Collaborative software architectures for interactive biomedical applications

Tirado Ramos, A.

Publication date
2007
Document Version
Final published version

Collaborative Software Architectures for Interactive Biomedical Applications

Collaborative Software Architectures for Interactive Biomedical Applications

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. J.W. Zwemmer
ten overstaan van een door het college voor promoties
ingestelde commissie, in het openbaar te verdedigen in
de Aula der Universiteit op 15 juni 2007, te 10:00 uur

door

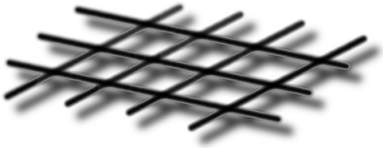
Alfredo Tirado-Ramos

geboren te Mazatlán, Mexico

Promotiecommissie:

Promotor:	prof. dr. P.M.A. Sloot
Co-promotor:	prof. dr. M.T. Bubak
Overige leden:	prof. dr. R.J. Meijer prof. dr. D.A. Kranzlmüller dr. S. Delgado Olabarriaga dr. D.G. van Albada
Faculteit:	Faculteit der Natuurwetenschappen, Wiskunde en Informatica

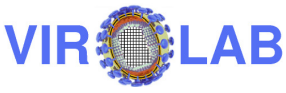
Section Computational Science



Universiteit van Amsterdam

The work described in this thesis has been carried out in the Section Computational Science of the University of Amsterdam, with financial support of:

- The University of Amsterdam,
- The European Union IST *CrossGrid* Project,
- The European Union IST *ViroLab* Project,
- The Netherlands Organization for Scientific Research *VL-e* Project,
- El Consejo Nacional de Ciencia y Technología (CONACYT).

ISBN 978-90-6464-143-5

Typeset with LaTeX 2ε, Printed by Ponsen & Looijen, Wageningen.

Author contact: *alfredo@science.uva.nl*

Numquam ponenda est pluralitas sine necessitate
—William of Ockham, 14th Century

Contents

List of Figures

List of Tables

1

Introduction

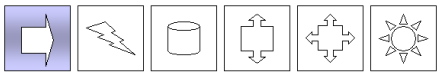
Many areas of biology and medicine are being revolutionized by the introduction of new experimental techniques, accompanied by informatics methodologies that fundamentally change the way that investigators do their work.

—Russ B. Altman and Teri E. Klein
"Challenges for Biomedical Informatics and Pharmacogenomics",
Ann. Rev. of Pharm. and Toxic., 42:113-133, 2002

1.1 Background

The most important goal in Biomedical Informatics is to advance the quality of health care and the breadth and depth of its reach in society. Working towards the purpose of covering the whole healthcare spectrum, from prevention to treatment to rehabilitation, experts in the field have been focusing in the last few years on complex issues such as large-scale data integration and resource interoperability [116,136]. These new foci of research are causing a technological revolution in the field, where unprecedented amounts of biomedical digital information produced by data-intensive applications are rapidly changing the way computer scientists think about and design sofware architectures.

It is widely acknowledged that larger amounts of digital data are being generated by next generation large-scale, collaborative e-Science experiments



[86]. New computational experiments in science and engineering need to cover the whole biomedical spectrum for the simulation of complete biological systems. The gradual introduction of electronic patient records and distributed bioinformatics data warehouses is expected to increase digital image storage and processing dramatically in the short to medium term. For instance, bioinformatics data storage systems used by the Protein Databank [27], Swiss-Prot [31], and the EMBL Nucleotide Sequence Database [146] projects include a variety of increasingly complex numeric, textual and image data, offering gene sequence data handling in the order of Gigabytes (10^9 bytes); this may even expand to the order of Petabytes (10^{15} bytes) when structure measurements of proteins are required.

1.1.1 The Data Digitalization Challenge

Biomedical informatics is steadily evolving into a research field that encompasses the use of all kinds of biomedical information in more integrative ways, from genomics and proteomics to medical data in clinical settings. The main fields comprising biomedical informatics research are *Medical Informatics* and *Bioinformatics*, whose subfields are firmly established in academia and industry (Table 1.1).

Table 1.1: Biomedical Informatics subfields at a glance; large institutional funding bodies such as the European Commission have in recent years started to fund a number of initiatives to explore such potential synergies between the different biomedical informatics subfields, as shown by Martin-Sanchez et al. [115]

Medical Informatics	**Bioinformatics**
Health Information Systems	Genomics
Medical Decision Support Systems	Proteomics
Language and Classification Systems	Structural Bioinformatics
Telemedicine	Sequence Databases
Statistics	Computational Analysis

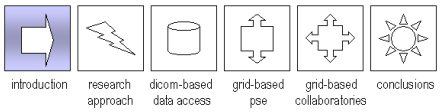
As P. Miller [120] showed during the American College of Medical Informatics Symposium in 2000, recent technological developments offer biomedical informatics researchers easy access to the potential cross-fertilization of bioinformatics with the clinical world. Miller presents the correlations be-

tween individual genetic variation and clinical risk factors, as well as differential response to treatment. R. Altman [8] also presents a clear case for the complementarity of clinical informatics and bioinformatics research, tracing the intersection of the fields back to the Dendral Project [143] in the 1960s. Altman identifies a number of "affinity groups" in biomedical computation, which may be of particular interest to system architects:

- Image acquisition and analysis,
- structural biology and genetics bioinformatics,
- biomechanical modeling for macroscopic systems,
- computer-assisted interventions and robotics,
- data modeling, statistics, and informatics,
- networked and computer-enabled education.

The interest in biomedical informatics has resulted in notable increases in the number of scientific publications on the field in the last few years. For instance, Ammenwerth and de Keizer [10] found that while the number of papers *on medical informatics alone* available in the PubMed online repository in 1983 (including hits between 1982 and 2002) was already around 45,000, the overall number of published medical informatics evaluation studies increased steadily during that period of time (Figure 1.1).

Medical Informatics, often also called Health Informatics, requires data models and software architectures for the creation, maintenance and communication of image-based medical records [28], complex decision support systems [19], and controlled medical vocabularies [54] among others. The first references to organized informatics for medicine go back to the early 1950's with Gustav Wagner and his Deutsche Gesellschaft für Medizinische Dokumentation, Informatik und Statistik [130]. The field, though, is usually considered to be rooted in the early work on foundations of decision making by Von Neumann and Morgenstern [158], and in the seminal work by Ledley et al. [106], on the reasoning foundations of medical diagnosis. It was in France in the 1960's that the term *Informatique Medicale* first appeared in academia. Expert systems eventually started to become a focus of attention, particularly in the United States, with MYCIN [44] a prime example. These developments were followed by the development of tools for the support of text-based clinical records, such as the popular MUMPS programming system by Pappalardo et al. [111]. Eventually, models for image format and communications such as the Digital Imaging and Communications for Medicine (DICOM) model started to offer support for basic image-based connectivity and data transfer.



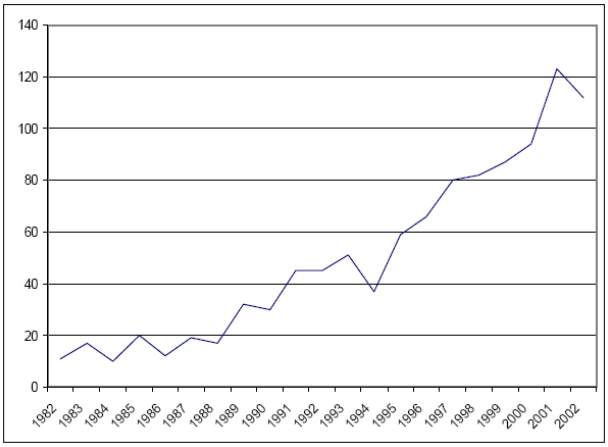
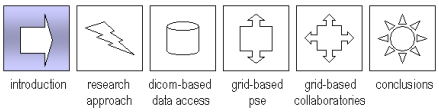


Figure 1.1: The overall number of published evaluation studies on medical informatics for the period 1982 - 2002 in the PubMed system, which shows 10-fold increase and suggests increasing interest in the field, from Ammenwerth E, de Keizer N., "An inventory of evaluation studies of information technology in health care: Trends in evaluation research 1982 - 2002", Methods Inf. Med. 2005; 44:44-56

Bioinformatics usually focuses on the use of techniques from mathematics and computer science to address complex biological problems such as sequence alignment [53, 87], comparative and functional genomics [46], proteomics [129], and prediction of gene expressions [135]. As put forward already in the 1970s by P. Hogeweg [88]: "(Bioinformatics is) The science of information and information flow in biological systems, especially of the use of computational methods in genetics and genomics." The field can be considered to have started with the 1977 DNA sequencing of Φ -X174 by Sanger et al. [133], and took noticeable impulse with the eventual development of alignment search tools such as BLAST by Altschul et al. [9], and the first full-fledged genome annotation software system, TIGR Assembler, by White et al. [148]. The recent completion of the Human Genome Project has provided an explosive boost to the field.

Physicians, technologists and scientists require easier access to incrementally larger and more complex resources, as well as methods for the efficient extraction of biomedical knowledge from available data, leading us to the following observation:



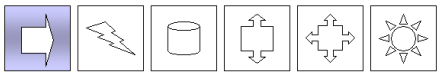
Due to digitalization, we now face a biomedical data deluge: large amounts of medical and genomic digital data have been produced in recent years by applications using data-driven information management systems, data integration and distribution models.

1.1.2 The Changing Nature of Biomedical Informatics Data

The nature of data generated by biomedical applications is changing rapidly. For instance, Critchlow et al. [56], from the Lawrence Livermore National Laboratory discuss how new knowledge acquisition in the field is being driven by a move to a more complex discovery-verification paradigm, where informatics support is increasingly crucial to handle new correlations between biological components and their digitized data.

Most medical imaging modalities nowadays produce digital data [1], making the issues related to large storage capacity more complex than ever. There is a growing range of applications producing multi-dimensional digital images available to the clinician at present, such as X-ray, ultrasound, Magnetic Resonance Imaging (MRI) and Computer Tomography (CT), among others. Some of the latest technology on CT scanning modalities allows nowadays the creation of three-dimensional (3D) multislice patient data, like in the case of Philips' Mx8000 Multislice scanner which provides a quad-multislice system with maximum rotation time of 0.42 seconds per turn.

More than textual patient data, medical images represent the major amount of digital data collected for medical purposes. However, medical images are not sufficient by themselves, as they may need to be interpreted and analysed in the context of the patient's medical record. Patient management (diagnosis, treatment, continuing care, post-treatment assessment) is rarely straightforward. There are a number of factors that can make patient management based on medical images particularly difficult: medical data may be distributed over a number of acquisition sites, with data concerning one patient not necessarily located in a single location nor accessible through a unified interface [2]. Physicians most often have no simple way to access the medical records of all their patients, and the patients' digital images they require are often part of very large data files with complex data and structure that may include annotations, regions of interest, and other types of clinically significant metadata. In many cases, no single imaging modality suffices, since there are many parameters that affect the appearance of an image, and their complementary information is captured by different physical acquisition systems [114]. Such systems increasingly communicate via a local network with what is commonly



called Picture Archiving and Communications Systems (PACS), sort of standardized image data repositories within hospitals or across distributed sites.

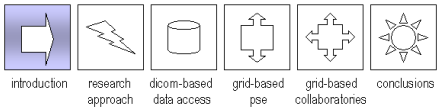
There are valid reasons for the distributed nature of biomedical data. For instance, medical image data are used in diagnosis, continuing care, and therapy planning. For diagnosis, medical images acquired in a medical centre are sometimes visualised and interpreted immediately after acquisition by a technologist before being sent to a physician for second viewing [59, 149]. These two readings normally take place in different offices, and possibly even in different sites. For therapy follow-up, even more clinicians may be involved as images acquired at different times may be acquired in different radiology centres and several physicians may need to read them. For therapy planning and assisted intervention, images may also need to be accessible from the intervention room [50, 157]. Increasing complexity in data leads to the following observation:

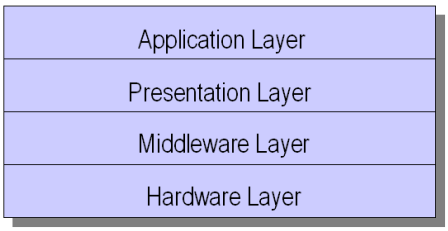
The nature of biomedical data is evolving: we now have to process increasingly distributed, heterogeneous, and complex kinds of data.

1.1.3 From Image Transfer to Interactive Collaboratories

Computational science technologies and architectures are evolving to address ever more complex demands from biomedical scientists. Software design is currently receiving increased attention by biomedical researchers [20, 126], suggesting the emergence of software architecture design as a major discipline in biomedical informatics [155, 160]. This takes the design problem, as Garlan et al. put it, "beyond the algorithms and data structures of the computation: designing and specifying the overall system structure emerges as a new kind of problem" [75].

The design of interactive collaboration within complex architectures aims to enhance the productivity and effectiveness of multidisciplinary biomedical applications. These applications result from the convergence of different approaches to architectural design, such as relational, object-oriented, component, and client/server technologies. These technologies allow computer scientists to spread software components across organizations via computer networks that communicate as a unified whole, and modeled in different granularities and levels of formalization (Figure 1.2). Computer systems have gone through an impressive progression in the last 30 years. This transition, ranging from computer mainframes to symmetrical multi-processing (SMP) and





(a) layered representation

```
System client_server = {
  Component client = {Port sendRequest}
  Component server = {Port receiveRequest}
  Connector rpc = {Roles {sender, receiver}
}
Attachments : {
    client.sendRequest to rpc.sender;
    server.receiveRequest to rpc.receiver
    }
}
```

(b) description language

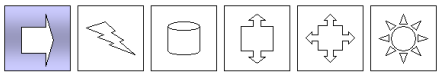
Figure 1.2: Generic software architectural representations; clockwise: (a) a layered middleware-centric model of a multi-tier software architecture, and (b) description of a generic Client/Server software architecture in the ACME description language [74]

non-uniform memory architecture (NUMA) machines to high-performance clusters, is the product of significant advances in hardware technology.

Hardware advances have also extended to developments in biomedical modality technologies [5, 66], computer based drug design [100, 162], digital storage capacity and mining [62], as well as medical imaging and simulations [60]. Computation intensive applications that analyze 3D and 4D digital images or simulate medical surgery scenarios are certainly good news for biomedical scientists and clinicians, but pose great demands on system design. System architects now have to design systems with increasing high demand for large computational power and access to highly distributed and heterogeneous data sources.

In Medical Informatics, new approaches are being researched to improve interoperability and integration of distributed applications with legacy systems, such as Steward's work on architectural approaches for medical data integration [145]. Bioinformatics issues have historically been related to improving collaboration frameworks for bioscientists and computational scientists [48, 96], as well as access to distributed databases consisting of heterogeneous types of growing amounts of data. For instance, Grethe et al. [78] describe the collection of large amounts of heterogeneous data as the first step in the biomedical experimental process; sharing and processing such data, as they show, requires non-trivial developments on the architectures that support the scientific process.

Access to such sheer amounts of biomedical data produced every day at clinical settings, together with the long-term archiving of data for pathology and epidemiology studies [63], represent big challenges for software archi-



tects. Distributed technologies, standards and models are driving the development of new approaches to software architectures in order to address these issues. One of the best examples of models for representation and transfer of digital biomedical image data is the Digital Imaging and Communications for Medicine (DICOM) model, the current default standard for biomedical image representation and access within hospitals worldwide [101]. The American College of Radiology (ACR) and the National Electrical Manufacturers Association (NEMA) created a joint committee in 1983 to address the problem of developing a standard model for communication between medical imaging equipment, its associated information and user applications [118]. The DICOM model was developed by this effort to meet the needs of manufacturers and users of medical imaging equipment on standard networks. It also provided a means by which users of imaging equipment may determine whether two devices are able to exchange information, facilitate communication in networked environments, and connection of PACS to specialized information systems such as Hospital Information Systems (HIS) and Radiology Information Systems (RIS). Initially known as ACR-NEMA in its versions 1.0 and 2.0, it became DICOM in its version 3.0, released in 1991. This version also offered an Object-Based design and support for the Transmission Control Protocol/Internet Protocol (TCP/IP) [144] and Open Systems Interconnection (OSI) [58] protocol stacks, describing an image format, a communications protocol between an image server and its clients, and other data access capabilities (Figure 1.3).

