography Description Language (WS-CDL) [Kavantzas et al. 2005]. There are other complementary efforts in the area of choreography languages, such as Business Process Modeling Notation (BPMN) defined by BPMI.org, Business Process Specification Schema (BPSS) defined by ebXML, IBM's Web Services Flow Language (WSFL), Microsoft's XLANG, Let's Dance [Zaha et al. 2006] and BPEL4Chor [Decker et al. 2007]. Orchestration describes how services interact at the message level, including the business logic and execution order of interactions. Orchestration is achieved via BPEL and other XML-based process standard definition languages. This distinction between orchestration and choreography is rather

artificial, and the consensus is that they should coalesce in a single language and environment.

Service composition focuses on models and methods supporting composition of several existing Web services into more complex processes which provide new functionalities. In a way, composition can be seen as programming in large with existing services serving as basic building blocks. WS-BPEL presents one possible service composition model allowing creation of complex business processes from existing services. Similarly to service discovery, the main focus of research has been in exploring methods of automated or semi-automated service composition [Rao and Su 2004, Alamri 2006]. A usual formulation of the automated composition problem assumes that there is a given set of existing services and a set of requirements and the goal is to find a composition (possibly the best under certain given criteria) which satisfies the specified requirements. Most of the approaches to automated composition are based on some form of automated planning [Sirin et al. 2002, Traverso and Pistore 2004].

A service may also be defined as a composition of other services constituting a workflow describing the choreography of several operations. Such a workflow may determine: the order of operation execution; what operations may be executed concurrently; alternative execution pathways (if conditional operators are included in the workflow modeling language). Furthermore, workflows are required to orchestrate the execution of several simple services that may be composed together for forming a more complex service.

The importance of Web Services has been recognized and widely accepted by the industry and academia. The Web is now evolving into a distributed device of computation from a collection of information, data and computational resources.

2.7 Semantic Web Services

The need for composing existing Web Services into more complex services is continuously increasing. Web Services technologies are being used as a means of

exploiting language- as well as platform-independence and achieving syntactic interoperability among cooperating programs. In many cases, Web Services offer little more than a formally defined invocation interface, with some human oriented metadata that describe what the service does, and which organization developed it.

Due to the strictly syntactic nature of WS standards, discovery, matching or composition of services must basically rely on string comparisons similar to classical keywords based search, since they do not deal with the meaning of services. Thus, for example, even if two services provide exactly the same functionalities and even if their interfaces are semantically compatible, automated matchmaking algorithms will consider them as completely different services if their descriptions do not match exactly. This semantic gap is making it difficult for requesters and providers to interpret or represent non-trivial statements such as the meaning of inputs and outputs or applicable constraints. This limitation may be relaxed by providing a set of semantic annotations that augment the service description.

Research in the Semantic Web Services area [McIlraith et al. 2001] is addressing these and similar problems of traditional Web Services by augmenting the existing Web services standards with additional semantic layer(s). The additional semantics focus on enabling "*... semantically transparent services which will make it possible for clients to successfully use services that are dynamically discovered without prior negotiations between client and service developers*" [Burstein et al. 2005]. Most of the proposed mechanisms rely on extending the current WS descriptions with descriptions in some logical formalism more suitable for capturing the explicit meaning of provided Web Services. Such formalisms typically rely on formal ontologies and research conducted in the Semantic Web area.

Current Semantic Web Services approaches combine Semantic Web technologies with existing Web Services standards. OWL-S [OWL-S Coalition 2003] is a service ontology expressed in OWL. OWL-S is a successor of the DAML-S ontology [Ankolekar et al. 2001], a DAML-based Web service ontology, which defines a core set of markup

language constructs for describing the properties and capabilities of Web Services in unambiguous, computer-interpretable form.

OWL-S covers three areas: Web services capability-based search and discovery, specification of service requester and service provider interactions (interaction protocol), and service execution. A *Service Profile* describes what the service does in terms of its capabilities and it is used for discovering suitable providers, and selecting among them. A *Process Model* specifies (a) how the service works and (b) how clients can interact with the service by defining a requester-provider interaction protocol. OWL-S process model pertains mainly to describing service providers. However, its constructs can be used to describe the requester as well. A *Grounding* links the Process Model to the specific execution infrastructure. Currently, a grounding for WSDL and SOAP is available, which maps OWL-S processes to WSDL operations and data structures and allows for sending messages in SOAP. Recently, also a grounding of OWL-S to SAWSDL [Farrell and Lausen 2007] was developed [Paolucci et al. 2007].

Web Service Modeling Ontology (WSMO) [Roman et al. 2005] presents another popular approach. WSMO is an ontology for describing services expressed in the WSML language, based on Frame Logic [Kifer et al. 1995]. A significant difference between WSMO and OWL-S is the approach to modeling complex workflows. While in OWL-S, control flow is based on a notion of processes and explicit control constructs known from programming languages, in WSMO the Abstract State Machine [Gurevich 1995] methodology is used. WSMO also provides an integrated execution environment WSMX [Cimpian and Mocan 2005]. Recently, WSMO-Lite [Vitvar et al. 2008] was developed, which is a lightweight service ontology, expressed in RDFS, for annotation of WSDL and SAWSDL based services.

While the previous approaches follow a top-down direction to WS description, developing a full-fledged model for service annotations, a recently published W3C recommendation Semantic Annotations for WSDL and XML Schema (SAWSDL) [Farrell and Lausen 2007] presents a rather bottom-up approach to WS annotations. SAWSDL extends WSDL by allowing WSDL documents to refer to semantic concepts

from ontologies, which can formally define service operations or their input and output parameters. SAWSDL was based on Web Services Semantics (WSDL-S) [Akkiraju et al. 2005]. In contrast to WSMO and OWL-S, SAWSDL is neutral with respect to a specific annotation language or formalism (i.e. developers may annotate their Web Services with their choice of language e.g. UML or OWL).

Electronic Business XML (ebXML) [ebXML 2001] is a widely adopted standard from OASIS and United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT) and ebXML registry implementations have essentially replaced UDDI registries. One of the important characteristics of ebXML compliant registries is that they provide mechanisms to define and associate semantics with registry objects. The basic semantic mechanisms of ebXML Registry are classification hierarchies consisting of classification nodes and the predefined associations among classification nodes. These are all registry objects which can be assigned properties through a slot mechanism. Given these constructs, considerable amount of semantics can be defined in the registry.

In contrast to the OWL-S approach, the work presented in [Dogac et al. 2005] takes a data management approach for service discovery by exploiting the metadata and the query mechanism of the ebXML registry and shows how ebXML registry semantics support can be further enhanced to make it OWL aware. The work presented in this paper extends the ideas given in [Dogac et al. 2004] and provides a complete model on how OWL Lite constructs can be stored to ebXML registries and queried to facilitate semantic discovery of Web Services. These predefined stored queries provide the means to exploit the enhanced semantics stored in the registry and to retrieve knowledge through queries.

3 IMPLEMENTING THE KNOWLEDGE INFRASTRUCTURE

In this chapter we propose an ontology-based conceptualization of the domain knowledge and introduce knowledge representation techniques in 3D shape modeling. We present our ontology development methodology and briefly describe the structure and motivation behind the context-dependent ontologies developed during the AIM@SHAPE project.

We also present the overall architecture of the Digital Shape Workbench (DSW), a unified platform for modelling, storing, processing and reasoning about shape resources. Moreover, we present in detail the design and implementation of the Ontology & Metadata Repository (OMR), the knowledge base back-end of the DSW and its role in the development of the Semantic Search mechanism.

3.1 Ontology Development for the Representation of 3D Objects

The use of 3D data is not only related to the visual aspects and rendering processes of models, but it also involves an adequate representation of domain knowledge to exploit and preserve both the expertise used for their creation and the information content carried.

3D semantic modeling includes activities like the acquisition and reconstruction of objects, providing high geometric accuracy in the digital models, extraction of features and/or model properties, portability and flexibility in applications, while minimizing human interaction during the modeling process. The association with semantics is also crucial to visualize the models properly and retrieve them efficiently from large repositories. Efficient retrieval implies equipping 3D content with metadata related to both the whole object and its subparts, developing automatic metadata extraction tools and shape similarity mechanisms to compare objects, providing best practices assisting the processing phase and beyond.

Our development of the Digital Shape Workbench (DSW) [AIM@SHAPE 2006b] required the conceptualization of the chosen application domains and the precise characterization of the resources. This effort can be seen as an initial step towards contributing to the goals of the Semantic Web as the mean to share scientific resources.

The complexity and the extent of the 3D domain makes the use of a single monolithic ontology unsuitable and it leads the way in building a framework where different ontologies are adopted to represent different facets of specific domain applications and usage scenarios.

One of our objectives was to formalize the domain knowledge into context-dependent ontologies and introduce knowledge management techniques in 3D shape modeling, with the aim of making explicit and sharable the knowledge embedded in digital models.

The formalization of knowledge related to the geometric and structural representation of digital models is the basic step in building application-specific ontologies. The geometry and structure of an object always remain the same while its descriptions can differ according to the various contexts, and it should be captured by a set of metadata that fully describes the properties of the representation used (see section 3.1.2).

3.1.1 Development Methodology

Ontology development still remains a not well understood process. This is due to the size of the knowledge we are usually faced with, for a specific scientific domain, and the complexity of the problem of storing and managing this knowledge in a flexible and efficient way. During the ontology creation process different perspectives on the domain have to be brought together and a consensus within a group of people with different backgrounds has to be reached. Also, because of the nature of the problem, ontology development is a rather subjective matter i.e. there is not a single correct ontology for a

specific domain. The resulting ontology structure depends on the usage, scope, anticipated extensions, etc.

Much like in the design of object-oriented software, there is not a single correct methodology for designing and building an ontology. There are, however, a few methodologies that are well established and seem to perform better than others.

In order to design the ontologies described in section 3.1.2, we adopted the methodology introduced in the On-To-Knowledge project [Sure et al. 2003], which is characterized by the early specification of the requirements through the formalisation of competency questions (i.e. questions that should be answered using only the information included in the ontology) and an iteration of a refinement phase, an evaluation phase and a maintenance phase, as shown in Figure 3.1.

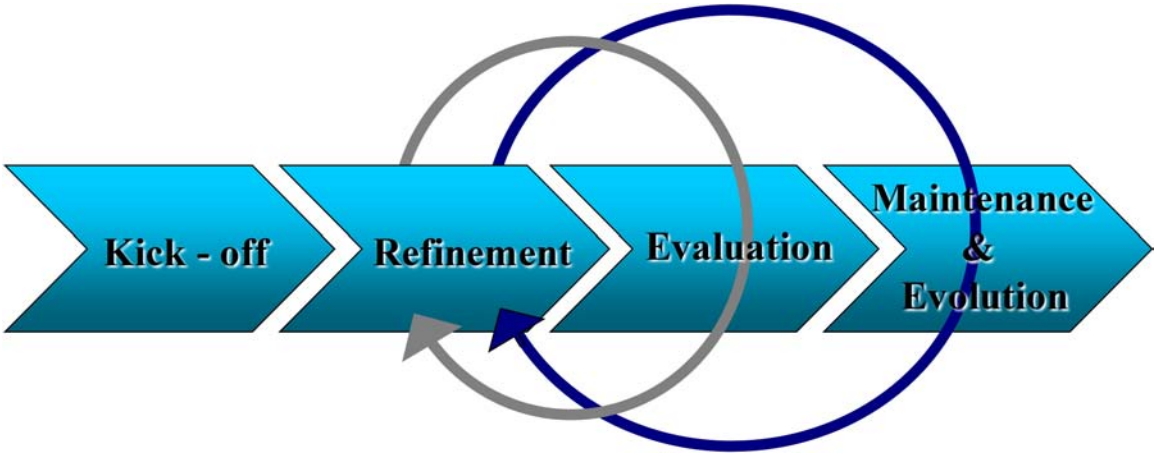


Figure 3.1: The On-to-Knowledge methodology

The On-To-Knowledge methodology defines an iterative process comprised of four phases:

Kick-off phase: Capturing the ontology. The kick-off phase is probably the most important phase since it formulates the initial concrete idea of how the ontology structure is going to be. There are two steps in this phase. During the first step, a set of concepts (e.g. entities, attributes and processes) are identified as well as the relationships to each other; this is referred to as a *conceptualisation*. This step identifies the domain of the ontology, some usage scenarios, knowledge sources (domain experts, glossaries and

dictionaries, etc.), potential users and applications, and basic questions the ontology should be able to answer. These basic questions are called *Competency Questions* (CQs) and essentially constitute the expressiveness requirements for the ontology. CQs indicate the scope of the ontology and can also be used as reference during the evaluation phase. Ideally, CQs should be defined in a modular manner, so that higher level (more complex) questions are comprised of lower level questions (less complex).

In the second step a semi-formal hierarchy of the concepts and relations is built. Three different approaches exist for this stage:

- The top-down approach models the most general concepts first and then produces more refined concepts. This approach produces a high-quality, fine-grained ontology but may not cover all domain information.
- The bottom-up approach identifies the most specific concepts first and produces generalizations to obtain the higher level concepts. This approach produces an ontology that is less-detailed than that produced by the top-down approach, but is more complete and addresses all available domain information.
- The middle-out approach identifies the most important concepts and obtains the rest either by specialization or generalization. This is the most natural approach, concentrating on what is important and better controlling the desired level of detail.

Refinement phase. In the refinement phase a representation language is chosen and the ontology is recorded in a formal manner. The result of this phase is a formal representation of the ontology described in the previous phase. What makes a representation language appropriate or not, is its expressive power and its support for reasoning. In our case, the OWL-DL language was used for the representation of shape knowledge in all the developed ontologies. OWL has been chosen as it currently represents a standard for designing ontologies and has enough expressive power for our

needs. Most of the ontology development and refinement work was done using the Protégé[8] open source ontology editor.

Evaluation phase. During the evaluation phase the ontology is constantly checked to evaluate its compliance with the expectations/requirements. Usage scenarios, requirements specifications and competency questions are used as reference for the evaluation.

Maintenance and Evolution phase. During this phase several maintenance tasks are performed as the ontology changes and evolves over time.

Each phase can also be distinguished by the degree of formality by which the vocabulary, that constitutes the ontology, is created and the meaning is specified. Four, somewhat arbitrary degrees of formality can be identified:

- Highly informal: the vocabulary is expressed loosely in natural language;
- Semi-informal: the vocabulary is expressed in a restricted and structured form of natural language, greatly increasing clarity by reducing ambiguity;
- Semi-formal: the vocabulary is expressed in an artificial formally defined language;
- Rigorously formal: the vocabulary is meticulously defined with formal semantics, theorems and proofs of such properties as soundness and completeness.

Following the ontology development guidelines stated above, the ontologies presented in section 3.1.2 were developed and modeled with the assistance of domain experts. It was an iterative and time-consuming process that took several months to reach the desired consensus among the experts. The ontology development process followed is shown in Figure 3.2.

[8] Protégé official web site: http://protege.stanford.edu/

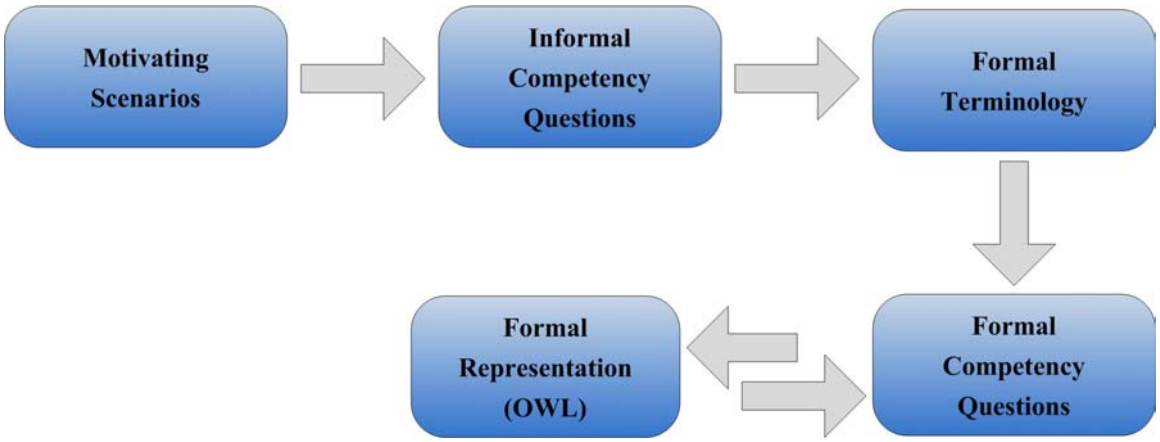


Figure 3.2: The ontology development process

The progress and evolution of the ontologies were measured by the amount of knowledge captured according to the requirements specification; i.e. to be able to answer as many as possible of the specified competency questions or possibly a richer set of CQs. A qualitative evaluation of the developed ontologies is presented in Chapter 5.

3.1.2 Description of the Developed Ontologies

The introduced framework relies on the conceptualization of different aspects of an object. To be more precise, 3D objects can be described as a simple resource, and/or by its geometry, its structure, its semantics, depending on the application domain, as depicted by Figure 3.3. For example, there are cases where more emphasis should be put on capturing and representing information about the usage of an object (implicit information), which can be even more important than its geometrical attributes (explicit information). Implicit information is information that can be inferred from explicit information about a resource. The information that a teapot can be grasped by a virtual human in an animation can be deduced by the fact (explicit information) that the tea pot is specified to have a handle and by the fact that handles provide a part of the object that can be grasped.

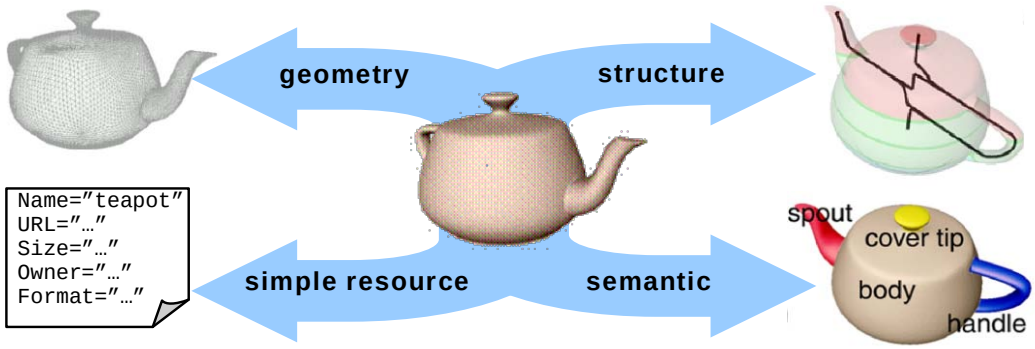


Figure 3.3: Depending on the application domain a shape may be described either as a simple resource, or by its geometry, its structure and its semantics [Albertoni et al. 2005]

Ontology-driven metadata provide an expressive characterization of shapes, offering different levels of representation of a resource. Due to the intrinsic complexity of shapes, ontology-driven metadata are necessary in order to reach a sufficient level of expressiveness, providing a thorough characterization of shapes by storing: *(i)* the information related to its history, such as the acquisition devices and techniques for creating it or the tools for transforming it (its *past*, e.g. for documentation), *(ii)* the information held by the shape itself (its *present*) and *(iii)* the information related to its capabilities and potential uses, such as the possible steps that can be performed or the tools that can be used (its *future*, e.g., for acquisition/process planning).

The knowledge carried by digital representations of 3D objects can be divided into three levels of granularity with respect to their knowledge content. At the *geometric level*, a digital shape is represented by coding its form using a suitable geometric representation scheme (see Figure 3.3). At the *structural level*, a digital shape gives an abstraction, identifying the portions or segments that are relevant and how they are connected to each other. The process of structuring a digital shape requires the geometric or morphological analysis of the geometrical representation, and it is often related to the extraction of relevant form features. At the *semantic level*, a digital shape makes its interpretation explicit in a given knowledge domain, for example, by associating an object to a specific class or by associating a semantic label to specific portions of the shape.

In order to express in a formalized way the knowledge about digital shapes that is common to all domain-specific scenarios used, we have specified a high level ontology for digital shapes – the Common Shape Ontology (CSO) [Vasilakis et al. 2010; Vasilakis et al. 2007]. It targets different kinds of multimedia content, ranging from 2D/3D images to videos, 3D models and 3D animations, and maintains top-level information that is sharable and usable in different domains.

A common ontology, also known as meta-ontology, is an ontology that is created to integrate ontologies across domains in order to support some common purpose. In this sense a common ontology has the potential to achieve interoperability across domains and encapsulate the fragmented knowledge, captured by the different ontologies, under a unified ontological framework. In this case, the common goal is the ability to reason, to re-use existing knowledge and to extend and create new knowledge about shape resources and tools.

As promising as this might seem, the overall integration of existing ontologies is one of the biggest problems we face in developing a comprehensive methodology for building common ontologies. The difficulty lies in achieving the necessary integration and consensus while keeping the structure simple in order to increase its potential for reuse.

The CSO deals with 3D models as a key resource type and it has been designed and used for a full characterization of shapes in the Shape Repository [AIM@SHAPE 2006b]. An overview of the CSO high-level structure (level-one classes), where the most important concepts are shown, is given in Figure 3.4. The ontology includes fifteen (15) object properties of which thirteen (13) are among the four top-level classes.

An account of the level-one class properties (datatype properties) of the ontology is given in Figure 3.5. In this diagram displaying class properties, a property type with an asterisk denotes multiple occurrence of the property-value triplet, while absence of the asterisk denotes single occurrence.

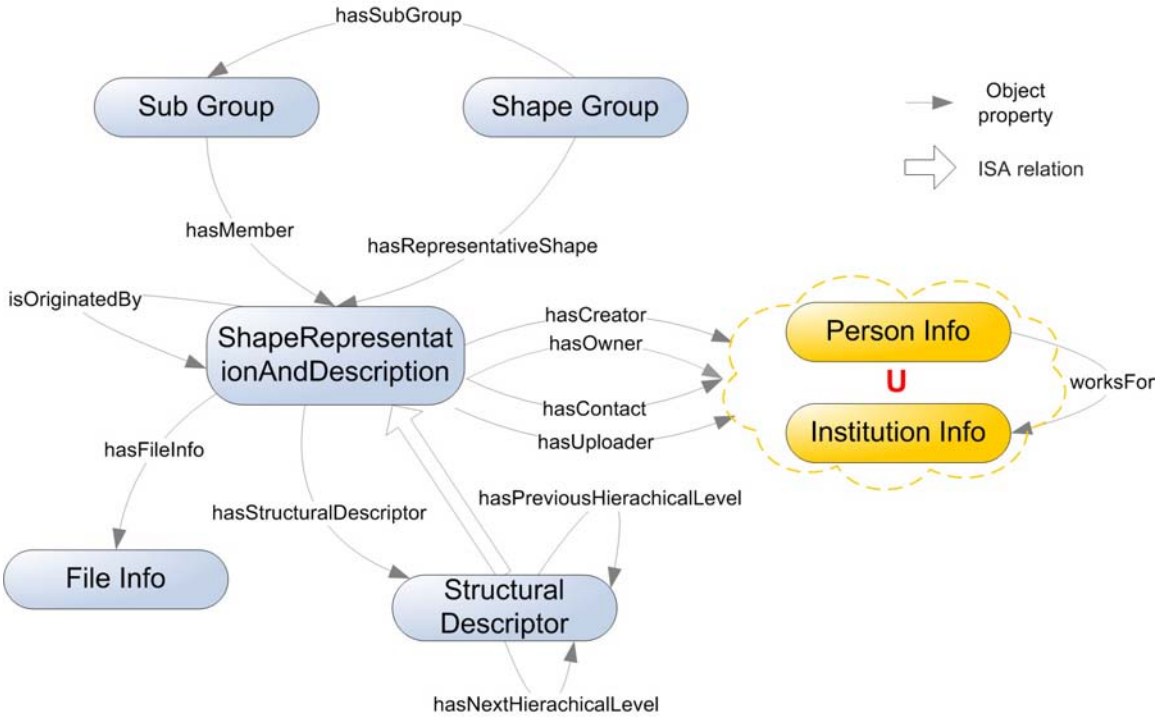


Figure 3.4: An overview of the high-level structure of the Common Shape Ontology.

File Info	
hasFileFormat	string
hasFileName	string
hasFileSize	integer
hasFileURL	string *
hasPurpose	string

Institution Info	
hasAddress	string
hasName	string *
hasShortName	string
hasURL	string *

Shape Representation	
hasAvailabilityLevel	string
hasContentType	string *
hasCopyright	string
hasCreationDate	string
hasDescription	string
hasGalleryImageURL	string *
hasKeyword	string *
hasLastModificationDate	string
hasLicense	string
hasModelURL	string
hasName	string
hasOrigin	string
hasQuality	integer
hasRelatedResource	string *
hasStatus	string
hasSynthesisDescription	string
hasThumbnailImageURL	string
hasUploadDate	string

Shape Group	
hasDescription	string
hasName	string

Person Info	
hasAddress	string
hasEmail	string *
hasFax	string
hasName	string *
hasPhone	string
hasTitle	string *
hasURL	string *

Sub Group	
hasDescription	string
hasLevelNumber	integer

Figure 3.5: Level-one class properties (datatype properties) of the CSO

The *Shape Representation* class hierarchy is shown in Figure 3.6. Different class levels are painted in different colour in the diagram. It can easily be seen that the structure is quite extensive and contains a lot of specialized classes and enough information to differentiate between all types of shapes in the domain of interest. The classification of the classes was made using mainly geometric characteristics of the shapes but also the domains of use (see the level-two classes in Figure 3.6).

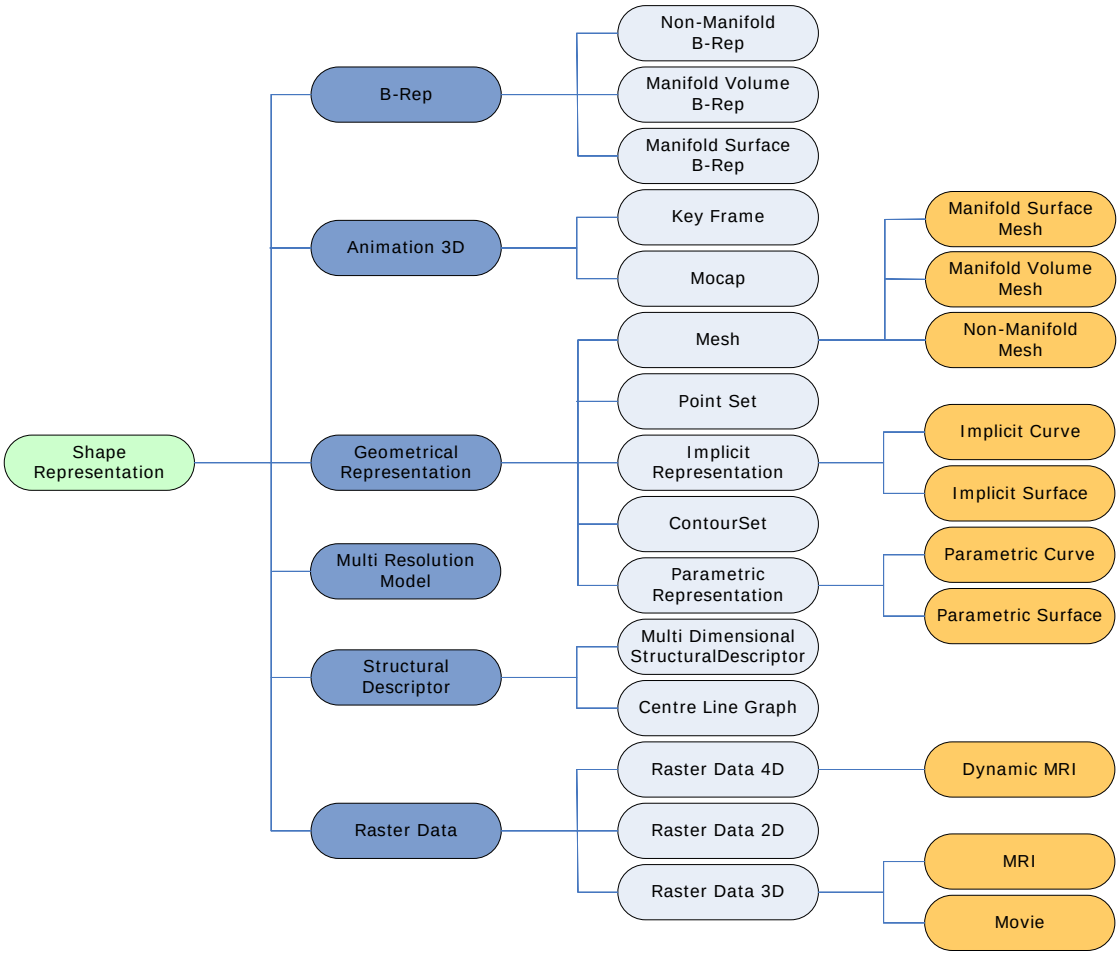


Figure 3.6: The *Shape Representation* class hierarchy of the CSO

This ontology is detailed enough to be used and instantiated directly, but also general enough to be referred to and extended by other specialized domain ontologies (all domain

ontologies import it) - hence the name “common”. In fact it constitutes the foundation for the three domain specific ontologies developed, namely the Virtual Human Ontology [Gutirrez et al. 2007], the Shape Acquisition and Processing Ontology [Albertoni et al. 2006] and the Product Design Ontology [Catalano et al. 2008]. Our aim was to address multiple contexts and applications in a semantic-aware level of representation where the shape knowledge can be exploited.

The *Virtual Human Ontology* (VH) aims at organizing the knowledge and data related to research and applications in the field of virtual environments and humans: the modeling and analysis of virtual human body, and their animation and interaction with virtual objects. It describes complex 3D entities not only at the geometric level, but also at the structural and semantic level. The goal of this description is to simplify the composition of virtual humans by non-experts and to facilitate sharing of useful information by domain experts in order to promote reusability and scalability.

The *Shape Acquisition and Processing Ontology* (SAP) intends to conceptualize the knowledge pertaining to the development, usage and sharing of hardware tools, software tools and shape data by researchers and experts in the field of shape acquisition and processing. The fundamental goal of the ontology is to formalize the knowledge related to the acquisition and processing of a shape.

The objective of the *Product Design Ontology* (PDO) is to guide researchers and experts in the development of tools and methods for: supporting industrial product design and engineering analysis, dealing with knowledge concerning shape processing methods and algorithms, and knowledge about processes and workflows regarding product development phases.

The following two usage scenarios [Vasilakis et al. 2010] outline how the Common Shape Ontology and the domain-specific ontologies are being used. The first scenario is related to a human body acquisition for creating an animated virtual human starting from a real person.

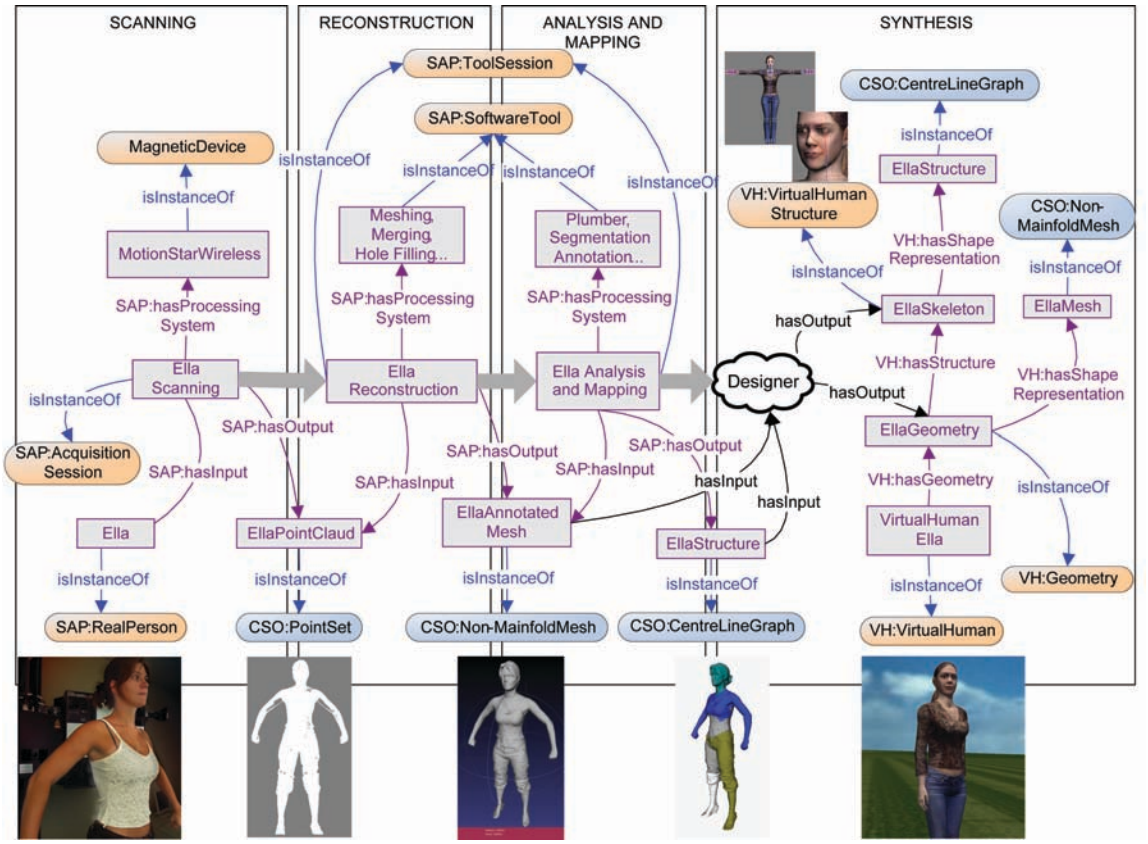


Figure 3.7: Acquisition and Processing of a Human Shape [Vasilakis et al. 2010]

In Figure 3.7 the different concepts and relations involving the human shape acquisition are depicted. The "*SAP:*" prefix denotes a concept belonging to the Shape Acquisition and Processing Ontology, the "*VH:*" prefix means that the concept is formalized in the Virtual Human ontology, while the "*CSO:*" prefix means that the concept belongs to the Common Shape Ontology. The scenario is presented as a workflow of actions (scanning, reconstruction, analysis and synthesis) for obtaining an animated virtual human (instance of *VH:VirtualHuman*) from a real object (the human person – instance of the concept *SAP:RealPerson*). Every action is foreseen in the conceptualization of the corresponding domain ontology.

During this creation pipeline the history of the shape is stored in the CSO. This allows us to answer competency questions such as: *What shape originated from shape 'EllaMesh'? What kind of structure conforms the skeleton of this Virtual Human? Which*

shapes were generated from the shape "EllaPointClaud"? Furthermore, the SAP and VH also serves in answering domain specific questions, e.g: *Which software was used to annotate the shape "EllaAnnotatedMesh"? Which is the real person used to create the animated virtual human 'Michela'?, Under which lighting conditions did the real person create this virtual human*? *What animations can be used by this virtual human?*

The second scenario is associated with the product design process related to the engineering analysis, also referred to as *simulation phase*. It evaluates the physical behavior of any engineering component of a product, which is subject to various kinds of loads and conditions, ranging from structural analysis to thermal and electrical analysis, and so on. Figure 3.8 presents the workflow followed to perform a simulation on a mechanical part.

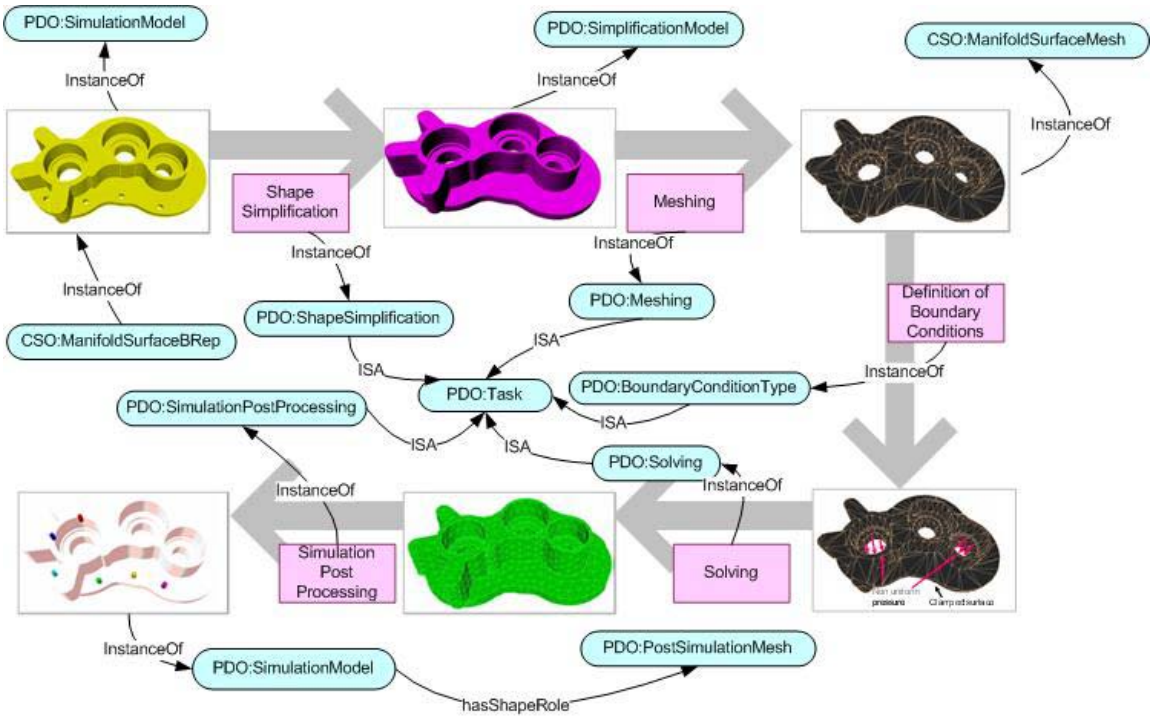


Figure 3.8: Tasks of the product design simulation process. Some elements in the diagram refer to concepts in both the PDO and the CSO. All the boxes are instances of concepts which are sub-concepts of PDO:Task [Vasilakis et al. 2010]

Concepts that are shared by all domain ontologies have also lead to the creation of a Common Tool Ontology (CTO) related to shape processing tools. An overview of the CTO level-one (high–level) structure is given in Figure 3.9. An overview of the level-one class properties (datatype properties) of the ontology is given in Figure 3.10.

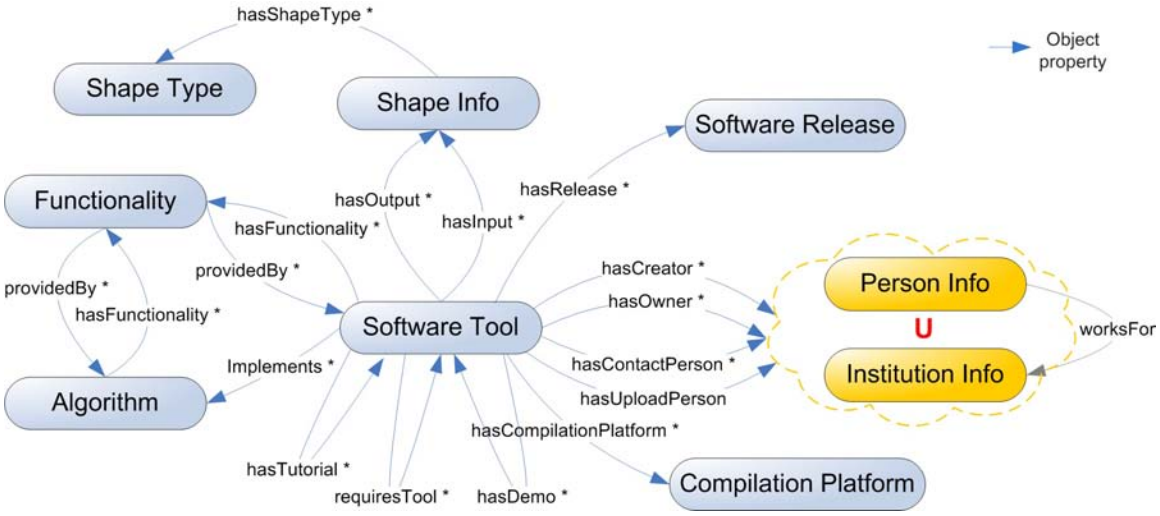


Figure 3.9: The high-level structure of the Common Tool Ontology.

Software Tool	
hasBenchmarkURL	string *
hasBugsRepository	string *
hasDescription	string
hasDevelopersMailingList	string *
hasDevelopmentStatus	string
hasExecutionPlatform	string *
hasHardwareRequirements	string *
hasInputParameter	string *
hasLastModifiedDate	string
hasLicense	string
hasName	string
hasReference	string *
hasReviewURL	string *
hasScreenshot	string *
hasSourceCodeURL	string
hasUILanguage	string *
hasUIType	string *
hasUploadDate	string
hasURL	string *
hasUserComment	string *
hasUsersMailingList	string *
writtenWithProgramming Language	string *

Shape Info	
hasFormatConverter	string *
hasShapeExample	string *
hasShapeFormat	string *
hasShapeSelfIntersection	boolean
hasSpaceDimension	string *
isValidated	boolean

Compilation Platform	
supportsCompiler	string *
supportsOS	String *

Person Info	
hasAddress	string
hasEmail	string *
hasFax	string
hasName	string *
hasPhone	string
hasTitle	string *
hasURL	string *

Algorithm	
hasAverageSpaceComplexity	string
hasAverageTimeComplexity	string
hasReference	string *
hasWorstSpaceComplexity	string
hasWorstTimeComplexity	string
isValidated	boolean

Software Release	
hasContent	string *
hasDownloadRestrictions	boolean
hasDownloadURL	string *
hasInstallationInstructions	string
hasPrice	string
hasReleaseDate	string
hasReleaseNotesDocument	string
hasUninstallationInstructions	string
hasVersion	string

Figure 3.10: Level-one class properties (datatype properties) of the CTO

The *Functionality* class hierarchy is used to define what the functionality of an *Algorithm* or a *Software Tool* is. This hierarchy is shown in Figure 3.11. An instance of a class in the *Functionality* hierarchy is essentially a leaf and as such it represents the most specific functionality with respect to the path in the hierarchy that has been chosen. For example, *Structuring Functionality* is more general than *Subdivision Functionality*, which in turn is more general than *Triangulation Functionality*. Thus, along this path *Triangulation Functionality* constitutes the largest degree of specialization possible for defining the functionality of a specific tool.

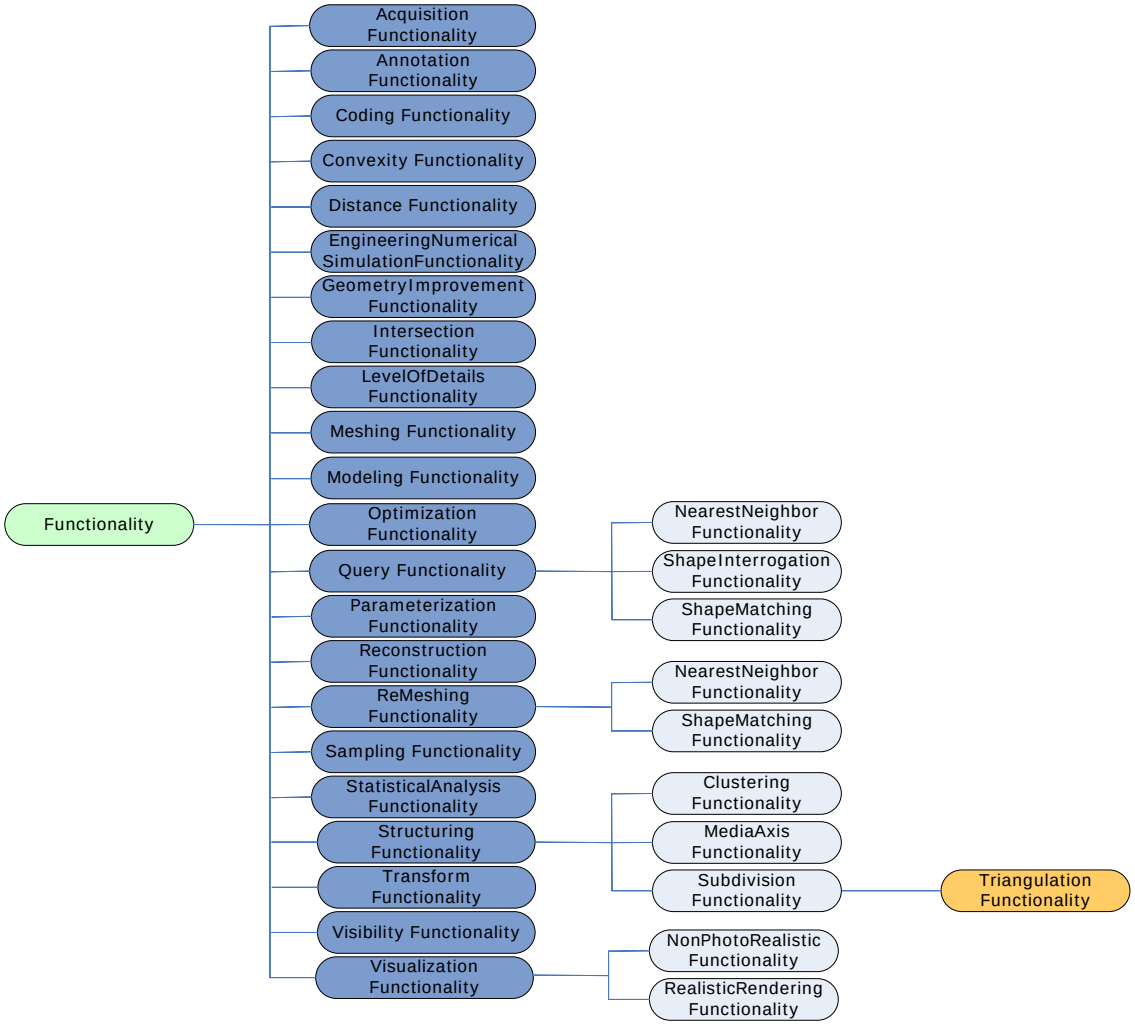


Figure 3.11: The *Functionality* class hierarchy of the CTO

The Common Tool Ontology basically models information related to software tools. A tool can be further specialized to the concepts of a *Library*, a *Macro*, a *Plugin*, and an *Independent Application*, as shown in Figure 3.12. The most important properties of a tool are also shown in the associated text sheet. For the purposes of our prototype implementation of Geometry Processing Web Services (discussed in Chapter 4), we extended *Software Tool* by adding a subclass called *Web Service*.

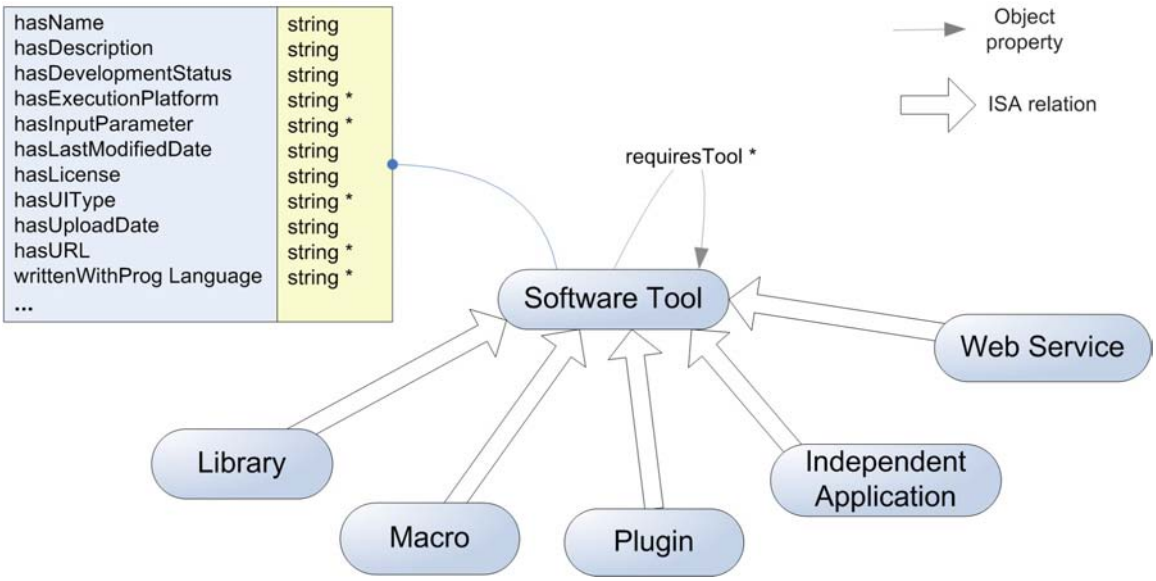


Figure 3.12: The subclasses of the *Software Tool* class and some of its properties

The CTO can therefore support the description of workflows or in general it can be used (e.g. it can be imported) in other ontologies where the notion of tools and services is considered important. A simple example of reusing the concept of *SoftwareTool* in workflow related scenarios is evident in the Product Design Ontology (PDO). In the PDO we have defined the notion of *Task* to model product design chains where each task is supported by a specific software tool instance or by a class of specific algorithms. This can be seen in Figure 3.13, which depicts the various relations between the PDO and the two common ontologies.

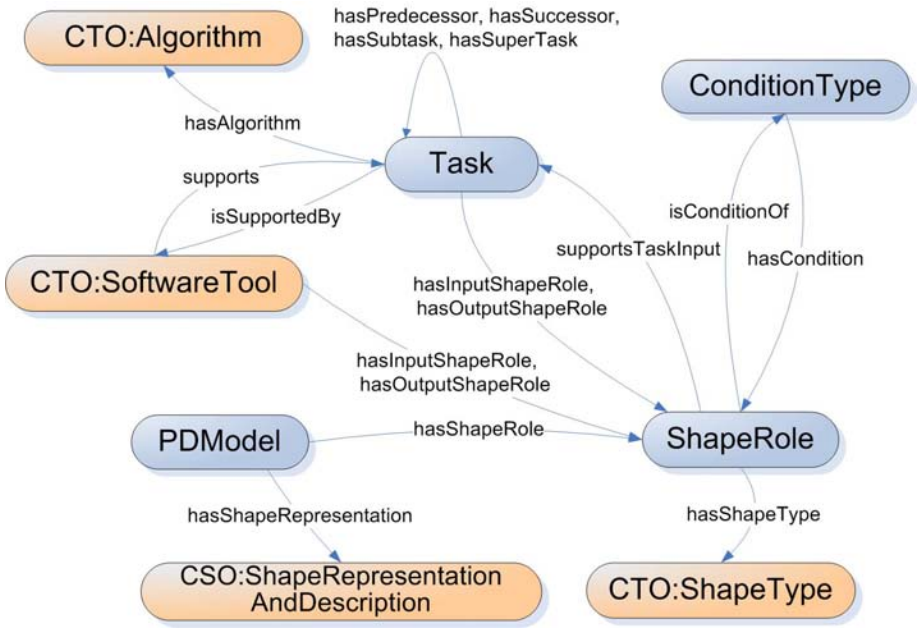


Figure 3.13: Modeling the *Task* concept

The class *PDO:ShapeRole* formalizes the relationship between the type of the geometric representation of a shape and the specific information associated with a particular task of the product design workflow. The *PDO:PDModel* class has been introduced to model the role of a specific shape, which is represented by an instance of *CSO:ShapeRepresentationAndDescription*.

A *PDO:Task* performs a certain design operation and is associated with specific instances of *PDO:ShapeRole* and *PDO:PDModel* as its input and output, respectively. To operate on digital shapes, dedicated software tools and classes of algorithms have been devised and formalized in the PDO through the relations *PDO:supports, PDO:isSupportedBy,* and *PDO:hasAlgorithm.* To be able to search for tools that apply on and provide for a given shape role, the CTO has been extended by the PDO by introducing two relations between the classes *CTO:SoftwareTool* and *PDO:ShapeRole*; these relations are *hasInputShapeRole* and *hasOutputShapeRole*.

Unlike the development process regarding the domain ontologies which was based on a combination of *top-down* and *middle-out* approach, the methodology used for the development of the common ontologies was a *bottom-up* approach (see section 3.1.1).

The primary role of CSO and CTO ontologies is to express, in a formal manner, the knowledge about digital shapes and the associated tools, which are common to all domain-specific scenarios addressed in the three domain ontologies. In other words, the common ontologies constitute the result of our efforts to integrate the various domain ontologies. An overview of the rationale, the structure and the objectives for creating the common ontologies have been presented in this subsection. A brief description of the domain ontologies and two usage scenarios have also been given.

An ontology-based representation of digital models, as shown with the description of the developed ontologies, is able to provide an expressive characterization of shapes at different levels of abstraction and can ensure that existing tools such as Description Logic reasoners [Baader et al. 2003], can be used to reason on the repository and deduce information that is implicit in a digital model.

3.2 An Overview of the Digital Shape Workbench

One of the main achievements of the AIM@SHAPE project was the implementation of the Digital Shape Workbench (DSW) where resources and knowledge are integrated into a unified framework [Pitikakis 2010]. The DSW is intended to be used by the scientific community, mainly expert users (from universities, research institutes and the industry) and students. The Digital Shape Workbench is aimed at laying the basis for a common research platform (an e-Science environment) for modeling, storing, sharing, processing and reasoning about 3D resources (shape models, software tools & services, publications etc.) and their related knowledge.

The main features of the DSW are:

(a) storage for scientific resources (i.e. 3D models, software tools, bibliographic references etc.);
(b) management of resource metadata;
(c) management and maintenance of ontologies; and
(d) advanced searching, browsing and downloading of resources.

The DSW consists of the data repositories (namely the Shape Repository and the Tool Repository), the knowledge management system (Ontology & Metadata Repository), and a number of different ways of discovering, searching and browsing resources (Semantic Search Engine, Geometric-based Search Engine, Digital Library of publications). All the above components are geographically distributed across Europe. The overall architecture of the DSW is shown in Figure 3.14.

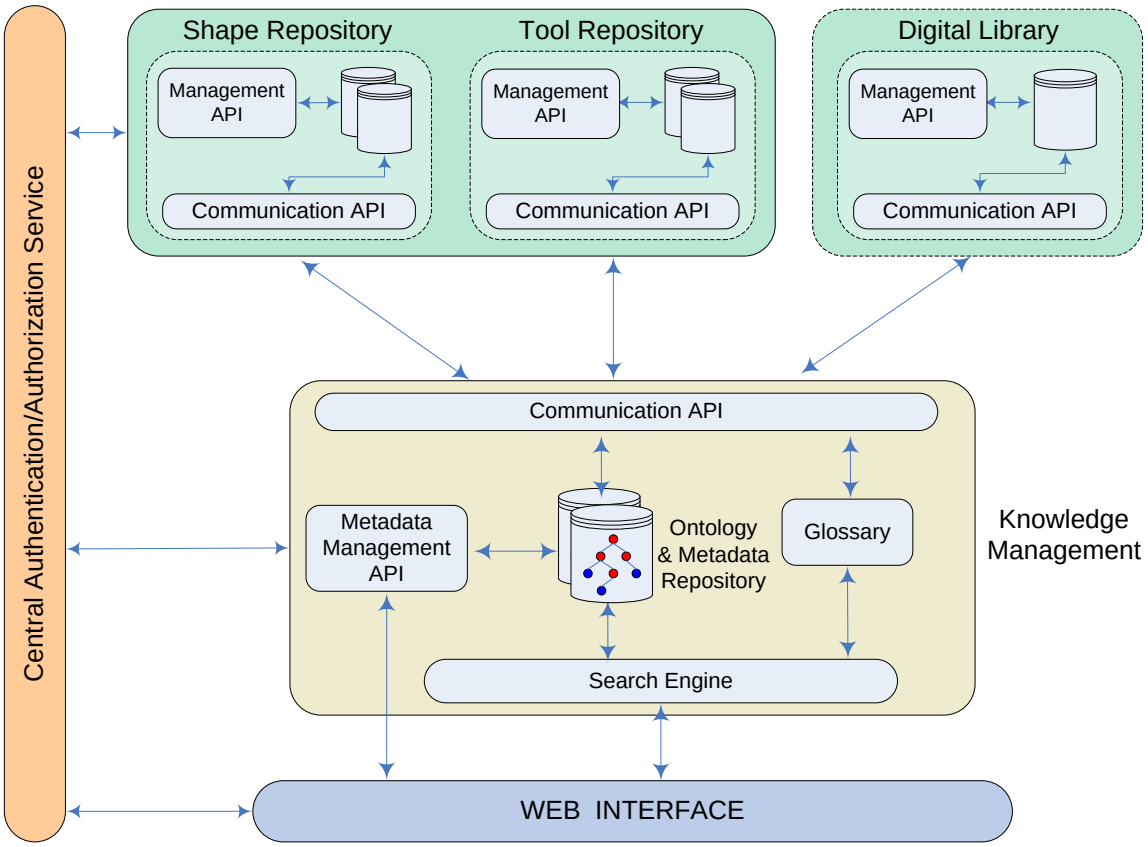


Figure 3.14: The overall architecture of the DSW.

The Shape Repository is a repository open to the research community and populated with a collection of digital models. Its primary goal is to include a variety of standard test cases and benchmarks, enabling efficient prototyping as well as practical evaluation on real-world or large-scale models. The Shape Repository has some unique features that

differentiate it from other repositories. It contains high quality models categorized and documented through ontology-driven metadata. It is important to note that existing repositories do not allow end-users to add new resources to the repositories. Our repository provides this functionality for authorized users (or user groups). During the upload procedure, it is required to classify the shape according to the CSO ontology and fill in the appropriate ontology-driven metadata information. For some types of shapes (e.g. meshes) it is also possible to automatically fill in some of the metadata values (e.g. number of vertices, number of faces etc.).

When a model is downloaded, its thumbnails and metadata are automatically bundled into a package, which is then sent to the user on agreement to the applicable licenses. It is also possible to choose the quality of the downloaded model.

Another unique feature of the Shape Repository is the support of "group models". It sometimes becomes necessary to have different versions of the same model with slightly altered properties, e.g. different re-meshing algorithms applied on the same model. Models logically related to each other can be stored as a single group with a representative model at the topmost level. Other models can be stored in lower levels and each level is accompanied by a "level description" which gives an idea of how models in that level are related to each other. Similarly, each group is accompanied by a "group description". With constant addition of models the repository has rapidly grown and it now contains more than one thousand shapes (521 models and a total of 1180 shapes).

The Tool Repository is an inventory of software tools that can be used in different stages of digital shape processing. It contains a variety of freeware tools and libraries developed by different research teams and vendors. The tools in the Tool Repository are organized based on a functionality hierarchy. For every tool a brief specification of its usage, limitations and capabilities are provided as metadata. Currently, there are more than eighty (80) software tools and libraries in the repository.

The Digital Library aims at the creation of a common repository of scientific references and technical reports which integrate the bibliographies of the participating Institutes and will become a reference point for publications on the topic of shape

modeling and semantics. Each item in the Digital Library is described by metadata. It does not store any copyrighted full text, but it provides a URI to the resource at the publisher site. The goal of the Digital Library is to strive for quality; entries are carefully checked for their relevance, completeness and accuracy and there are more than two thousand eight hundred (2800) references in the field.

The DSW also offers a common vocabulary/glossary of terms concerning digital shapes. It contains a selected set of over three hundred seventy (370) terms and their definitions which are distinctive in the domain of shape modeling.

The Search Engine provides a facility aiding in the discovery of shape resources and tools, as well as references to these resources. The fundamental components of the DSW search mechanisms are the inference mechanism for semantic-based searching [Vasilakis et al. 2005] and the geometry-based search mechanism.

The aim of the inference engine is not to restrict the user in searching only for shape resources (i.e. models, tools, references etc.) but to be able to acquire other information relevant to the resource as well, using semantic criteria based on the ontology and its' instances. This information may be either explicitly or implicitly described. Explicit information (metadata) includes datatype property values (i.e. specific alphanumeric values of properties), object property values (i.e. relations between classes or instances) or logical expressions/combinations of both. Implicit information can be inferred from explicit information about a resource.

To help the user in making efficient and appropriate queries we have developed a graphical user interface, see Figure 3.15, that abstracts the query API and the underlying reasoner logic. The visual interface provides the means to search in an intuitive and straightforward way without sacrificing the flexibility or expressiveness of the queries.