DICOM's model is gradually being made interoperable within related data models such as the Health Level Seven (HL7) [21] clinical and administrative standard for healthcare services [49]. HL7's Reference Information Model (RIM) and object-oriented development methodology provide an explicit representation of the semantic and lexical connections for information to be carried across information systems. DICOM was a milestone in the state of the art for Medical Informatics modeling and communications. Nevertheless, a number of important issues hamper its scalability within large distributed organizations that are emerging in biomedical informatics research. For instance, the first step in sending a DICOM image is to establish data connections between the two machines for the negotiation of the details of the image transmission, with security mechanisms between communication networks configured to allow passage of information between the sites. This naturally does not scale well. Also, since DICOM is specified in terms of individual components, when two systems support sending DICOM images, but neither knows how to receive and store them, then these devices will have trouble interconnecting, let

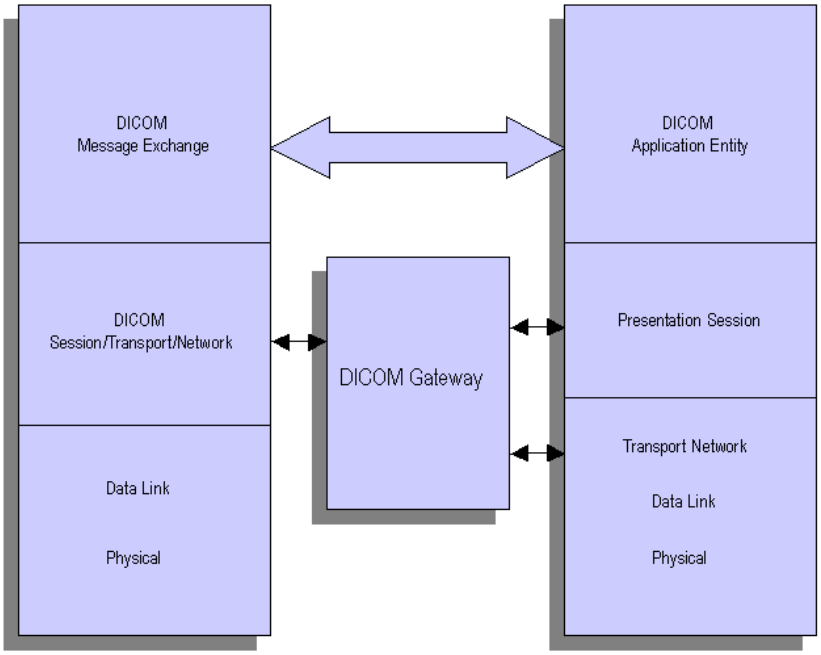


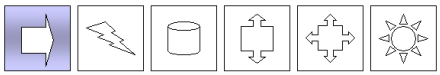
Figure 1.3: Representation of the DICOM communications model, where a subset of the OSI services working in conjunction with IP transport protocols for connection establishment and message transfering via gateways

alone interoperating.

There has been a number of efforts to extend the original DICOM model from its relational and static architecture into more dynamic and scalable distributed component architectures [13, 67], aimed to improve the prospects for distributed transparency and scalability. Nevertheless, new distributed architectures that build on the state of the art while addressing the complex nature of highly distributed biomedical applications securely and transparently are clearly needed.

Advances in Bioinformatics

Bioinformatics applications continuously increase in size, complexity and heterogeneity [16] as well. As put forward by T. K. Attwood [14], bioinformatics



will need interoperable applications that allow users the access to disparate data sources in order to enable knowledge-based inference and innovation. In Bioinformatics, main architectural issues include data acquisition, representation, archival, retrieval, compatibility, provenance, and consistency. A number of complex data-centric tools based on distributed architectures are being developed by researchers, though not at a comparable rate as gene and protein data are made available by the research groups all over the world. Like in medical informatics, there is a strong need for architectures that facilitate seamless integration and interoperability of new bioinformatics systems and applications with legacy systems and databases. There is already an important body of work in, e.g., distributed database query systems [57], standards for data representation and exchange [137], and even web-based architectures [73]. Such systems are designed to provide clusters that are tightly integrated with data servers, providing high volume processing. Nevertheles, these kinds of approaches to large scale integration and distributed data access have been found to be limited in terms of query power [98], and normally require a centralized design that does not scale well with distributed systems.

Interactive Collaboratories

Recent cutting-edge research efforts in the field are exploring the use of highly distributed software architectures for the setup of dynamic, loosely-coupled data storage and computational resources, as in Cannataro's Proteus [47], or Sloot's ViroLab [142]. ViroLab is a good example of such systems, where a decision support system for the interpretation of genotypic resistance is being integrated within a distributed virtual laboratory. The aim there is to achieve seamless access to large amounts of experimental hospital data from databases and archives within a large, distributed and dynamic virtual infrastructure, allowing hierarchical data flow models (Figure 1.4) to be seamlessly integrated.

This kind of applications assist clinicians and researchers in choosing effective therapeutic alternatives, using large amounts of highly distributed and heterogeneous experimental hospital data from databases and archives [138]. Efforts like these explore the boundaries in new kinds of software architectures that leverage, instead of ignore, the dynamic characteristics of highly distributed virtual organizations. These issues become particularly relevant when they involve the integration of resources from *highly* distributed infrastructures, such as the ones formed by large, loosely-coupled organizations that offer unified access to distributed computational services and resources. Computationally intensive and distributed Problem Solving Environments (PSEs)

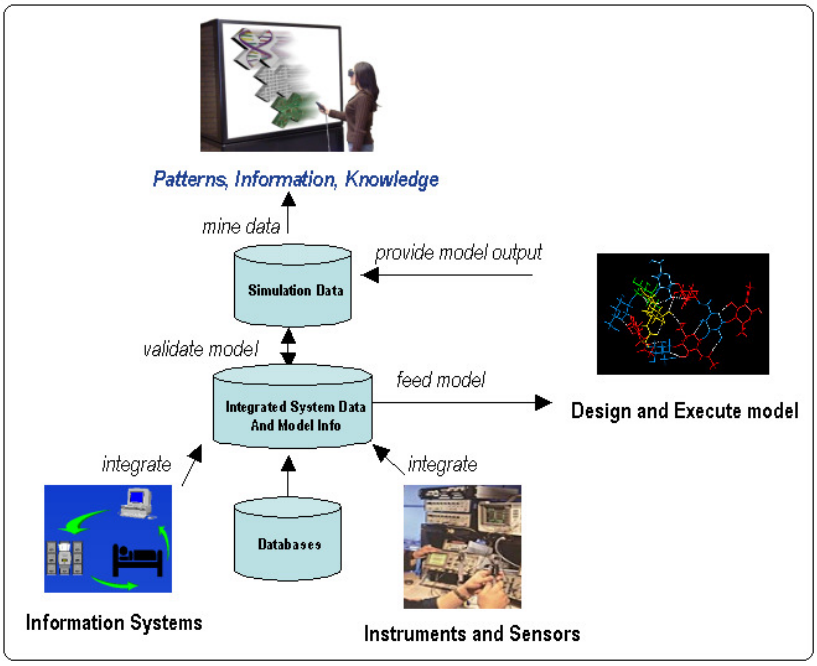
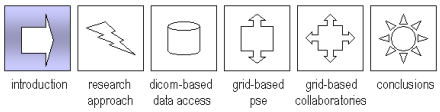


Figure 1.4: General architecture for conducting e-Science research: information systems integrate available data with data from specialized instruments and sensors into distributed repositories. Computational models are then executed using the integrated data, providing large quantities of model output data, which is mined and processed in order to extract useful knowledge [142]

are a good example of such systems.

As argued by Houstis et al., *Problem Solving Environment* means different things to different people [90], but can be succinctly described as "a computer system that provides all necessary computational facilities to solve a target class of problems". Houstis suggests that such facilities should be accessible without the need for specialized knowledge of the underlying computer hardware or software system. One of the more interesting recent approaches to address these issues, from the distributed computing viewpoint, is that of *Grid computing*, as defined by Foster, Kesselman, et al. in their seminal works on the basis of Grid architectures [71] and a service-based approach to Grids [69]. Grid computing offer PSEs the concept of dynamic *virtual organizations*, where members may pool their their resources transparently and make them avail-



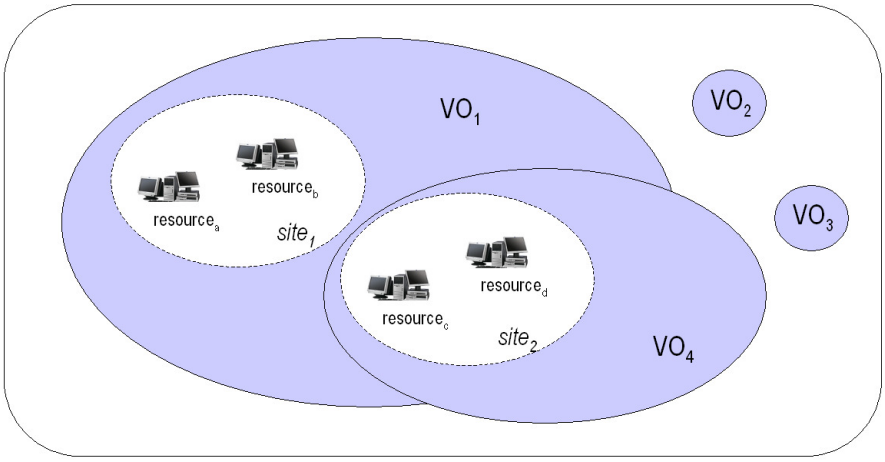


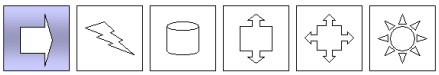
Figure 1.5: Graphical representation of a multi-VO Grid infrastructure, where different, dynamic topologies allow members of the Grid VO to share their resources securely and transparently.

able to a trusted virtual community.

Grid Computing

Grid Computing [68] is a distributed computing paradigm that offers system architects the possibility to design architectures that provide secure and seamless access to multiple distributed data and computational resources. This is achieved by allowing users to share such resources within (possibly multiple) Grid Virtual Organizations (VOs) (Figure 1.5), supported by middleware based on open standards.

The vision of Grid Computing, or utility computing, as it is also known, is based on the access to computing and data resources from a virtual infrastructure that mimics the electrical industry's power Grid. That is, users should not have to own or care where available computing resources are, as long as there is a reliable and secure way to access them. In comparison, tightly-coupled systems based on relational and classic object-oriented models impose overhead in order to enable communication, and require higher levels of understanding between systems. Loosely-coupled systems like Grids allow distributed systems to be able to create composite services and to disassemble easily into their functional components.

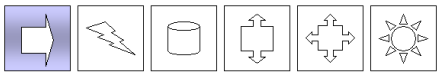


Grids are often classified as *Computational Grids* or *Data Grids*. Computational Grids aggregate the power of individual computers into the computational capabilities of virtual supercomputers. Data Grids, on the other hand, aim to create aggregated virtual repositories that give users access vast amounts of data and storage capacity previously unavailable, using transparent higher level services such as replica location services. Nevertheless, as shown by Nemeth et al. [123], fundamental differences between Grids and more traditional kinds of distributed systems (Table 1.2) provide a strong rationale for rethinking current approaches to distributing computing for complex problem-solving.

Table 1.2: A comparison between conventional distributed environments and Grids, from Z. Nemeth and V. Sunderam, "Characterizing Grids: Attributes, Definitions, and Formalisms", *Journal of Grid Computing*, 1: 9-23, 2003

Conventional Distributed Environments	**Grids**
A virtual pool of computational nodes	A virtual pool of resources
Access to all the nodes in the pool	Access to the pool but not to individual nodes
Access to a node means access to all resources in the node	Access to a resource may be restricted
The user is aware of the capabilities and features of the nodes	The user has little or no knowledge about individual resources
The nodes belong to a single trust domain	Resources span multiple trust domains
The elements in the pool are 10-100, more or less static	Elements in the pool »100, dynamic

The Grid was originally proposed as a layered architecture providing a lower layer called the *fabric* that comprises the hardware and interfaces to local control of computational, storage, and network resources. On top of the fabric, a *connectivity* layer contains the core communication and authentication protocols for network transactions. Next, the *resource* layer provides the information and management protocols required for the secure negotiation, initiation, monitoring, control, and accounting of Grid resources. On top of this, the *collective* layer contains the directory, co-allocation, scheduling, brokering, monitoring, diagnostics, data replication, workload, discovery and



other global services that are not associated with specific resources. Finally, the top layer of the model is reserved for the *application* layer, which contains the user applications that run on the Grid infrastructure. Such infrastructure usually provides secure access to resources via a standard security mechanism, such as Public Key Infrastructure (PKI) [150], that *authenticates* members of a VO. Authorization is delegated to local resource owner policy [70]. A set of base middleware components [68] takes care of data management, execution management, information services, and security services, while higher level services such as resource brokers [89] may be deployed to hide the lower level infrastructure from the user.

This evolution from simple data management all the way to interactive collaboratories can be based on loosely-coupled and potentially concurrent approaches such as Grid computing, which provides new toolboxes for building distributed system architectures. System-level integration of complex simulation and other services into widely distributed scientific collaboration is now a real possibility. This is a fundamental change in the way system architects and biomedical scientists may approach new challenges in the field, and leads us to the final observation in this chapter:

> *The changing nature of biomedical data requires a paradigm change: new technological advances in simulation, visualization, and collaboration as supporteded by Grids can be required for producing accurate and meaningful biomedical information.*

1.2 Objectives of this Work

Connectivity between distant locations, interoperability between diverse systems and resources, and high levels of computational performance are some of the most promising features of the Grid. In the case of biomedical applications, issues such as remote access to patient data, medical knowledge bases, advanced visualization technologies and specialized medical instruments are of the most importance. For these applications, Grid technology provides dedicated support such as security, distributed storage capacity, and high throughput over long distance networks. Besides these immediate benefits, the computational resources of the Grid provide the required performance for large scale simulations, complex visualization and collaborative environments, which are expected to become of major importance to many areas of medicine.

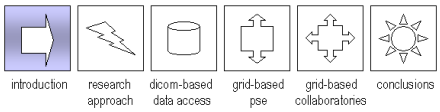
In this work we specifically investigate issues related to dynamic sharing of digital resources such as computational power, data, applications and instruments for supporting the acquisition of biomedical information. *Computer science in general, and the Grid computing paradigm in particular, provide the logic, the language and the tools needed to study and understand modern complex biomedical systems*. In this chapter, we identified a number of observations about the current state of the art in technology for the support of biomedical informatics research.

Complex interactivity using distributed, multi-dimensional biomedical resources clearly requires new approaches that identify environments, architectural constructs and tools for reasoning about the system. This need for new approaches lead us to the research questions to be addressed in this work:

- ***Is there a distributed environment for collaborative biomedical informatics in the state of the art in data communications and distributed technology?***
- ***What new modeling abstractions, methods and tools can we use to reason about such kind of environment's architecture?***
- ***Can we identify temporally invariant components and aspects of data interaction which are present in biomedical informatics applications?***

These questions provide the basis for determining the scientific aims of this thesis, by logically extending them in the form of specific objectives. These objectives are defined as follows:

- Objective 1: *To design and prototype software architecture models that support distributed virtual laboratories for interactive collaboration (i.e., collaboratories), building on the state of the art, and reason about them, using runtime components that enable the virtualization of distributed resources within dynamic trust domains.*
- Objective 2: *To identify software components for biomedical informatics that may remain invariant against next generations of technology, and once they are identified, map them to distributed systems.*



We take a progressive approach towards these objectives, from a system architecture viewpoint, analyzing and validating a number of incrementally complex distributed system architectures for biomedical informatics applications.

1.3 Thesis Roadmap

This work is divided up into six chapters, of which *chapter one* is the present introduction. In *chapter two* we describe our approach to the methodologies used in this work for the analysis of information models and architectures, and the rationale behind our approach. We describe a formal approach, based on an actor model of computation, as well as a feasibility approach based on prototyping, laying the baseline we use to address the stated *objectives 1* and *2*. *Chapter three* focuses on case studies in which we describe the DICOM model and elaborate on some of its most notable shortcomings, and extend its functionality using object-oriented and component models for interoperability of data access, therefore beginning our progressively more complex prototyping and analysis required to address *objectives 2* and *3*. *Chapter four* presents our experiments and in-depth analysis of a complex biomedical problem solving environment within an interactive Grid infrastructure, where we use Grids for addressing our requirements for interactive biomedical applications in highly distributed environments. We use application benchmarking and online monitoring to characterize the prototype, addressing *objectives 2* and *3*. In *chapter five* we conclude our progressive analysis with a distributed bioinformatics support-decision system in which collaborative discourse among biomedical researchers is a main area of concern. We finalize in *chapter six*, where we elaborate on our results and present our conclusions and elaborate on what we consider potentially interesting future work in the field.

2

Research Approach

More generally, the π calculus is a formal calculus, while the Actors model, in spirit closer to the approach of physics, sets out to identify the laws which govern the primitive concepts of interaction

—Robin Milner
Turing Award lecture, 1993

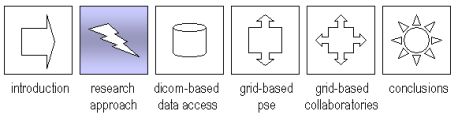
2.1 Introduction

In this chapter we lay out our roadmap or approach to the analysis of distributed software architectures for biomedicine. We first describe some of the most popular and potentially useful methods for architectural representation of distributed and concurrent systems, and then elaborate on the rationale for the approach chosen in this thesis. This approach is focused on the use of feasibility studies coupled with a formal abstraction for architectural analysis. The aim here is twofold: to find a methodology that eases the analysis of biomedical systems based on distributed computing models ranging from relational, to object-oriented and component-based, to service-oriented computing, and use such methodology to reason about a number of application architectures in a series of increasingly complex use cases.