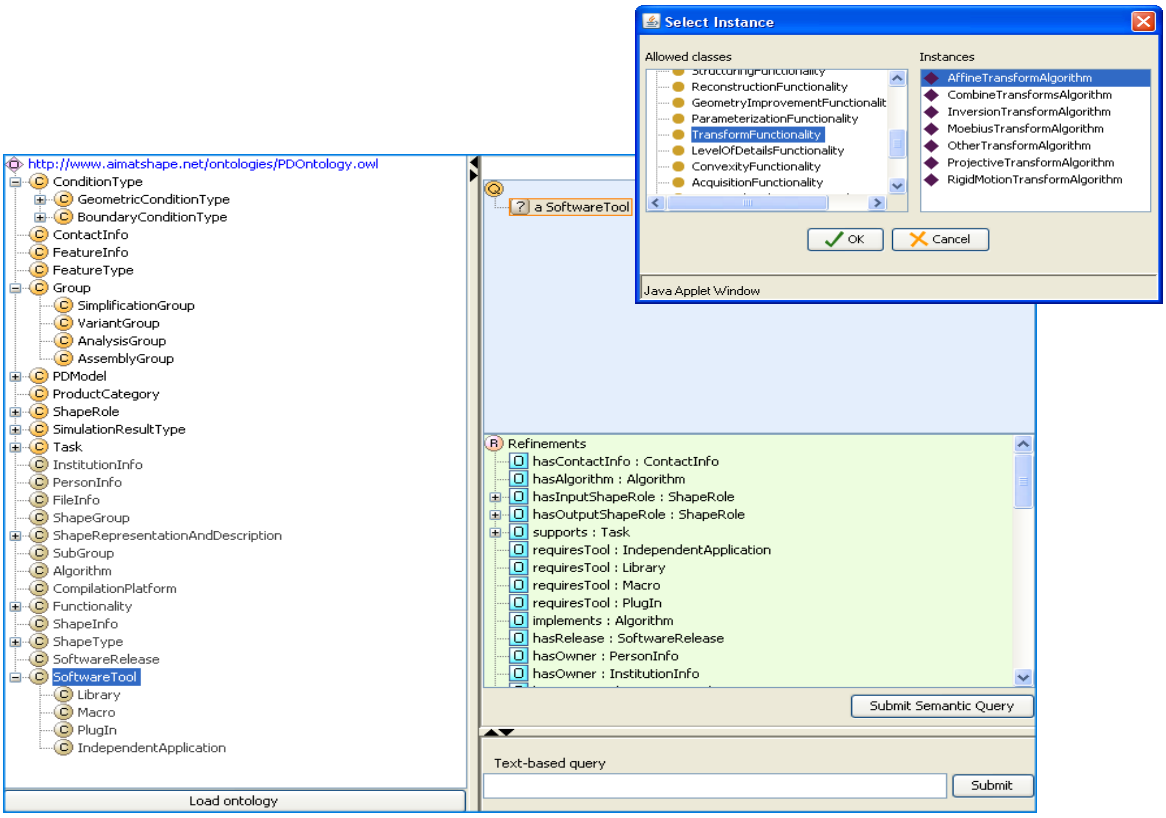


Figure 3.15: The Semantic Search Engine user interface

The user can express some semantic criteria based on which the search mechanism will extract certain information from the knowledge base. The user is guided in forming these criteria by visualizing all possible options and query refinements, depending on the current context, in an intuitive way. It is also possible to explicitly store and reuse predefined user-defined queries. This is helpful in displaying the wealth of information that can be accessed and the range of interactions that are possible.

The geometry-based search mechanism is able to search the Shape Repository according to different similarity criteria and matching mechanisms (e.g. global and partial matching, sub-part correspondence, part-in-whole) and according to different perspectives (e.g. geometry-based or structure-based). Geometric similarity between shapes is evaluated as a distance between shape descriptors that capture relevant properties of the shape. Tools for computing shape descriptors and for computing

distances between shape descriptors are implemented as stand-alone tools that can be plugged into the Geometric Search Engine (GSE). The GSE works with a variety of matching criteria depending on the shape descriptors and the comparison methodologies available.

An extensive user manual for the Search Engine is available online[9]. This is essentially a guide describing the graphical interface for the Semantic Search Engine but also includes relevant information regarding the Geometric Search Engine as well. The online manual presents information to help the user in:

- Getting familiarized with some basic notions related to ontologies.
- Understanding what is involved in creating meaningful queries.
- Taking advantage of the graphical interface to submit efficient queries.

There is a section for quick start tutorials which enables the user to quickly get started in creating, refining and submitting some simple queries. There is also a section for Frequently Asked Questions presenting some valuable information regarding common issues that may be encountered during querying the knowledge base using the Semantic Search Engine (SSE). The information presented in this section is also likely to help a new user to quickly start using the search engine.

3.3 The Ontology and Metadata Repository

In this section, the design and implementation of the framework that provides persistency and ultimately inference support for domain queries will be described [Pitikakis et al. 2010]. The Ontology and Metadata Repository (OMR) constitutes the knowledge base back-end that conceptualizes and provides persistency services for the stored knowledge. As it is depicted in Figure 3.14, it lies in the heart of the DSW and is responsible for effectively integrating all other components.

[9] AIM@SHAPE Search Engine Online Help: http://dsw.aimatshape.net/sse/help/help_toc.html

The OMR in essence is an ontology management system that allows storing, editing and accessing ontologies and ontology-driven metadata. The search mechanism's expressiveness and flexibility on performing the search is only constrained by the way that information is stored in the OMR and by the choice of the logical formalism that is used to describe this information. Hence, the OMR is closely related to the inference engine, which is necessary for semantic-based searching.

3.3.1 Design and Implementation of the OMR

The methodology for designing the knowledge base consists of several phases:

- the selection of the appropriate logical formalism that will enable the use of existing tools and reasoners to perform inferences on resources (i.e. shapes, tools, services or related references in our case);
- the selection, or implementation, of a storage back-end for persisting metadata and ontology resources;
- the selection of a specific inference engine that meets the criteria posed in the previous phases;
- the selection, or implementation, of a query language and the development of the appropriate interface for discovering resources.

The desired features of the knowledge base are: (a) to be *expressive* enough, in order to allow for powerful inferences on ontologies; (b) to be *declarative*, being independent of the specific ontology schema or contents; and (c) to be *intuitive*, by aiding the user in query composition in a user-friendly and transparent manner.

An ontology management system is comparable to a database management system (DBMS). It allows storing and processing of ontologies and metadata via a standard interface and relieves from the burden of deciding how the ontology is stored and accessed, how queries are processed and optimized, how query results are retrieved, etc. Ontology editing capabilities are not viewed as a critical component of an ontology

management system since there are well established independent, stand-alone ontology editors such as Protégé [Noy et al. 2000].

The requirements we are most interested in for such a management system are the loading of potential large ontologies (including their instances) to the storage system, the loading speed of these data into the inference engine (performance issues) and the efficient browsing and viewing of both ontologies and metadata. Ontology management and evolution is also a significant concern. Transactions like inserting, updating, and deleting metadata as well as altering the ontology structure, are considered as vital operations.

The knowledge-base was designed in a layered manner, dividing the responsibilities into various inter-dependent but loosely-coupled components. The architectural elements that comprise the software components of the Ontology and Metadata Repository are identified in Figure 3.16.

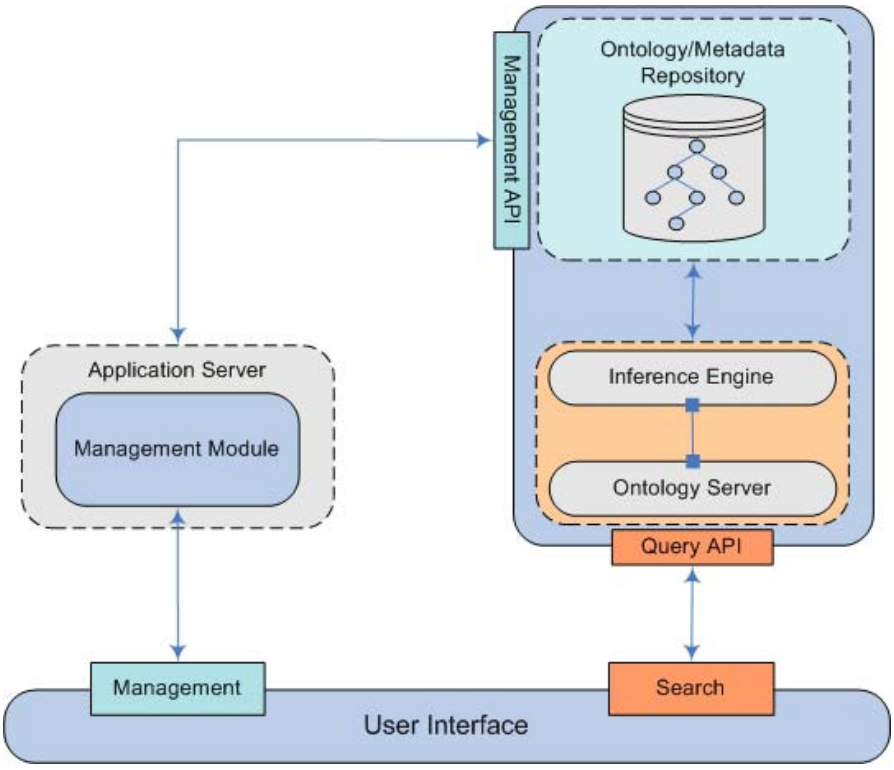


Figure 3.16: The Ontology and Metadata Repository general design architecture

As shown in the diagram, the core of the knowledge base is comprised of the Ontology and Metadata Repository and the Semantic Search Engine (inference engine and ontology server). There are two APIs for accessing knowledge base functionality: the Query API and the Management API. The search engine interfaces with the knowledge base through the Query API while the Management module provides functionality for managing the ontology schema and metadata.

A more detailed description of the OMR architecture is shown in Figure 3.17. This architecture and most of its components is generic enough to be used for other kind of knowledge as well, and not only for 3D content.

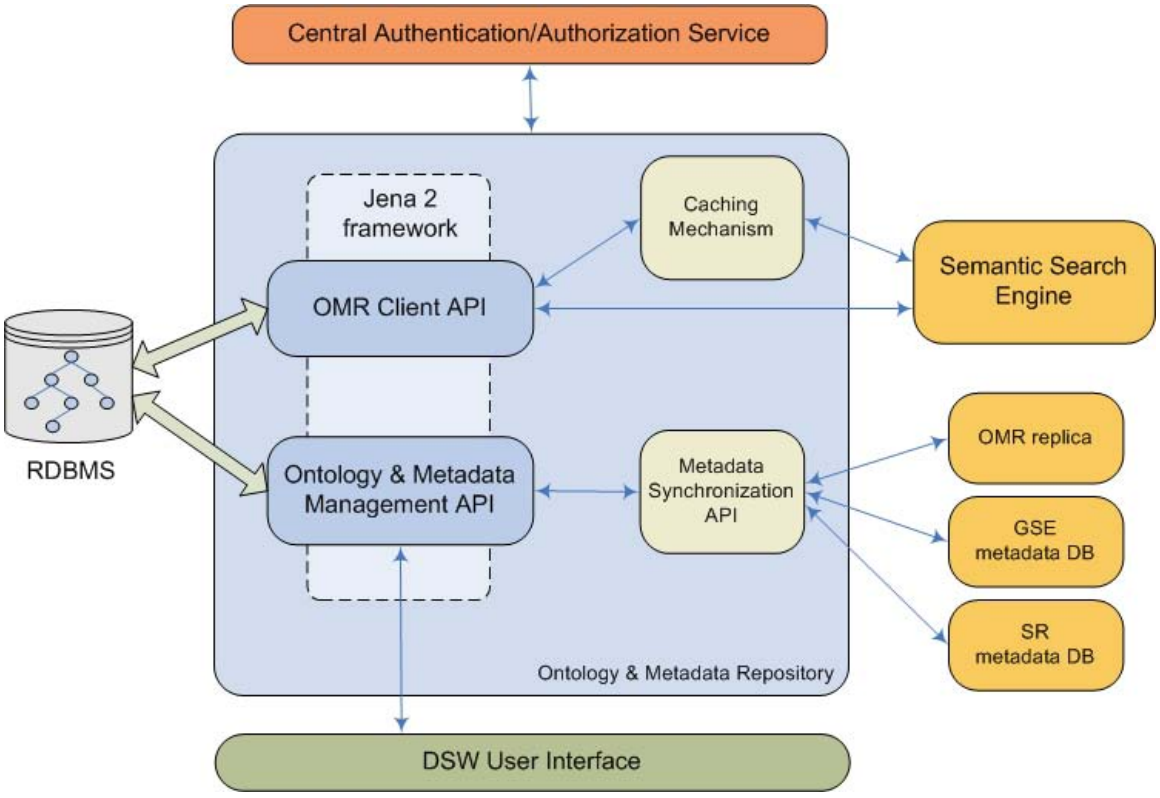


Figure 3.17: Detailed description of the OMR architecture

The ontology and metadata Management Module provides the basic persistence services. More specifically, it provides a set of ontology & metadata management and maintenance activities and is capable of storing and handling multiple ontologies.

The Management module, which is the core part of the repository, was developed on top of the Jena 2 framework [McBride 2002] to provide management and persistence services. Jena is available as open source Java software under a BSD license and contains a rich set of features for dealing with RDF and OWL, including OWL-based ontology processing. Jena provides a semantic layer which abstracts a lot of the underlying knowledge that is required to work with the OWL syntax. Jena's fundamental class is the ontology model, an API for dealing with a set of RDF/OWL triples. Jena ontology models are created from OWL files and stored persistently to a RDBMS using the appropriate API. In our implementation we used PostgreSQL[10] as the underlying RDBMS i.e. a metadata repository for enabling the basic persistence services in a scalable and reliable fashion.

The Management module extents the Jena API and provides the following:

(a) An API capable of storing multiple ontologies in the repository and providing a set of ontology management and maintenance activities like: inserting an ontology (from an OWL file or URL) in the repository; deleting a selected ontology from the repository; cleaning the RDBMS i.e. removing everything (ontologies and metadata) from the persistent storage; export/download stored ontologies and metadata to OWL files.

(b) A Metadata Editing API, providing a set of metadata management and maintenance activities as follows: insertion of instances (individuals) to a specific class of a selected ontology after filling in the required metadata; deletion of existing instances from a selected ontology model; editing/updating metadata values of a selected instance.

(c) An API for browsing/listing all the stored ontology structures (schemas) as well as all the instances (individuals).

[10] http://www.postgresql.org/

The Management Module's web interface was developed using the Java Server Pages framework (JSP). Apart from the central storage of ontologies and metadata, there are also:

(a) a complete Ontology & Metadata Repository (OMR) replica,

(b) a database containing (among other things) part of the OMR metadata information to be used by the Geometry-based Search Engine (GSE),

(c) a database containing a copy of all the Shape Repository (SR) related metadata.

All of the above are hosted at distributed locations and in order to keep all the metadata synchronized, an API for this synchronization has been developed. The update of all the metadata repositories is synchronous i.e. all the metadata information are updated concurrently to all the repositories.

A combination of local in-memory and persistent storage approach was used. While storing everything in-memory cannot be an effective method for manipulating extremely large volumes of metadata, it is very useful for caching purposes. Caching of ontology schemas is necessary because with imported ontologies the resulting schema that is being loaded can be (and in fact is for the AIM@SHAPE purposes) quite large. More specifically, caching is performed for two distinct reasons concerning the search framework:

(a) The ontology schema that is extracted from the database is cached to minimize loading time from the persistent storage (database) to the inference engine.

(b) The representation of the schema that is being used by the user interface (UI) framework. Caching the UI representation of the schema improves the necessary initialization time significantly.

Since ontologies were developed by different groups of people (domain ontologies and common ontologies) and continuously evolved over time, we needed a way to keep track of ontology changes. Ontology evolution and versioning is admittedly a hard research issue and beyond the scope of this work. We implemented a basic set of functionalities to store and manage different versions of the same ontology (updates).

Some additional information about the ontologies was also maintained like: imported ontologies (e.g. the Common Shape Ontology and Common Tool Ontology), version number, date of insertion and ontology model name. Moreover, a tracking mechanism has also been implemented for logging changes to resource metadata through the DSW user interface. All changes are kept for backup and security purposes.

One of the most important aspects of searching is "*knowing how to search*". In our case, knowing how to search is related to the user's comprehension of the domain that is being conceptualized. It also depends on the user's comprehension of the way that domain knowledge has been structured according to the developed ontologies.

Our approach for displaying and navigating through ontologies is to use a hyperbolic tree visualization technique. A hyperbolic tree visualization tool[11] was developed for these purposes (the Java applet shown in Figure 3.18) which can support management and browsing actions like displaying, inserting, editing and deleting metadata just by right clicking on a class/node in the hyperbolic tree.

[11] http://dsw.aimatshape.net/ontologies/ontologies.jsp

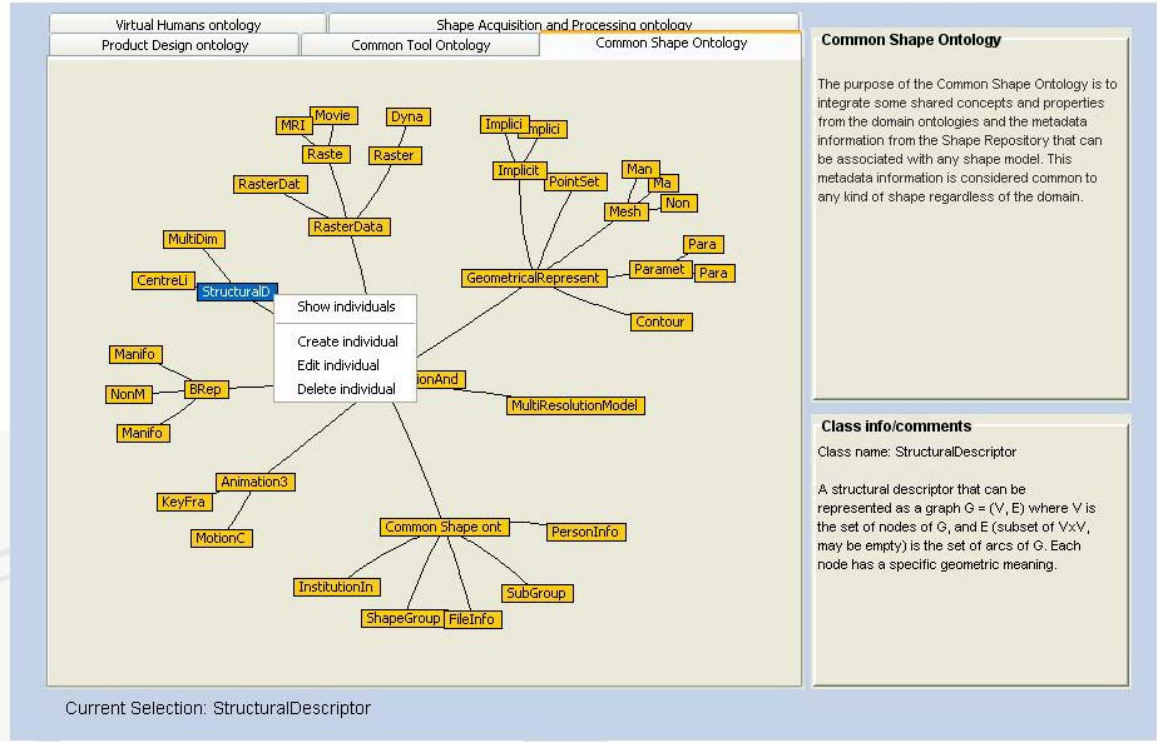


Figure 3.18: The hyperbolic tree visualization tool

Context-sensitive help has been integrated in the hyperbolic tree for any selected node i.e. the appropriate help content is displayed when a user selects a class (see bottom right panel in Figure 3.18). This help text is dynamically generated from the ontology schemas by reading the RDF comments that may be contained in class definitions.

Context-sensitive help has also been integrated in other OMR web user interface. For example, when inserting or editing metadata, comments are displayed as floating tooltips for every property in the ontologies that has relevant RDF comments. A typical example is the allowed values of a property or the definition/meaning of a property. Furthermore, the range of a property value (e.g. integer, string etc) is displayed as an indication of expected values. The range is also dynamically extracted from the ontologies (i.e. the property definitions).

To summarize, the following functionalities are offered by the OMR:

(a) uploading metadata regarding shape models, tools, and bibliographic references,

(b) browsing/listing all the stored ontology structures (schemas) as well as all their instances (individuals), either in a web page or through the hyperbolic tree applet,

(c) export/download stored ontologies and metadata to OWL files,

(d) editing of resource metadata,

(e) uploading, maintenance and version tracking of OWL ontologies,

(f) interaction with metadata extraction tools for automatic metadata generation,

(g) support for pluggable reasoning modules (inference engines).

3.3.2 Integration with the Semantic Search Mechanism

Jena has a set of internal/build-in reasoners (a "micro" and "mini" reasoner for OWL Lite and one for OWL DL), but for our purposes they are insufficient performance wise and their inference API is limited. Jena can, however, be easily combined with existing reasoner implementations and allows direct access and communication with the reasoner. In our case, the Racer Pro[12] v1.9.1 description logic (DL) reasoner [Racer Systems GmbH & Co.; Fikes et al. 2003] was utilized for its inference and deductive reasoning facilities.

To ensure independence of reasoner specific functionality, the layer titled as Query API has been built on top of Racer (see Figure 3.16). It is the protocol that binds the Semantic Search Engine interface to the inference engine. Upon the specification of this protocol lies the flexibility and openness of the design. The benefits from choosing a protocol that is a standard, or has the potential to become one, are obvious. One could easily plug in a different inference engine that implements the same protocol and the system would still work with no required modifications. In other words, the system is truly agnostic of the actual reasoner implementation that is being used.

We have selected nRQL [Haarslev et al. 2004] as the ontology query language for the knowledge base. The drawback to this design decision is that nRQL is not a standard and can only be used when communicating with back-ends that use some implementation

[12] http://www.racer-systems.com/products/racerpro/

of Racer. To mitigate this we have designed the Query API on top of nRQL following a layered architecture. The only element tied to nRQL is the nRQL Translator component of the API. By changing this component the API can easily be adapted to another communication protocol, for example OWL-QL [Fikes et al. 2003], much like changing a driver for a computing device would work. This ensures that the functionality provided by the Query API is not affected by changing the communication protocol and the implementation required for such a change is minimal. With this design we ensure that the functionality of the Query API is independent of reasoner specific functionality.

The integration of the Semantic Search Engine (SSE) and the OMR is considered a critical task since the ability of efficiently answering semantic queries is based on the communication of the SSE and the OMR. For that purpose, the development of an OMR client API for communicating with the SSE took place.

This API provides the necessary methods for loading ontologies (schemas and metadata) from the OMR to the SSE, getting class descriptions and other ontology information, exporting ontologies (schemas and metadata) to files in order to be used by the SSE etc.

The API for metadata and ontology management was extended to support the sharing of commonly used APIs between the OMR and the SSE.

The API was also modified to support caching of ontology schemas. Since performance is an important consideration for the SSE, cashing is necessary because with imported ontologies the resulting schema that is being loaded to the SSE inference engine can be fairly large. Therefore, the ontology schema is cached to minimize loading time from persistent storage (database) to the inference engine.

The Ontology Server API also supports remote management of the Ontology Server. In particular it provides support for loading specific ontologies from the OMR directly to the Inference Engine, restarting the Ontology Server and resetting the inference engine. All the above contribute to the improved fault tolerance of the Semantic Search Engine.

4 A SERVICE-ORIENTED APPROACH TO E-SCIENCE

Ontologies play a central role in empowering Web Services with semantics that can enable service discovery and composition. In this chapter, we propose a semantic organization and description of Web Services as an important requirement for enhancing the ability to compose processing chains (workflows) of Web Services. We adopt a knowledge-based approach that effectively utilizes the knowledge infrastructure presented in Chapter 3.

We address the issues of vertical and horizontal integration, by developing a conceptual and technological platform offering undemanding composition of complex workflows of services.

Next we discuss the overall Semantic Web Service interaction process and we partition it into three main phases. We then propose a generic service oriented middleware architecture for orchestrating composite services.

As proof-of-concept we provide a prototype implementation of shape processing Web Services which can be easily combined into workflows. Two workflow scenarios present the general conception of pre-processing, polygonal surface generation and reconstruction, and post-processing, demonstrating a part of the 3D model processing pipeline.

4.1 Semantically Enriched Services and Workflows

Scientists tend to collaborate and they most likely use the same resources during their collaboration. Resource sharing in a (predominantly) scientific community is the driving force of computational grids. Over the past several years, with the advent of the Open Grid Services Architecture (OGSA) [Foster et al. 2002] and the Web Services Resource Framework (WSRF), Service-Oriented Architectures (SOA) and Web Service technologies have been embraced in the field of High Performance Computing, offering

us the potential to make scientific infrastructures simpler to use, more cost effective to implement, easier to maintain and more importantly easier to reuse.

The next generation of the Web promises to deliver semantically enriched Web Services so that they can be directly utilized by application agents or other services. A great technological challenge is thus to support and enhance such activities through mechanisms that enable a flexible and efficient exploitation of resources, distributed in separate locations and managed by different organizations. It is worth to note that an important issue to consider is whether service composition (i.e. combining a set of operations in a specific way) provides an added value.

The semantic organization and description of Web services is a crucial requirement for enabling the effective composition of Web services. The industry and academic worlds have both proposed solutions that progress along different dimensions regarding Semantic Web Services. Academic research has been mostly concerned with expressiveness of service descriptions, while industry has focused on modularization of service layers.

We believe that the success of the Semantic Web evolution depends to a great extent on proving its scalability from simple application scenarios to knowledge-intensive and computationally-complex ones. As a case study, we address e-Science applications focused on the knowledge lifecycle of shapes such as geometry processing.

In this sense, our primary objective is to demonstrate how formalizing, sharing and re-using expert knowledge can effectively improve shape processing and scientific computations in general. Our aim is to provide a framework that can support the development of sophisticated (“intelligent”) services which can be utilized according to user’s needs or context.

This is achieved via a structured information space where scientific resources are documented with reference to domain specific ontologies, enabling efficient discovery and access both for humans and computer programs. Metacomputing is integrated in this information space by linking workflow definitions back to the ontology. It is thus possible not only to find resources (data and programs) that can be combined into

workflows but also to find workflows which will produce data objects of a given class that are automatically annotated with metadata according to the specific ontologies.

In Chapter 3 we proposed a knowledge management platform that constitutes a common semantic-based information avenue that address shape processing needs. This knowledge-based approach makes extensive use of domain knowledge and can be effectively combined with Web Service technologies offering a next generation e-Science framework.

4.1.1 Information and Computation Integration

Our approach towards a system architecture and implementation that combines the principles of Web Services and the Semantic Web to support scientific computing for a given application domain, addresses two types of integration: vertical and horizontal integration.

As we presented in [Lalis et al. 2005], *vertical integration* refers to the specification of the dynamic process of bringing together, at runtime, a set of functional elements in order to implement application-specific functionality. The aim is to focus on tuning the various parts of the system in order to fulfill the needs of the application in the best possible way.

Vertical integration serves the purpose of extending the information aspect to include the, typically dominant, computational aspect found in scientific environments. This is achieved by associating data concepts for which instances can be produced as a combination of data and program objects with the corresponding workflows. Hence, when searching for information and arriving at that particular data concept, it is possible not only to discover and access the currently available objects but also to generate new ones by launching a computation. It must be noted that the user does not have to be aware of the existence or the internal details of the computing; in the ideal case, the system may infer the input parameters of the computation as a result of the controlled searching environment. What is equally important, when a computation finishes the produced data

can be automatically annotated with metadata that capture its provenance, and can be properly stored in the ontology-driven knowledge base of the system.

Horizontal integration is about classifying the passive functionality at a semantic level, presumably at design-time, which could be useful for implementing a wide range of applications and/or intermediate system elements (Figure 4.1). In our case, the focus is on packaging this functionality in components that are well-documented and at the same time easily accessible though Web Services. Horizontal integration is implemented via ontologies that contain concepts (classes) and instances (objects) of scientific resources relevant to the particular application domain. The properties of these resources are described via metadata which also contain information about the location and protocols/interfaces through which they can be accessed or invoked. Such an organization greatly simplifies the task of navigation and searching in a large information space. The ontologies and metadata, being encoded in machine readable form, can also be queried to locate and access available resource objects.

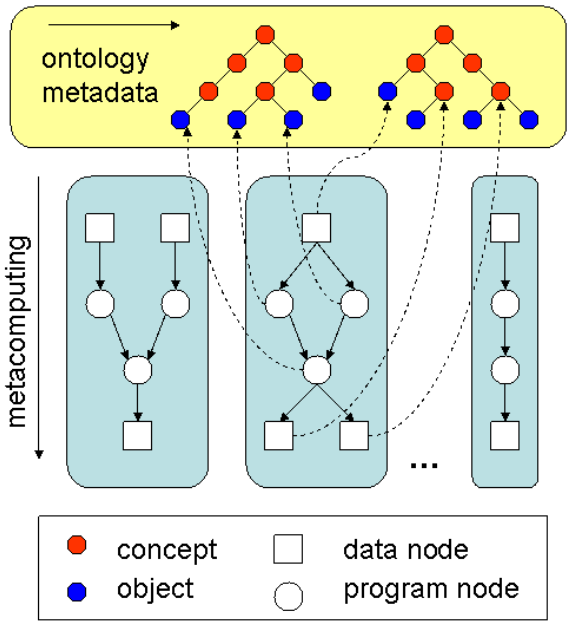


Figure 4.1: Horizontal and vertical integration for knowledge-centric computing

The vast majority of shape related applications or processes to date are largely vertically integrated, and without any documentation about their internal structure. However, there is little or no horizontal integration. Most applications are built in a monolithic custom way, using specially built data processing and information integration elements, sometimes even special sensor and computational infrastructure. What is probably worse, custom stand-alone development is perpetuated since little or no effort goes into making the individual functional components universally accessible and reusable.

While vertical integration seems to be the easiest way to implement a shape processing application (and probably still is at this point in time) in the long term this approach will inhibit fast and robust deployment. Moreover, considerable resources could be wasted just to re-invent the wheel, simply because no resources were invested to componentize individual elements and/or make them visible and accessible to others.

Admittedly, horizontal integration is a hard problem. It requires considerable experience to single out what is generally useful from a large collection of functional elements that may be available (or possible to implement). Notably, this is simpler to achieve for elements that sit on the lower levels of the chain. It becomes quite harder for elements that are higher up in the chain, since application semantics, which may differ considerably from case to case, play an increasingly important role in higher layers. The context of design and engineering adds even more complexity to this category of integration because it faces the difficulties related to shape processing as well as processing the semantics attached to them. This should not prohibit us to target better solutions, and by aggressively focusing on the specific stages of the process composition to take a first step to properly address this high complexity.