2.2 Approach

In-depth analysis of real architectural components and the distributed environment that bridges them into large scale, loosely-coupled and concurrent systems systems is, naturally, not a trivial task. In this thesis our approach is therefore to start with modeling, prototyping, and analyzing simple software architectures used for supporting distributed biomedical applications, and then move on to more complicated systems. Scalability and seamless resource sharing are at the heart of our approach to the design and analysis of interactive architectures. In the present work we focus on the two simplest kinds of interactivity which allow user analysis and response generation: level 1 linear sequencing or one way interactivity where users can monitor execution and see the output of the application online, once the application finishes execution; and level 2 linear sequencing with feedback or two way interactivity, where users can receive output from the running application online, synchronously while the application continues execution, and react to it. We do not address the third level of interactivity, where real-time steering of the running application takes place. As case studies for validation of our approach, we work with increasingly complex applications, as depicted in Figure 2.1.

Distributed computing is a rapidly changing, evolving field. It was not long ago that local and wide area network capacity hindered the efficient use of remote resources for computationally expensive applications. In this thesis we start our analysis with DICOM-based simple services for sharing digital medical image content, first in the context of enterprise data sharing interoperability and then in the context of a seamless data transfer service within teleradiology applications. We next model and implement a more complex, computationally intensive biomedical application for blood flow simulation [11, 12] which is part of a problem solving environment developed within the University of Amsterdam's Polder Project [94]. This biomedical environment, the Virtual Radiology Explorer (VRE) [25], provides important challenges for data access, image processing, simulation, and visualization that are computationally intensive, using parallel implementation technologies in order to get executed in a reasonable amount of time. For its implementation and validation we use middleware and higher level components offered by the European CrossGrid Project [43] infrastructure. Finally, we analyze a collaborative bioinformatics application, the ViroLab [142], a virtual laboratory for decision support in viral diseases treatment. This multi-scale collaboratory aims to facilitate medical knowledge discovery and decision support for, e.g., HIV



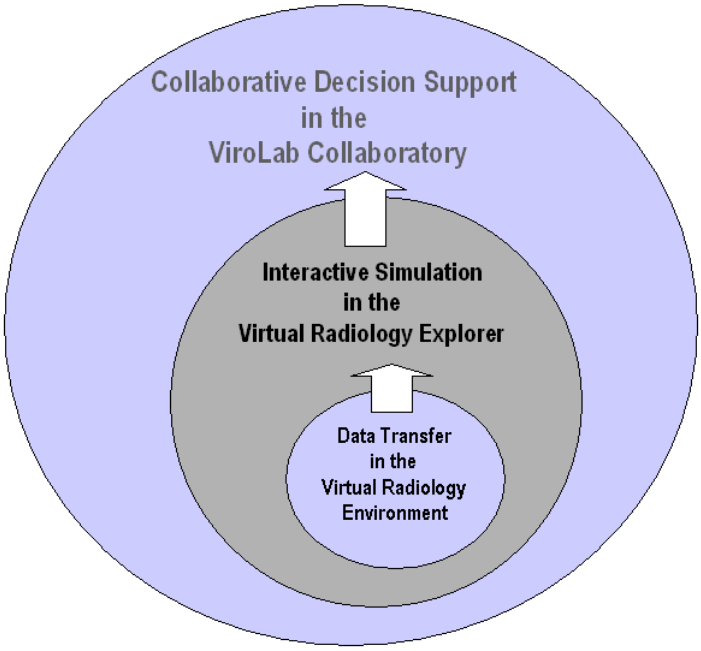


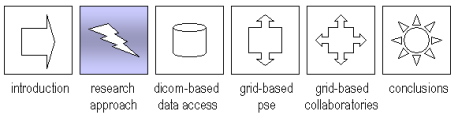
Figure 2.1: High-level representation of our approach to incrementally complex distributed system analysis: from a distributed application for digital image access, to an interactive simulation-centric problem solving environment, to a virtual collaborative laboratory on the Grid

drug resistance, using a Grid-based service oriented architecture to vertically integrate the biomedical information from viruses (proteins and mutations), patients (viral load) and literature (drug resistance experiments).

2.3 Methodology

We start our analysis of distributed biomedical applications by researching increasingly complex case studies, using the technique of feasibility study via prototyping. We expect this approach to answer basic questions on whether the technology needed for the system exists and how difficult they are to build.

There is a great deal of work in the literature related to software architec-



ture modeling languages. The Unified Modeling Language (UML) [35], for instance, provides standard, widely adopted high-level descriptional semantics. At lower levels of architectural description we find more formal description approaches such as the popular Petri Nets [127] method, and the classical π and λ process calculi. Nevertheless, while high-level formalisms like UML present a very clear view of the *macroscopic* properties of a system, low-level ones like Petri Nets and process calculi normally represent a very detailed *microscopic* representation of system components' control flow. Recent trends in distributed computing, such as the use of dynamic virtual organizations that offer users the access to large numbers of resources, possibly concurrently, may be better suited for *mesoscopic* architectural models that support distributed and potentially concurrent computation.

We complement our prototyping with one of such mesoscopic models of system representation, the *actor model* of concurrent computation. In this chapter we briefly present some of the techniques available in the state of the art, and offer the rationale for our chosen actor-oriented approach.

2.3.1 Prototyping

In software enginering, a feasibility study is an analysis of possible architectural solutions to a system's design, in order to arrive to the best possible fit. During this process, architects design and evaluate how an architecture will fit into the general approach, or whether new requirements can be met by a new approach that may be more efficient than the one currently used in the state of the art. A feasibility study can be used to build and analyze a new system, shedding very important light on whether the software architecture suits its purpose, technological advancement may render current approaches redundant, the technology base allows to cope with higher work loads or new types of process flows, users are satisfied with the quality of experience while working with the system, and other issues.

Feasibility studies can be carried out by *prototyping*. This is the process of putting together a basic working model, or prototype, in order to evaluate various aspects of the system's architecture and performance, and get early user feedback. Prototyping can be considered as an integral part of the system design process. In this thesis, we work with prototypes of complex biomedical informatics systems in a development stage, focusing on a relevant subset of the intended systems. We assume that the prototypes are intended for experimental purposes, in order to provide us with a working model for analysis using more formal models of architectural analysis.

2.3.2 Modeling Distributed Concurrent Systems

In distributed infrastructures for interactive biomedical applications the initial descriptions of the architectural vision for communication with the stakeholders are usually informal, and even ad-hoc. The common approach is to use diagrammatic constructs such as layered and flow diagrams, offering poor semantics as basis for later logical architectural designs like class diagrams. The situation is aggravated when applied to systems that are distributed and show concurrent behaviour. Collaborative systems may be necessarily loosely-coupled in order to support virtual organizations which dynamically share complex applications and components. For distributed applications to fully leverage current technologies, system architects have to address the integration of highly distributed and potentially heterogeneous data and computational resources. Also, distributed collaborative systems, just as standard local and non-local concurrent systems, rely on the use of shared resources and therefore require some form of concurrent access to such resources.

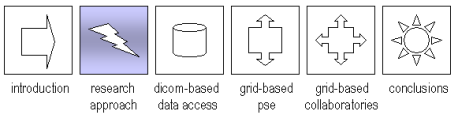
2.4 Existing Modeling Formalisms

We next briefly describe some formalisms that may be used to describe the prototyped distributed architectures in our use cases, or parts of them, particularly when concurrency and resource distribution are relevant.

2.4.1 Macro-level: The Unified Modeling Language

The Unified Modeling Language (UML) [35] is a way of specifying, visualizing, constructing, and documenting components of software systems. The conventional UML notation uses the basic principles of object-orientation to model system structure and behavior, defining components with object analysis and design concepts such as objects, classes, stereotypes, and relationships (Figure 2.2).

UML is not a method, though it was designed to be compatible with object-oriented software development methods such as OMT [15] and Booch [33]. The UML model provides a set of diagrams, which are partial graphical representations of a system's architectural design, used within standardized models such as:



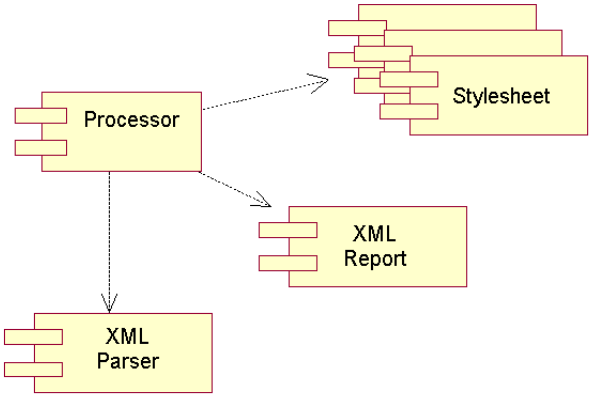


Figure 2.2: A simple UML component diagram, showing XML transformation components that are linked by "uses" relationships

- A functional model, which contains the functionality of the system from the user's viewpoint (Use Case diagrams),
- an object model, showing the structure and substructure of the system using objects, attributes, operations, and associations (Class diagrams),
- a dynamic model, containing the internal behavior of the system (Sequence diagrams, Activity diagrams, State Machine diagrams).

UML is widely recognized and used, though it is frequently criticized for a number of well-known deficiencies. For instance, UML is commonly considered as being too large and complex, using many diagrams and constructs that are redundant or infrequently used. Also, it is considered to have imprecise semantics, specified by a combination of an abstract syntax, some well-defined formedness rules, and basic English language, lacking the rigor of a language precisely defined using formal techniques. Finally, UML's value in approaches that compile the models to generate source or executable code may not be relevant since UML, as a language, does not exhibit Turing completeness and any generated source or executable code would be limited to what a UML interpreting tool can discern or assume.

2.4.2 Micro-level: Petri Nets and Process Calculi

Petri Nets (PN) is a graphical and mathematical language for modeling complex systems [127]. PN is a generalization of automata theory in which the concept of concurrency can be easily expressed. PN consist of Places and Transitions, as well Input and Output Arcs. Arcs connect Places with Transitions; Input Arcs go from a Place to a Transition while Output Arcs go from a Transition to a Place. Places may contain tokens to simulate the dynamic activities of the system; the number and type of tokens in each place define the current state of the system. Transitions are the active components that represent the activities that may change the state of the system; a Transition may only occur when all preconditions for the activity are fulfilled. When a Transition is fired, a number of tokens are removed from its Input Places and added to its Output Places; new tokens may be placed in the Input Places. The number of tokens removed depends on the cardinality of each arc. The interactive firing of Transitions in subsequent markings is known as Token Game (Figure 2.3).

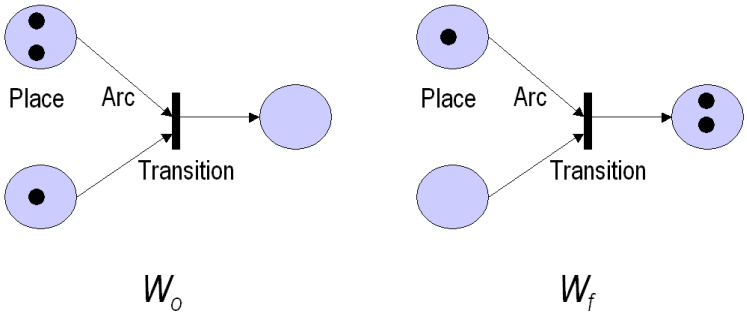


Figure 2.3: A simple Petri Net, showing an initial state or marking W_o and a final state W_f after one firing between Places and Transitions

The concept of timing may be included in PNs. Time may be associated with Tokens, Transitions, and Places. Transition-timed PN may be achieved by associating delays in the time between enabling and firing, or the time associated to the firing itself. In Place-timed PN, when a token arrives to a Place it can enable a transition only after a waiting time in the Place has elapsed. Transition and Place timing PN may be considered equivalent, since one model can be translated into the other; Transition timed nets may be considered the most commonly used. PN modeling including timing may be accomplished by as-

sociating firing delays with each transition, specifying the time that the transition has to be enabled before it may fire. If the delay is a random distribution function, such as in Poisson processes, the model is called Stochastic-PN and is analyzed with Markov processes techniques. If the delays are distributed in a deterministic fashion, the model is called a Timed-PN. Execution in Petri nets is nondeterministic, so multiple transitions can be rendered possible at the same time (any one of which can fire), and none are required to fire (they fire at will, between time 0 and infinity, or not at all). Since firing is nondeterministic, Petri nets are well suited for modeling the concurrent behavior of distributed systems.

A Petri net is a set:

$$\mathcal{P} = \langle S, T, F, N, M, W \rangle \tag{2.1}$$

... where:
S, is a set of places
T, is a set of transitions
F, is a set of arcs
N, is a set of tokens

The set F is subject to the constraint that no arc may connect two places or two transitions, $F \subseteq (S \times T) \cup (T \times S)$

$M : S \rightarrow N$ is an initial marking, where for each place $s \in S$, there are $n \in N$ tokens.

$W : F \rightarrow N^+$ is a set of arc weights, which assigns to each arc $f \in F$ some $n \in N^+$ denoting how many tokens are consumed from a place by a transition, or how many tokens are produced by a transition and put into each place.

Other models of concurrency, originally designed for message-passing that feature hierarchical composition (e.g. the Actor model) build on the rich Petri Net literature and experience. Nevertheless, there is an ongoing argument about the complexity and lack of compositionality in the model (the ability to compose hierarchically contained submodels), which can be a serious limitation of Petri nets because this deficiency limits modularity. In addition, Petri nets lack the notion of locality because input tokens of a transition disappear simultaneously, which limits the realism of the model.

On a more formal note, *process calculi* such as λ or π calculus offer detailed sets of mathematical formalisms for describing and analyzing properties of concurrent computational systems. The π calculus [121], for instance, was developed by Robin Milner specifically as a language for concurrent computational processes. It is based on a model for the representation and simulation of communication systems based on concurrent processing. Process calculi offer a formal, fine-grained model of computation that is well suited for the description of systems at the microscopic level of interaction.

2.5 Our Methodology Rationale

In this thesis we argue that *even though there have been important steps forward in the field of architectural design and analysis, the new conditions and challenges created by system-level e-Science require new approaches* that build on the state of the art while addressing current shortcomings.

Towards this end, the methodology used is to start by prototyping a number of software architectures for biomedical systems intended for experimental purposes, which provide us with fully working models. We then analyze the increasingly complex system architectures using a formal model of architectural analysis, in order *to reason* about them and understand, e.g., component concurrency and architectural invariability. The rationale for using such model, as elaborated next, is based on the fact that we identify an important gap in architectural design, where methods that provide a mesoscopic viewpoint to system analysis have not been throughly investigated in the literature of current state of the art.

2.5.1 Addressing The Mesoscopic-level Gap

In the previous section we described a couple of common methods of representation when it comes to distributed systems in biomedical informatics: the Unified Modeling Language and Petri Nets. We discussed how the UML offers an approach to represent macroscopic viewpoints of system design via a number of diagrammatic constructs, while also being able to delve into component details to a certain extent via class diagrams and other programmatic devices. UML offers good compositionality characteristics, as well as a practical (though informal) semantics based on object-oriented abstractions. We also discussed the widely used Petri Net model, an excellent approach to graphical modelling of microscopic viewpoints of discrete distributed system via, e.g.,

its strong support for potentially concurrent interaction among components, since it was originally intended for message-pasing communication systems. Petri Nets' popularity is also grounded in the model's formal semantics.

Nevertheless, one of the pressing problems in the field of biomedical informatics modeling and simulation is precisely the lack of integrative, mesoscopic level approaches to system design and analysis. Spatial, temporal and functional scales of new biomedical simulation systems require complex integration of data and hierarchically composed subsystem components, in order to study system-level problems related to disease. That is, with new complex system-level initiatives to e-Science (e.g., the Physiome Project [91, 92]), biomedical informatics research spans now across the spectrum, from molecules and genome to organs, and integrative to whole humans. This suggests the eventual integration of large quantities of distributed data, components and other resources, being accessed in a potentially concurrent way, in scales that researchers could not even consider until recently.

We identify a gap at the level of representation and analysis at the mesoscopic level of architectural design and analysis. New systems should be able to represent complex hierarchies of their own subsystems, with componets that may interact concurrently. These kind of characteristics are not easily represented in the current state of the art. For instance, while UML offers good macroscopic compositionality, its informal semantics lack concurrency support, rendering it not fully endowed for system-level analysis. Petri Nets offer a better tool for modeling more microscopic characteristics of system design, though the model's *token game* lacks compositionality and locality, hindering realistic system representation of nonsequential processes. Table 2.1 shows a high-level comparison between the approaches discussed before.

In this thesis we propose an actor model as a *mesoscopic* approach to system level design, based on the classical actor model of computation for potentially concurrent message-passing systems, which we next describe in detail.

2.5.2 The Actor Model

The actor model is a model of computation that resulted from the work of Hewit et al. [84, 85], where the universal primitives of computation are *actors*, which are message-passing entities. The model has been extensively used as a framework and unit of abstraction for the theoretical understanding of classic computation systems, where all entities are considered actors, as opposed to, e.g., object-oriented models where all entities are considered objects. Another key difference is that object oriented models are designed to be ex-

Table 2.1: Approach-level comparison and mesoscopic gap: a mesoscopic level of analysis, the Actor model, filling and complementing the identified gap between the macroscopic and microscopic levels of system architecture representation provided by UML and Petri Nets

UML	**Actor Model**	**Petri Nets**
Macroscopic	*Mesoscopic*	*Microscopic*
based on objects	based on message-pasing	based on message-pasing
compositionality	compositionality	no compositionality
informal semantics	formal semantics	fine-grain formal semantics
no locality	locality	no locality
no concurrency	concurrency support	native concurrency support

ecuted mostly sequentially, whereas actor oriented models are explicitly concurrent. That is, communications between actors may occur asynchronously, and can be carried out in parallel. Since highly distributed communications can involve arbitrary and dynamic patterns of interaction, concurrent computational models seem to offer an interesting approach to the problem.

Actor models were originally introduced in the 1970's, where the fundamental principles were laid out: everything is an actor, actors are operated upon by sending the messages so that they perform the required operation themselves, and events consist of actors sending and receiving messages. Multiple actors may execute concurrently, and each actor has a *behavior* (program) and a unique address. Actor languages are traditionally extensions to process calculi, including primitives for actor coordination. For instance, in one of the best known works in the field, Agha et al., describe actor's behaviour using a call-by-value functional language [4], with λ-calculus abstractions to represent the execution behavior when messages are exchanged. They use primitives such as *send(a,v)* (for sending messages with receiver *a* and contents *v*), *become(b)* (for capturing local state change by, e.g., altering the behaviour of the actor executing to be *b*), and *newadr* and *initbeh* (for actor creation). They also introduce an analogy between the actor primitives and reference primitives: *newadr* is an allocation primitive, *initbeh* and *become* update or alter the actor's state, and *send*'s effect depends on the state.

A simple example of actor behaviour in such model, as found in Agha's

work, can be expressed as:

$$\mathsf{b} = rec(\lambda y.\lambda x.seq(send(x.M), become(y))) \tag{2.2}$$

In this algebra an actor b offers an interface for a message from another actor. Such message M is sent, updating the actor's state; *rec* is here used as a call-by-value combinator. An actual expression to create such an actor and enact such behavior would look like:

$$\mathsf{e} = letactor\{x := newadr()\}seq(initbeh(x, b), send(x, a)) \tag{2.3}$$

Where *letactor* creates a new actor with initial behaviour b and address x, where *seq* is a syntactic expression for sequential composition.

Actor orientation complements and builds on other models such as object orientation by emphasizing communications and concurrency between components. That is, while object interfaces mediate transfer of control, actor interfaces mediate communication, emphasizing interaction between components distinctly from the specification of a component's behavior. This complementarity with existing technologies fits our requirement of building on the state of the art. The actor model is commonly used to separate functionality from component interaction, using hierarchy and model refinement to divide a model into nested sub-models, allowing for scalable compositionality that similar models like Petri Nets lack. Actor interfaces are also very useful to mediate high-level communication without assuming transfer of control.

We therefore use an actor model for a number of reasons. Actors are reported in the literature as being quite useful to understand system heterogeneity via their support for hierarchical composition and localized component interaction [112]. Such model component interaction is clearly distinguished from component behaviour [107].

The actor model we propose, like the original model it is based upon, facilitates conceptual modeling of Grid architectures by representing the *mesoscopic* level of architectural representation quite accurately. This is accomplished by capturing relevant microscopic component details (e.g., offers formal semantics derived from process calculi, clear locality of interaction, and native concurrency support), while still representing the macroscopic overview of the system architecture (e.g., supports high-level representation and boundless compositionality) as shown in Table 2.1. In this way, the architecture description acts as a proxy of the system, with an architecture that may be essentially flat, but represented hierarchically via actor composition and hierarchical refinement. We next elaborate on the specific actor model developed for

analysing distributed software architectures.

2.6 Architecture Representation

We next define our mesoscopic representation model for software architectures, in terms of a set of design primitives and hierarchical graphs. This model provides the basis for our approach to reasoning about the increasingly complex software architectures used as case studies in this thesis.

2.6.1 Design Primitives

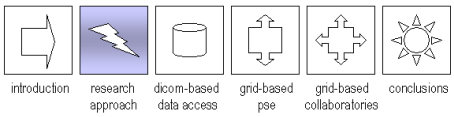
Actors are commonly modeled using visual syntaxes for diagrammatic specification of system structures, simplifying component deterministic expressiveness [108]. In this thesis we extend the Bowers-Ludäscher model for architectural definition [113]. We use the model's concurrent data modeling or structural data types for describing the high-level architectural interfaces between hierarchically-refined actors in distributed Grid systems. We next define a visual syntax and a language description for interface type expressions.

Visual Syntax

Following the graphical Bowers-Ludäscher actor notation [113], we define a visual syntax for architectural representation via a set of basic transformations or design primitives, modifying the original syntax in order to support architectural primitives, as opposed to the original model's intended orchestration semantics. We map define a set of visual basic transformations as shown in Figure 2.4, and defined as follows:

- $\mathbb{A} \rightarrow$ *actor introduction* primitive
- $\mathbb{D} \rightarrow$ *data connection* primitive
- $\Sigma \rightarrow$ *hierarchy refinement* primitive
- $\Phi \rightarrow$ *port introduction* primitive

It is important to note that Φ are abstract interface definitions. That is, actual implementation details, which are particularly useful for structured data typing, are left for developers to refine.



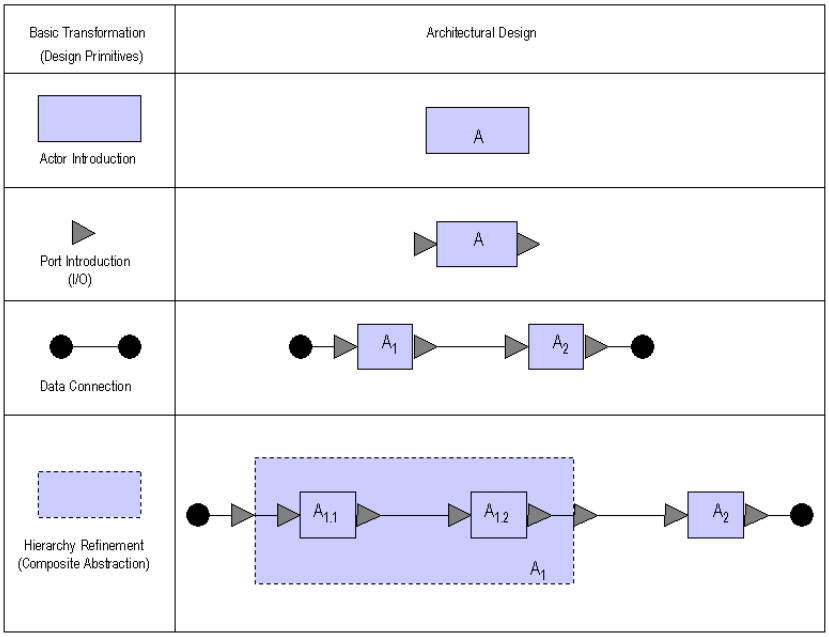


Figure 2.4: Visual Actor model syntax, showing the basic design primitives and their fit into architectural design. A number of basic transformations serve to describe the four main design primitive components of *actor introduction*, *port introduction*, *data connection*, and *hierarchy refinement*

Interface Language Description

We use XML Schema as a language for standard description of structural interface definition, constraining the actor *ports* and their types. XML Schema is a markup language for the description of XML document instances. It includes constraints on the structure and content of documents in a more precise manner than standard syntax constraints in XML, at a high level of abstraction.

We define actors' interface signatures using the XML Schema description shown in Figure 2.5. Here, we define a *stage* root element containing an unbounded number of *actor* elements, which contain an *actorType* ennumerated element definition which may take the *simple* or *composite* values, and an *actorSignature* sequence definition. The *actorSignature* in turn contains the *inPorts* and *outPorts* sequence definitions, where the actual *inPort* and *outPort*

unbounded elements and their contraints are defined. When an actor is of type *composite*, then it may contain an unbounded number of actors, and so on.

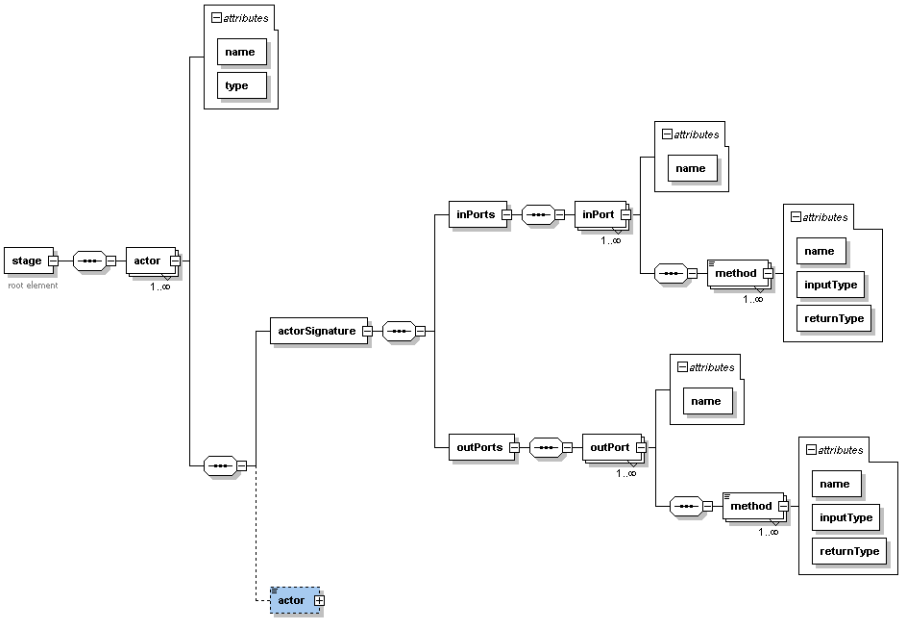


Figure 2.5: Actor interface definition XML Schema; an unbounded number of *actors* with attributes *name* and *type* within a *stage* contain *actorSignature* elements, with *inPorts* and *outPorts* sequences. The interfaces are defined in the *inPorts* and *outPorts*, which contain the actual methods and types to be refined by system developers. Additionally, *actors* may contain an unbounded number of other actors when they are of *composite* type

2.6.2 Hierarchical Architecture Graphs

For formal architectural descriptions we use the Bowers-Ludäscher [113] actor model, originally designed for workflow description. We modify the model and apply it to the description of distributed and potentially concurrent software architectures. We adopt the classic terminology of *Actors*, *Ports*, and *Data Connections*, where Actors contain internal state and procedures that follow an operational thread or execution path, and communicate with other actors

sending messages via their ports, through a data connection.

We define an architectural graph:

$$\mathcal{G} = \langle \mathbb{A}, \mathbb{D}, \Sigma, \Phi \rangle \tag{2.4}$$

where we have:
a set $\mathbb{A}$ of actors representing components, and
a set $\mathbb{D}$ of data connections connecting actors via data ports

Actors. Actors communicate by passing messages between their ports, where:

each actor $A \in \mathbb{A}$ has an associated set ports(A) of data ports
each $p \in$ ports(A) is input port or output port;
ports(A) = in(A) $\cup$ out(A)

Data Connections. Data connections $d \in \mathbb{D}$ are directed hyper edges that may specify a multicast-type of behaviour $d = \langle \mathbb{O}, \mathbb{I} \rangle$, connecting:

n output ports $\mathbb{O} = \{o_1 ... o_n\} \subseteq$ out($\mathbb{A}$),
with m input ports $\mathbb{I} = \{i_1 ... i_n\} \subseteq$ in($\mathbb{A}$)

The architectural graph includes a set of signatures Σ, where i/o signature Σ_A of A = ports (A)

Hierarchy Refinement. Architectural hierarchy refinement can be done by decomposing a subarchitecture into its composite actors $A_{\mathbb{A}}$, and possibly specifying subarchitectures $\mathbb{A}_A$ inside of them.

Nevertheless, the i/o signature of the composite actor (Σ_A) should match the i/o signature of the contained subarchitecture ($\Sigma_{\mathbb{A}}$). Therefore, if a composite actor $A_{\mathbb{A}} = \langle \mathbb{A}, \Sigma_{\mathbb{A}} \rangle$ has a subarchitecture $\mathbb{A}_A$ and a set of ports $\Sigma_{\mathbb{A}}$, the subarchitecture $\mathbb{A}_A$'s signature matches the composite actor's signature $\Sigma_{\mathbb{A}}$ = ports ($A_{\mathbb{A}}$). This hierarchical representation with composite actors defines *a flat system that is represented in a hierarchical way*.

Structural Data Typing. For modeling primitives to be consistent in data-centric models, structural data typing should be defined. We use an abstract description, where:

$p \in$ ports($\mathbb{A}$) may have structural data types

so, if $\mathbb{L}$ is a language for standard description (e.g., XML Schema)

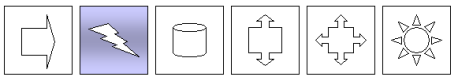
$s \in \mathbb{L}$ is an abstract type expression that describes the messages that p can accept.

Therefore, if a set of structural constraints Φ associates structural types from $\mathbb{L}$ to ports in $\mathbb{A}$.

This model offers a detailed proxy of the architecture, which we can use to reason about the system's components and their potentially concurrent interactions. We decided to leave semantic data typing and associated ontologies as future work for a number of reasons: as originally defined by Bowers-Ludäscher, semantics issues are relevant for orchestration concerns and ontology building, which are out of the scope of this work. While the relevance of the use of detailed semantics in order to *build* architectures is clear, their use for the *representation* of architectures and their proxies in order to reason about their component interaction is not.

2.7 Summary

In this chapter we describe some of the methods commonly used for modeling software architectures, particularly when issues of concurrency are of concern. We identify a gap in the state of the art where work on methods for mesoscopic level of system architecture is not yet complete. We leverage the actor approach in order to fil such gap, and develop an actor-based methodology for analysis of distributed and potentially concurrent architectural models. This methodology is aimed at addressing issues related to object, component, and distributed concurrent systems resulting from recent developments in computational science applied to biomedical research. These developments show trends for computational simulation taking a prominent place, pushing the evolution in the scientific method from classical experimental approaches in-vivo to in-vitro to nowadays progressively more in-silico. We are motivated by the real and specific problem space of resource access and interoperability, as well as by the emerging distributed technology base, which complements recent advances in the state of the art in distributed computing and computational science.



In the following chapters we apply our prototyping and analyzing methodology to a number of increasingly complex case studies, where we focus on issues ranging from transparent access of distributed radiological data and PACS interoperability, to distributed simulation and visualization within an interactive problem solving environment for the study of systolic haemodynamic flows, to a highly-distributed collaboratory for biomedical decision support in viral disease treatment. We also explore performance characterization of some of the deployed architectures, with new approaches to the use of methodologies and tools available from both data and computational Grid viewpoints. This allows us to investigate, in the context of specific biomedical informatics applications, a number of issues related to support for transport-level security and authentication, widely distributed data access and storage capacity, as well as high performance and throughput.

We expect that the process of defining these architectural designs results in a scalable and reusable approach to distributed architectures for biomedicine. Such approach should ease the work of scientists who need to execute their computationally intensive process flows just like they currently do in local computational and data farms.

3

Distributed Data Access*

3.1 Introduction

One of the main issues in modern biomedical informatics is the support for transparent data communications and integration using distributed resources in order to achive more transparent data access and proccessing. In the case of standard image-based medical informatics, this is normally based on the communications approach provided by the Digital Imaging and Communications for Medicine (DICOM) model. DICOM provides a standard interconnectivity and communications model for access to image and some non-image information stored within local Picture Archiving and Communications Systems (PACS) repositories, within the hospital information system or across the enterprise.

*The results presented in this chapter formed the basis for the following two papers:

- A. Tirado-Ramos, J. Hu, K.P. Lee, "Information Object Definition-based Unified Modeling Language Representation of DICOM Structured Reporting: A Case Study of Transcoding DICOM to XML", *Journal of the American Medical Informatics Association*, vol. 9, July 2002, pp. 63-72.
- R. Martinez, J.F. Cook, A. Tirado-Ramos, "Java-CORBA DICOM Adapter Service for DICOM-compliant Datasets", *First Latin American Conference on Biomedical Engineering, IEEE Engineering in Medicine and Biology Society*, November 1998, pp. 621-622.