We believe that investing effort in achieving horizontal integration yields substantial benefits. In fact, it may be the case that most of the resources that were regarded as being “highly application specific” can be implemented as (simple) refinements of more generic components. This integrated approach allows users (and computer programs) to interact with and explore a large but well organized information space in an accurate

way. Even complex data objects can be promptly found or generated on demand, without dealing with the technical details. With fast enough computing and communication infrastructure (or lightweight enough computations) it would be hard to distinguish between retrieving available data from a repository and generating it via metacomputing. Last but not least, efficient searching is not limited to data objects alone. It becomes possible to search for program objects based on the data objects required as input and produced as output, as well as for composite service descriptions based on their data and program components and results.

We can identify three kinds of semantic descriptions: data semantics, functionality semantics and user task (process) semantics. Data semantics of scientific objects are captured by ontology-driven metadata. Functionality semantics are formalized by the application of (Semantic) Web Services. Processing task semantics are formalized through workflows.

The notion of a workflow, defined as an ordered sequence of tasks or activities, related by data and control flow relationships, is used to describe the computational aspect of the creation of processing pipelines. A task is typically performed by executing a program, enacting a human/machine action, or invoking another process. Programs, persons, machines, and data used to perform workflow processes are called *workflow resources*. The composition of Web service workflows is a burden due to the complexity of the composition process. Helping composers to simplify or even automate this process plays a major role in enabling the envisioned Semantic Web Services paradigm.

4.1.2 Ontology-Driven Service Discovery and Composition

The aim of research efforts around Semantic Web Services is to facilitate easy and perhaps automated handling of web services. Initial web service efforts failed to hold the promise of effortless or even automatic interaction and dynamic composition of web services. The reason is that the web service technology stack does not supply sufficient

means for describing web services in a way that supports generic mechanisms for discovering, composing, and executing web services.

By augmenting web services with rich semantic descriptions many aspects of their management can become automatic. Specifically, web service discovery, composition and mediation can become dynamic, with software agents able to reason about the functionalities provided by different web services, to locate the best ones for solving a particular problem and to automatically compose the relevant web services into workflows and build applications dynamically.

Service composition is a far from trivial problem. It must be as less manual as possible (preferably fully automatic) and deal with discovering complex service combinations, conditions and preferences requirements, and with heterogeneous results provided by several services. The purpose of discovery in the context of abstract workflow mediation is to provide mechanisms for finding services which can deliver the missing pieces of information or can bridge the identified incompatibilities by exploiting the knowledge stored in the system. Notice that in open dynamic environments discovery is a pure necessity because process components simply cannot be assumed to know all suitable workflow components. While service discovery is an active research field, it turns out that traditional discovery approaches are not well suited for use in the context of scientific computations.

From the process mediation perspective, new services need to be discovered based on the identified needs. During this phase, “gaps” are identified which cannot be bridged by using any known services. To bridge such gaps, new services need to be discovered. Very often a specific gap cannot be bridged by any existing single service and a combination of several services must be used. This introduce a problem, because, traditionally, matchmaking algorithms [Bellur & Kulkarni 2007; Klusch et al. 2006; Paolucci et al. 2002] used in service discovery components consider only one service as a suitable candidate, while combinations or compositions of services are not considered. Therefore, when querying existing components with a service request based on the

identified gap, usually no match can be found, although some existing combination of services (workflow) would be a suitable match.

The reason why usually only one service is considered as a valid match is motivated by the requirements service discovery components need to fulfill: service registries are expected to store large numbers of services and at the same time the best matching set of services for a given query has to be retrieved in a timely manner. Such a combination of requirements makes it difficult to employ full-fledged composition algorithms [Alarmi et al. 2006; Traverso & Pistore 2004; McIlraith et al. 2002] during the discovery process simply because their time complexity is unacceptable for discovery purposes (composition algorithms usually assume a rather small number of services).

Developers of scientific tools are not experts in Grid technologies, irrespective of the simpler and more consistent APIs being provided. From their perspective, the services that are most relevant are services that perform a scientific operation and where the semantics of the operations are defined in terms of the domain science. The tool developers would rather focus on the algorithms and the appropriate interfaces to present to the scientific end-users. [Foster 2005] introduces the term Service-Oriented Science. In his words, “Grid technologies can accelerate the development and adoption of service-oriented science by enabling a separation of concerns between discipline-specific content and domain-independent software and hardware infrastructure.” Enabling access to scientific applications and tools using a service-oriented approach allows the developers of scientific tools to focus on the domain science, and delegate the management of the complex back-end resources to others who are more proficient in SOA middleware.

We address these problems by developing a conceptual and technological platform able to encapsulate 3D resources into a semantic layer for efficient storage, retrieval and reuse of these resources [Pitikakis et al. 2005]. This objective is achieved by providing access to existing scientific and computational resources and by allowing users to assemble these resources together in order to generate complex functionalities based on Web Service workflows. The capability to compose complex services strongly depends on the conceptualization of the domain. The conceptualization of the relevant concepts

and the relations among them are represented by ontologies able to capture expert knowledge.

The development of such a knowledge-based system involves:

- Formalization of domain knowledge through ontologies, starting from the basic building blocks (tools, services, applications, data sets), that allow a shared understanding of the domain and make explicit descriptions of generic functionalities.
- Grouping of basic functional elements in order to implement application-specific functionality. This is addressed by the development of Web Service workflows.
- Incorporation of semantics into service descriptions and service composition through the use of loosely coupled, reusable software components.
- Knowledge discovery that can help the users assemble relevant information for effective decision-making and semantically enriched services, to improve their capability to perceive and utilize system knowledge. This can also help the users discover and assemble services into processes for easier and better quality of workflow executions.

4.2 Phases of Semantic Web Service Interaction

Seamless service composition requires clearly identifying application requirements (e.g., business goals, constraints, and preferences), unambiguously describing the services, precisely and formally specifying the orchestration model, and adopting flexible composition schemes. The central idea in our approach for supporting service composition is the use of ontologies to semantically enrich services and a selection framework to define explicit and implicit (ontology based) dependencies to build composition schemas and instances. Services are usually either data services or computational (process-based) services. Data services are about exposing data through

services e.g. the Amazon Simple Storage Service[13] (Amazon S3). Computational intensive services are mostly concerned with providing an added value resulting from computations that involve input data e.g. the Amazon Elastic Compute Cloud[14] (Amazon EC2).

Clients and services can act as agents, where, in this context, "agents" include requesters (clients), service providers, and middleware. We assume the following general capabilities of agents in Semantic Web service environments:

- All agents can access published ontologies, and can communicate using messages whose content is represented and can be interpreted in terms of these ontologies.
- Service providers publish semantic descriptions of their service capabilities and interaction protocols, which clients can use and interact with when selecting appropriate services.
- Requestor agents can use these semantic service descriptions as guides to construct processing workflows.

Clients can utilize the published semantic descriptions of services to help them decide which services to use and how to interact with them. As a result of this style of interaction, clients are able to discover and substitute services with similar functionality e.g. different implementations of the same or new algorithm.

The overall process of discovering, composing and interacting with a Semantic Web Service, in general, includes three phases:

- *Candidate service discovery* is the search for available services that can (potentially) accomplish a client's task. This process is mostly done manually e.g. querying a service registry, however it can be at least partially automated possibly using ontology-driven semantic searching.
- *Service engagement* includes the process of interpreting candidate Web service enactment constraints (partly or fully described in each service's published semantic description), service orchestration and negotiating with prospective

[13] http://aws.amazon.com/s3/
[14] http://aws.amazon.com/ec2/

services until reaching an agreement. This phase concludes when both service and client agree to the service provision terms in an explicit or implicit service contract. Negotiations could include service price, the quality and timeliness of service, security and privacy, and so on.

- *Service enactment* is the process that finally completes the service execution. It is also possible to monitor the service process's status during execution.

Figure 4.2 gives an overview of the process and relationships among these three phases. A client starts with a task (goal) that it intends to accomplish via a service request. Correspondingly, services have explicitly represented semantic descriptions of the task functionality they were designed to provide (often in exchange for some form of compensation). In scientific computing, functional requirements are probably the most important aspect of service discovery. Providers must be able to describe the capabilities and constraints on offered services.

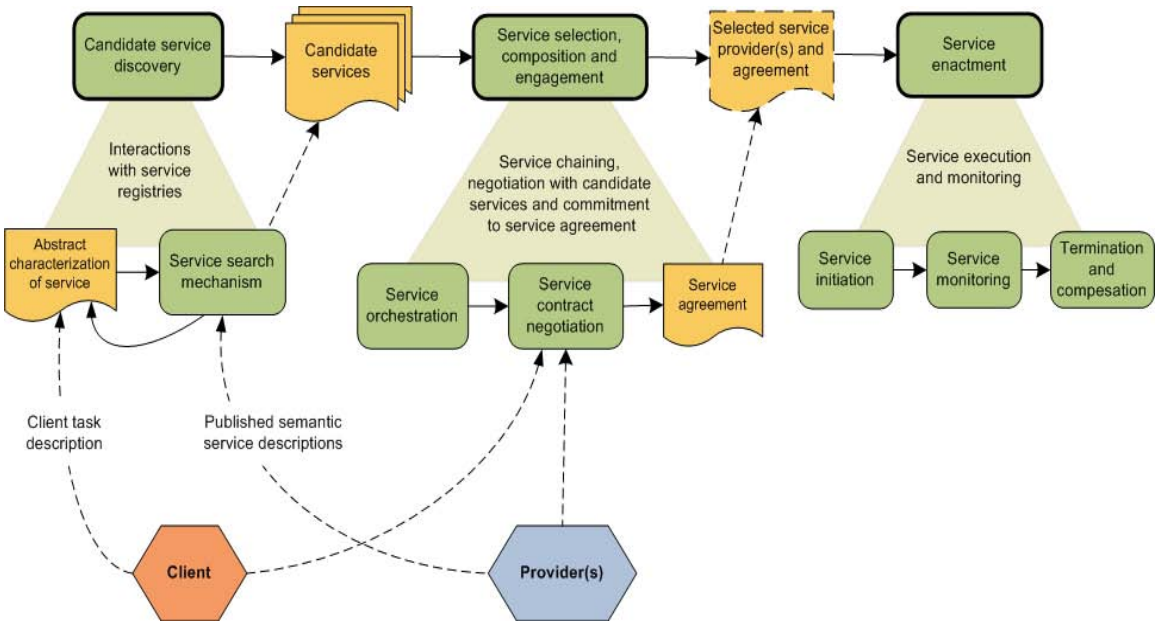


Figure 4.2: Main phases of the service interaction process (discovery, engagement and enactment)

In the overall service-utilization process, service requests (messages from clients to prospective services) and service orchestration are part of the engagement stage. Service engagement is the initial phase of interaction between a requestor and a potential provider. The result of this phase is an agreement, explicitly or implicitly, between requester and provider that a specific service will be provided. Functional requirements (e.g. contract negotiations and service agreement) and architectural requirements (negotiation protocols, negotiation services, auditing services) can vary according to the complexity of the negotiations. A provider can simply honor these service requests with no official acknowledgement or agreement, in which case we move directly to the enactment stage. Alternatively, these requests can result to multistep contract negotiations and service agreement. These issues are beyond the scope of this thesis. In our framework, the discovery and engagement processes lead to an implicit service contract.

Interactions during the service enactment stage include initiating the service activity, monitoring the service execution, and confirming service completion. If a contract was formed, it is also possible to address compensation issues. Functional requirements for service enactment include: choreography interpretation and execution, dynamic service composition, process-status monitoring and event notification. Other requirements include service-failure handling and compensation, dispute resolution and compliance, and audit tracking.

We propose an approach for the composition of services into processing workflows, which consists of the following four conceptually separate stages: specification, matchmaking, selection, generation and execution. The specification phase enables high level descriptions of the desired compositions. The matchmaking phase uses composability rules to generate composition plans that conform to composers' specifications. By composition plan, we refer to the list of component services and their interactions with each other (plugging operations, mapping messages, etc.) to form the composite service. Composers select a generated workflow (selection stage) possibly based on some quality of service (QoS) criteria (e.g., performance, cost, etc). Using the selected plan, a detailed description of the composite service is generated (generation

stage) and executed using a service engine. In the next sections we present the design, development and deployment of such an architecture.

4.3 Overview of our Service-Oriented Architecture

Composing services together is a new challenge for SOA middleware meeting e-Science environments and scientific computing. The variety of services requires the development of models, techniques and algorithms in order to create composite services and execute them. Composing services is technically performed by chaining interfaces using a syntactic or semantic method of matching.

The specification of a service composition requires dealing with two major issues: service description and orchestration. Services are perceived as black-box components with well-defined interfaces, they are accessed using XML message exchanges and metadata in the form of WSDL are used to describe abstract interfaces and concrete endpoints.

The execution order and conditions of such a workflow are specified through an orchestration model/language. The generation of composite services requires a clear and unambiguous description. In a SOA-enabled environment, we build application workflows by orchestrating various services in the right order (sequencing, conditional behavior etc.) via BPEL. BPEL supports the description of abstract and executable processes. Let us continue with a brief description of the essential to our study BPEL components. The reader is referred to [Alves et al. 2007] for more details.

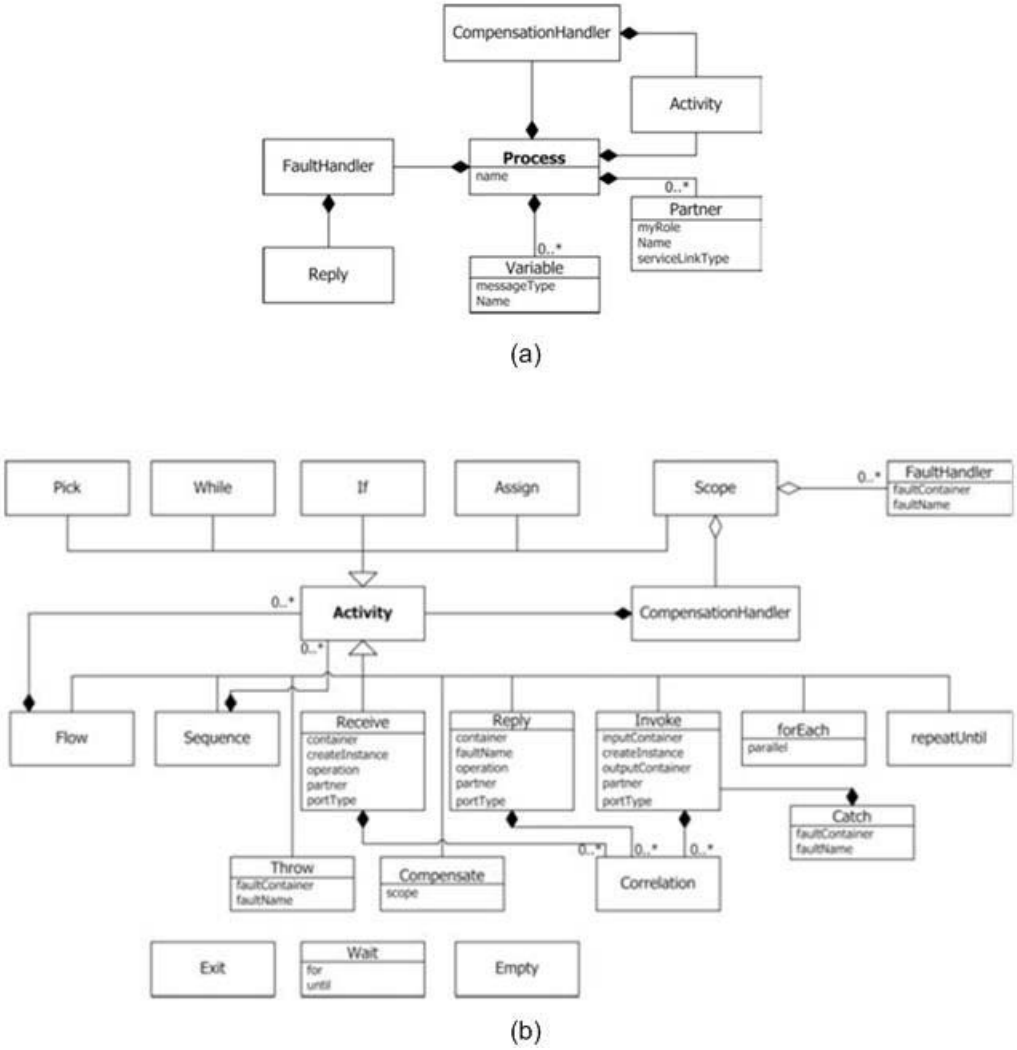


Figure 4.3: (a) BPEL process metamodel, (b) BPEL activity metamodel

The overall structure of BPEL consists of partner / partner links, variables correlation sets, fault handlers, event handlers and compensation handlers. Figure 4.3(a) shows the BPEL 2.0 process metamodel. *Partner links* are parties that interact with the business process. *Variables* describe data that are used by the business process. *Correlation* sets are sets of properties shared by messages in a correlated group. *Fault handlers* are activities that must be performed in response to faults. *Event handlers* are invoked

concurrently if the corresponding event occurs. *Compensation handlers* are wrappers for a compensation activity.

Figure 4.3(b) describes the BPEL 2.0 activity metamodel. *Basic activities* are atomic activities for describing a process with BPEL. <*assign*> updates the values of variables or partner links with new data. An <*invoke*> is a synchronous (request/response) or asynchronous call of a web service. <*receive*> waits for a message to arrive. If the message requires a response, it is send back by a <*reply*>. <*throw*> signals a fault from inside the business process, which is processed by fault handlers. <*exit*> quits the behavior of a business process instance. <*wait*> stays for a given time period or until a certain time has passed. <*empty*> is a no-operation activity, that means it is used to specify that no action is executed. This activity is used for fault handlers that consume a fault without acting on it. Other use cases for the empty activity include synchronization points in a flow, or placeholders for activities that are to be added later.

Structured activities contain other activities for defining the recursive composition of complex processes. <*sequence*> executes the activities one after another. <*while*> indicates that an activity is to be repeated until a certain stop criteria is met. <*switch*> selects one branch out of several activity branches. <*flow*> executes activities concurrently or in any different order. Dependencies between activities are defined by <*links*>. <*pick*> blocks and waits for a suitable message to arrive or for starting a time-out alarm. <*scope*> handles a set of activities for describing a nested activity. A scope may be associated with a fault handler, an event handler or a compensation handler.

The orchestration of executable processes is then deployed and executed by a workflow engine in a Java Business Integration (JBI) enabled platform (see Figure 4.4). JBI [Christudas 2008] provides an open application integration framework, is built on a Web Services model and provides a pluggable architecture for different Service Engines (SE). It is a standard meta-container for integrating service containers that can host service producer and consumer components (service units). OpenESB[15] implements an Enterprise Service Bus (ESB) runtime using JBI as the foundation. This allows easy

[15] http://java.net/projects/open-esb/

integration of web services to create loosely coupled enterprise class composite applications. It also provides various tools for the development, deployment, and management of composite applications.

The Java EE Service Engine acts as the bridge between Java EE applications and JBI. A Java EE application archive (ear/war/jar) can be packaged in a JBI composite application and be deployed in a JBI server. We are using the JBI runtime that has been integrated with the GlassFish application server.

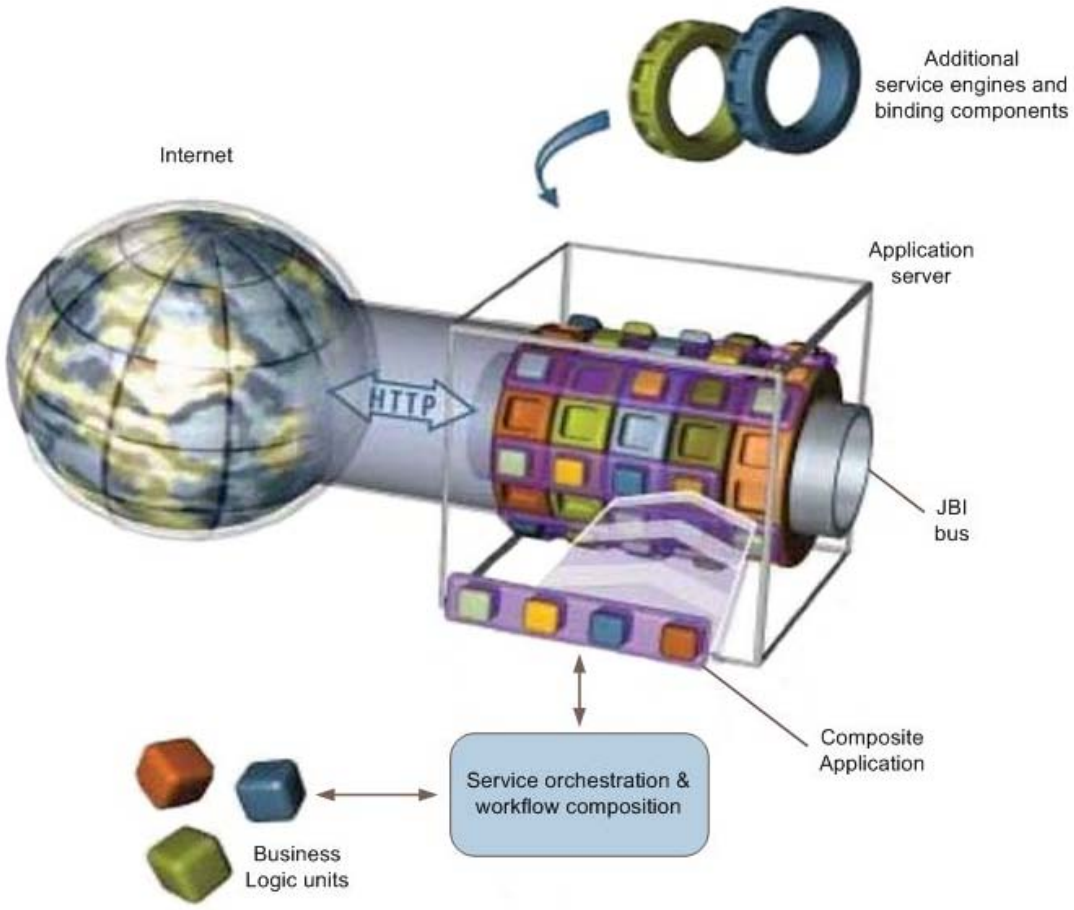


Figure 4.4: Service oriented JBI architecture

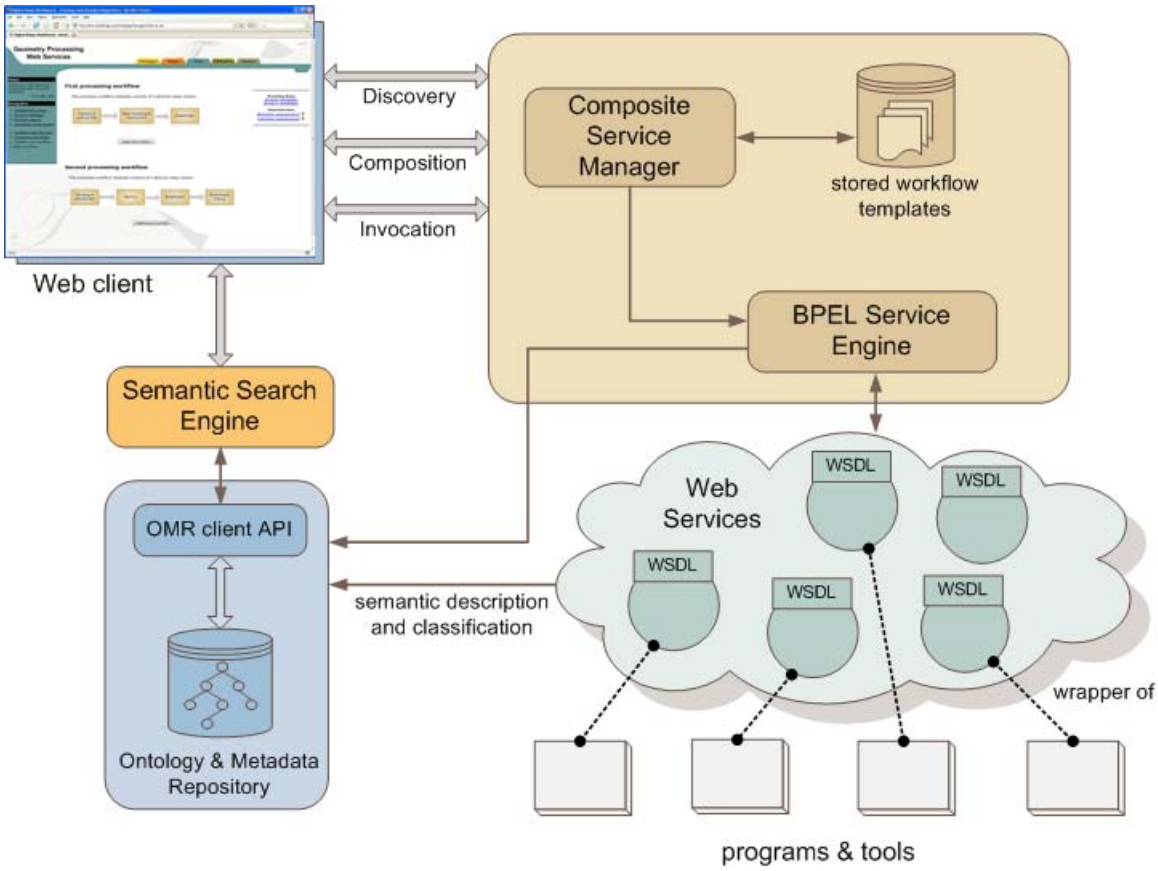


Figure 4.5: The proposed semantically enriched Web Service architecture

We present our system architecture in Figure 4.5. The architecture is organized into three layers. The first layer includes a set of scientific applications and tools. The second layer contains associated Web Services. These Web Services could be either simple or composite. Some of the simple Web Services just wrap applications and tools defined in the first layer. Composite services are generated as processing chains (workflows) of simple Web Services. The third layer includes the service composition component, the BPEL service engine and a Web client GUI and manager. Three types of requests are submitted through the GUI: discovery, invocation, and composition of Web Services into workflows. During the service composition process, there is communication with the knowledge base for discovery and selection of services. It is also possible to utilize stored workflow templates (abstract workflow definitions) for the generation of new processing

services. These abstract workflows are stored in the form of deployed BPEL modules. In addition, the metadata of the workflow results are stored back to the knowledge base for future reference and reuse. The knowledge base contains ontologies for describing domain constraints, objectives, and preferences.

4.3.1 Dynamic Workflow Composition and Execution of Web Services

Dynamic selection and composition of loosely-coupled Web Services is increasingly used to automate either scientific or business processes. Management challenges such as dynamic adaptation of running process instances, discovery and selection of services to be used have to be addressed.

There are three different ways to invoke a Web Service:

- static binding
- dynamic binding
- dynamic invocation

With static binding the client is compiled and binded at development time. This binding is tightly bound to one and only one service implementation. It provides the fastest performance but gives the least flexibility.

With dynamic binding the only part of the client that is compiled at development time is the interface to a service type (i.e. the WSDL *portType* definition). At runtime the client can bind to any service implementation that supports that *portType*. It generates a dynamic proxy from the service's WSDL binding at runtime and casts it to the interface. There is a slight performance reduction though, but in exchange you win a lot of flexibility. Using this technique, the application is able to connect to any number of different service implementations without modification.

With dynamic invocation, there is no need to compile anything at development time. Instead everything is done at runtime. The application retrieves and interprets the WSDL at runtime and dynamically constructs calls. This method gives the most flexibility, but

also requires a much more sophisticated client and there is a decrease in terms of performance, which occurs on each invocation.

Partner links describe interfaces (messages and operations), transport protocol, and most importantly, the location of each service to be used. BPEL processes call these external services using information stored in the *partner links*. The partner links define operations and message types that make up the interface to the service using *portTypes* in WSDL. *portTypes* also indirectly define the transport used to communicate with the service (bindings) and the location of the service (service address).

Commonly, process designs in BPEL include static *partner links* that refer to a single external process selected by the developer at design time. This approach is appropriate for most systems; however, scientific workflow processes are more complex. They could interact with multiple external services and define multiple *partner links*, and some of these *partner links* might not be known at design time. As a result, all potential callouts and logic for deciding which *partner links* to use is usually built inside the business process itself, unnecessarily complicating that process. Furthermore, as additional services are added, the resulting process grows more and more unwieldy, as any changes to the *partner links* require modification of the entire process.

In our implementation we are using the dynamic binding technique. This approach eliminates the need to anticipate and manage all relationships at design time. The BPEL language supports the concept of dynamic binding of *partner links* by shielding processes from web service changes and letting the system manage *partner links* dynamically at runtime.

The WS-Addressing[16] standard provides a mechanism called endpoint references (EPR) that allows selecting one of the available services or even defining new services at runtime. The process statically depends on the interface information defined in the *portType* whereas an endpoint reference (which maps the binding to the service) allows us to redefine the service location dynamically. In essence, the endpoint reference is a dynamic alternative to the static service element defined in the WSDL. In our case, the

[16] Web Services Addressing 1.0 – Core http://www.w3.org/TR/ws-addr-core/

process/workflow designer can remain isolated from the decision about which services to call as long as those services conform to a standard (common) interface.

In our approach we use our knowledge-based framework for dealing with service composition and orchestration. Instead of creating hard-coded service discovery and routing logic, we utilize dynamic knowledge-driven service selection and binding mechanisms, triggered by predefined knowledge captured in an ontology.

We focus on a service provisioning and adaptation model that supports the process of assigning particular concrete service instances to the constituent activities of an abstract process (see section 4.4.1 for more details). This adaptation is transparent because it preserves the original functional behavior of the process.

This framework is used for:

(a) Service selection to control how concrete service endpoint references are assigned to abstract process activities. Specific policies for service endpoint selection could be used to find the best suited service instead of manual selection.

(b) Controlling runtime rebinding by user-defined service preferences and constraints to enable automatic knowledge-based selection of service endpoints. Usually policies attached to process activities are matched with services in order to find an optimal configuration that satisfies required user preferences and constraints. The selection could cover several QoS criteria such as performance, accuracy, reliability, availability, cost etc.