3.1.1 Problem Statement

Advances in networking technologies push the state of the art in the field: characteristics such as bandwidth, latency and throughput in high-speed networks can make network performance almost comparable to the bandwidth offered by the internal links of commodity computers. Software architects have to reflect such technological evolution of the communications fabric in their designs, gradually moving from monolitic software architectures to more generic component approaches [117, 151]. The continuous increase in the already large amounts of digital data produced by modern medical imaging modalities underlines a number of issues in software architectural design.

For instance, digital modalities that are connected to networked PACS require support for more scalable and transparent access to patient data, straining hospital process flows. Furthermore, the use of legacy architectures and models may hinder scalability and transparency in distributed biomedical systems and their enterprise-wide interoperability.

We propose that new architectural approaches are needed for the representation of abstract layers between biomedical applications, users and the hardware fabric required across geographical and organizational boundaries.

3.1.2 Chapter Organization

In this chapter we introduce the issues related to the evolution of software architectures for biomedical informatics, which we address at length in the context of Grid computing in subsequent chapters. We design and prototype two representative systems, and apply the modeling and analysis approach stated in *chapter 2*. We focus on the issues of enterprise interoperability and data transfer, where we use state of the art in medical informatics and apply object oriented and component models for architectural representation. We also use the actor model of architectural representation in order to identify high and low level constructs to be used as comparison with more advanced distributed systems, such as the ones in *chapters 4* and *5*. We start our analysis of biomedical informatics architecture by using two medical informatics case studies. We first analyze a case study where we prototype an Extensible Markup Language (XML)-based application using object-oriented modeling, bridging the DICOM relational model to standard object-oriented technologies. For this, we specifically use the DICOM Structured Reporting relational model as initial framework. In the second use case, we analyze the architecture of a more complex distributed application. We design and prototype data

access components for this distance radiology system, using the Common Object Request Broker (CORBA) model to access distributed medical image data. This prototype was deployed as part of a large virtual environment for distributed access to patient and image information. We finalize this chapter by presenting a discussion and summary of our results.

3.2 DICOM Communication

The American College of Radiology (ACR) and the National Electrical Manufacturers Association (NEMA) created a joint committee in 1983 to address the problem of developing a standard interface between medical imaging equipment, its associated information, and user applications. The DICOM model aims at interconnecting devices on standard networks, and provides a means by which users of imaging equipment may determine whether two devices claiming conformance are able to exchange information. The model was intended to facilitate communication in networked multi-vendor environments, connecting PACS to Hospital Information Systems (HIS) and Radiology Information Systems (RIS) within hospital data flows, as shown in Figure 3.1.

DICOM was designed to rely on relational *Entity-Relationship* technology. Its model and tables of attributes are called Information Object Definitions (IOD), and define the data structures used in DICOM that describe Information Objects (IO) such as patients and images involved in radiology operations. The basic entity-relationship diagram for a Radiology Department serves as the basis for most of the additional models, including both the data items required in a given scenario and how these items interact and are related to each other.

The DICOM communications model defines Services that use constructs called Operators and Notifications. Sets of generic Operations and Notifications are called DICOM Message Service Elements (DIMSE), which are the communication message services. The combination of an IO and a Service is called a Service-Object Pair (SOP), which is the atomic unit of DICOM functionality [†]. When an Information Object is used with a set of services, a SOP Class is defined (Figure 3.2).

In DICOM, a device may serve as a Service Class User (SCU), or as a Service Class Provider (SCP), following the client/server paradigm. Service Classes are basically built up from a set of operation primitives operating on IODs. The Storage Service class provides the basic support for transfer of

[†]Optional DICOM SOP classes enable annotation, image overlays, and enhanced reporting.

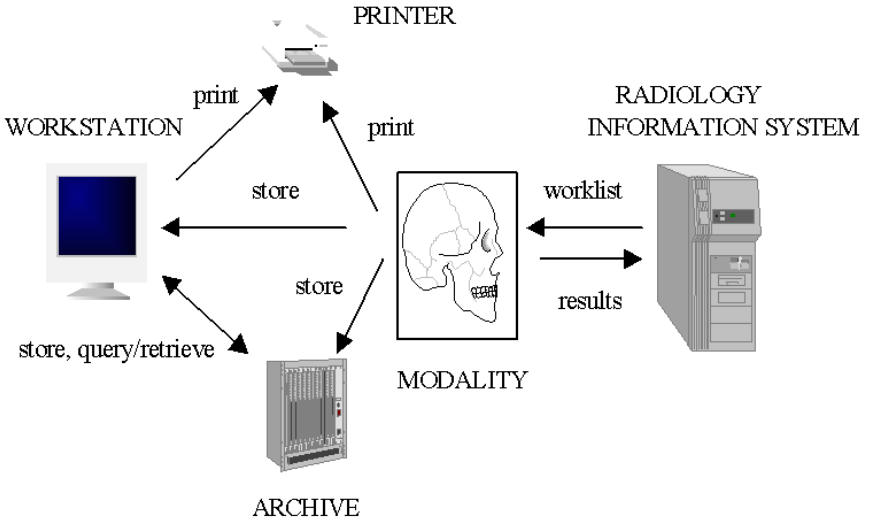


Figure 3.1: Data flow in the DICOM model, from work stations to short and long term archives, printers and the hospital's information systems

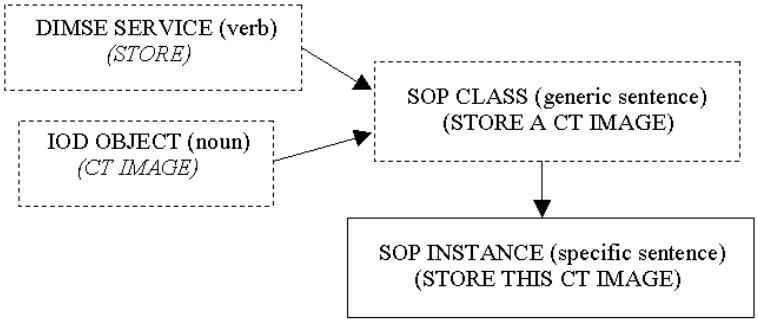


Figure 3.2: DICOM services and "object" definitions result in Service-Object Pair classes, an object-based construct which is the atomic unit of DICOM functionality

images between DICOM applications. To retrieve images from DICOM applications, the Query/Retrieve Service class supports basic operations to access and move images based on simple search criteria, such as get all of the images of a particular patient. The Patient, Study, and Results Management Service classes were also designed to support communication between DICOM compliant PACS and a separate HIS or RIS.

3.3 Improving Interoperability

We first discuss a simple data-centric example, where data communications within a PACS, based on the DICOM SR standard, is extended for enterprise-wide communication using object-orientation.

Image communications models such as DICOM are commonly used to integrate and facilitate communication among image-acquisition, waveform (non-image digital data related to an examination), archiving, and information system components. For applications to handle information from DICOM objects, DICOM tools are required for decoding and encoding the messages. Systems in departments other than these often do not support DICOM but use other communication protocols.

The goal of an integrated electronic medical record can be facilitated by object-oriented representations and Web-based interfaces. Common scenarios include the retrieval by a radiologist of images stored in a PACS and their display for diagnostic interpretation or post-processing. At the workstation, the radiologist can then create structured reports that can be mapped from DICOM to open technologies such as XML. These XML-based reports can offer enterprise access to the key information and related images via a mobile device or a light-weight viewing terminal. In this way vital information can be passed from system to system, and made available as needed at the point of care, with the aggregated value of hierarchically structured information as opposed to natural language format.

Clinicians often prefer an outline report with hierarchic standardized vocabularies and structures over a natural language format. The current usage of standard formats for this purpose is minimal at best, with most of the effort going instead into voice recognition and capture of narrative reports. The DICOM Structured Reporting (DICOM SR) specification is intended to address the structuring of captured data, supporting and structuring conventional free-text reports commonly used in diagnosis. It provides the capability to structure information to enhance the precision, clarity, and value of clinical

documents. The DICOM SR specification supports a semantically rich representation of image and waveform content, that enables experts to share textual and coded data linked to images and waveforms, as well as knowledge about non-linguistic evidence [29]. The purpose of DICOM SR is to improve the expressiveness, precision, and comparability of documentation of diagnostic images and waveforms, so that critical features can be denoted unambiguously by the observer and retrieved selectively by reviewers. This way, findings can be expressed as textual or coded information, numeric measurement values, and references to spatial or temporal regions of interest.

We design an object-oriented model to explore the representation of the relational DICOM SR Information Object Definition (IOD) hierarchy, macro representation, its characteristics of recursion, and some of its constraints. We use a full object-oriented approach within the boundaries of current object-oriented modeling technologies [34], and prototype this architecture using XML technologies. We then move on to the analysis and prototype of components for a more complex, fully distributed system, where we apply the lessons learned and use an actor model for architectural representation.

3.3.1 DICOM SR Modeling

Some attempts have been made to model the process of creating structured reports by using explicitly stated criteria. Some of these attempts have resulted in concept models that support structured data entry and image retrieval, providing a model for analyzing sets of natural- language reports [23, 30]. This is precisely the focus in DICOM SR. For developers who are not DICOM-literate, it is relatively difficult to understand its information model. Information object definitions in DICOM are based on entity-relationship concepts. Interfacing between such relational technologies and distributed, object-oriented applications can present a significant semantic and language barrier for application developers and system architects. We approach this problem via the Unified Modeling Language (UML). UML offers a way for specifying, visualizing, constructing, and documenting the artifacts of software systems, as discussed in *chapter 2*.

We follow the conventional UML notation and syntax, using the basic principles of object-orientation to model system structure and behavior. We define classes and class responsibilities with object-oriented analysis and design concepts such as objects, classes, stereotypes, and relationships. We then prototype it using XML. XML eases access to imaging, demographics, and waveform data using open component technology, addressing interoperabil-

ity and system integration issues. DICOM SR is intended to support the interchange of expressive compound reports in which the critical features shown by images and waveforms can be unambiguously annotated by the observer, indexed, and retrieved selectively by subsequent reviewers. DICOM IODs define the data structures that describe information objects, or logical representations of real world objects, such as patients and images involved in radiology operations.

DICOM SR introduces DICOM Services and IODs used for the transmission and storage of structured reports. DICOM IODs are representations of real world entities (e.g., images and reports) represented in the specification as templates of attributes. The DICOM SR (SOP) definitions allow users to link text and other data to particular images and waveforms and to store the coordinates of findings so that they can see exactly what is being described in a report. These DICOM SR IODs and corresponding DICOM SR Storage SOP Classes enable the query and retrieval of DICOM SR SOP Instances as Instance-level entities, following the DICOM Query/Retrieve model. Information object definition modules contain attributes, which in turn may refer to other attributes or to attribute groupings called Macros. The modeling decisions and mapping rules for mapping the different DICOM SR elements into UML and XML data type definitions start with the SRDocument IOD the root (Figure 3.3).

Furthermore, DICOM SR provides particular recursion characteristics, as present in the SR Document Content module via the Document Relationship Macro. The issue of recursion, from a developer's viewpoint, is a key property that allows multiple containment within structured reports, an important property in numeric-intensive reporting such as in ultrasound applications. The DICOM SR specification reflects this complex property by the cross-referencing of DICOM Macros; this representation makes it hard for those who are not DICOM-literate to understand this property. We approached this problem by modeling the reference relationship to Content Item, as well as its relationship by containment, to reflect this reciprocal recursion. This is a key difference between the various SR SOP Classes as defined in the specification (Figure 3.4).

Additional constraints[‡] to the UML model may be represented by new modeling technologies and artifacts, and are out of the scope of this work.

[‡]For instance, DICOM Type 1 and 2 attributes are mapped as Required; Type 3 attributes are mapped as Optional; for Type 1C and 2C attributes, which are required under certain conditions, a number of conditionality-based rules are applied.

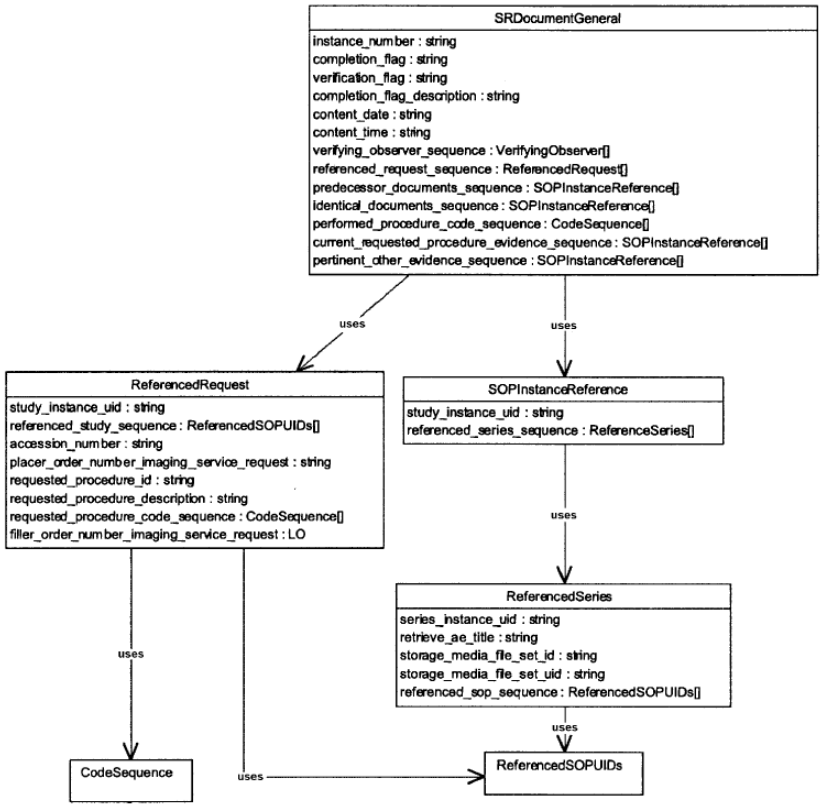


Figure 3.3: An object-oriented class diagram, where the basic DICOM SRDocument-General links to CodeSequence via the ReferencedRequest class

3.3.2 XML-based Prototype

Early attempts to represent medical information contained in structured reports focused on generalized and ad-hoc languages for representation [95]. Our approach is based on the detailed representation of structured reports using an open standard, the XML. We generate XML type descriptions based on this DICOM SR UML model using a number of rules and modeling decisions, e.g.:

- UML classes are mapped to the XML data type Elements,

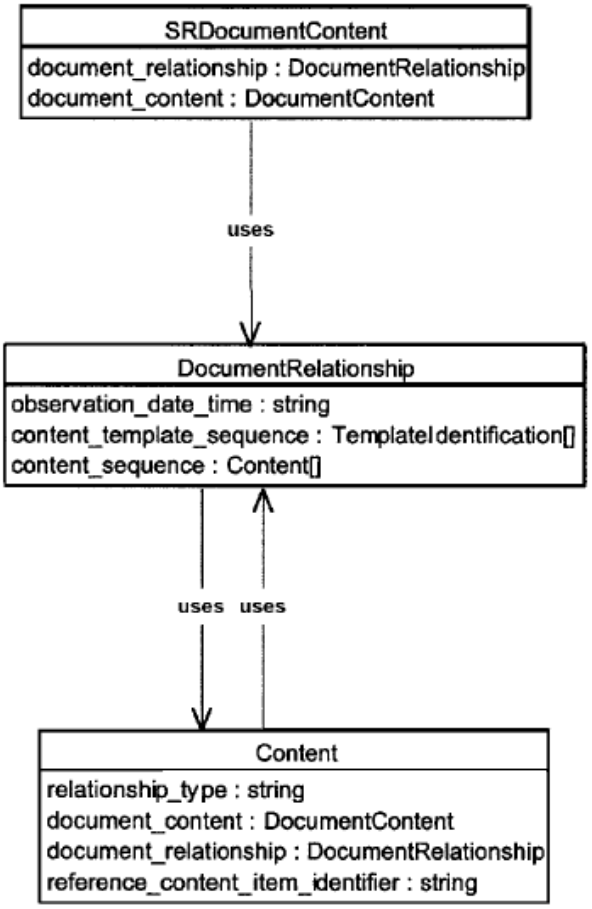
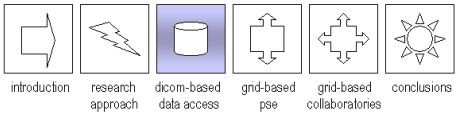


Figure 3.4: Document Relationship Macro, showing DICOM SR's recursive relationship by containment within the SRDocumentContent class; class SRDocumentContent has a "use" relationship with class DocumentRelationship, which itself defines attributes such as document_relationship



- UML class attributes are mapped to XML data type Elements,
- UML Association and Uses relationships are mapped to the XML data type Elements as relationships by containment,
- each atomic attribute is mapped to an element, which contains five attributes: codingScheme, codeId, type, value and label.

We generate the actual XML data type representation by mapping the class structure to the data type framework, taking the model as a reference. This definition provides the basis for an XML translator engine based on Extensible Stylesheet Language Transformations (XSLt) [99]. That is, an XSLt engine may take as input patient demographics and reports in a native or XML format and DICOM SR XSL transformations. The engine then generates an XML output document that was compliant with the DICOM SR data type discussed previously. Finally, a second translation module may convert DICOM SR XML documents into DICOM standard binary files when required.

3.3.3 Towards Enterprise Interoperability

In this enterprise data transfer case study we modeled and prototyped an object-oriented component representation of parts of the relational DICOM specification. This approach, using the UML modeling approach and XML-based open technologies to interface image and non-image digital patient information, enables interoperable data transfer to various clinical specialties as well as leverages Web-aware applications and technologies, *within the enterprise*. This component-based approach proves not only quite helpful in understanding issues related to migrating from monolitic relational models for data access to open, scaleable object-oriented enterprise data access using a structured reporting model, but also provides a context for further investigation on distributed DICOM and other legacy data. Next, we expand our analysis with an example of a component service within a software architecture for distributed teleradiology applications.

3.4 Components for Data Access

In this second case study, we move from simple data access within the enterprise to distributed communications extending the DICOM communications model. For this case, we rely on the CORBA communications model,

where DICOM relational components are extended by a distributed component model.

Healthcare provision has experienced a revolutionary transformation in recent years due to the application of new telecommunication technologies to the transmission of medical information. Distributed systems for hospitals and rural clinics now make possible to handle patient information between remote terminals in multimedia fashion, with audio, text, images, and full-motion video being transferred from one site to another.

Medical records are not necessary normally online. The rapid development of some of the most important distributed applications demands now a wider approach to these kinds of medical information services. For instance, in distributed biomedicine applications, technologies such as PACS are now frequently linked to online medical records systems such as HIS and RIS. In this way, physicians or technologists may use PACS to extract patient data and communicate with the HIS and RIS to handle medical imaging information.

Clinical specialties such as radiology and cardiology continue to be a strong focus of distributed medical informatics research. We next analyze an effort on component approaches for distributed teleradiology applications within a large distributed computing environment, a CORBA-based virtual radiology environment. The design of the storage and archiving systems in this virtual environment is based on a distributed computing environment using a set of open middleware protocols. Within this virtual environment, a seamless radiology department is virtualized, where patient cases could be acquired at any medical treatment facility and sent to any other that had available radiology reading resources. For this case, the lack of object-oriented support in DICOM limits the interoperability of the distributed DICOM gateways with distributed objects present in the network. Also, DICOM interfaces are generally tightly-coupled to specific programming languages and operating systems.

The integration of legacy data sets based on the DICOM relational model into the distributed environment is not trivial. We next design and prototype a component service within the system architecture that extends DICOM basic functionalities and provides seamless access to clinical datasets by users in regards to operating system, distributed platform, and legacy database access.

3.4.1 Actor Analysis

We present an actor-based representation of the main architectural components for this use case, following the actor model presented in *chapter 2.3*. With this approach we aim to emphasize communications and concurrency

between components. We also aim at separating functionality from component interaction, and comparing the architecural components of increasingly complex distributed biomedical informatics case studies shown in *chapters 4 and 5*.

Actor-based Grid Architecture

In the actor design, we identify three levels of abstractions for component representation, and start our analysis at the first level. We identify as main components a set of four actors, *client*, *server*, *object broker* and *interface*, as shown in Figure 3.5, where:

- *client* offers an interface for external input, in this case from the system's user, and two abstract output interfaces: one to initialize *object broker* and one to initiate interaction with *interface*,
- the composite *interface* offers two, possibly concurrent, input interfaces: one from *client*, and one from the composite *server* actor to get data previously requested. *Interface* also provides an output interface to *server* actor, by which it may request data,
- *object broker* is shown as providing one input interface via which it can be initialized by *client*, and one output interface to perform CORBA object bindings from the broker's interface repository.

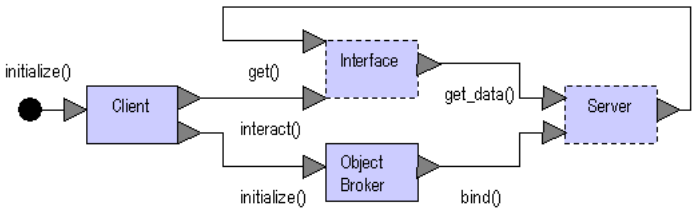


Figure 3.5: First level of actor containment, showing the interactions between *client*, *server*, *object broker* and *interface*

In the second level of containment we find the internals of the first level's composite *interface* and *server*, as shown in Figure 3.6. *Interface* is shown as composed by four components, *ui*, *session manager*, *patientInfo manager* and *retriever*, where:

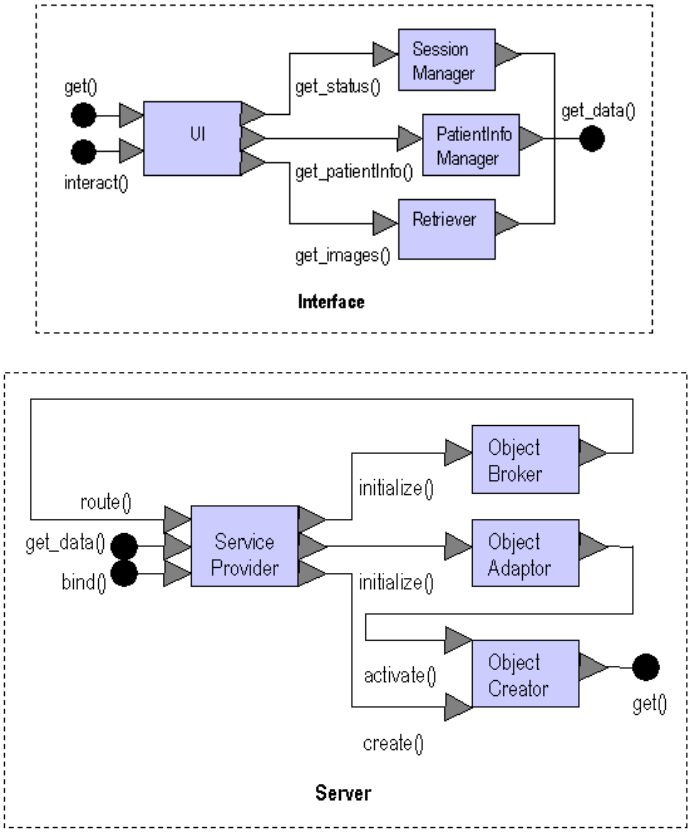


Figure 3.6: Second level of containment, detailing the *interface* and *server* composites

- *ui* represents the user interface, and provides both external interfaces to the upper level, as well as abstract output interfaces to *session manager* for accessing current status of patient study, *patientInfo manager* for getting patient's demographics and image metadata, and *retriever* for image data transfer,
- the other three actors all offer output interfaces for data provision to the upper level *server*.

The *server* composite actor is shown as containing actors *service provider*, *object broker*, *object adaptor* and *object creator*, where:

- the *service provider* actor provides three input interfaces, two of them to access it from the upper level *interface* and *object broker*, and one from the *object broker* for routing object requests and responses,
- *object broker* and *object adaptor* are initialized by *service provider* via their input ports, while the *object adaptor* offers an output interface into the composite *object creator* for object activation,
- *object creator* offers input interfaces to *service provider* for object creation, and an output interface to the upper level *interface* for object transfer

Actor Signatures

The architecture's actor port signatures are defined in XML, based on the XML Schema actor interface definition described in *chapter 2*, for matching of composite actor signatures and basic structural data typing between actor ports. A code fragment of this architecture's actor signatures is shown in Figure 3.7. There, a *stage* contains actors *client*, *server*, *object broker* and *interface*. Furthermore, it shows the second level composite *interface* internals, containing simple actors *session manager*, *patientInfo manager*, *retriever* and *ui*. It also shows inPort signature matching between *interface* and *ui*.

This approach is based on *eq. 2.6*, which states the use of a language $\mathbb{L}$ for standard signature description, with $s \in \mathbb{L}$ describing the messages that a given port can accept. In this case, the language used is defined by the XML Schema defined in Figure 2.3.

That is, we use this actor model of computation for the representation of concurrent behaviour in order to identify patterns of distributed component behaviour and how such components evolve and interact across levels of complexity, compared with the use cases in *chapters 4* and *5*, and discussed in the thesis *conclusions*.

3.4.2 CORBA-based Prototype

Large distributed environments such as the virtual environment in this case study present special requirements. Enhanced security, for instance, may introduce firewalls, gatekeepers, and other intermediate gateways in the path of DICOM file transmission. In DICOM, communication between entities is commonly considered on a direct association basis. Conventional gatekeepers and gateways usually allow access by Telnet, the File Transfer Protocol (FTP), or other applications that may not be able to interpret the contents of

```
<?xml version="1.0" encoding="UTF-8"?>
<stage xmlns="http://staff.science.uva.nl/~alfredo/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://staff.science.uva.nl/~alfredo/ C:\WINNT\profiles\software\MYDOCU~1\ALTOVA~1\actor1.xsd">
  <actor type="composite" name="interface">
    <actorSignature>
      <inPorts>
        <inPort name="get">
        <inPort name="interact">
      </inPorts>
      <outPorts>
        <outPort name="get_data">
      </outPorts>
    </actorSignature>
    <actor type="simple" name="ui">
      <actorSignature>
        <inPorts>
          <inPort name="get">
          <inPort name="interact">
        </inPorts>
        <outPorts>
          <outPort name="get_status">
          <outPort name="get_patientInfo">
          <outPort name="get_images">
        </outPorts>
      </actorSignature>
    </actor>
    <actor type="simple" name="session manager">
    <actor type="simple" name="patientInfo manager">
    <actor type="simple" name="retriever">
  </actor>
  <actor type="composite" name="server">
  <actor type="simple" name="client">
  <actor type="simple" name="object broker">
</stage>
```

Figure 3.7: Actor port definitions: code fragment of the actor's port signatures for the distributed medical access application actor architecture described in *Figures 3.11-3.12*, showing higher level *interface* matching lower level *ui* input ports

submitted data nor allow proper synchronization. The need for a DICOM-compliant data access service that wraps the medical information into object entities was identified.

The use of CORBA-based distributed object environment technologies offers several benefits in this context. For instance, interfacing to the legacy databases by developing CORBA wrappers allows access to the database data structures without disturbing the existing databases.

The client of a CORBA object has an object reference to issue method requests. If the server object is remote, the object reference points to a stub function, which uses the Object Request Broker (ORB) logical bus. The stub code uses the ORB to identify the machine that runs the server object and asks that machine's ORB for a connection to the object's server. When the stub code has the connection, it sends the object reference and parameters to the skeleton code linked to the destination object's implementation. The client has no knowledge of the CORBA object's location, implementation details, nor which ORB is used to access the object. Different ORBs communicate via

the Internet Inter ORB Protocol (IIOP) [132] or the General Inter ORB Protocol (GIOP) [55]. CORBA allows the connection among the distributed objects and their integration with the rest of the distributed environment. The medium that CORBA uses to perform this integration is the Interface Definition Language (IDL), which describes interfaces between distributed components and their bindings.

For this distributed computing use case we design and prototype a service based on the CORBA model in order to wrap legacy DICOM data and related information. Our prototype architecture provides seamless access to medical images and related information via an interface to some DICOM services, such as the Query and Retrieve DICOM image-oriented Service Classes. The service offers access to clinical images and related information based upon patient and study identifiers, with a Client service that queries Server services connected to a commercial ORB. The uncompressed, DICOM-compliant datasets are wrapped and transferred through the network via both pure CORBA objects, as well as Java-based sockets, using the stubs and skeletons produced by the IDL specification implementation. This offers an interface to the Retrieve Service Class. A Store Service Class interface is available from a freeware DICOM image viewer application to store the clinical information in formats which are compatible with more widespread image standards and office type computing equipment, such as Tagged Image File Format (TIFF) or Graphics Interchange Format (GIF). The datasets are packaged as binary components that remote viewing workstations interacting with the roles of as DICOM Service Class User (SCU) or Service Class Provider (SCP) may access transparently via method invocations (Figure 3.8).

3.4.3 Experiment Results and Conclusions

In our implementation the server is designed to initialize the ORB and the Basic Object Adaptor (BOA), create the object implementations and activate them, and wait for incoming messages. Socket and stream objects are used to transfer the requested image file. The client application is designed to get the session status information, patient demographics and image data, wrapped as CORBA objects.

The program initializes the ORB, binds to the object implementations, queries the needed information, displays it, and retrieves the data files. In this architecture, the ORB is in charge of the requests to the server that implements the requested objects and returns the results to the client. Access to information about the patient, study hierarchy, and series is achieved by offering a trans-

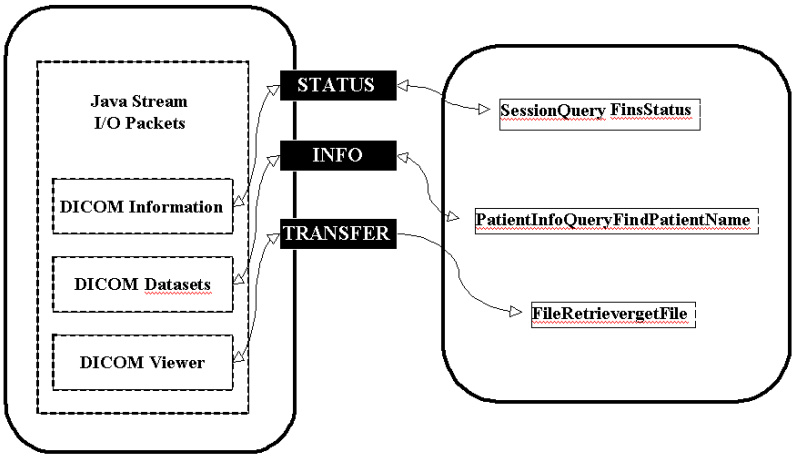


Figure 3.8: CORBA-based access service, showing the session status, patient information, and data transfer interfaces. The application communicates with the distributed object that is performing the operation; the data is passed from the client to the server and is associated with a particular operation on a particular object, using clearly defined inteface definitions, data is then returned to the client

parent interface to the Query Service Class. The DICOM-compliant datasets are wrapped and transferred through the network via both pure CORBA objects and an interface to the Retrieve Service Class.

We prototyped a basic working data access component within the virtual environment in order to evaluate various aspects of the application's architecture. For instance, during our data transfer experiments we found similar performance in terms of time delay in the inquiry and transmission processes when comparing the transmission of images and text files by CORBA objects and Java sockets, as shown in Figure 3.9. Here, the transfer of digital images and diagnosis information via pure CORBA objects proved faster than in the case of sockets, both while creating new objects and with existing ones. The experimental environment was set up in a shared 100Mb Ethernet, with a 3COM dual speed hub. We used SUN Microsystems ULTRA1/3Dcreator with 256 MBytes RAM running SOLARIS 2.5.1, 200 MHz Pentium PCs with MMX and 96 MBytes RAM running under Windows NT 4.0 ServerPack3.

We designed and implemented a component-based service that extends basic DICOM functionality with component and object oriented transparency and scalability, and showed that the added overhead of the implementation

	Object not yet created	*Object created already*
Binding to SessionManager	0.470 (0.440 - 1.973)	0.431 (0.421 – 0.433)
Binding to PatientInfoManager	0.035 (0.030 – 0.041)	0.022 (0.020 – 0.022)
Getting the status information	0.090 (0.090 – 0.091)	0.055 (0.050 – 0.080)
Getting the DICOM information	0.091 (0.090 – 0.093)	0.076 (0.070 – 0.102)
Image Transmission (65-514 KB)	0.245 (0.240 – 0.360)	0.242 (0.240 – 0.360)
Diagnosis Transmission (1 KB)	0.222 (0.220 – 0.431)	0.222 (0.220 – 0.431)
Viewer Transmission (1031 KB)	0.550 (0.220 – 0.711)	0.550 (0.220 – 0.711)

(a) Transfer times in seconds, using CORBA Objects and Java Sockets

	Object not yet created	*Object created already*
Binding to SessionManager	0.470 (0.440 - 1.973)	0.431 (0.421 – 0.433)
Binding to PatientInfoManager	0.035 (0.030 – 0.041)	0.022 (0.020 – 0.022)
Binding to FileRetriever	0.421 (0.401 – 0.551)	0.381 (0.360 – 0.411)
Getting the status information	0.090 (0.090 – 0.091)	0.055 (0.050 – 0.080)
Getting the DICOM information	0.091 (0.090 – 0.093)	0.076 (0.070 – 0.102)
Image Transmission (65-514 KB)	0.094 (0..091 – 0.111)	0..092 (0.07 – 0.061)
Diagnosis Transmission (1 KB)	0.082 (0.062 – 0.120)	0.033 (0.032 – 0.036)
Viewer Transmission (1031 KB)	0.762 (0.521 – 1.432)	0.426 (0.421 – 0.462)

(b) Transfer times in seconds, using pure CORBA Objects

Figure 3.9: Data transfer experiment results, showing average image transfer time, using a) hybrid and b) pure CORBA objects; we found similar performance, in terms of time delay in the inquiry/transmission processes, while comparing the transmission of images and text files by both communication paradigms. Contrary to what was expected, the transfer of images and diagnosis information via pure CORBA objects has proved not only completely seamless, but also faster than in the case of CORBA-sockets

is minimal. A number of issues still need to be fully addressed. For instance, the use of virtualized storage repositories distributed across organizational boundaries, not tied to specific technologies or interface implementations is quite relevant. These kinds of virtualized resources can be based upon complex hierarchies, with records containing references to loosely coupled repositories, and may not be trivial to access via, e.g., CORBA bindings. Also, the fact that multiple users may access a limited number of reliable servers *con-*

currently can greatly affect resource reliability.