Ontology-driven knowledge controls the way discovery, selection, and binding are managed at runtime. Dynamic (late) binding is the process of transparently mapping an abstract service to a concrete service instance at runtime. Web Services registries do not provide support for dynamic binding. Hence, the service client is responsible for service selection and rebinding. Typically, the client queries the registry based on certain criteria, retrieves a list of services and then manually selects one service and tries to invoke it. The limitations of this approach become evident especially when dynamic rebinding

becomes necessary. The main problem is that the service client is responsible for taking all corrective actions (e.g., re-querying the registry for new service bindings, polling the registry for possibly new services, etc.). This requires considerable client-side code since current registries (and their APIs) do not provide support for implementing such dynamic binding.

Furthermore, the logic for discovering and selecting the best service (according to some criteria) cannot be easily encoded to the registry. Most registries are organized according to some kind of taxonomy and are using keyword search for service discovery. For example, the *seekda!*[17] free search engine for Web Services (with a directory containing more than 28500 services at the moment) uses a fully automated crawling process aggregating information to a semantic model (RDF triples). The collected metadata are filtered into an optimized index and a full text index. It has a basic keyword search facility which can be further refined by specifying the country, provider name or tag of a Web Service. Browsing for Web Services is done by using tag clouds, the country name of the provider, Web Service usage (most used services) or chronological order (most recent services).

These approaches are focused on business / commercial Web Services and are usually insufficient for scientific Web Services where relationships between classes need to be defined or semantic searching is required. Because of these limitations, we are not using UDDI or ebXML registries. Instead, we rely on our own ontology-based repository (OMR) for managing and querying the services metadata.

Our approach aims to address the aforementioned limitations of current approaches. Dynamic binding is handled transparently in combination with the service discovery. The service selection and (re)binding are enforced by our knowledge-based middleware. The latter queries our repository to find services matching the given criteria. Based on the available services and knowledge stored to or inferred from the ontology (possibly gathered from prior interactions), the user selects the service that best fulfils the functional (or potentially QoS) constraints.

[17] http://webservices.seekda.com/

Our runtime platform relies on the OpenESB[18] BPEL service engine, which can be easily integrated with the NetBeans[19] IDE, to execute a process and is able to trigger dynamic binding when needed to support composite scientific workflows.

4.3.2 Shape Processing Knowledge

Knowledge related to the acquisition and processing of a 3D model characterizes its lifecycle. Due to the intrinsic complexity of 3D models, ontology-driven metadata are necessary in order to reach a sufficient level of expressiveness and semantic impact. Metadata provide a thorough characterization of models by capturing information related to the processing history of an object, the possible actions that can be performed to it (e.g. smoothing, simplification, enhancement etc.), or the tools/services that can use it as input or produce it as output.

In addition, the formalization of the relations among tasks (e.g. workflow steps), programs/tools (e.g. Web Services) and data (e.g. 3D models) is necessary to specify processing pipelines. As already mentioned in section 3.1.2, the Common Tool Ontology (CTO) can support the description of workflows by, for example reusing the concept of *SoftwareTool*.

In Figure 3.13, the concept *Task* is connected to the *ShapeRole* (and *ShapeType*) via the *hasInputShapeRole* and *hasOutputShapeRole* relations. Each *Task* also is connected to a previous or next task (via the *hasPredecessor, hasSuccessor* relations) and provides a well-defined functionality or software tools that implement this functionality (via the relation *isSupportedBy* and *hasAlgorithm*). The concept *Functionality* represents all the possible different tasks a tool can provide (e.g. Triangulation, Voxelization, Filtering etc). Instances of these functionalities can have references to specific tools in the Tool Repository or Web Services. This way, we could be able to answer questions such as: What are the possible steps we can do starting with a points cloud? Through what sequence of steps is it possible to reach a surface mesh? Is there a converter tool/service

[18] http://java.net/projects/open-esb/
[19] http://soa.netbeans.org/

able to convert an OFF file into a PLY file? Which type of shape (or format type) does this tool support? Which are the tools/services with functionality "xyz"?

The basic idea is that the ontology should maintain and provide all the possible "*template set of actions*" (depicted as workflow templates in Figure 4.5) we can activate considering a given input and/or output type.

Typically, a requestor has a specific task that needs to be achieved at a particular time on a particular shape, whereas service providers publish generalized descriptions of their service capabilities that will enable clients to find them. For effective service selection to occur, capability queries should be more abstract than the specific tasks of the agent, but they should include information about how the task should be achieved, and under what constraints. This use of abstract functionality descriptions is one of the reasons we developed the *Functionality* class hierarchy of the *Common Tool Ontology*.

4.4 Case Study: Scientific Applications as Services and Scientific Workflow Composition

In the absence of automated methods for service composition, the user invests considerable time and effort visiting numerous sites, determining appropriate service providers, entering his preferences repeatedly, integrating or aligning the different type of results coming from different tools. We would prefer that the user enters information once and receives the expected results from the most appropriate services with minimal additional assistance.

In our earlier work [Houstis et al. 2005; Houstis et al. 2003; Houstis et al. 2002], we developed a prototype for providing e-services in the field of ocean engineering (statistical processing of long-term time series of wave parameters and prediction of surface waves for geographical areas of interest). The system architecture combined metacomputing with ontology-driven resource documentation in order to achieve integrated and efficient information and computation management. Based on our

experiences, discovering and utilizing distributed scientific resources is of key importance to the development of an e-Science environment.

Our objective is to demonstrate how formalizing, sharing and re-using expert knowledge can effectively improve scientific computing in general and shape processing in particular. By embedding semantics and domain knowledge in the different stages of processing, we enhance the 3D processing pipeline, allow the re-use of valuable resources (e.g., existing content, processing tools/services, workflows), contribute to efficient resource discovery and support the composition and execution of scientific workflows.

Composing services into a workflow, given specific target functionalities, implies to discover the services that provide one or more of the functionalities required; retrieve the definition of these functionalities (or actions); integrate them into the result service and handle input and output constraints. Note that semantic web service composition focuses on the services functionalities and tries to build a chain of tasks that fulfills the user's requirements.

To address such an example, we have to deal with the following main problems:

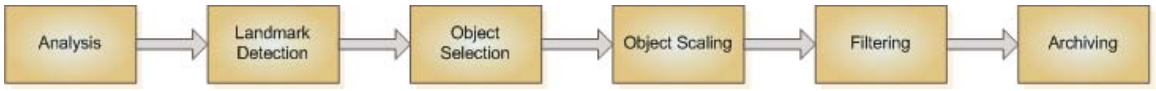
- the services discovery and selection;
- the service composition and orchestration according to a specific functionality;
- the relief of the user from any machine-processable task, i.e. to automate the composition process as much as possible.

At this stage, it is very desirable to have real world examples of possible application tasks in mind. In the following paragraphs we shortly describe some scenarios in the context of Cultural Heritage and we graphically depicted them as workflows.

Acquisition and Reconstruction - Consider the task of generating a digital model from a physical object (e.g. a statue, a fountain, a building etc.). The digital model may be further processed depending on the application task: visualization, simplification at different levels of detail, deformation, prototype generation, etc. For instance, the use of a

laser scanner to acquire a façade of a building or a white light pattern projection scanner scanning in close range 3D artifacts produces a huge amount of data. Usually more than one scanning session is necessary to acquire the whole façade or artifact before an accurate digital model is produced. Each scan session produces a data-set (point cloud), and different data-sets have to be registered and merged in order to produce the whole digital raw data from which the final digital model is obtained. Several post processing tools can be applied to the digital model, depending on the application task (e.g., visualization, prototype generation, validation and certification).

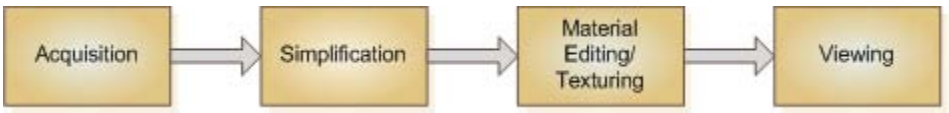
Virtual Restoration of 3D objects - The (semi-) automatic restoration of missing parts of a 3D artifact (e.g. a statue or an ancient building) can be achieved, based on the selection of semantically similar parts from a database. In this case, for instance, an analysis based on the human body could capture information on the body landmarks of the statue (e.g. the Venus of Milos). An arm could be selected elsewhere, and its measurements could be adapted to fit a missing arm of the statue. A process could produce the fitting arm, and another process could make the actual merging. Another application could be to seek for fitting parts and fragments of historical artifacts distributed in locations all over the world. The purpose of such applications could be purely scientific but also could used for training and entertainment.



Virtual Reconstruction - Another application scenario could be the reconstruction of historical artifacts and whole historical buildings or even whole cites. Virtual reconstruction is based on information sources like historical texts, drawings and other

2D information, and the comparison with historically similar artifacts or buildings, which could be assessed applying semantically driven queries for historical information available. Existing 3D digital models and shape processing tools would enable more feasible approaches to such reconstructions. The tools required for this task would include assembling of different digital models, performing geometric modeling activities, and editing of material properties.

Virtual Presentation - Many 3D models are generated for public use in virtual exhibits (either in real museums or in virtual museums on the Web). For this purpose, besides the acquisition of the 3D models, the visualization of the 3D data has to be supported. Special attention must be given to the complexity of the data that typically are visualized with standard 3D viewers on off-the-shelf computers. Model simplification tools are applied in such situations. To obtain an appealing visual quality, tools for applying 2D textures to the 3D geometry need to be available. Other tools, the so called "shaders", generating visual effects for simulating different lighting situations would also be of interest.



4.4.1 Development of Web Service Workflows

With the use of the MeshLab[20] mesh processing system, we exported several 3D shape processing tools as Web Services and we utilized them in our proof-of-concept prototype implementation. MeshLab is a free and open-source general-purpose mesh processing system, which was developed by ISTI-CNR in the framework of the EPOCH Network of Excellence [EPOCH 2008]. It is designed to help the flow and adaptation of 3D models that typically occur in the pipeline when processing 3D scanned data in the context of Cultural Heritage. Specifically, MeshLab provides a set of tools for editing, cleaning, healing, inspecting, rendering and converting the resulting meshes. MeshLab can be considered as an intuitive mesh viewer application, where a 3D object, stored in a variety of formats, can be loaded and interactively inspected in an easy way, by simply dragging and clicking on the mesh itself. Once a mesh is loaded, the user can work on it by selecting appropriate actions from a large set of parametric filters that perform smoothing, re-meshing and simplifying tasks either automatically or by means of interactive tools.

MeshLab provides many mesh processing functionalities. A subset of these functionalities cover the so called “mesh cleaning” needs, offering tools to correct the geometric/topological imperfections (noise) that often affect 3D scanned data and 3D models in general. Typical examples are removal of duplicated, unreferenced vertices, null faces, small isolated components, coherent normal unification and face flipping, erasing of non-manifold faces and massive automatic filling of holes. There are also many mesh inspection tools for analyzing and assessing in an intuitive, visual, and measurable way the quality and the correctness of the examined meshes. A set of tools have been recently added for implementing the full 3D scanned data processing pipeline: from the raw sources obtained by the hardware acquisition devices to the final clean, ready-to-be-used 3D model. This processing pipeline includes a subsystem for the alignment of 3D meshes that allows to precisely registering many different raw range

[20] http://meshlab.sourceforge.net/

maps and a set of three different algorithms for surface reconstruction that merge the multiple range maps into a single mesh.

To enable web access to a selection of seventeen of the aforementioned tools, we wrapped them into Web Services using the command line version of MeshLab (meshlabserver) which takes as input an XML formatted "filter script" i.e. sequences of filtering actions.

State-of-the-art technologies were used for the implementation and deployment of these Web Services. Table 4.1 gives a summary of these technologies.

Table 4.1: Enabling technologies for the implementation and deployment of our MeshLab Web Services

Product	Version	Role
Java EE 6	1.6.0 u11	Platform for the Java programming language
GlashFish Enterprise Server	2.1.1	Application server
NetBeans	6.5	Integrated Development Environment (IDE) for software developers
Jena 2	2.3	A Semantic Web Framework for Java
PostgreSQL	8.1	Database (RDBMS)
Tomcat	5.5.15	Application server

We used the SOA module of the NetBeans IDE (integrated with OpenESB and GlashFish[21]) for the development of our Web Services, the BPEL process modules, the composite applications and the web user interfaces. The Web Services and all the other web modules are deployed using the GlashFish application server that provides not only server-side infrastructure for deploying and managing services, but also client-side API for invoking those services.

[21] The GlashFish application server. http://glassfish.java.net/

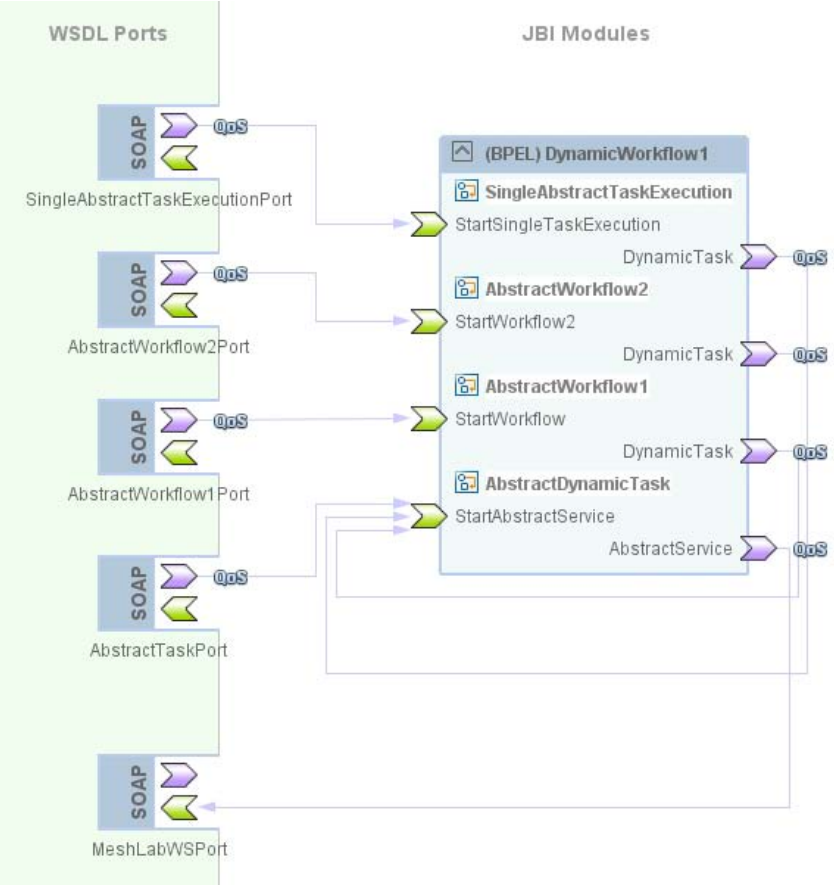


Figure 4.6: Design view of the developed BPEL processes.

Figure 4.6 gives an overview of the developed service endpoints (WSDL ports), the JBI module with all our BPEL processes and the connections between them (WSDL bindings). This design view has been created using the Composite Application Service Assembly (CASA) editor provided with NetBeans.

The designer views of our BPEL process diagrams are shown in the following figures. Figure 4.7 illustrates the designer view (BPEL editor from NetBeans) of our abstract task implemented with dynamic endpoint reference (dynamic binding). The purpose of this abstract task is to be able to invoke any kind of Web Service implementing the same interface.

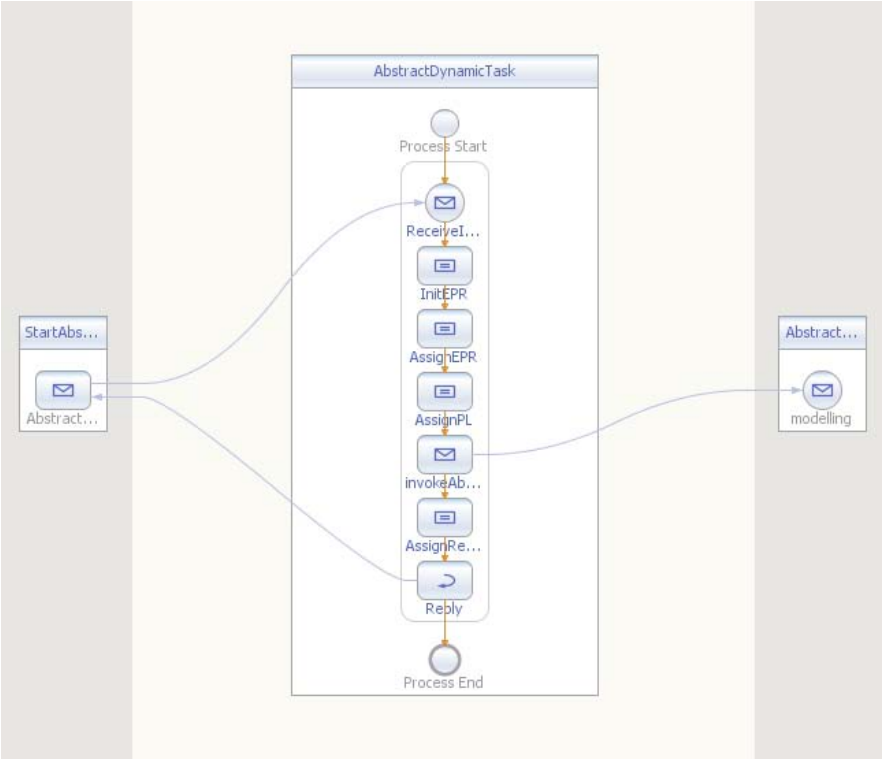


Figure 4.7: BPEL editor view of our abstract task with dynamic binding.

Figure 4.8 gives the design of a single task execution service which invokes the abstract task process shown in Figure 4.7, i.e. the *partner link* on the right in Figure 4.8 corresponds to the *partner link* on the left in Figure 4.7. This can be considered as a special case workflow containing only one step.

The designer's views of our two prototype workflow scenarios are given in sections 4.4.2.1 and 4.4.2.2 respectively.

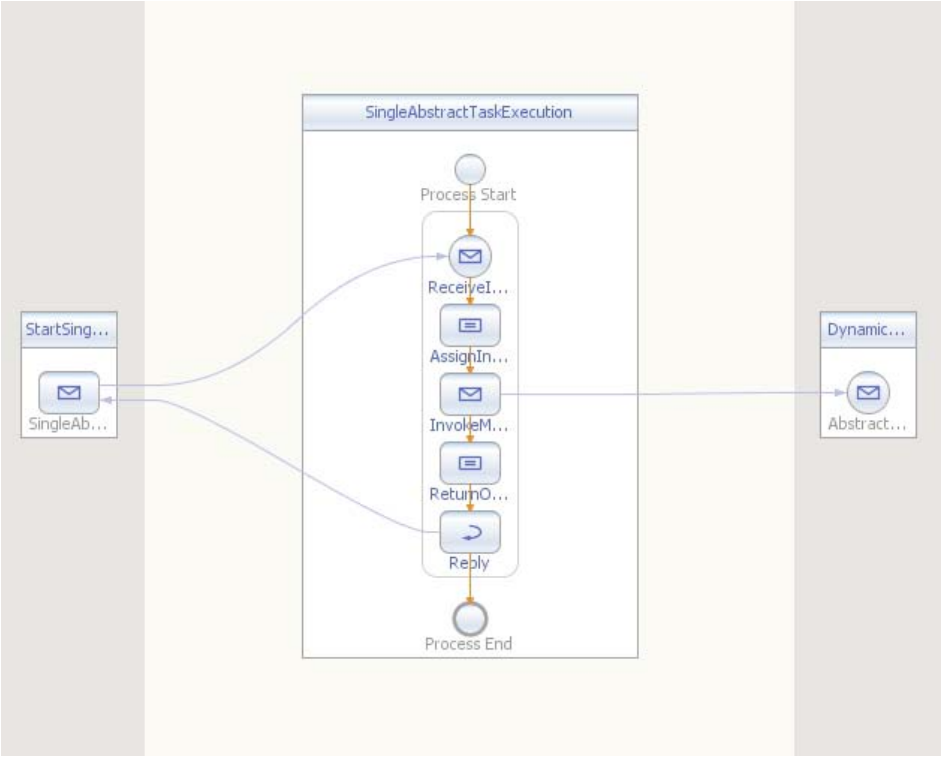


Figure 4.8: BPEL editor view of a single abstract task execution step.

Each of the developed Web Services is described by a set of metadata information and is classified according to the *Functionality* hierarchy of the *Tool Common Ontology* (TCO). Every service is associated to a *Software Tool* instance in the TCO ontology. In fact, we added an extra subclass to the TCO hierarchy to specifically describe Web Services.

The list of currently available Geometry Processing Web Services is shown in Figure 4.9. The user interface dynamically generates this list from the instances of the TCO class *Web Service*. Additional information about each service is provided when the mouse pointer goes over the service name (tooltips). This information is also dynamically generated from the values of the datatype property called *hasDescription*. By selecting a single service and pressing the execution button, the specific Web Service is invoked.

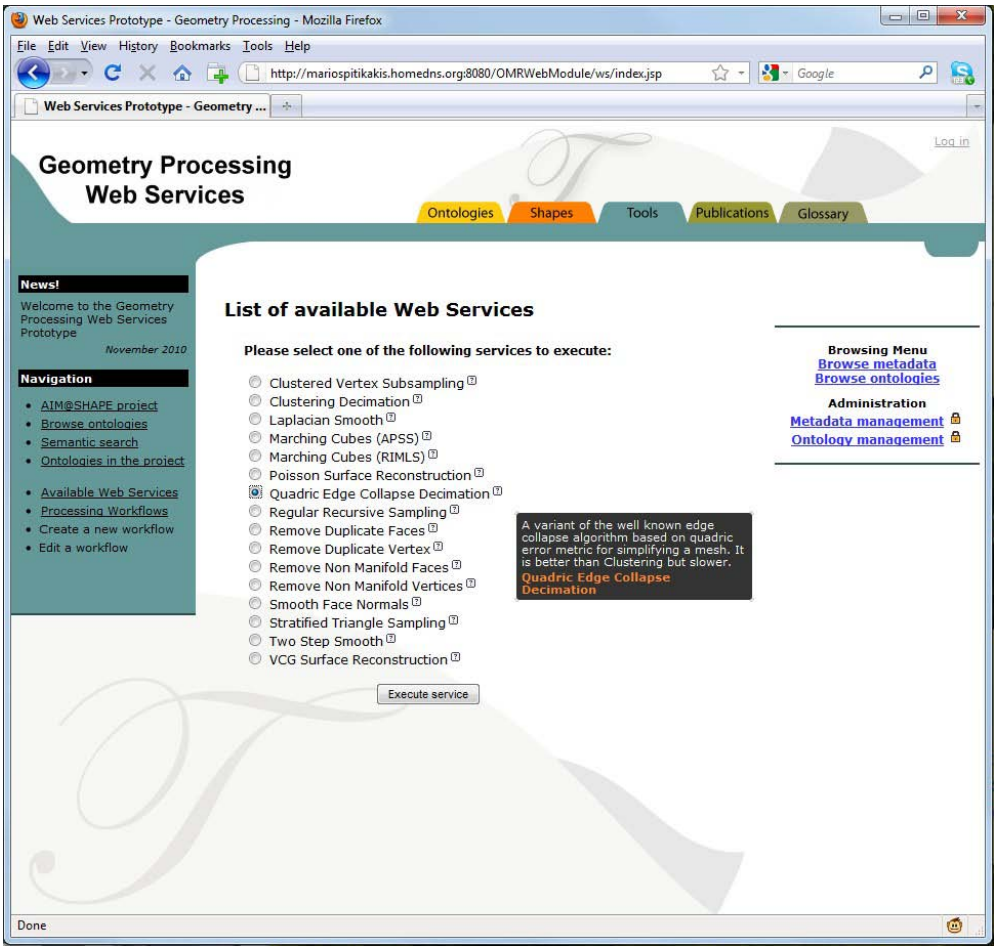


Figure 4.9: The user interface of a single Web Service selection and execution.

In addition, the execution of a Web Service automatically produces the appropriate metadata for the resulting (output) model which are stored to the knowledge base through the OMR API.

Two geometry processing workflows were developed (more details are given in subsections 4.4.2.1 and 4.4.2.2) where service discovery and selection is done semi-automatically. The user can select appropriate services using a functionality-based discovery method.

4.4.2 Surface Reconstruction and Modeling Workflow Scenarios

The reconstruction of precise surfaces from unorganized point clouds derived from laser scanner data or photogrammetric image measurements is a hard problem, not completely solved and problematic in case of incomplete, noisy and sparse data. The generation of polygonal models that can satisfy high modeling and visualization demands is required in different application domains. The goal is always to find a way to create a computer model of an object which best fits the reality and specific application needs. Polygons are usually the ideal way to accurately represent the results of measurements, providing an optimal surface description. Many methods have been developed to create a regular and continuous (triangular) mesh representation from a point cloud. Given the polygonal surface, various techniques can be used for post-processing operations (smoothing, texturing) and for the visualization of the 3D model. Before going into details regarding the modeling (and visualization) aspects of 3D models, the reader may find a short list of important terms in Appendix A.

The conversion of the measured data into a consistent polygonal surface is generally based on four steps:

(a) Pre-processing: in this phase erroneous data are eliminated or points are sampled to reduce the computation time. Editing operations on the measured points are very important before generating a triangular surface. The pre-processing operations usually are:

- data sampling, based on the curvature of the points or uniformly apply. In case of scanner data, this step is mandatory in order to reduce the input redundancy (downsampling) and to remove a certain amount of errors introduced because of to the scanning device limitations.
- noise reduction and outliers rejection: statistical methods are applied taking into consideration the surface curvature and trying to preserve the measured features. In case of image matching results, wrong correspondences can be removed automatically or manually with visual inspection.

- holes filling: gaps in the point clouds are closed adding (manually or automatically) new points and using the curvature and density of the surrounding points.

(b) Determination of the global topology of the object's surface: the neighborhood relations between adjacent parts of the surface has to be derived. This operation typically needs some global sorting step and the consideration of possible "constraints" (e.g. breaklines), mainly to preserve special features (like edges);

(c) Generation of the polygonal surface: triangular (or tetrahedral) meshes are created satisfying certain quality requirements, e.g. limit on the meshes element size, no intersection of breaklines, etc. A triangulation converts the given set of points into a consistent polygonal model (mesh). This operation usually generates vertices, edges and faces (representing the analyzed surface) that meet only at shared edges. Two types of information are encoded in the created meshes: the geometrical information (i.e. the position of the vertices in the space and the surface normals) and the topological information (i.e. the mesh connectivity and the relations between the faces). Finite element methods are used to discretize the measured domain by dividing it into many small "elements", typically triangles or quadrilaterals in two dimensions and tetrahedra in three dimensions. The triangulation in 3D is called tetrahedralization or tetrahedrization. A tetrahedralization is a partition of the input domain into a collection of tetrahedra that meet only at shared faces (vertices, edges or triangles). Tetrahedralization results are much more complicated than a 2D triangulation.

(d) Post-processing: when the model is created, editing operations are commonly applied to refine and perfect the polygonal surface. The created polygons usually need some refinements to correct imperfections or errors in the surface. Moreover spikes can be removed with smooth functions. These operations (mainly manually) vary from single triangles editing to surface corrections:

- edges correction: faces can be splitted (divided in two parts), moved to another location or contracted.

- triangles insertion: holes can be filled constructing polygonal structures that respect the surrounding area; incomplete meshes can also be repaired with radial basis function or with volumetric approach.
- polygons editing: the number of polygons can be reduced, preserving the shape of the object or fixing the boundary points. The polygonal model can also be improved adding new vertices and adjusting the coordinates of existing vertices.

The following workflow scenarios demonstrate the general conception of pre-processing, polygonal surface generation or reconstruction and post-processing using two actual 3D model processing pipelines.

4.4.2.1 First Workflow Scenario

In the pipeline of processing 3D data, after range maps have been aligned, merged and registered, it is often required to get a mesh. An abstract workflow description is defined as follows: Cleaning of scanned data → Mesh smoothing and reconstruction → Simplification. This is a predefined workflow template that can be found in our prototype implementation and is one way to capture expert knowledge.

We start from a quad-remesh of a human hand taken from the AIM@SHAPE Shape Repository (model name: "Pierre's hand"). The original data set contains range images at medium resolution, and the palm was scanned at the highest possible resolution (some fingerprints are even visible) with a Minolta scanner and registered using Minolta software. The model (depicted in Figure 4.10a) is composed of 218012 vertices and 393384 faces.

Chances are high that the object is surrounded by a lot of extra points that do not need to be included in the final mesh or there are many duplicate vertices. The easiest way to remove these is to clean the model, resulting a cleaner mesh and smaller file size. After a cleaning procedure, 21110 duplicated vertices were removed.

The medium complexity mesh produced, as it often happens, is topologically “dirty”. There are several hundred of small holes (some of them are visible in Figure 4.10b) that make difficult any kind of parametrization. So the next step is to build a watertight, coarser but topologically sound model. Poisson surface reconstruction is a good method for this task and generates a mesh of 376414 of faces. Figure 4.10c depicts the watertight Poisson reconstructed surface with all the holes filled.

After that a further simplification step is needed to reduce the model size to a more reasonable number of faces. The full accuracy model is not required for most application domains since small details or features have minor influence on the overall quality of the result. A model with the same overall shape and topology is usually enough in most cases. The simplified watertight model usually has up to 50K triangles. We use a Quadric based Edge Collapse strategy and we choose to produce a model with 40K faces (see Figure 4.10d). It is very hard to distinguish the differences between Figure 4.10c and d but the file size is considerably reduced.