3.5 Summary

In this chapter we design, prototype, and anayze two increasingly complex case studies: a UML model of relational DICOM-based data transfer, and a CORBA-based component extension of DICOM services for large scale data sharing. These case studies illustrate previous advances in extending the functionality of distributed biomedical informatics applications for image-based data access and interoperability (Table 3.1). We show that distributed computing technologies such as CORBA and XML can be used to extend software architectures based on legacy models, such as the relational DICOM model, without heavy loss of performance. We model, implement and deploy these systems using an actor representation to provide a clear hierarchical representation of the system, with component hierarchy and abstract interface mapping and matching. Later in *chapter 6* of this thesis we use this analysis to compare base actor components with increasingly the more complex prototype implementations found in *chapter 4* and *5*.

Table 3.1: Extending the standard DICOM information model with object models (using UML) and component implementation (using XML and CORBA), which enhanced scalability, transparency for loosely-coupled data access across distributed networks

State of the art	**Contribution**
DICOM-based distributed data access	*Component and Object-based distributed data access*
DICOM	XML, CORBA
Local PACS network connectivity	Distributed connectivity with resource transparency
Relational model	Component-based model
Tightly-coupled implementation	Transparent method invocation
Monolitic software architecture	Object-oriented software architecture
Virtual environment	Scalable virtual environment with support for legacy data
Low latency for tightl-coupled store and forward	Low latency for tightl-coupled store and forward

4

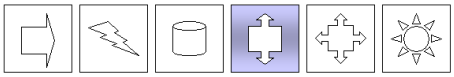
Interactive Grids*

4.1 Introduction

The biomedical informatics field is currently moving towards the intensive use and support for more complex data types, user and simulation-centered new requirements for analysis, collaboration and interactivity, while build-

*The results presented in this chapter formed the basis for the following four papers:

- A. Tirado-Ramos, P.M.A. Sloot, A.G. Hoekstra, M. Bubak, "An Integrative Approach to High-Performance Biomedical Problem Solving Environments on the Grid", *Parallel Computing*, special issue on High-Performance Parallel Bio-computing, vol. 30, 2004, pp. 1037-1055.
- A. Tirado-Ramos, G. Tsouloupas, M.D. Dikaiakos, P.M.A. Sloot, "Grid Resource Selection by Application Benchmarking: a Computational Haemodynamics Case Study", *International Conference of Computational Science 2005*, Atlanta, USA, May, 2005, series Lecture Notes in Computer Science, vol. 3514, Springer-Verlag, pp. 534-543.
- A. Tirado-Ramos, D.J. Groen, P.M.A. Sloot, "On-line Application Performance Monitoring of Blood Flow Simulation in Computational Grid Architectures", *18th IEEE Symposium on Computer-Based Medical Systems*, special track Grids for Biomedicine and Bioinformatics, Trinity College Dublin, Ireland, June 23-24, 2005, pp. 511-516.
- A. Tirado-Ramos, J.M. Ragas, D.P. Shamonin, H. Rosmanith, D. Kranzlmueller, "Integration of Blood Flow Visualization on the Grid: the FlowFish/GVK Approach", *2nd European AcrossGrids Conference*, Nicosia, Cyprus, January 28-30, 2004, series Lecture Notes in Computer Science, vol. 3165, Springer-Verlag, pp. 77-79.



ing on previous research and state of the art. One of the most prominent and relevant developments in distributed system architecture research is the *Grid Computing* model. In the next chapter we analyze in depth a case study of Grid architectures for collaborative biomedical environments that require open distributed services and tools for users requiring large scale, interactive biomedical simulation systems.

Grid technology offers a unified means of access to different and distant computational and data resources. Connectivity between distant locations, interoperability between different kinds of system architectures, and high levels of computational performance are some of the most promising characteristics that Grids offer to complex, simulation-centric biomedical applications.

In this chapter we discuss a novel architecture for interactive biomedical applications running on a Grid, focusing on simulation-centric issues on production-type Grid infrastructures. We begin by defining the context, a problem-solving environment for blood flow simulation.

4.1.1 Problem Statement

Cardiovascular disease is a leading cause of death in the developed world. For instance, about every 30 seconds someone in the United States suffers a coronary event [64]. Vascular diseases affect arteries and veins, with vascular disorders generally falling into two categories: aneurisms and stenoses. An aneurismal disease is balloon-like swelling in the artery walls. Stenosis is a narrowing or blockage of the artery. The purpose of vascular reconstruction is to redirect and increase blood flow or repair a weakened or aneurismal artery if necessary.

The best treatment is not always obvious, though, because of the complexity of the vascular disease of a patient and sometimes of other diseases that the patient may have. An interactive application in which patient-specific situations and several treatments can be simulated and tested in near real time can provide useful clues for surgeons and assist students in understanding the complexity of the treatment.

4.1.2 Chapter Organization

In this chapter we define the main architectural requirements for a Grid-based problem solving environment, prototype a representative system, and analyze the system's behaviour. We use the actor-based approach described in

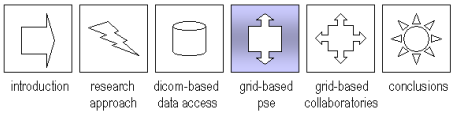
chapter 2 to analyze component behaviour of the implemented system, and finalize by describing a typical use case scenario using the prototype. We then use this prototype for further investigations on performance monitoring and application-level benchmarking for resource selection, as well as distributed visualization. To this end, we port the prototype to an interactive problem solving environment for computer simulation of pre-operative planning of vascular reconstruction developed by the University of Amsterdam [26]. For the experimental setup, highly distributed computational, storage and Grid service resources are used to access medical image repositories. We use the European CrossGrid framework and testbed; this allows us to build on available achievements from other European Grid projects such as European DataGrid and the Large hadron collider Computing Grid (LCG) [105].

4.2 Simulation-based Biomedicine

Even though digital technologies have noticeably improved physicians' workflows, the verification of operation plans can be one of the most complicated tasks in the process (Figure 4.1). There are several imaging techniques that can be used to detect vascular disorders. For instance, three-dimensional (3D) data acquired by Computed Tomography or Magnetic Resonance Imaging can be converted into a set of two-dimensional (2D) slices that can be displayed and evaluated from various perspectives and at different levels. Magnetic Resonance Angiography is a technique for imaging blood vessels. This technique is expensive, though popular among specialists working in cardiovascular diseases because of its ability to non-invasively visualize the disease.

We use the Virtual Radiology Explorer (VRE) application, which puts a user at the center of an experimental cycle controlled by a computer, and allows him to apply his expertise in-silico to find better solutions for treatment of vascular diseases. The aim of the VRE is to provide end users with an intuitive virtual simulated environment to access medical image data, visualize it, and explore patient vascular condition. Naturally, since this kind of medical image processing is usually a complicated and resource intensive task, additional computational resources are needed. Furthermore, the VRE supports smart-agent based interaction, used for real-time simulation [163].

The VRE contains an efficient parallel computational hemodynamics solver that computes pressure, velocities, and shear stresses during a full systolic period. The simulator is based on the Lattice-Boltzmann method (LBM), a mesoscopic approach for simulating fluid flow based on the kinetic Boltz-



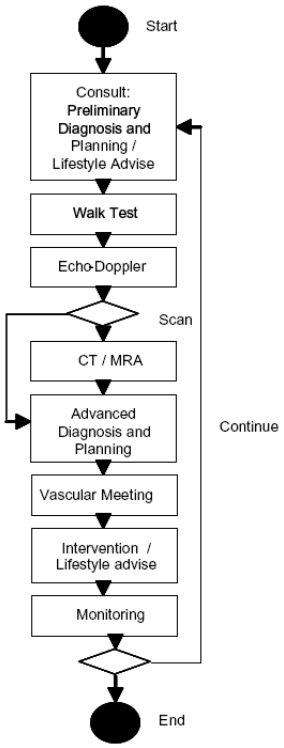


Figure 4.1: Simplified task trajectory of vascular diagnosis and treatment; digital technologies now allow better workflows, like not having to go back to radiography to develop film, the ability to produce multiple originals instead of second films of questionable quality, and so forth

mann equation [147]. In order to convert the medical scans into LBM meshes, the raw medical data is first segmented so that only the arterial structures of interest remain in the data set. The segmented data set is then converted into a mesh that can be used by the LBM solver; boundary nodes, inlet and outlet nodes are added to the Grid using a variety of image processing techniques. The simulator generates the patient blood flow parameters using Grid resources. In order to allow for parallel execution, the simulation volume is divided into several sub-volumes, and each sub-volume is processed concurrently. For visualization the VRE uses a semi-immersive wall as a projection environment [26], in addition to a virtual reality environment where the pa-

tient's data obtained from the imaging modality is visualized as a 3D stereoscopic image, together with the graphical interpretation of the simulation results [24]. A user can then manipulate the 3D images of arteries, patient's body and blood flow structures in virtual reality, an environment where users interact freely in a 3D space with entities within it. The working prototype of the VRE is provided with a multi-modal interface described in [164].

4.3 Architectural Requirements

Scalability and seamless resource sharing are at the heart of Grid-based architectural design. To this end, we base the requirements on a simulation-centric interactive biomedical application, and the facilities offered by current production-level Grid infrastructure and services. We next elaborate on the main architectural requirements related to the use and support for *virtual organizations*, *service orientation* and infrastructure *interoperability*.

4.3.1 Virtual Organizations

The central concept in Grid computing is the idea of *Virtual Organizations* (VOs). VOs are multiple trust domains, where interaction is based on mutual authentication via federated domains supported by their respective security mechanisms. Trust in the VOs is commonly established via a Public Key Infrastructure (PKI). Every entity in the system is issued a "certificate" that links an identifier (the persons name, or a DNS name) to a piece of unique cryptographic data (an RSA keypair, for instance). These certificates usually have a limited lifetime when stored in a file, or are carried on hardware tokens like smart-cards and USB keys. In practice this usually requires the use of asymmetric cryptography technology, which currently means a Public Key Infrastructure (PKI), such as the one provided by the Globus Toolkit's Grid Security Infrstructure (GSI). Every entity in the system is issued with a "certificate" as token, which links an identifier to a piece of unique cryptographic data.

GSI takes care of the mapping of credentials, via temporary proxies or gateways. A set of keys is normally used as tokens by Grid users; a *private key* that is kept secret and a *public key* that is made publicly available: data which has been encrypted with the public key can only be decrypted with the private key, and vice versa. Policy definition, though, is left to the local owner's trust policy of the resource to be accessed, which should provide the means for single authentication and delegation. It is important to note that GSI decisions

are made at the local level, where a periodically updated LDAP-based directory service manages VO users, in accordance with local security policy.

VOs support Grid middleware, the Grid software that links applications and layers in the Grid architectural model. A number of middleware implementations are currently available, of which Globus [68], Cactus [7], Legion [79] and Unicore [65] are the most mature. In this thesis we use the Globus core middleware since it is the most suitable tool for our approach due to its open source nature, service orientation, loosely-coupled transport-level data transfer. The Globus Alliance is a community of organizations and individuals developing the technologies behind the popular Globus Toolkit, an open source software toolkit used for building Grid systems and applications. The toolkit has evolved from its standard version 2, enhanced by multiple projects all over the world such as the European Data Grid (EDG) and the Large Haedron Collider Computing Grid (LCG), to the present web services-oriented version 4, still under development.

4.3.2 Service Orientation

The recent Open Grid Services Architecture (OGSA) [69] extends the Web Services terminology to include Grid concepts, and to manage the creation and termination of resources as services. Its main focus is on the definition of abstract interfaces that allow services to cooperate. Grid Services, as defined by OGSA, integrate Grid technologies from Globus toolkits with Web Services mechanisms to construct a Grid-based distributed framework. That is, a Grid Service instance is a potentially transient service that conforms to a set of conventions, expressed as Web service description interfaces, extensions, and behaviors. Grid Services provide controlled management of the distributed and often long-lived state that is commonly required in sophisticated distributed applications.

The release of the Globus Toolkit 4 (GT4) provides a simple approach to implementing OGSA. The GT4 is currently being deployed among new Grid development projects, and is expected to become a de-facto standard for Grid services. More mature frameworks such as the CERN's Large Hadron Collider Computing Grid (LCG) [105] have been successfully deployed by production-type Grid infrastuctures like the European CrossGrid project [43], providing scientists with a production testbed that is maintained continuously. Other toolkits such as the Common Component Architecture toolkit (CCAT) [109], which aims to create a standard component architecture for high performance computing, currently use the Globus toolkit as their base foundation.

4.3.3 Interoperability

High-performance Grid distributed computing continues to evolve and to become a standard tool for data access and computational job submission in scientific organizations. Virtual scientific communities and organizations are currently being created and maintained which support large, distributed and diverse information sources. Access to such resources is inherently complex, and is aggravated by structural heterogeinity of both the resources and the software infrastructures that support them.

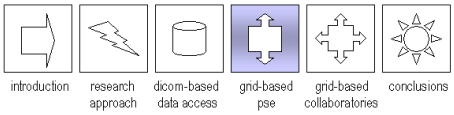
In the case of biomedical applications data are usually available in heterogeneous formats and from various legacy sources, and computational job submission is often supported by infrastructure-specific frameworks that may be required to intercommunicate; as Rambadt et al. [131] state: "Different (Grid) projects focus on different aspects and it is only natural to combine them". Wide-area access to biomedical information and computation usually requires higher degrees of interoperability. Currently available Grid technology supports data access and computational job sumission within specific toolkits, though the diverse biomedical informatics tools that generate and consume data rarely come from within a single source or project [45], requiring resource and infrastructure interoperability in order to access resources seamlessly across Grid virtual organizations. A. Ouksel et al. [125] differentiate four types of interoperability: semantic, syntactic, system, and structural. In Grid computing, while there are a number of efforts currently at work in the fields of semantic [36,77], syntactic [89] and system [65] interoperability, there is still much work to do into the issues related to structural interoperability (e.g., seamless access to a set of shared infrastructure services).

4.4 System Architecture

We next discuss an architectural approach, mainly as applied within the framework offered by the European CrossGrid Project [153], a production Grid testbed with resources distributed across 16 European sites and production services for interactive computation.