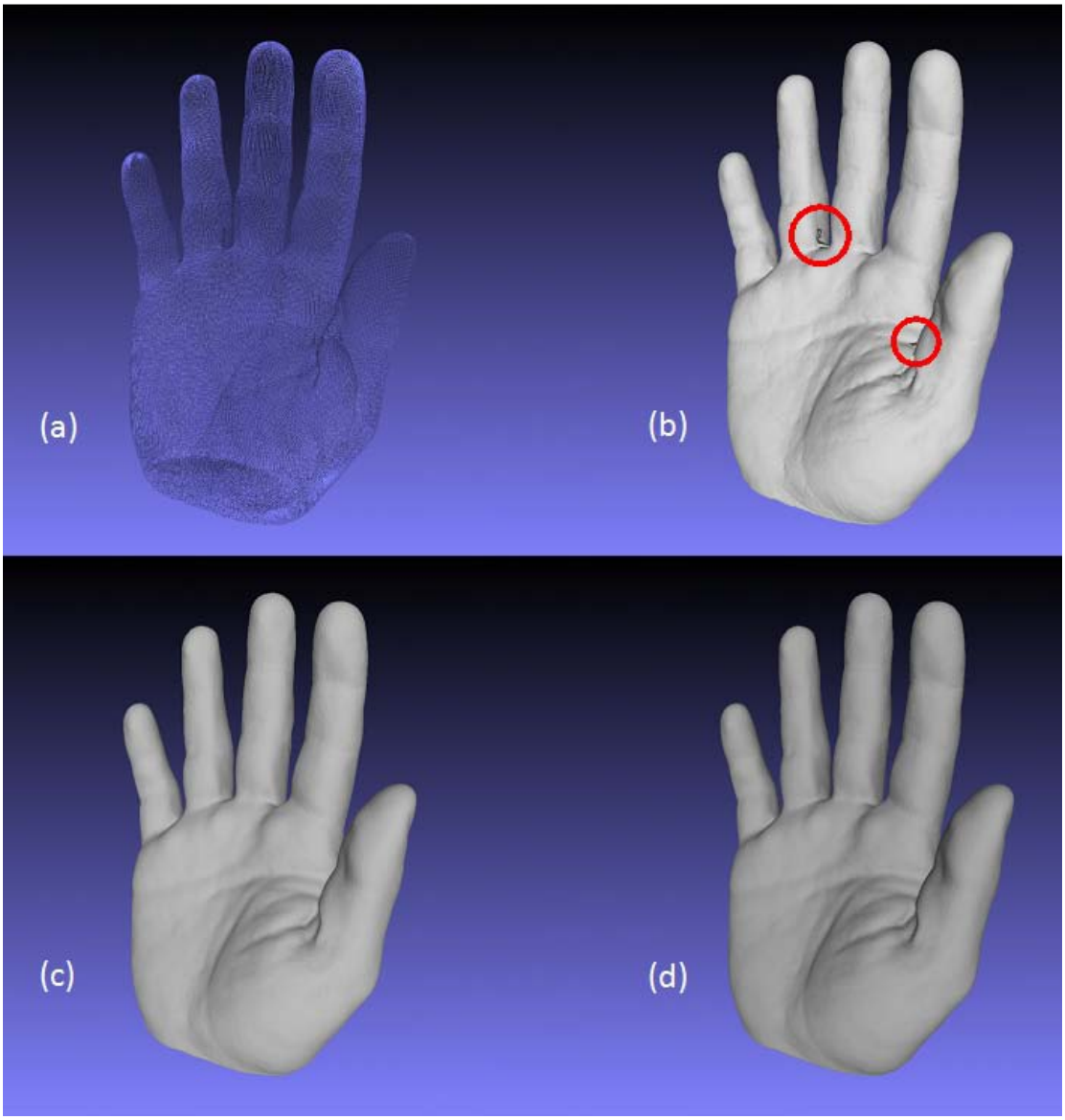


Figure 4.10: (a) Original data set, (b) model with holes after cleaning procedure, (c) watertight model after hole filling and (d) model after simplification

To summarize, from the abstract workflow description we have selected and executed a pipeline of specific web services from a variety of different choices for each processing step that can found in our system. The execution order of the selected web service instances is: Remove Duplicated Vertex → Poisson Surface Reconstruction → Quadric based Edge Collapse.

The designer view of the above abstract workflow scenario (BPEL process diagram) is shown in Figure 4.11.

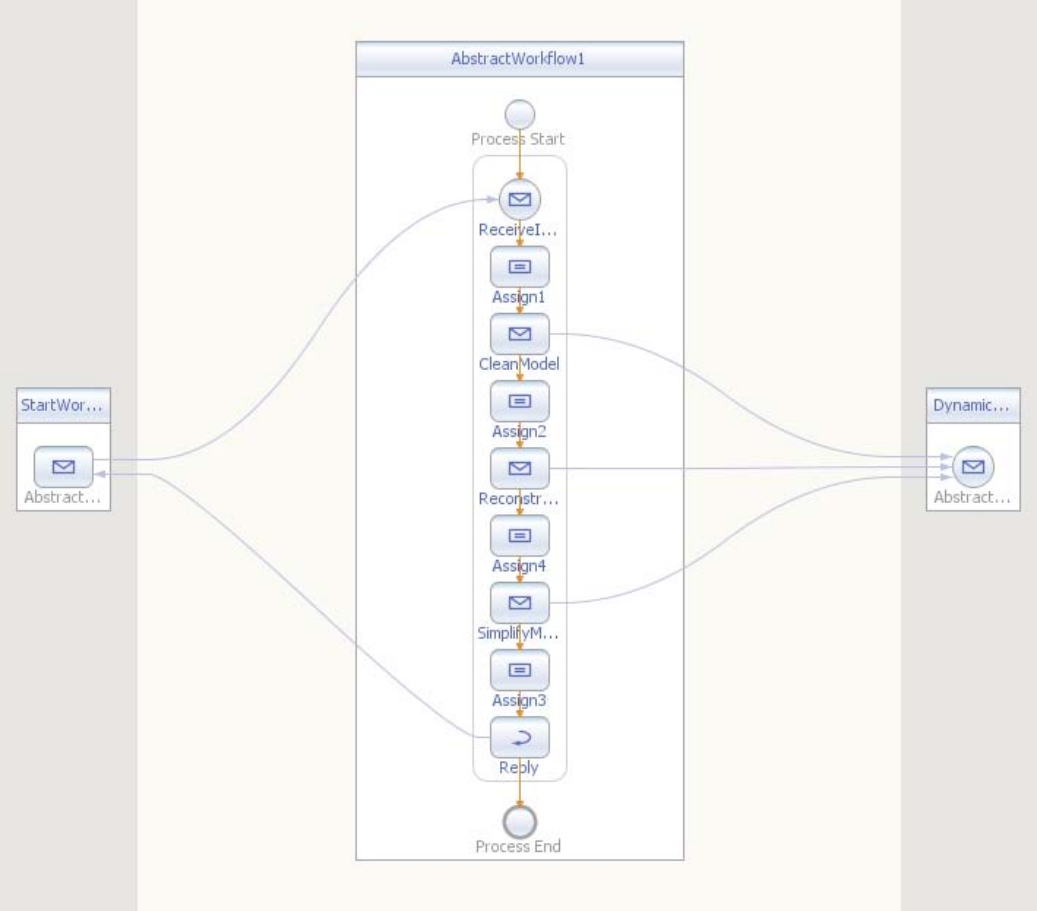


Figure 4.11: BPEL editor view of our first abstract workflow scenario.

The web interface of the first workflow scenario is presented in Figure 4.12:. The web page shows the abstract workflow definition (functionality descriptions) in a diagram and the user is prompted to assign concrete service instances to each task of the abstract process. The selection of the specific Web Service instances is done from drop-down menus that are dynamically generated using the *Functionality* property of the *Software Tool* class. For example, there are four available Web Services for cleaning a mesh: Remove Duplicate Faces, Remove Duplicate Vertices, Remove Non Manifold Faces and Remove Non Manifold Vertices.

After the Web Services selection, the user uploads the input model and initiates the workflow execution. As described in section 4.3.1, the services are dynamically binded to each task of the abstract BPEL process (shown in Figure 4.11).

Finally, the execution of the workflow automatically produces the appropriate metadata for the resulting (output) model i.e. processing history, documentation and other details (e.g. number of vertices, number of faces, model origin, file size and file format, location and URL etc.). These metadata instances are stored back to the knowledge base.

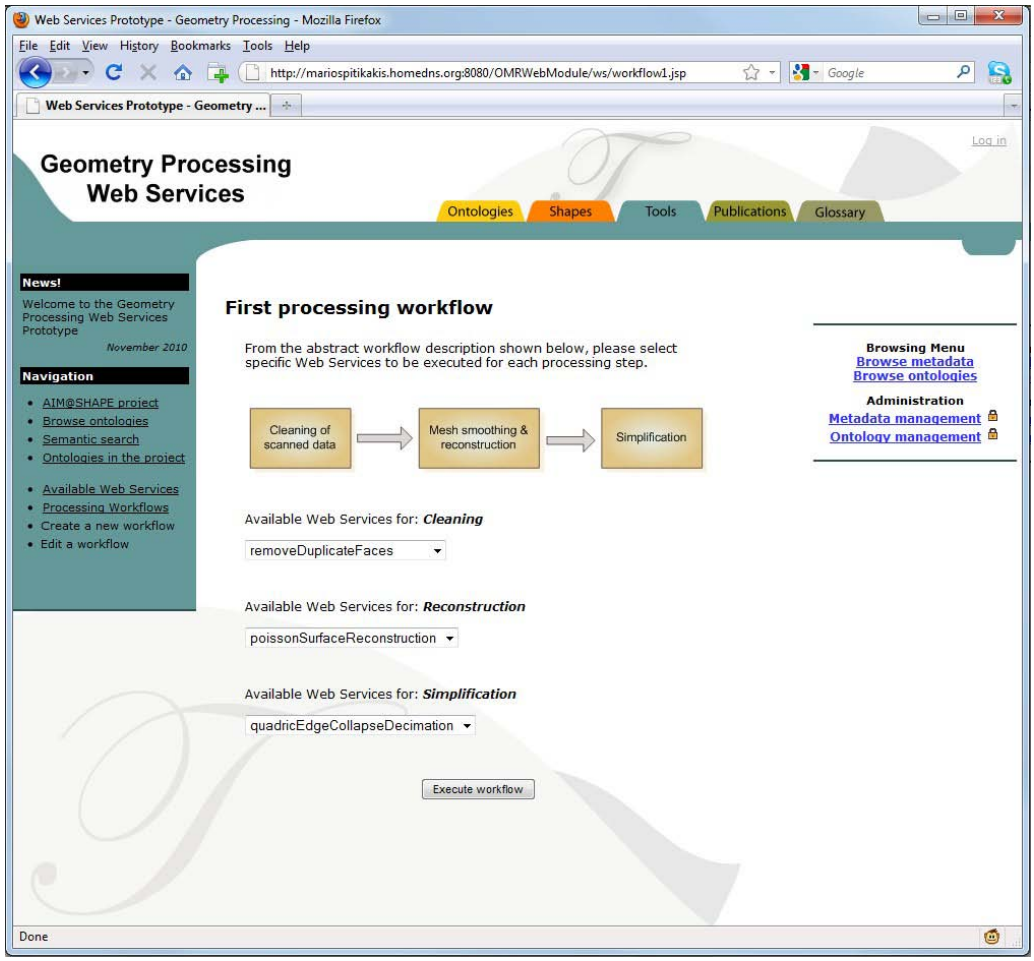


Figure 4.12: The web interface of the first workflow scenario.

4.4.2.2 Second Workflow Scenario

The abstract workflow description for this example is defined as follows: Sampling of scanned data → Meshing → Simplification → Smoothing & Fairing.

Starting from the same human hand model considered in the previous example, we apply a sampling technique called Clustered Vertex Subsampling and we produce a point cloud composed of 10892 vertices (see Figure 4.13a).

The next step is to create a mesh from the point set by using a marching cubes algorithm. To be more specific, we use an algebraic point set surfaces (APSS) variant which is based on local fitting of algebraic spheres and requires points equipped with oriented normals. Figure 4.13b depicts the generated triangular mesh of 58468 vertices and 116932 faces.

After that, we apply a simplification step to reduce the model size to 30K faces (see Figure 4.13c), using again the Quadric based Edge Collapse algorithm. Finally, to create a smoothed mesh, we choose to apply the Smooth Face Normals algorithm. The resulting mesh is shown in Figure 4.13d.

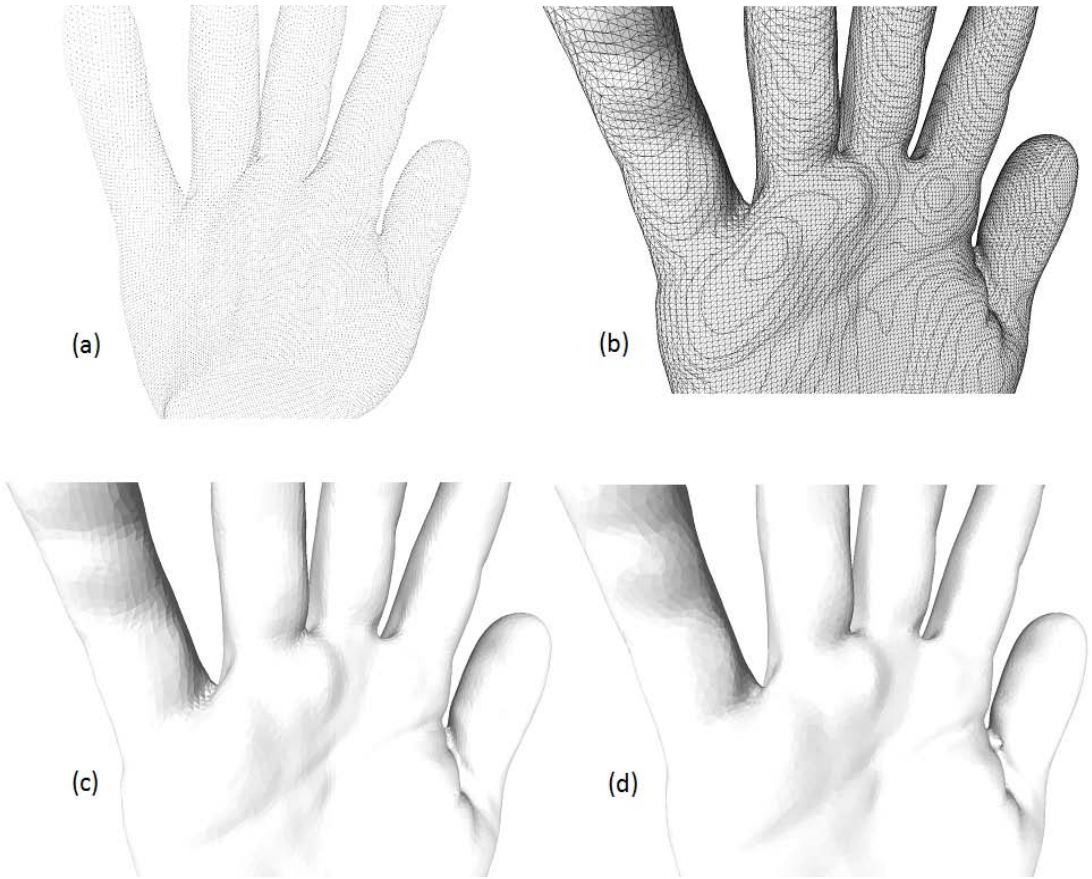


Figure 4.13: (a) Sampling of the original model, (b) mesh generation from point cloud, (c) simplification and (d) smoothing & fairing

In this abstract example workflow scenario we executed a pipeline of specific web services selected from a variety of different web service choices. The execution order of the selected web services instances is: Clustered Vertex Subsampling → Marching Cubes (APSS) → Quadric based Edge Collapse → Smooth Face Normals.

The designer view of the above abstract workflow scenario (BPEL process diagram) is shown in Figure 4.14.

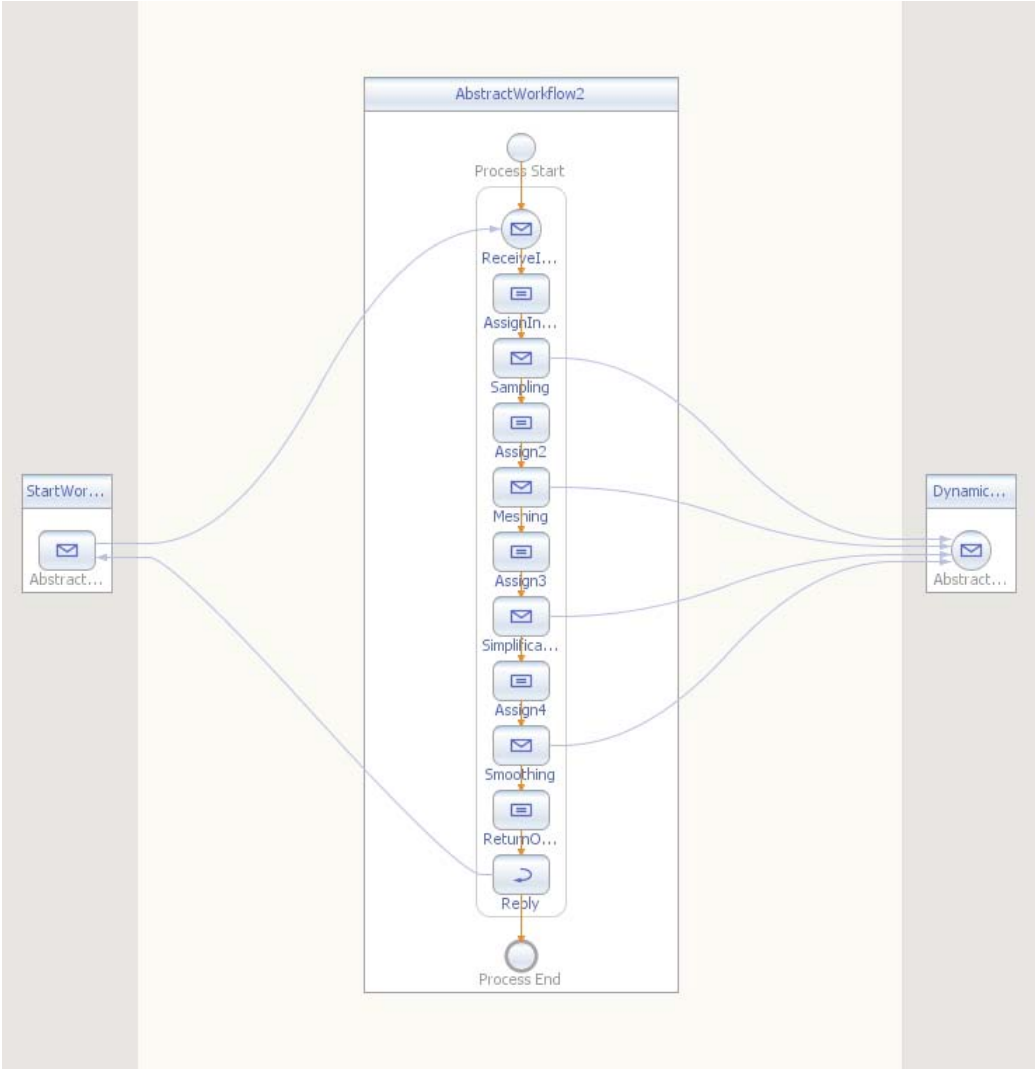


Figure 4.14: BPEL editor view of our second abstract workflow scenario.

The web interface of the second workflow scenario is presented in Figure 4.15: and has the same functionality and usage as presented in the previous section.

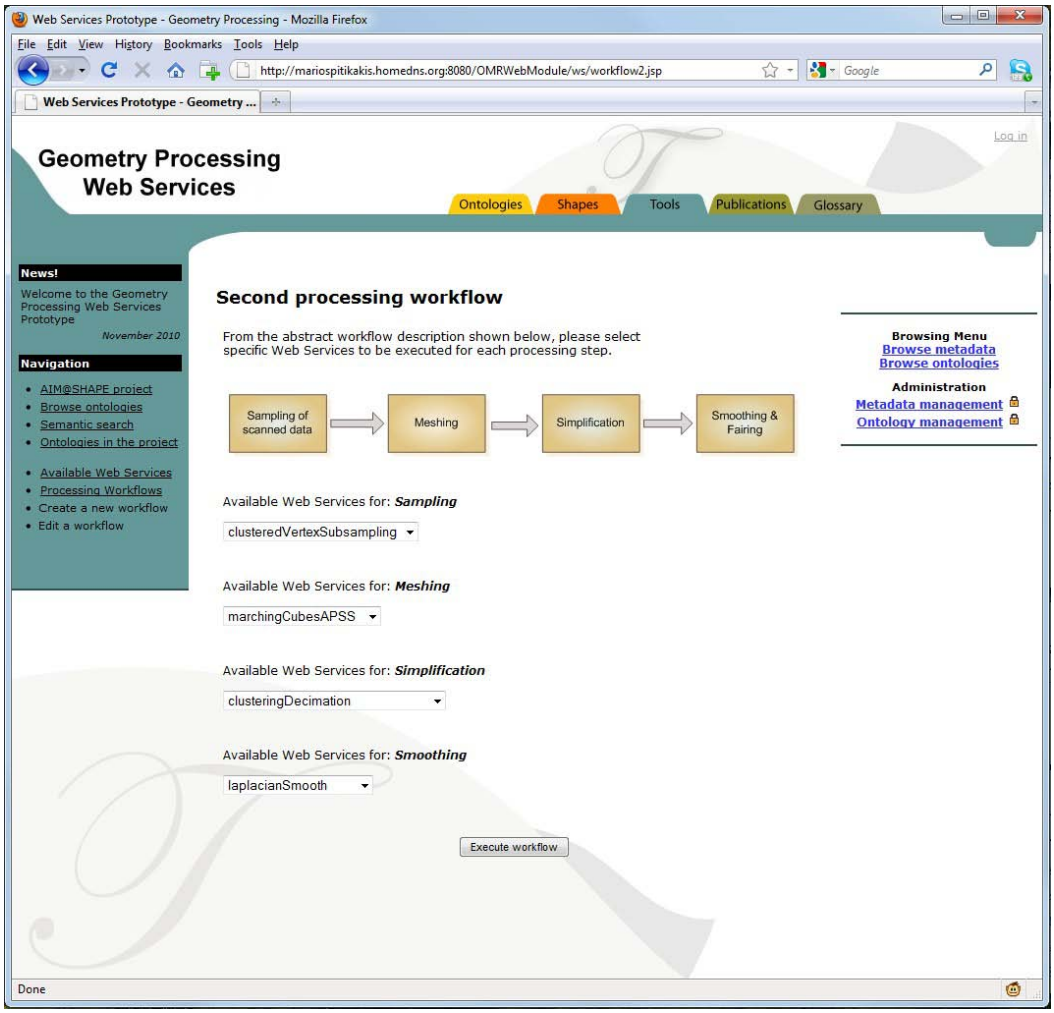


Figure 4.15: The web interface of the second workflow scenario.

5 PROTOTYPE SYSTEM EVALUATION

5.1 Quantitative Performance Evaluation

5.1.1 Evaluation of the Knowledge Infrastructure

Testing and validation of the OMR and inference engine with large scale ontologies and metadata have been conducted. As a measure of scalability of the knowledge base we provide here an indication of the volume of metadata currently stored in the OMR. There are over 1100 shape models, more than 80 tools, more than 220 instances of Users and Institutions, with a total of over 4300 context-dependent instances in the domain and common ontologies. These also provide a solid test base for the evaluation of the efficiency of the Semantic Search Engine.

Our implementation choice of using a persistent storage solution (RDBMS) for storing ontologies and related metadata reduced many of the OMR performance issues to the performance of the underlying database management system. Scalability and reliability in an RDBMS are already solved problems. The number of ontologies loaded does not affect significantly the performance of the OMR (e.g. load time, query execution) since each ontology is stored in separate tables (set of tables).

To evaluate the performance of the OMR we have contacted a series of tests on a commodity desktop computer with the following characteristics: Intel Pentium 4 Processor at 3.2GHz; 2GB of RAM; Windows XP Professional OS; Java SDK 1.6.0 update 13; Apache Tomcat 5.5; PostgreSQL 8.1 RDBMS.

Table 5.1 shows the estimated data average loading/insertion elapsed time for the five ontologies considered above with different schemas and instance sizes. Each of the ontologies was loaded ten times and the calculated average load time is presented in seconds with respect to its number of classes (column 2), its number of properties (column 3) and the number of instances it contains. As expected, the persistent storage (PostgreSQL) performs relatively well and displays good scalability in data loading. The

results show that, as expected, the loading time is a function of both the complexity of the ontology and the number of instances.

Table 5.1: Performance data for OMR load time with respect to ontology parameters.

Data set	Ontology information		Number of instances	Average Load Time (in seconds)
	#classes	#properties		
Info Common ontology (ICO)	2	9	289	4.95
Product Design ontology (PDO)	75	72	130	5.97
Virtual Humans ontology (VHO)	53	73	186	6.07
Common Tool ontology (CTO)	70	15	769	15.43
Common Shape ontology (CSO)	35	153	2929	105.50

Table 5.2 presents a list of three query test results and the estimated average response time measure in seconds (again the average is computed from ten measurements). These queries represent the most common actions expected from an ontology management system or knowledge base. The increased response times concerning the last two queries are mainly caused by the imported ontologies. The CSO and CTO ontologies import the ICO ontology, the VHO ontology imports the ICO and CSO ontologies and the PDO ontology imports the ICO, CSO and CTO ontologies. In addition, the PDO extends some classes of the CSO ontology. Therefore, the response time of such complex class hierarchies with such volume of instances is expected to be significantly higher.

Finally it is worth to mention that all times reported in this section are in accordance with recent evaluation studies [Babik & Hluchy 2008].

Table 5.2: Performance data for OMR queries.

Action	Average Response Time (in milliseconds)				
	ICO	CSO	CTO	VHO	PDO
Find and display all instances of a class	0.150	0.234	0.164	0.220	0.231
Retrieve an instance and display all property values	0.255	11.396	3.226	10.617	17.012
Find and display the ontology class hierarchy	0.317	3.798	6.320	8.845	21.306

The Digital Shape Workbench has been running for the past five years while both the Shape Repository and the Ontology & Metadata Repository (OMR) continue to increase in size. The AIM@SHAPE Shape Repository has became one of the most cited repositories for 3D models due to the accuracy and detail of the metadata and its availability and openness to the public. It has an access count exceeding three millions visitors, over 120.000 3D model downloads and a world-wide dissemination and acceptance. All of the above were actively supported by the OMR.

5.1.2 Evaluation of the Prototype Web Services and Workflows

We have developed seventeen (17) shape processing Web Services and two (2) 3D model processing pipelines (workflows) in our proof-of-concept prototype implementation. Web Service providers can easily add new services from the web interface by creating a *Web Service* instance in the *Common Tool Ontology* and filling in the required metadata information – most importantly the URL of the Web Service (provided that it implements the same interface as the other Web Services) and the *Functionality* (algorithm category).

To evaluate the performance of our prototype system we have deployed our Web Services on a low-end PC with an Intel Core Duo T2400 processor at 1.83GHz and 1GB

RAM, running a GlashFish v2.1.1 application server (more details on the installed software can be found in section 4.4.1).

We contacted a series of execution tests on a set of test models from the Shape Repository of AIM@SHAPE. More specifically, we selected four (4) manifold surface meshes of hands in different postures and with different geometric characteristics and file sizes, as shown in Table 5.3.

Table 5.3: Model information on our set of test models.

Model name	Model information		
	File size (in MB)	# vertices	# faces
Model 1: 759_hand-olivier-isotropic.off	3,7	53054	105860
Model 2: 354_gipshand2-273k.obj	10	136663	273060
Model 3: 737_pierre_hand_quad_graphite.ply	19,9	351048	701543
Model 4: 787_uniform-350kv.off	30,6	218012	393384

The average execution times (in milliseconds) of all our test models when inserted as input to the first and second prototype workflow are given in Table 5.4 and Table 5.5 respectively. The execution times are reported per Web Service and the total execution times of the workflows do not include the upload time of the input model.

The results show that, as expected, the performance (execution time) of these Web Services depends on several factors like the geometry of the input model (e.g. the number of vertices, the number of faces etc.), the file size of the model (that affects the memory and disk usage) and of course the algorithm itself (which is wrapped as a Web Service).

Table 5.4: Performance data for the execution of the first workflow scenario.

Web Services	Average execution time (in milliseconds)			
	Model 1	Model 2	Model 3	Model 4
Remove Duplicated Vertices	1724	7719	2406	13781
Poisson Surface Reconstruction	7085	11302	7331	16797
Quadric Edge Collapse Decimation	3181	11901	9342	25247
Total execution time	11990	30922	26904	55825

Table 5.5: Performance data for the execution of the second workflow scenario.

Web Services	Average execution time (in milliseconds)			
	Model 1	Model 2	Model 3	Model 4
Clustered Vertex Subsampling	1707	7903	2817	12116
Marching Cubes (APSS)	11341	14138	14638	20888
Quadric Edge Collapse Decimation	3165	11891	8801	24947
Smooth Face Normals	1027	1404	647	1003
Total execution time	17241	35335	19079	58954

5.2 Qualitative Evaluation

5.2.1 Qualitative Evaluation of Ontology Development

Ontologies are the cornerstones of the system since they formalize the relevant semantic aspects of models, tools and services. When evaluating information retrieval techniques, two important indicators are precision and recall:

- *Precision* indicates the percentage of the retrieved information that are relevant to the initial user's need.
- *Recall* indicates which percentage of relevant information was actually retrieved.

Ontology-based representation allows inferences on its content, and this is a powerful way to increase precision and recall; ontologies enable reasoning (improve precision) while the use of the hierarchical classification enables the search mechanism to retrieve information that would not have been found if the queries were only at term/instance level (improve recall).

Ontology usability and exploitability can be considered from the two points of view, of the user and of the designer:

- From the user point of view, ontologies do improve the system behaviour and effectiveness. The system relies on this conceptual structure to perform meaningful inferences and improve its “intelligence”. One condition, however, is the development of suited interfaces to make that conceptual structure transparent through the system’s infrastructure.
- From the designer point of view, it introduces efficient coding standards and high-return and high-performance techniques. Ontology-based design does ease the integration phase by enabling semantic coupling. It makes explicit the semantic consensus and captures it in an external logical representation (the ontology object itself). However the additional design burden can be prohibitive if it is not thoroughly planned, controlled and restricted to effective needs.

A set of criteria was chosen to guide us in the qualitative evaluation process of the developed ontologies. These criteria and their short definition are shown in Table 5.6.

Table 5.6: Qualitative evaluation criteria for the development of the ontologies

Criteria	Definition
Modularity	Degree to which the ontologies are subdivided in separated parts or independent parts.
Maintainability	Degree to which the ontologies are amenable to changes after they have been created.
Documentation	Ability of the ontologies to hold documentation information.
Reusability	Degree to which the ontologies or parts of them could be reused with few or no adaptations.
Extensibility	Capability of the ontologies to facilitate the addition of new concepts and functionalities.
Expressiveness	Capability of the ontologies to offer meaningful representation to the user.
Feasibility	Degree to which the design of the ontologies is realistic.

Maintainability also includes the adaptability and customization which had to be tackled; maintenance cycles on the ontologies were a heavy recurrent burden all the more since they required a greater consensus. The first draft of the ontologies was a good step for a feasibility study, but refining, validation and checking was very time consuming. Revisions of the ontologies were triggered by feedback from experts or what had not been conceptualized or formalized properly. An ontology is *a living object*, the maintenance of which has consequences beyond its own lifecycle: it has an impact on everything that depends on it. Stored ontologies in the OMR had to be versioned and the knowledge base consistency had to be maintained. Therefore, although the problem of the ontology evolution itself is a hard one, one should consider the fact that ontologies provide building blocks for modelling and implementation.

As presented in section 3.1.2, the developed ontologies are divided into two categories, which are interesting to consider for versatility, modularity and reusability concerns:

- The top-level common ontologies that are abstract and general and therefore, potentially highly reusable.

- The domain-specific ontologies that tend to be application and scenario specific with internal complex concepts; they are potentially reusable in the context of the same application domain, but they may require rework and extensions to reflect the conceptualization of different end-users.

The reusability of parts of an ontology does not mean that it will not require adaptations and updates; it only means that the part can be used as a starting point for a new application in order to save time. Extensibility is an intrinsic characteristic of ontologies where notions can be subsumed by more general notions and/or can be specialized by more specific concepts.

Extensibility concerns the three characteristics of the coverage of an ontology: exhaustivity, specificity and granularity. Exhaustivity is extended by enlarging the domain of application or the cases considered in the scenarios. Specificity is extended by detailing the identification of conceptualized concepts. Granularity is extended by detailing the axiomatization.

Ontologies provide the semantic grounding for all the systems' interactions. The Search Engine relies on the fact that the ontology provides a non-ambiguous shared conceptual vocabulary to describe the resources and express the patterns of searched information. The shared ontologies are therefore, the cornerstone of interoperability in the system at the semantic level. The ontologies are formalized in OWL and therefore, can be exchanged with and used by other systems (portability).

One of our aims was to reduce the complexity of the ontologies, especially the top levels that introduce some abstract concepts. If users are overloaded with details or lost in the depths of the hierarchy, they will never use it. Therefore, accessibility is a critical factor for the adoption of the ontology by end-users. User interfaces also play an important role in this matter. We investigated these aspects of visualization and browsing of ontological structures and semantic searching user interfaces and developed web interfaces that enable users to interact with the ontologies. As shown in Chapter 3, we developed Java applets for the visualization, browsing and searching of the ontologies

(i.e. the hyperbolic tree visualization tool, the semantic searching engine user interface) and the OMR web interface for managing ontologies and metadata.

The work of building an ontology is a tough one: it is extremely time consuming and the commitment/consensus needed from the users/experts is difficult to obtain even in a small group, and becomes more difficult for larger communities. The ontology evolves in prototype cycles, slowly refining and ever changing. The ontology in itself can grow without major problems, but the scalability to larger projects or communities raises serious time and logistic concerns. The precision needed for the ontology and the available resources have to be considered in feasibility studies before any project is started.

We used specific application scenarios to capture in a natural and efficient way the context specific requirements. This was vital for the integration of the specific aspects of knowledge management involved in our case. A qualitative evaluation of each domain ontology was necessary and constituted a technical judgment of the ontologies and their associated environment with respect to a frame of reference. This frame of reference consisted of the requirements specifications, the competency questions (CQs), expert knowledge of the domain, etc., for each of the ontologies being developed. Such a qualitative evaluation has taken place, at least partly, as the various ontologies were tested to answer the competency questions initially defined. However, a more rigorous evaluation procedure needed to be imposed at the ontology development phases.

For this purpose the notion of an *evolution step* with regards to ontology development was introduced. An evolution step signifies a well-defined progress of an ontology by showing that it takes a concrete step towards completeness by capturing a richer set of -the initial- competency questions and adhering to the requirements specifications that have been specified. Evolution steps had a constant duration of two months.

Measuring progress can only be done with respect to competency questions and use-case scenarios. Both CQs and scenarios constitute requirements on the expressiveness and the amount of knowledge that needs to be captured by the ontologies. Therefore, the

evolution of domain ontologies must necessarily be justified through the presentation of new CQs that can be answered by each ontology and the modification of its structure in order to capture the new knowledge that is required.

Competency questions were initially specified in an informal way in natural language. To be able to use them as a qualitative measure of the ontology development process, however, they needed to be logically consistent and unambiguous. This implies that a formal way of describing CQs was necessary. Formal competency questions constitute a required step towards the definition of formal axioms (axioms, rules and restrictions in OWL), which facilitate the validation of the ontology. CQs were specified in a formal manner in every evolution step using *nRQL* (new Racer Query Language). The nRQL definition for a CQ can be retrieved when posing the query using the Semantic Search Engine. The Search Engine first creates the query in nRQL format that can be easily retrieved before submission and evaluation by the inference engine. This ensured that CQs were properly defined, were logically consistent and unambiguous.

5.2.2 Questionnaire Survey

During the FOCUS K3D project [Falcidieno et al. 2008], a questionnaire survey was contacted with the aim of collecting information on the methods and practices for 3D content use and sharing in various domains. Around eighty (80) people answered our questionnaires from nineteen (19) different countries coming both from Industry and Education/Research. Part of the goal of the questionnaires were to identify the use of current methodologies for handling 3D content and coding knowledge related to 3D digital content in different contexts and also to identify possible gaps and new areas of research. We summarize here some of our findings that also validated our efforts in developing the DSW.

Although the creation and/or capture of 3D models does not appear to trouble professionals (users, developers, scientists, creators of 3D content, publishers/dealers of 3D repositories, etc.), however the management of 3D collections is at an early stage or

non existent at all. Our survey results confirmed the current situation and revealed some specific problems and shortcomings related to 3D knowledge management.

Commonly, people deal with geometric and structural aspects of 3D models while the semantic aspect appears only for the management of more complex models. Almost everybody is aware of knowledge technologies, even if half of the people questioned claim not to be familiar with them. Concerning the adopted knowledge technologies, there is a wide preference for databases and metadata and some of them use quite often taxonomies and ontologies but not for 3D content.

Obviously the answers vary widely from scientists, researchers, developers, designers, project managers, to publishers and dealers of 3D content. Also concerning the actual amount of data, there is a huge variety ranging from just a few models (e.g. ten) to very large collections (e.g. thousands of models). Most of the data are stored on file servers or proprietary repositories. Often rather primitive approaches are used for handling 3D content (e.g. using file and folder names to encode information about the contained data) and that there is an apparent lack of information about knowledge technologies or at least a common misunderstanding or misuse of the related terminology. Tools that are specifically designed for organizing, browsing, and searching 3D content are commonly not used.

People from the Archaeology, Cultural Heritage and Medicine domains claim to be dissatisfied with the way they manage and store 3D content, while CAD/CAE & Virtual Product Modeling and Gaming & Simulation people are quite satisfied but they would expect an improvement. Areas mentioned for improvement or for which they feel that the use of 3D knowledge technologies can provide a solution include functionalities related to the documentation and identification of objects and parts of them, automatic extraction of geometric and semantic information from models, better visualization, and improved search using semantic and/or geometric criteria.

A noteworthy observation is that 3D collections are becoming more and more demanding in terms of management, preservation, and delivery mechanisms.

We also had several ad-hoc meetings where we demonstrated the functionality of the DSW and its browsing and searching facilities were evaluated. We mostly got positive feedback however there were some concerns about the usability of the Semantic Search Engine and browsing through ontologies. People find it difficult to browse or search the OMR without being familiar with the underlying ontology structure. But this was expected since the DSW addresses expert users with knowledge of the represented domain.

6 CONCLUDING REMARKS

This thesis focused on a knowledge-based approach to resource management, discovery and service composition, by making explicit and sharable the knowledge embedded in scientific resources. We provided the groundwork for the technical realization of an e-Science environment in the field of 3D shape processing and demonstrated the potential of such a framework in terms of establishing a novel, knowledge-based and semantically enriched solution for a collaborative and problem solving environment. We believe in the complementarily between knowledge management and Web Services.

In this chapter we summarize the results of this dissertation, present the main contributions of our work and discuss how the ideas presented here can be applied to the general scientific resource sharing, discovery and service composition problem. Finally, we discuss future research directions.

6.1 Contributions – Summary of Results

Our work intends to define a framework for capturing invaluable expert knowledge, sometimes implicitly contained in scientific objects and 3D resources in particular, that is mostly undocumented or is expressed in an inappropriate for our needs manner, and therefore hard to be reused or automated, with the goal of fostering the next generation of semantically-enabled systems and services. This work is based on an ontology-driven framework for formalizing, processing, managing and sharing knowledge about scientific resources. We believe that the addition of explicit semantics to 3D shape resources can significantly improve their discovery and reuse.

We summarize below the main contributions of this dissertation:

Achieving semantic interoperability - We proposed an ontology-based conceptualization of the domain knowledge and introduced knowledge representation

techniques in 3D shape modeling. The knowledge carried by digital representations of 3D objects was divided into three levels of granularity with respect to their knowledge content: the geometric level, the structural level and the semantic level.

The developed common ontologies represent our efforts towards the formalization of knowledge related to the geometric representation of shapes and constitute the basic step to build application-level ontologies. This reflects the fact that while the description of a shape can vary according to various contexts, its geometry remains the same and should be captured by a set of metadata that fully describe the properties of the geometric representation used.

In section 3.1 we presented the adopted ontology development methodology and briefly described the structure and some motivating scenarios behind the developed context-dependent ontologies, and showed how the common ontologies can be reused in practice to address context-specific challenges. Through the domain ontologies developed we demonstrated how to extend and adapt the common ontologies to conceptualize knowledge in different application areas.

The structure of the common ontologies is also used by the Shape Repository and Tool Repository to maintain semantic information for shape models and tools, where each object represents an instance of a particular class in the specified ontology. This demonstrates how semantic integration is utilized to facilitate the infrastructure and the tools/services, taking advantage of the information that are associated with digital shapes.

With the common ontologies we have demonstrated a systematic and formal approach for capturing a comprehensive description of shape-related content, from the geometrical information up to the knowledge pertaining to the context when a shape is used. We believe that the common ontologies represent a solid step for exploring the true potential of digital shapes and promote the reuse of shape information at much greater extent.

Through the ontology development we have demonstrated that preserving such expert knowledge and making it widely available enhances the value of each object and strengthens its potential for reuse in diverse application areas. Summarizing in a more

specific manner we demonstrated the usefulness of knowledge management technologies in developing an approach to formalize shape knowledge and contributing to the definition of an ontological framework for representing shapes.

Design and implementation of an ontology-driven knowledge base - The design and implementation of the Ontology & Metadata Repository (OMR) has been presented in detail (section 3.3), which constitutes a first step in creating a large-scale framework for promoting the use of knowledge technologies for managing, searching and retrieving 3D content. The primary goal of the OMR is the formalization and sharing of knowledge about 3D resources and their applications. The need for such a framework seems to grow exponentially as was discussed in the introductory and motivation sections of this thesis.

The OMR constitutes the knowledge base back-end for storing, querying and reasoning with such machine processable information. It supports the semantic searching framework, hence is closely related to the inference engine. The OMR lies in the heart of the Digital Shape Workbench (DSW) and is responsible for effectively integrating all DSW components.

Design of the Digital Shape Workbench architecture - We addressed the challenge of formulating an e-Science environment and its requirements from a perspective that unifies two distinct instances of resource sharing, namely Semantic Web and Web Services. A middleware architecture was designed in order to achieve a semantic-oriented representation of scientific resources able to provide a higher level of interaction, facilitate knowledge sharing and reuse, and ensure the preservation and exploitation of resource knowledge. In section 3.2 we presented the overall architecture of the Digital Shape Workbench (DSW), a unifying and highly extendable and truly distributed framework that supports modelling, storing, processing and reasoning about shape resources.

Efficient discovery and composition of scientific workflows - There is a semantic gap in current Web Services technologies. They do not deal with the definition of the meaning of services, since they provide only syntactic-level descriptions of their functionalities (no formal definition of what the syntactic definitions might mean),

making it difficult for requesters and providers to interpret or represent non-trivial statements such as the meaning of inputs and outputs or applicable constraints.

Understanding the semantics of Web Services is a key requirement for the discovery and composition of workflows. Composing services together is a new challenge for SOA middleware meeting e-Science environments and scientific computing.

In Chapter 4 we proposed an ontology-based organization and description of Web Services as an important requirement for enhancing the ability to compose processing chains (workflows) of Web Services. We addressed the issue of combining vertical and horizontal integration and discussed the main phases of the overall Semantic Web Service interaction process. Finally, we propose a generic service oriented middleware architecture for orchestrating, managing and executing composite and semantically enriched services.

Prototype implementation of scientific application services - As proof-of-concept we provided a prototype implementation of shape processing Web Services together with all the needed components to combine them into workflows. We used several shape processing tools/filters from the MeshLab open-source mesh processing system, wrapped them into Web Services and dynamically invoked them using BPEL processes. Two workflow scenarios were presented demonstrating the 3D model processing pipeline. This way, we proved that semantically enriched Web Services can improve the discovery and composition of workflows in knowledge-intensive and computationally-complex application areas.

6.2 Directions and Future Work

Important issues in knowledge management, resource discovery and service composition remain to be solved in the fields of Semantic Web and Web Services related to e-Science. The research directions described in this section cover both general unresolved problems but also specific future work that can be considered as a natural continuation of this thesis.

The first major research direction concerns 3D knowledge management. It has been predicted that 3D is bound to become the fourth wave of digital multimedia communication, where the first three waves were sounds in the 70's, images in the 80's, and videos in the 90's. 3D digital shapes are therefore expected to take a central role in the Semantic Web in the coming years, with high potential impact in several emerging application areas like Medicine & Bioinformatics, Gaming & Simulation, CAD/CAE & Virtual Product Modeling and Archaeology & Cultural Heritage.

Therefore, effective 3D resource discovery is a major challenge. For instance, a next generation search mechanism for 3D content could integrate content-based geometry-driven criteria with concept-based semantic-driven ones (see Figure 6.1). This search mechanism can be based on an innovative query formulation that supports geometric, semantic and combined search modalities. For example, it could be possible to pose queries such as *"find the 3D models in the repository that represent a vase with handles, and whose handles are globally similar in shape to a given query model"*. The words *"vase"* and *"handle"* could refer to semantic annotations of objects and be resolved via a

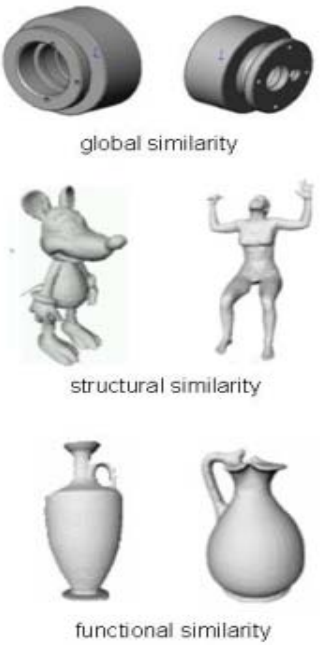


Figure 6.1: Retrieval of 3D content by combining geometric, structural and semantic criteria [Courtesy of IMATI-CNR].

semantic search, whereas "*handles are globally similar in shape*" could be resolved by applying a geometric search to the models selected by the semantic search. This kind of query formulation interface would provide the unprecedented capability of capturing the intuitive notion of similarity expressed by the user in the query session.

Whereas text and image digitization and management are rather mature technologies, the technology necessary to benefit from 3D knowledge management is not yet fully exploited. There are still many areas in which further progress in research and consolidation of approaches is required. Methods and tools are required for making digital shapes machine understandable and not just human-understandable as they are today. Developing semantic mark-up of content, in addition to intelligent agents and ontology infrastructures for 3D content can contribute to this direction [Bustos et al. 2007].

Semantically enriched descriptions, indexing, and retrieval will allow the deep integration of 3D content into Digital Libraries, and fascinating new applications can be envisioned. Intelligent 3D acquisition will automatically segment and interpret any scanned objects, semantically enriching the data. New possibilities to work with 3D data will emerge, recognizing not only the low-level shape features, but also its structure and semantics. Once 3D editors are made aware of semantic properties of the models, the composition of new 3D objects based on existing content will be greatly simplified. 3D search engines will become highly intelligent, utilizing the available semantic shape analysis methods and the many different similarity notions on all conceptual levels.

Current limitations are mostly related to the methodologies used and the lack of specialized tools for 3D knowledge management. The areas in which progress would be particularly important and might be achieved in the relatively short term are:

- Developing solutions for improved re-use of 3D datasets by end-users. There is a need to support better and easy re-use of 3D objects as well as their use in a variety of contexts. For example, enhancing 3D repositories so as to exploit and reuse semantic annotations

- Developing easy-to-use authoring tools for 3D knowledge management. Digitizing objects in 3D format is only a first step. The creation of collaborative and problem solving environments (easy-to-use toolkits) for end-user interaction will certainly require the involvement of knowledge management technologies.
- Automate the knowledge extraction from shapes. Manual annotation of semantics is inconvenient in general, even more in the case of such complex objects as 3D shapes due to the mass of information involved and the complexity of their formal representation. In order to facilitate automatic annotation we can define rules for associating semantics to lower-level knowledge in shapes. We can use ontology structures to define these rules but we also need tools to actually perform the semantic annotation. For some specific types of digital shapes (namely, manifold surface meshes, non-manifold meshes, multi-dimensional structural descriptors and key frames), different automatic annotation tools have been recently developed [Papaleo et al. 2007] in order to extract useful information from a specific digital shape. One step further has been taken in [Attene et al. 2007] with the development of a prototype system for the semi-automatic annotation of shapes and shape parts in a context specified by an ontology.
- Moving towards non-textual, semantic documentation and processing of 3D content. With the growth in 3D content there is an ever increasingly pressing need for progress in the field of semantic processing of such content. Development of ontologies is crucial to 3D content categorization, indexing, detailed search and retrieval, and enhanced access.
- Undertaking ontological modelling to establish a broader base of domain knowledge, which is systematically encoded.
- Developing tools for the integrated management of the metadata related to 3D objects. There are specific aspects of these metadata that are unique to the digital domain including the digital object structure and parameters, the relationship of the digital object to the physical object (e.g., coordinate systems and metrics); the

digital provenance (i.e. processes used to create the model from the original data, tools and parameters employed) etc. The digital provenance is a vital part of the long-term usability of the digital object, providing information for migration to new formats in the future. These tools would be based on the ontological framework mentioned earlier.

- Developing solutions for improved search. There is a need to better support the end-user in identifying and re-using 3D digital objects. Search and retrieval will be used in a variety of contexts with different search criteria.

The second research direction investigates in more detail matters of Service-oriented Architectures (SOA) from the perspective of scientific applications. Our near-term focus is on semantics for user-directed service interactions: users should specify what exactly they want from a service (its functionality), rather than the details of how to discover it. By using information from published semantic service descriptions we can mediate interactions with classes of functionally similar services (or workflows of services), even when their detailed interfaces differ.

As the technologies mature, we anticipate that methods now being explored will support more widespread use of mechanisms for automated service discovery and matchmaking. A key element of this phase will be the development of shared, extensible, community-wide ontologies for describing capabilities and services. Open-source ontologies for different kinds of services and products will enable broad-based, automated, service discovery in the same way search engines now make it easy to discover new Web sites.

A major issue when defining a composite service is whether its component services are composable. For example, it would be difficult to invoke an operation if there is no matching between the parameters requested by this operation (e.g., data types, number of parameters) and those transmitted by the client. In general, a model has to be defined by a set of composability rules. Each rule compares a specific feature of interacting Web services. Since those features are relevant to different aspects of Web Services (syntactic,

semantic, behavioral, and qualitative), these rules can be organized into several levels. Due to the sheer heterogeneity of scientific Web Services, it is not always possible to find services that are fully composable. Composers may, in this case, select services that are partially composable and then adapt their requests based on the results returned by a composability process.

In the broader context, monitoring is recognized as a natural part of Service-oriented architectures. In SOA systems the main stress is usually put on monitoring of performance and availability metrics. The monitoring sub-system should be able to support tasks such as measuring and evaluation of Quality of Services (QoS) metrics, enforcement of Service Level Agreements (SLA), notifications and auditing of service performance, alert-based reporting on the level of adherence to the SLA, or sending automatic notifications and allow graceful exception handling when the SLAs break down. To address these challenges in the context of semantically enriched Web services, an ontology for specification of individual events can be developed. During service execution, the execution infrastructure emits semantically annotated events specific to the state of process model execution. The content of emitted event instances describes the execution context in the time when the event occurred and other information relevant to the given event type. The content is annotated by the same domain ontology concepts that are used in the service definition itself, which allows a more flexible events detection techniques than those derived from a simple syntactic key-words matching. We could also employ semantic reasoning for detection of primitive events based on matching their event type and the content.

In our current system architecture, we did not deal with accounting and quality of service constraints, which should be addressed as the resource utilization grows. Although our framework is fault tolerant by design, we have not alocated a significant amount of time in investigating the limits at which it breaks down. We plan on addressing this in our future efforts.

We identify the following directions for future research: dynamic composition of Web Services, dependable service composition, support of mobile services, grid and cloud services, and semantic monitoring.

Dynamic Composition of Web Services - The number of services to be integrated may be large and continuously changing. Web Service composition requires flexibility to dynamically adapt to changes that may occur in partners' applications. Services must be able to respond rapidly to operational changes (e.g., server load) or possibly the availability of alternate services that provide “similar" functionality. The support of dynamic composition will facilitate the establishment of *on demand* and *real-time* partnerships. Services will not statically be bound to each other. New services with relevant features should be dynamically discovered and assembled. Currently, relationships among component services are established mostly at development time. While Web Service technologies provide capabilities for defining services, they are clearly not sufficient to facilitate the establishment of dynamic relationships and more research effort is needed in this direction.

Dependable Composition of Web Services - Transaction support is required to provide reliable and dependable execution of composite services. Traditional transaction management techniques are not appropriate in the context of composite services. The participants of a composite service may be heterogeneous and autonomous. They may not be transactional and if they are, their transactional features may not be compatible with each other. In addition, participant services, for different reasons (e.g., quality of services), may not be willing to comply with constraints such as resource locking, until the termination of the composite service execution. New transaction techniques are required in the context of Web Services. For instance, it is important to extend the description of services by explicitly describing transactional semantics of Web Service operations. An example is to specify that an operation can be aborted without effect from a requester's perspective. It is also imperative to extend service composition models to specify transactional semantics of an operation or a group of operations. An example is to specify how to handle the unavailability of a participant service. The effective handling

of transactional aspects at the composite service level should be facilitated by exploiting the transactional capabilities of participant services.

Grid and cloud Services - Grid and cloud computing is another research area we would like to explore in the future. The aim of research on grids is to provide scalable and transparent methods for accessing resources in distributed environments. Grid concepts and technologies were first developed to enable scientific collaborations. Applications include collaborative visualization of large scientific datasets and computationally demanding data analyses. Just like the Web began as the technology for scientific collaboration and was widely adopted in various application areas such as e-commerce and e-government, a similar trajectory is expected for Grid and cloud technologies for e-Science.

One of the challenges is the support of dynamic integration of resources. These resources must be discovered on-the-fly, selected when requests are submitted, and released after the requests are fulfilled. In both grids and Web Services, we often need to integrate resources across distributed, heterogeneous, and autonomous systems. One possible way to realize grids on the Web is to view grid resources as Web Services. In this way, grids can be defined by reusing, assembling, and coordinating existing resources. The challenge is to define techniques for semantically discover, select, create, and assemble grids using Web Service composition techniques.

Semantic Monitoring - While the advantages of semantically annotated Web Services are recognized in the context of service discovery and composition, little effort has been invested into studying possibilities of Semantic Web services monitoring (semantic monitoring). For example, a mediation component should be able to dynamically execute the process models and also be able to interpret the flow of execution and its results, including possible failures and their causes.

APPENDIX A: 3D Modeling Terminology

A short list of important terms concerning 3D Modeling, which do not cover all the aspects of the presented subjects in Chapter 4.4, is given in this Appendix.

Breakline: Feature line or polyline representing a ridge or some other feature (e.g. folds) that the user wishes to preserve in a mesh made up of polygonal elements. Therefore a breakline is a singularity on a surface, like an edge to which the polygons should conform to, i.e., not intersect.

Fairing: Today the word *fairing* relates to designing and editing smooth curves and surfaces. Before computers the smooth curves describing the hulls of ships were faired by using an elastic thin wood ruler, called the spline.

Level of detail (LoD): it is the amount of information (detail) or complexity of any object that is displayed at any particular time. The LoD is usually a function of the distance of the object from the viewer.

Manifold: A (separable Hausdoff) *k*-dimensional topological space *M* in which each point has a neighbourhood which is homeomorphic either to the *k*-dimensional open ball, or to the half-ball.

Marching cubes: An algorithm that extracts an implicit surface represented by a 3D regular grid of data values. This grid is often referred to as a grid of voxels. The eight corners of a voxel give a total of alternative 256 different configurations for the corner status. By standard operations these can be group into 15 alternative configurations. Triangulations representing these 15 can be pre-computed, and thus efficiently used when tracing the implicit surface described by the grid of voxels.

Mesh: It is a collection of triangular (or quadrilateral) contiguous, non-overlapping faces joined together along their edges. A mesh therefore contains vertices, edges and faces and its easiest representation is a single face. Sometimes it is also called TIN, Triangulated Irregular Network. Finite element methods are generally used to generate a

surface mesh. A grid-like polygonal subdivision of the surface of a geometric model. More formally, it is an Euclidean cell complex such that any k-cell of Γ, with $k<d$, bounds at least one d-cell of Γ.

Modeling: The (mathematical) construction and computer implementation of an object, by defining points in a 3 dimensional array. This array is based on the X, Y and Z axis of geometric space. Then, different sets of these points are mathematically 'joined' by e.g. lines, to create polygons and the polygons joined to create objects. The simplest result is usually displayed as a wireframe model.

Point Cloud: A set of uncorrelated points, usually in 3D, which have to be further elaborated to obtain a 3D model.

Range image: A grid of distances (range points) that describe a surface in Cartesian (height field) or cylindrical coordinates. A range scanner senses 3D positions on an objects surface and returns an array of distance values.

Reconstruction: The process of making a surface geometry or volume type description of a 3D object based on measured points coordinates on the outer and inner surfaces of the 3D object. Automatic reconstruction is a very challenging process, as the point sets can have wrong points, can miss points or do not contain a sufficient amount of point to describe critical surface regions accurately. If a point set is produced by combining measurement from different camera positions, there is often drifting of the data between the measurements. Such drifting will often result in false surface features.

Registration: A rigid transformation that brings points of one range image into alignment with portions of a surface it shares with another range image.

Remeshing: There is no precise definition, since it often varies according to the targeted goal or application. Nonetheless, it is possible to say that, given a 3D mesh, remeshing consists in computing another mesh, whose elements satisfy some quality requirements, while approximating well the input. *Quality* has several meanings. It can be related to the sampling, grading, regularity, size and shape of elements. Often a combination of these criteria is desired in real applications. Some remeshing techniques proceed by altering the input, and some generate a new mesh from scratch.

Rendering: Usually realistic drawing of 3D objects using computer technologies. In order to render an object, certain properties like transparency, color, diffuse or specular reflection and refraction must be assigned to it and to the surrounding environment. Two common rendering techniques are called *ray tracing* (it renders the model object by object (or better pixel by pixel), testing if a ray intersects any object in the scene and calculating the light intensity in that particular direction) and *scanline* rendering (it renders the image of an object as a series of horizontal or vertical lines). Ray tracing is not suitable for real-time visualization. Scanline does not produce as realistic results as ray tracing, but it is frequently used in animation packages where the image quality of the single frames is not so important. Rendering can be geometry-based or image-based (usually called 'Texture mapping').

Sampling: The process of obtaining a sequence of point coordinates from a 3D model.

Splines: A piecewise polynomial function that can have a locally very simple form but at the same time be globally flexible and smooth. Splines are very useful for modeling arbitrary functions and are used extensively in computer graphics for free-form curves and surfaces representation. A class of parametric curves and surfaces is the Non-Uniform Rational B-Spline (NURBS) curve or surface. They are the generalization of non-rational B-splines, which are basis of polynomial function based on rational Bézier curves.

Surface: A compact, connected, orientable 2 or 3D manifold, possibly with boundary. A surface without boundary will be called a closed surface. A surface can be geometrically represented in implicit form (locus of the solution of the function F(x,y,z)=0) or parametric form (a collection of parametric patches properly joined together). Surfaces, which cannot be described in an analytic form, are called free-form surfaces.

Triangulation: Denoting $|S|$ the topological space underlying a simplicial complex S, ($|S| = \cup_{\sigma \in S} \sigma$), then a triangulation of the topological space M is a case of simplicial complex S such that $|S| = M$.

Improperly, the term triangulation is also used to denote the geometric realisation of a bi-dimensional simplicial complex (also known as *triangle mesh*); while tetrahedralisations are a subclass of three-dimensional simplicial complexes.

Water tightness: In general there are small gaps between the surface elements describing the shells (inner and outer) in a model. In a simulation model such small gaps are not allowed, the elements constituting the simulation model has to have an exact match, or in other words be water tight.

APPENDIX B: List of Publications

The following list of publications are authored or co-authored by Marios Pitikakis and are related to the work presented in this dissertation.

In Scientific Journals

M. Pitikakis, M. Vavalis and C. Houstis, "Semantically Interoperable 3D Scientific Objects", Knowledge Engineering Review (KER), 2010, accepted (to appear).

G. Vasilakis, A. Garcia-Rojas, L. Papaleo, C. E. Catalano, F. Robbiano, M. Spagnuolo, M. Vavalis, M. Pitikakis, “Knowledge-based representation of 3D media”. International Journal of Software Engineering and Knowledge Engineering (IJSEKE), Vol. 20, No. 5, August 2010, pp. 739-760.

C. Houstis, S. Lalis, M. Pitikakis, G. Vasilakis, K. Kritikos, A. Smardas, "A grid service-based infrastructure for accessing scientific collections: The case of the ARION system", The International Journal of High Performance Computing Applications, Vol. 17,No. 3, Fall 2003, pp. 269-280.

In Book chapters

M. Pitikakis, “A semantic based framework for managing, searching and retrieving 3D resources”, pp. 195-204. Heritage in the Digital Era, ISBN 978-1-907132-25-4, Multi-Science Publishing, 2010.

C. Houstis, S. Lalis, E. Vavalis, M. Pitikakis and G. Vasilakis, Chapter VII “Advanced Middleware for e-Science Applied to Environmental Integrated Systems”, pp. 76-93. Information Systems for Sustainable Development, ISBN 1-59140-342-1**,** Ideal Group Publishing, 2005.

In Conference Proceedings and Workshops with review

B. Falcidieno, M. Pitikakis, M. Spanguolo, M. Vavalis, C. Houstis, “FOCUS K3D: Promoting the use of knowledge intensive 3D media”, 14th International Conference on

Virtual Systems and Multimedia Dedicated To Digital Heritage (VSMM'08), Lemessol, Cyprus, October 20-26, pp. 329 – 333, 2008.

G. Vasilakis, A. G. Rojas, L. Papaleo, C. E. Catalano, F. Robbiano, M. Spagnuolo, M. Vavalis and M. Pitikakis, "A common ontology for multi-dimensional shapes", 1st Workshop on Multimedia Annotation and Retrieval enabled by Shared Ontologies (MAReSO 2007), Genoa, 5 December, 2007.

B. Falcidieno, M. Spagnuolo, M. Pitikakis, G. Vasilakis, A. Garcia-Rojas, L. Papaleo, "AIM@SHAPE: Research Advantages and Future Contributions", Poster, 1st International Conference on Semantics and Digital Media Technology (SAMT 2006), Athens, Greece, 6 - 8 Dec., 2006.

G. Vasilakis, M. Pitikakis, M. Vavalis and C. Houstis, "A Semantic Based Search Engine for 3D Shapes: Design and Early Prototype Implementation", 2nd European Workshop on the Integration of Knowledge, Semantic and Digital Media Technologies, (EWIMT2005), London, UK, 30 Nov. – 1 Dec., pp. 391 – 397, 2005.

M. Pitikakis, C. Houstis, G. Vasilakis, and M. Vavalis, "A Knowledge Management Architecture for 3D Shapes and Applications", 10th Panhellenic Conference on Informatics (PCI2005), Volos, Greece, 11-13 November, 2005, Springer, LNCS 3746, pp. 360 – 370, 2005.

R. Albertoni, L. Papaleo, M. Pitikakis, F. Robbiano, M. Spagnuolo and G. Vasilakis, "Ontology-based Searching Framework for Digital Shapes", IFIP WG 2.12 & WG 12.4 International Workshop on Web Semantics (SWWS'05), Agia Napa, Cyprus, 1-2 November, 2005, Springer, LNCS 3762, pp. 896-905, 2005.

J. B. Stav, H. Tsalapatas, M. Pitikakis, D. Korbetis, "System Design of an Independent European AVD-Knowledge Base", 4th European Conference on e-Learning (ECEL'05), Amsterdam, Netherlands, 10-11 November, pp. 493-504, 2005.

S. Lalis, C. Houstis, M. Pitikakis, G. Vasilakis, and M. Vavalis, "Providing Support for Integrated Scientific Computing: Metacomputing meets the Semantic Web and the Grid", 5th IEEE International Symposium on Cluster Computing and the Grid (CCGRID), Cardiff, UK, 9-12 May, pp. 631-637, 2005.

C. Houstis, M. Pitikakis, G. Vasilakis and M. Vavalis, "Semantic Web Technologies for Searching, Retrieving and Adding Value to Large Scale Scientific Data", PV-2004 Symposium, Ensuring the Long-Term Preservation and Adding Value to the Scientific and Technical Data, European Space Agency, ESRIN Centre, Frascati, Italy, 5 - 7 October 2004.

C. Houstis, S. Lalis, V.Christophides, D. Plexousakis, E. Vavalis, M. Pitikakis, K. Kritikos, A. Smardas, "A Data, Computation and Knowledge Grid: the case of the ARION system", 5th International Conference on Enterprise Information Systems (ICEIS2003), Angers, France, 23-26 April, pp. 359-365, 2003.

C. Houstis, S. Lalis, V.Christophides, D. Plexousakis, E. Vavalis, M. Pitikakis, K. Kritikos, A. Smardas, Ch. Gikas, "A Service infrastructure for e-Science: the case of the ARION system", 14th International Conference on Advanced Information Systems Engineering (CAiSE 2002), E-Services and the Semantic Web workshop (WES2002), Toronto, Canada, 27-28 May, 2002, Springer, LNCS 2512, pp. 175-187, 2002.

REFERENCES

3D-COFORM (2010), 'Tools and expertise for 3D collection formation, FP7-IST Collaborative project', Website. http://www.3d-coform.eu

AceMedia (2005), 'Towards a common multimedia ontology framework report, FP6-IST Integrated Project', Website. http://www.acemedia.org

Aduna Open Source project (2009), 'Sesame: an open source RDF framework', Website. http://www.openrdf.org

AIM@SHAPE (2005), 'FP6-IST Network of Excellence: Advanced and innovative models and tools for the development of semantic-based systems for handling, acquiring, and processing knowledge embedded in multidimensional digital objects', Website. http://www.aimatshape.net.

AIM@SHAPE (2006*a*), 'The Digital Shape Workbench', Website. http://dsw.aimatshape.net

AIM@SHAPE (2006*b*), 'Shape Repository', Website. http://shapes.aimatshape.net

Akkiraju, R., Farrell, J., Miller, J., Nagarajan, M., Schmidt, M.T., Sheth, A. and Verma, K. Web Service Semantics. http://www.w3.org/Submission/WSDL-S/

Alamri A., Eid M. and Saddik A. El. (2006). Classification of the state-of-the art dynamic web services composition techniques. *International Journal of Web and Grid Services*, 2:148 – 166, 2006.

Albertoni, R., Papaleo, L., Robbiano, F. and Spagnuolo, M., (2006), Towards a conceptualization for shape acquisition and processing, *in* 'Proceedings of the 1st Workshop on Shape and Semantic', Matsushima, Japan, pp. 85–90.

Albertoni, R., Papaleo, L., Pitikakis, M., Robbiano, F., Spagnuolo, M. and Vasilakis, G., (2005) , Ontology-based Searching Framework for Digital Shapes, OTM Workshops, Springer, LNCS Vol. 3762, pp. 896-905.

Alves, A., Arkin, A., Askary, S., Barreto, C., Bloch, B., Curbera, F., Ford, M., Goland, Y., Guízar, A., Kartha, N., Liu, C.-K., Khalaf, R., König, D., Marin, M., Mehta,

V., Thatte, S., van der Rijn, D., Yendluri, P. and Yiu, A. (2007). Web Services Business Process Execution Language Version 2.0, OASIS Standards Specification, April 2007. See: http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html

Ankolekar, A., Burstein, M. H., Hobbs, J. R., Lassila, O., Martin, D. L., McIlraith, S. A., Narayanan, S., Paolucci, M., Payne, T. R., Sycara, K. P. and Zeng, H. (2001). DAML-S: Semantic markup for web services. In Isabel F. Cruz, Stefan Decker, Jérôme Euzenat, and Deborah L. McGuinness, editors, SWWS, pp. 411–430.

Arnaud, R. and Barnes, M. (2006), *Collada: Sailing the Gulf of 3D Digital Content Creation*, AK Peters Ltd.

Arndt, R., Troncy, R., Staab, S., Hardman, L. and Vacura, M. (2007), Comm: Designing a well founded multimedia ontology for the web, *in* '6th International and 2nd Asian Semantic Web Conference (ISWC2007+ASWC2007)', pp. 29–42. **URL:** *http://data.semanticweb.org/conference/iswc-aswc/2007/tracks/research/papers/29*

Attene, M., Robbiano, F., Spagnuolo, M. and Falcidieno, B. (2007). Part-based Annotation of Virtual 3D Shapes. In: Proceedings of Cyberworlds, NASAGEM Workshop.

ATK project (2008), '3store: an RDF triplestore', Website. http://threestore.sourceforge.net

Austin, D., Barbir, A., Ferris, C. and Garg, S. (2004). Web Services Architecture Requirements, W3C Working Group Note 11. See: http://www.w3.org/TR/wsa-reqs/

Baader, F., Calvanese, D., McGuinness, D., Nardi, D. and Patel-Schneider, P., eds (2003), *The description logic handbook: theory, implementation, and applications*, Cambridge University Press, New York, NY, USA.

Babik, M. and Hluchy, L. (2008), 'A testing framework for owl-dl reasoning', Fourth International Conference on Semantics, Knowledge and Grid, pp. 42–48.

Barros, A., Dumas, M. and Oaks, P. (2006). Standards for Web Services Choreography and Orchestration: Status and Perspectives. C. Bussler et al. (Eds): BPM 2005 Workshop, Springer, LNCS 3812, pp. 61-74.

Beckett, D. (2004). RDF/XML Syntax Specification (Revised). Recommendation, World Wide Web Consortium (W3C). See http://www.w3.org/TR/rdf-syntax-grammar/

Bellur, U. and Kulkarni, R. (2007). Improved matchmaking algorithm for semantic web services based on bipartite graph matching. In: *IEEE International Conference on Web Services*, pages 86–93.

Berners-Lee, T. (1998). SemanticWeb Road map. Internal note, World Wide Web Consortium (W3C). See http://www.w3.org/DesignIssues/Semantic.html

Bloehdorn, S., Simou, N., Tzouvaras, V., Petridis, K., Handschuh, S., Avrithis, Y., Kompatsiaris, I., Staab, S. and Strintzis, M. G. (2004), Knowledge representation for semantic multimedia content analysis and reasoning, *in* 'Proc. of European Workshop on the Integration of Knowledge, Semantics and Digital Media Technology (EWIMT)', London, U.K., November 25-26, 2004. **URL:** *http://www.image.ece.ntua.gr/publications.php*

Brickley, D. and Guha, R. (2004). RDF Vocabulary Description Language 1.0: RDF Schema. Recommendation, World Wide Web Consortium (W3C). See http://www.w3.org/TR/2004/REC-rdf-schema-20040210/

Burstein, M., Bussler, C., Finin, T., Huhns, M.N., Paolucci, M., Sheth, A.P., Williams, S. and Zaremba M. (2005). A semantic web services architecture. Internet Computing, IEEE, 9(5), pp. 72–81.

Bustos, B., Fellner, D., Havemann, S., Keim, D., Saupe, D. and Schreck, T. (2007), Foundations of 3D digital libraries - current approaches and urgent research challenges, *in* 'Proc. 1st International Workshop on Digital Libraries Foundations DLF'07)', DELOS Network of Excellence on Digital Libraries (www.delos.info), pp. 7–12.

Catalano, C., Camossi, E., Ferrandes, R., Cheutet, V. and Sevilmis, N. (2008), 'A product design ontology for enhancing shape processing in design workflows', *Journal of Intelligent Manufacturing*, 20(5):553-567, 2008.

Catton, C., Sparks, S. and Shotton, D. (2005), The imagestore ontology and the bioimage database: semantic web tools for biological research images, The International Conference on Computer as a Tool, EUROCON 2005, pp. 257–264.

Chen, D.-Y., Tian, X.-P., Shen, Y.-T. and Ouhyoung, M. (2003), 'On visual similarity based 3D model retrieval', *Comput. Graph. Forum* **22**(3), pp. 223–232.

Chinnici, R., Moreau, J.-J. , Ryman A. and Weerawarana S. (2007). *Web Services Description Language (WSDL) 2.0 -- Part 1: Core Language*, W3C Recomentation, 26 June, 2007. See: http://www.w3.org/TR/wsdl20/

Christudas, B., A. (2008). *Service Oriented Java Business Integration*, Packt Publishing, ISBN 1847194400.

Cimpian, E. and Mocan, A. (2005). WSMX process mediation based on choreographies. In Business Process Management Workshops, pp. 130–143

Clement, L., Hately, A., von Riegen, C. and Rogers, T. (2004). *Universal Description, Discovery and Integration (UDDI) protocol*, OASIS Standards Specification. See: http://www.oasis-open.org/committees/uddi-spec/doc/spec/v3/uddi-v3.0.2-20041019.htm

COLLADA (2009), '3D Asset exchange schema', Website. http://www.khronos.org/collada.

Daras, P., Zarpalas, D., Tzovaras, D. and Strintzis, M. G. (2004), 'Shape matching using the 3D radon transform', *International Symposium on 3D Data Processing Visualization and Transmission*, pp. 953–960.

Dean, M. and Schreijber, G. (2004). OWL Web Ontology Language Reference. Recommendation, World Wide Web Consortium (W3C). See http://www.w3.org/TR/owl-ref/

Decker, G., Kopp, O., Leymann, F. and Weske, M. (2007). BPEL4Chor: Extending BPEL for Modeling Choreographies. ICWS 2007, pp.296-303.

Dogac, A., Kabak, Y. and Laleci, G. (2004). "Enriching ebXML registries with OWL ontologies for efficient service discovery," in Proc. of RIDE'04, Boston, March 2004.

Dogac, A., Kabak, Y., Laleci, G.-B., Mattocks, C., Najmi, F. and Pollock, J. (2005). "Enhancing ebXML Registries to Make them OWL Aware", Distributed and Parallel Databases Journal, Springer-Verlag, Vol. 18, No. 1, pp. 9-36.

Doulaverakis, C., Kompatsiaris, Y. and Strintzis, M. (2005), Ontology-based access to multimedia cultural heritage collections – the REACH project, The International Conference on Computer as a Tool, EUROCON 2005, Vol. 1, pp. 151–154.

Dublin Core (2006), 'Dublin core metadata initiative', Website. http://dublincore.org

ebXML Technical Architecture Team (2001). ebXML technical architecture specification v1.0.4. online: http://www.ebxml.org, February 2001.

EPOCH (2008), 'Excellence in processing open cultural heritage, FP6-IST Network of Excellence', Website. http://www.epoch-net.org

Falcidieno, B., Pitikakis, M., Spagnuolo, M., Vavalis, M. and Houstis, C. (2008). FOCUS K3D: Promoting the use of knowledge intensive 3D media, *14th International Conference on Virtual Systems and Multimedia Dedicated To Digital Heritage* (VSMM'08), Lemessol, Cyprus, October 20-26, pp. 329 – 333.

Falcidieno, B., Spagnuolo, M., Alliez, P., Quak, E., Houstis, C. and Vavalis, M. (2004), Towards the semantics of digital shapes: the AIM@SHAPE approach, *in* 'European Workshop on the Integration of Knowledge, Semantic and Digital Media Technologies EWIMT 2004, November 25-26, 2004, London, UK, IEE.

Farrell, J. and Lausen, H. (2007). Semantic annotations for WSDL and XML Schema. http://www.w3.org/TR/sawsdl/

Fensel, D., van Harmelen, F., Horrocks, I., McGuinness, D. and Patel-Schneider, P. (2001). OIL: An Ontology Infrastructure for the Semantic Web, *IEEE Intelligent Systems*, March/April 2001,Vol. 16,No. 2.

Fielding, R., T. (2000). Architectural Styles and the Design of Network-based Software Architectures. PhD thesis, University of California, Irvine.

Fikes, R., Hayes2, P. and Horrocks, I. (2003), OWL-QL - A language for deductive query answering on the semantic web, Technical Report KSL-03-14, Knowledge Systems Laboratory, Computer Science Department, Stanford University. **URL:** *ftp://ftp.cs.pitt.edu/chang/revs/c916.pdf*

FOCUS K3D (2008), 'FP7-IST coordination action', Website. http://www.focusk3d.eu

Foster, I., Kesselman, C., Nick, J. and Tuecke, S. (2002). Grid Services for Distributed System Integration, *Computer*, 35(6).

Foster, I. (2005). Service-Oriented Science, American Association for the Advancement of Science.

Funkhouser, T., Kazhdan, M., Min, P. and Shilane, P. (2005), 'Shape-based retrieval and analysis of 3d models', *Commun. ACM* **48**(6), pp. 58–64.

Galinski, J., Kaya, A. and Moller, R. (2005), Development of a server to support the formal semantic web query language OWL-QL, *in* I. Horrocks, U. Sattler & F. Wolter, eds, 'Proc. International Workshop on Description Logics'.

Gruber, T. R. (1993), A translation approach to portable ontology specification, *Knowledge Acquisition* 5(2), pp. 199–220.

Gurevich, Y. (1995). Evolving algebras 1993: Lipari guide. Specification and Validation Methods, pp. 9–36.

Gutirrez, M., Garca-Rojas, A., Thalmann, D., Vexo, F., Moccozet, L., Magnenat-Thalmann, N., Mortara, M. and Spagnuolo, M. (2007), 'An ontology of virtual humans: Incorporating semantics into human shapes', *The Visual Computer* **23**(3), pp. 207–218.

Haarslev, V., Möller, R and Wessel, M. (2004). Querying the Semantic Web with Racer + nRQL. In Proceedings of the KI-2004 International Workshop on Applications of Description Logics (ADL'04).

Hendler, J. (2003), 'Science and the semantic web', *Science* **299**(5606), pp. 520–521.

Horrocks, I., van Harmelen, F., Patel-Schneider, P., Berners-Lee, T., Brickley, D., Connoly, D., Dean, M., Decker, S., Fensel, D., Hayes, P., Heflin, J., Hendler, J., Lassila, O., McGuinness, D., and Stein, L. A. (2001). DAML+OIL. http://www.daml.org/2001/03/daml+oil-index.html

Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., and Dean, B. G. (2003). SWRL: A semantic web rule language combining OWL and RuleML. see http://www.daml.org/2003/11/swrl/

Houstis, C., Lalis, S., Vavalis, E., Pitikakis M. and Vasilakis, G. (2005). Advanced Middleware for e-Science Applied to Environmental Integrated Systems, *Information*

Systems for Sustainable Development, ISBN 1-59140-342-1**,** Ideal Group Publishing, pp. 76-93.

Houstis, C., Lalis, S., Christophides, V., Plexousakis, D., Vavalis, E., Pitikakis M., Kritikos, K. and Smardas, A. (2003). A Data, Computation and Knowledge Grid: the case of the ARION system, 5th International Conference on Enterprise Information Systems (ICEIS2003), Angers, France, 23-26 April, pp. 359-365.

Houstis, C., Lalis, S., Christophides, V., Plexousakis, D., Vavalis, E., Pitikakis M., Kritikos, K., Smardas, A. and Gikas, Ch. (2002). A Service infrastructure for e-Science: the case of the ARION system, 14th International Conference on Advanced Information Systems Engineering (CAiSE 2002), E-Services and the Semantic Web workshop (WES2002), Toronto, Canada, 27-28 May, 2002, Springer, LNCS 2512, pp. 175-187.

Humanoid animation (2009), 'H-anim',Website. http://h-anim.org/Specifications/H-Anim200x

Hunter, J. (2001), Adding multimedia to the semantic web - building an mpeg-7 ontology, *in* 'In International Semantic Web Working Symposium (SWWS)', pp. 261–281.

Hunter, J., Koopman, B. and Sledge, J. (2003), Software tools for indigenous knowledge management, *in* 'Proceedings of the Museums and the Web, Charlotte, 19-22 March 2003'. **URL:** *http://www.archimuse.com/mw2003/papers/hunter/hunter.html*

IBM Corporation (2004), 'IODT: Integrated ontology development toolkit', Website. http://www.alphaworks.ibm.com/tech/semanticstk

International Council of Museums, 'CIDOC Conceptual Reference Model', Website. http://cidoc.ics.forth.gr/

ISO FDIS 21127 (2007), 'Information and documentation - A reference ontology for the interchange of cultural heritage information', Website. http://www.iso.org/iso/catalogue_detail.htm?csnumber=34424

ISO-IEC Moving Picture Experts Group working group (2004), 'MPEG-7 overview', Website. http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm

Jiantao, P. and Ramani, K. (2007), 'An integrated 2D and 3D shape-based search framework and applications', *Computer-Aided Design & Applications* **4**(6), pp. 817–826.

Kavantzas, N., Burdett, D., Ritzinger, G., Fletcher, T., Lafon, Y. and Barreto, C. (2005) Web Services Choreography Description Language Version 1.0, W3C Candidate Recommendation 9 November 2005. See: http://www.w3.org/TR/ws-cdl-10/

Kazhdan, M., Funkhouser, T. and Min, P. (2004) 'A comparison of text and shape matching for retrieval of online 3d models', *Lecture Notes in Computer Science Including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics*, **3232**, pp. 209–220.

Kifer, M., Lausen, G. and Wu, J. (1995). Logical foundations of object-oriented and frame-based languages. Journal of the ACM, 42(4), pp.741–843.

Kiryakov, A., Ognyanov, D. and Manov, D. (2005), OWLIM a pragmatic semantic repository for OWL, *in* 'In Proc. of Int. Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2005), LNCS 3807', Springer-Verlag LNCS series, pp. 182–192.

Klusch, M., Fries, B. and Sycara, K., P. (2006). Automated semantic web service discovery with OWLS-MX. In: *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pp. 915–922.

Lagoze, C. and Hunter, J. (2001), The abc ontology and model, *in* 'DCMI '01: Proceedings of the International Conference on Dublin Core and Metadata Applications 2001', National Institute of Informatics, Tokyo, Japan, pp. 160–176.

Lalis, S., Houstis, C., Pitikakis, M., Vasilakis, G. and Vavalis, M. (2005). Providing Support for Integrated Scientific Computing: Metacomputing meets the Semantic Web and the Grid, *5th IEEE International Symposium on Cluster Computing and the Grid* (CCGRID), Cardiff, UK, 9-12 May, pp. 631-637.

Lee, R. (2004), 'Scalability report on triple store applications', Website. http://simile.mit.edu/reports/stores

Leifman, G., Meir, R. and Tal, A. (2005), 'Semantic-oriented 3d shape retrieval using relevance feedback', *The Visual Computer* **21**(8-10), pp. 865–875.

McBride, B. (2002), 'Jena: a semantic web toolkit', *IEEE Internet Comput.* **6**(6), pp. 55–59.

McIlraith, S. A., Son, T. C., and Zeng, H. (2001). Semantic web services. *IEEE Intelligent Systems*, 16(2), pp. 46–53.

McIlraith, S. and San, T. C. (2002). Adapting Golog for composition of semantic web services, In: *Principles of Knowledge Representation and Reasoning*, Morgan Kaufmann, San Francisco, California, pp. 482–493.

Min, P. (2004), A 3D Model Search Engine, PhD thesis, Princeton University.

Noy, N., Fergerson, R. & Musen, M. (2000), The knowledge model of Protege-2000: Combining interoperability and flexibility, *in* 'EKAW '00: Proceedings of the 12th European Workshop on Knowledge Acquisition, Modeling and Management', Springer-Verlag, London, UK, pp. 17–32.

OWL Services Coalition (2003). Semantic Markup for Web Services (OWL-S). http://www.daml.org/services/owl-s/1.1/

Paolucci, M., Wagner, M. and Martin, D. (2007). Grounding OWL-S in SAWSDL. In Bernd J. Krämer, Kwei-Jay Lin, and Priya Narasimhan, editors, ICSOC, volume 4749 of Lecture Notes in Computer Science, Springer, pp. 416–421.

Paolucci, M., Kawamura, T., Payne, T. R. and Sycara, K. P. (2002). Semantic matching of web services capabilities. In: *International Semantic Web Conference*, Springer, LNCS Vol.2342, pp. 333–347.

Papaleo, L., De Floriani, L., Hendler, J. and Hui, A. (2007). Towards a Semantic Web System for Understanding Real World Representations, In: Proceedings of the *10th International Conference on Computer Graphics and Artificial Intelligence*, Athens (GREECE), 30-31 May, 2007.

Papazoglou, M. (2007). *Web Services: Principles and Technology*, Pearson Education Series, ISBN 0321155556.

Papazoglou, M. (2003). Service-Oriented Computing: Concepts, Characteristics and Directions, Proceedings of the Fourth International Conference on Web Information Systems Engineering (WISE), pp.3-12, December 10-12, 2003.

Papazoglou, M., Traverso, P., Dustdar, S. and Leymann, F. (2007). Service-Oriented Computing: State of the Art and Research Challenges, *Computer*, V.40 N.11, pp.38-45.

Pelz, C. (2003). Web services orchestration and choreography. *IEEE Computer*, *36*(8), pp. 46–52.

Pereira, F. (2001), The MPEG-21 Standard: Why an open multimedia framework?, *in* 'Proceedings of the 8th International Workshop on Interactive Distributed Multimedia Systems', pp. 219–220. **URL:** *http://data.semanticweb.org/conference/iswc-aswc/2007/tracks/research/papers/29*

Pitikakis, M. (2010). A semantic based framework for managing, searching and retrieving 3D resources, *Heritage in the Digital Era*, ISBN 978-1-907132-25-4, Multi-Science Publishing, pp. 195-204.

Pitikakis, M., Vavalis M. and Houstis, C. (2010). Semantically Interoperable 3D Scientific Objects, *Knowledge Engineering Review*, accepted (to appear).

Pitikakis, M., Houstis, C., Vasilakis, G. and Vavalis M. (2005). A Knowledge Management Architecture for 3D Shapes and Applications, 10th Panhellenic Conference on Informatics (PCI2005), Volos, Greece, 11-13 November, 2005, Springer, LNCS 3746, pp. 360 – 370.

Prud'hommeaux, E. and Seaborne, a., (2008), 'SPARQL Query Language for RDF', Website. http://www.w3.org/TR/rdf-sparql-query

Racer Systems GmbH & Co. (n.d.), 'Renamed abox and concept expression reasoner RACER', Website. http://www.racer-systems.com

Rao, J. and Su, X. (2004). A Survey of Automated Web Service Composition Methods. *In Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition, SWSWPC 2004*, San Diego, California, USA, July 6th, 2004.

RDFS (2009), 'Resource description framework schema', Website. http://www.w3.org/TR/rdf-schema

Roman, D., Keller, U., Lausen, H., de Bruijn, J., n Lara, R., Stollberg, M, Polleres, A., Feier, C., Bussler, C., and Fensel, D. (2005). Web Service Modeling Ontology. Applied Ontology, 1(1), pp.77 – 106.

Sable C (2003), Robust Statistical Techniques for the Categorization of Images Using Associated Text, Ph.D. thesis, Columbia University.

SALERO (2006), 'Semantic audiovisual entertainment reusable objects, FP6-IST strep', Website. http://www.salero.info

SCHEMA (2002), 'FP6-IST Network of Excellence: Content-based semantic sense analysis and information retrieval', Website. http://www.iti.gr/SCHEMA

SCULPTEUR (2005), 'Semantic and content-based multimedia exploitation for European benefit, FP7-IST STREP', Website. http://www.sculpteurweb.org

Seaborne, A. (2004), 'RDQL - A Query Language for RDF', Website. http://www.w3.org/Submission/2004/SUBM-RDQL-20040109

Sirin, E., Hendler, J. and Parsia, B. (2002). Semi-automatic composition of web services using semantic descriptions. In *Web Services: Modeling, Architecture and Infrastructure workshop in ICEIS 2003*, Angers, France, April 2003, pp. 17-24.

Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A. and Katz Y. (2007). Pellet: A practical OWL-DL reasoned, *Web Semantics*, 5 (2), pp. 51-53.

SOAP (2007). Simple object access protocol (SOAP 1.2), W3C Recommendation. See: http://www.w3.org/TR/SOAP

Steinberg, D., Budinsky, F., Paternostro, M. and Merks, E. (2008), *EMF: Eclipse Modeling Framework*, 2nd edn, Addison-Wesley Professional.

Sure, Y., Staab, S., Studer, R., and Ontoprise Gmbh (2003), 'On-To-Knowledge Methodology (OTKM)', *Handbook on Ontologies*, International Handbooks on Information Systems, Springer, pp. 117-132.

Theodoridou, M., Tzitzikas, Y., Doerr, M., Marketakis, Y. and Melessanakis, V. (2010). Modeling and querying provenance by extending CIDOC CRM, Journal Distributed and Parallel Databases, Springer Netherlands, Vol. 27 pp. 169–210.

Traverso, P., and Pistore, M. (2004). Automated composition of semantic web services into executable processes. In: Sheila A. McIlraith, Dimitris Plexousakis, and Frank van Harmelen, editors, *International Semantic Web Conference*, Springer, LNCS Vol.3298, pp. 380–394.

UDDI Consortium (2001). UDDI executive white paper, November 2001. See: http://www.uddi.org/pubs/UDDI_Executive_White_Paper.pdf

Vasilakis, G., Garcia-Rojas, A., Papaleo, L., Catalano, C. E., Robbiano, F., Spagnuolo, M., Vavalis, M. and Pitikakis, M. (2010). Knowledge-based representation of 3D Shapes, *International Journal of Software Engineering and Knowledge Engineering*, Volume 20, Issue 5 (August 2010), pp. 739-760.

Vasilakis, G., Garcia-Rojas, A., Papaleo, L., Catalano, C. E., Robbiano, F., Spagnuolo, M., Vavalis, M. and Pitikakis, M. (2007). A common ontology for multi-dimensional shapes, In: *1st Workshop on Multimedia Annotation and Retrieval enabled by Shared Ontologies* (MAReSO 2007), Genoa, 5 December, 2007.

Vasilakis, G., Pitikakis, M., Vavalis, M. and Houstis, C. (2005). A semantic based search engine for 3D shapes: design and early prototype implementation, In: *2nd European Workshop on the Integration of Knowledge, Semantics and Digital Media Technologies*, London, UK, pp. 391–397.

Veltkamp, R. C. (2001), Shape matching: Similarity measures and algorithms, *in* 'Shape Modelling International', pp. 188–197.

VICTORY (2006), 'FP6-IST STREP', Website. http://www.victory-eu.org

Vitvar, T., Kopecký, J., Viskova, J. and Fensel, D. (2008). WSMO-lite annotations for web services. In Sean Bechhofer, Manfred Hauswirth, Jörg Hoffmann, and Manolis Koubarakis, editors, ESWC, volume 5021 of Lecture Notes in Computer Science, Springer, pp. 674–689.

W3C RDF Data Access Working Group (2009), 'Joseki: a SPARQL server for Jena', Website. http://www.joseki.org

X3D (2009), 'Extensible 3D', Website. http://www.web3d.org/x3d

Zaha, J.-M., Barros, A., Dumas, M. and ter Hofstede, A. (2006). Let's Dance: A Language for Service Behavior Modeling. OTM Conferences 2006, pp.145-162.