4.4.1 Grid Infrastructure

The European CrossGrid testbed provided a production level infrastructure built for the support of interactive e-Science. It contained sites ranging from



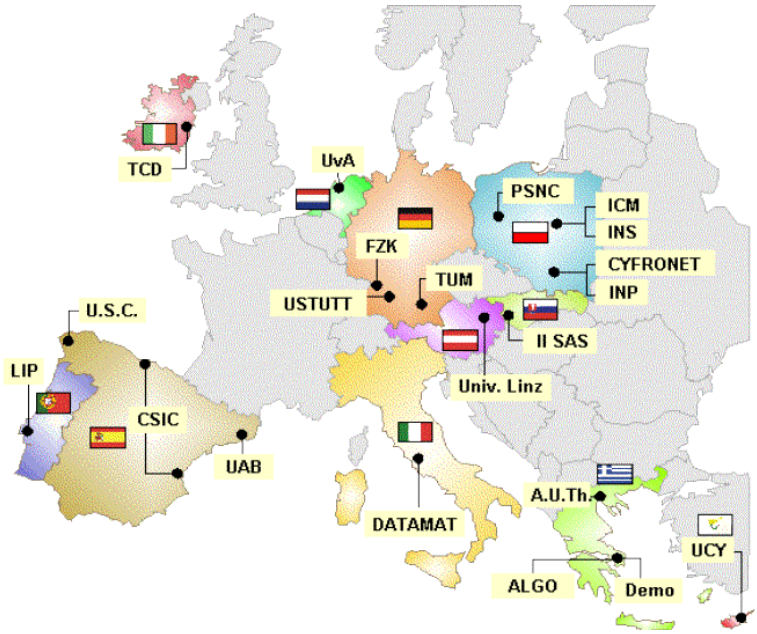


Figure 4.2: The CrossGrid testbed: a distributed high-performance computational network with clusters offering computational resurces, Grid services and tools across Europe.

relatively small computing facilities in universities to large research computing centers, offering an ideal mixture to test the possibilities of an experimental Grid framework (Figure 4.2). National research networks and the high-performance European network, Geant [76], assured interconnectivity between all sites. The network included a local step, typically inside a research center or university via Fast or Gigabit Ethernet, a jump via a national network provider at speeds that ranged from 34 Mbits/s to 622 Mbits/s or even Gigabit, and a link to the Geant European network at 155 Mbits/s to 2.5 Gbits/s. The CrossGrid team focused on the development of Grid middleware components, tools and applications with a special focus on parallel and interactive applications. The added value of this project, therefore, consisted in extension of the Grid to interactive applications. Interaction, in this context, refers to the presence of a human in a processing loop, and a requirement for near real-time response from the computer system. The CrossGrid testbed extended the European Data Grid (EDG) experience on testbed setup

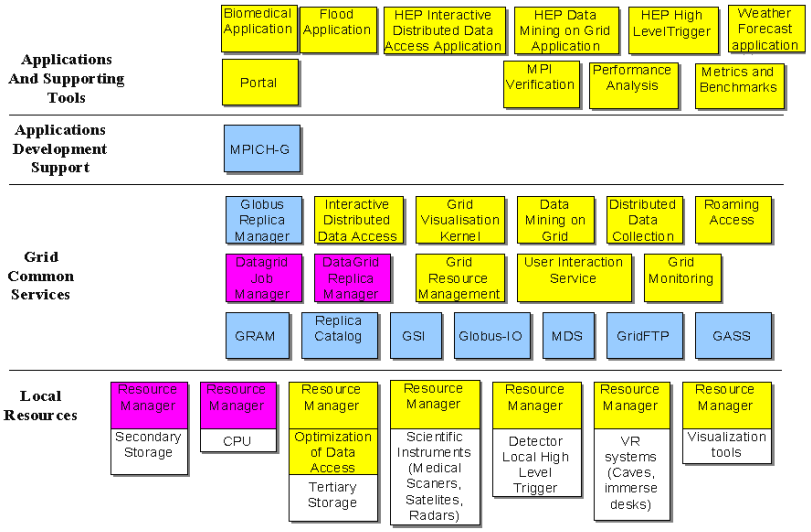
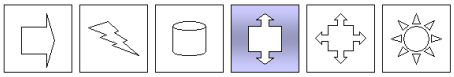


Figure 4.3: The CrossGrid testbed: layered architectural view, with local resources at the bottom fabric layer, a set of common Grid services and middleware from Globus, DataGrid and CrossGrid, application development support for cross-site job submission, and a set of interactive applications such as the biomedical VRE, from M. Bubak, M. Malawski, K. Zajac, "The CrossGrid architecture: Applications, tools, and Grid services", *International Conference on Computational Science (ICCS)*, Vol. 2657/2003, pp. 207-213

and Globus middleware distributions (Fig 4.3).

The CrossGrid testbed architecture and minimum hardware requirements were modeled after the LCG2 specification [105], with each site offering at least five system component roles:

- a *gatekeeper* that provided the gateway through which jobs are submitted to local farm nodes,
- *worker nodes* (WN) or local farm of computing nodes where computation could be actually executed; the combination of a Gatekeeper with its WNs is usually called a *computing element* (CE),
- *storage element* (SE) or storage resource that included a Grid interface ranging from large hierarchical storage management systems to disk pools,



- *user interface* (UI) machine, used by end-users to submit jobs to the Grid CEs,
- *local configuration* (LCFG) server, used to install, configure and maintain the above systems from a single management system.

The CrossGrid testbed included a set of tools and services such as monitoring tools, development tools, a remote access server, portals and a parallel resource broker. It supported the Message Passing Interface (MPI), a popular, widely supported standard for writing parallel programs using message passing [80]. Since the support of the CrossGrid Resource Broker for parallel applications using the MPI implementation used in a Globus environment, MPICH-G2 [97], was still being deployed at the time of the experiments, we based most of the work on the standard MPICH-P4 device, via the CrossGrid Resource Broker. For MPICH-G2 jobs we submitted the jobs directly, using the Globus job submission capabilities.

We incorporated the VRE application into the Grid via CrossGrid's portals. One of the key components of the CrossGrid architecture was the Grid portal offered by the Poznan team, the Migrating Desktop (MD) [104]. This generic portal produces transparent user work environments, independent of the system version and hardware, allowing the user to access Grid resources and local resources from remote computers, via a back-end access service. It allowes the user to run applications, manage data files, and store personal settings, independent of the location or the terminal type. With the use of the portal we achieved secured Grid access, node discovery and registration, Grid data transfer, application initialization, medical data segmentation, segmented data visualization, computational mesh creation, job submission, distributed blood flow visualization, and bypass creation.

The CrossGrid testbed provided transparent access to the distributed medical data of interest from medical image repositories acting as Grid Storage Elements (SEs) and managed by CrossGrid's Replica Manager. For the specific case of the interactive biomedical application, the main concerns were clear: the simulation-centric application and available distributed visualization services could be treated like black boxes initially, with the burden of connectivity and interoperability resting on the interactive services needed to run the solver on the Grid.

For some of the experiments we used the Dutch Distributed ASCI Supercomputer (DAS-2) (a smaller national wide-area distributed computer of 200 Dual Pentium-III nodes) before moving to the larger CrossGrid testbed. This national distributed machine consisted of clusters of workstations, which

were interconnected by SurfNet, the Dutch university Internet backbone for wide-area communication. Myrinet, a popular multi-Gigabit LAN, was used for local communication.

The DAS-2 machine was used for research on parallel and distributed computing by five Dutch universities: University of Amsterdam, Vrije Universiteit Amsterdam, Delft University of Technology, Leiden University, and University of Utrecht. The cluster at the Vrije Universiteit contained 72 nodes, the other four clusters had 32 nodes (200 nodes with 400 CPUs in total). Each node contained two 1-GHz Pentium-IIIs, at least 1 GB RAM (1.5 GB for the nodes in Leiden and UvA, and 2 GB for two larger nodes at the VU), a 20 GB local IDE disk (80 GB for Leiden and UvA), a Myrinet interface card, a Fast Ethernet interface. The nodes within a local cluster were connected by a Myrinet-2000 network, which is used as high-speed interconnect, mapped into user-space. In addition, Fast Ethernet was used as OS network. The five local clusters were connected by the Dutch university Internet backbone.

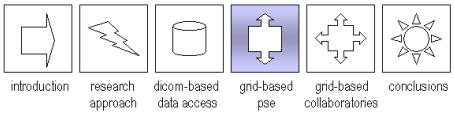
4.4.2 Actor Analysis

We approach architecture analysis of this Grid-based system by identifying the hierarchical components to build the system prototype, as well as the interfaces between such components. We take a system perspective that reflects the concurrent nature of Grid systems, and use a graphical model for the representation of Grid software architecture, which is based on the actor model discussed in *chapter 2*.

Actor-based Grid Architecture

The actor representation of the interactive, simulation-centric biomedical system is shown in Figure 4.4. We identify three actors that form the system or *stage*: *Interactor*, *Simulator* and *Visualizer*, communicating through messages via their input/output interfaces. The Interactor receives external input from the user-in-the-loop, interacting with the Simulator and the Visualizer via the *simulate* and *visualize* interfaces, while at the same time receiving feedback from the Visualizer via the *render* interface.

Furthermore, the Simulator is shown as providing an interface to the Visualizer for transfering the output from the computation. Since our focus is on the representation of the interactive characteristics of biomedical systems, we treat the Simulator and Visualizer kernels as black box components, and concentrate on the Interactor.



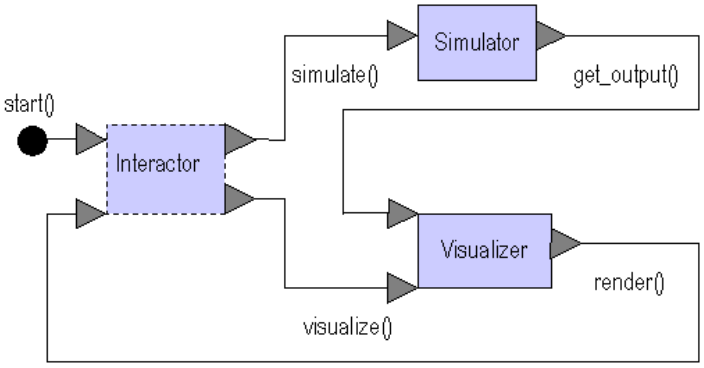


Figure 4.4: Representation of a high-level interactive, simulation-centric biomedical application; *Interactor*, *Simulator* and *Visualizer* actors exchange processes, where Interactor and Simulator may access Visualizer concurrently to allow for greater levels of Grid interaction

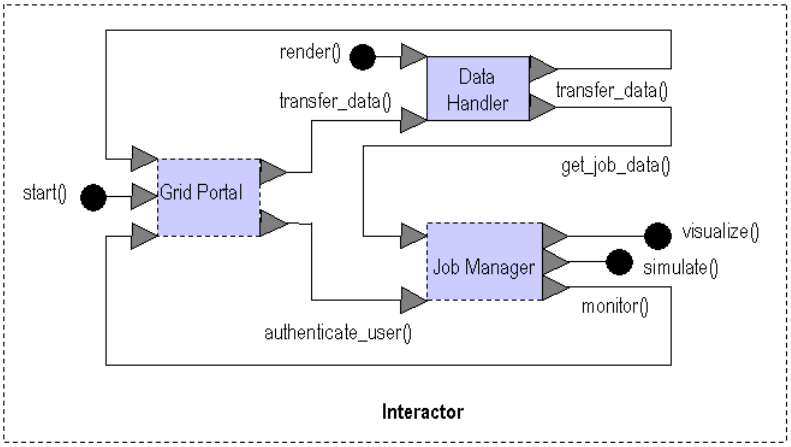


Figure 4.5: Expansion of the *Interactor* actor from the previos diagram; the *Data Handler* actor, that may perform image rendering and pre-data management services, interacts with the *Grid Portal* and *Job Manager* actors

As can be seen in Figure 4.5, the Interactor is composed by three actors: *Grid Portal*, *Data Handler* and *Job Manager*. Both the *Data Handler* and the *Grid Portal* receive external input from the upper level of actor containment in the form of security certificates, image preparation directives, and so on. *Grid Portal* communicates with *Data Handler* via an ouput interface for data transfer, and to *Job Manager* via another interface for user authentication, while getting messages from both via an input interface from the *Data Handler* and the *Job Manager*. The *Data Handler* offers an interface to the higher level Visualizer actor for image rendering, while communicating with the *Job Manager* for data access. The Job Manager communicates with the higher level actors providing the output interfaces for starting the simulation job and output visualization.

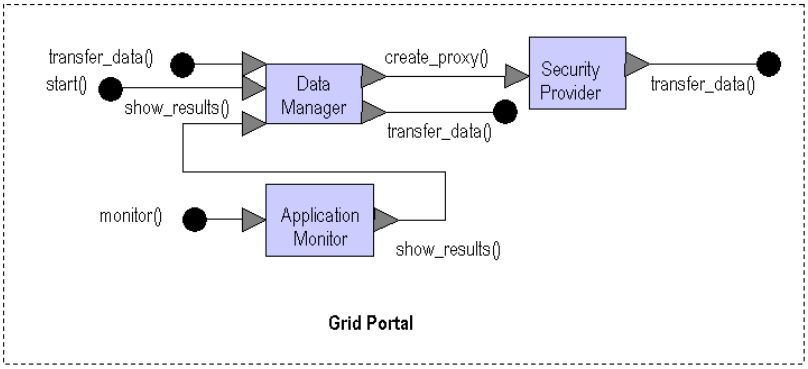
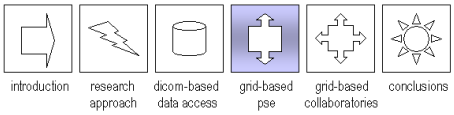


Figure 4.6: Detailed *Grid Portal* actor, which provides Grid security, data management and application monitoring services to the system

The *Grid Portal* actor is composed by the *Data Manager*, *Application Monitor*, and the *Security Provider* actors, as shown in Figure 4.6. *Data Manager* offers external input interfaces to the upper level actors for data transfer and initialization, and to the *Application Monitor* for online monitoring, while offering output interfaces to *Security Provider* for proxy creation, and to the upper level for data transfer.

The Security Provider offers an output interface to the higher level Job Manager for data transfer as well. Also, *Application Monitor* offers a single input interface to the upper level for initialization.

The Job Manager actor is composed by the *Infrastructure Monitor* and the *Job Submitter* actors (Figure 4.7). It is the Job submitter that takes the input



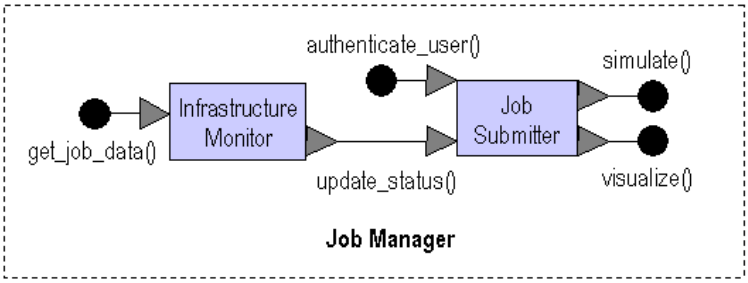


Figure 4.7: *Job Manager* actor, providing Grid job submission, logging, and bookkeeping services via its *Job Submitter* and *Infrastructure Monitor* actors

from higher level actors, via interfaces for upper level simulation and visualization service initialization, while offering input interfaces to the *Infrastructure Monitor* for status update and to the upper level for user authentication. Finally, the *Infrastructure Monitor* offers the *Job Manager*'s input interface to the upper level, for job data transfer.

Actor Signatures

The actors' port signatures are based on the XML Schema actor interface definition described in *chapter 2*, for matching of composite actor signatures and basic structural data typing between actor ports. A code fragment is shown in Figure 4.8. There, a *stage* is defined, containing actors *simulator*, *visualizer* and *interactor*. The composite *interactor* contains actors *data handler*, *job manager*, and *grid portal*. The composite *grid portal* is shown as containing actors *data manager*, *security provider*, and *application monitor*. The figure also shows some of the actors' signatures, such as *grid portal*'s and *data manager*'s matching in-Port *transfer_data* and *start* abstract interfaces.

Here, as in *chapter 3*, we use the actor model of computation for the representation of concurrent behaviour in order to identify patterns of distributed component behaviour across levels of complexity.

4.4.3 Architectural Instantiation

We next present a high-level UML model of the Grid architectural design, based on the previous actor model, in order to map component interaction

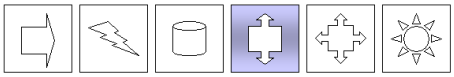
```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <stage xmlns="http://staff.science.uva.nl/~alfredo/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:schemaLocation="http://staff.science.uva.nl/~alfredo/actor1.xsd">
4     <actor type="composite" name="interactor">
5       <actorSignature>
23      <actor type="simple" name="data handler">
43      <actor type="composite" name="grid portal">
44        <actorSignature>
45          <inPorts>
46            <inPort name="start">
49            <inPort name="transfer data">
52            <inPort name="monitor">
55          </inPorts>
56          <outPorts>
57            <outPort name="transfer data">
60            <outPort name="authenticate user">
63          </outPorts>
64        </actorSignature>
65        <actor type="simple" name="data manager">
66          <actorSignature>
67            <inPorts>
68              <inPort name="transfer data">
71              <inPort name="render">
74              <inPort name="show results">
77            </inPorts>
78            <outPorts>
79              <outPort name="create proxy">
82              <outPort name="transfer data">
85            </outPorts>
86          </actorSignature>
87        </actor>
88        <actor type="simple" name="security provider">
105       <actor type="simple" name="application monitor">
122     </actor>
123     <actor type="composite" name="job manager">
146   </actor>
147   <actor type="simple" name="simulator">
161   <actor type="simple" name="visualizer">
178 </stage>
```

Figure 4.8: Actor port definitions: code fragment of the actor's port signatures for the CrossGrid biomedical application actor architecture, as described in *Figures 4.4 - 4.7*

to actual tools and infrastructure as provided by CrossGrid. Here, we focus on a conceptual level of architectural design, providing intuitive abstract component and interaction diagrams for the instantiation of the highly distributed components. We underline architecture scalability and communication among stakeholders, which can be used as a basis for subsequent actor-based architectural analysis.

Generic Components

We distinguish three different granularities of architectural views: Conceptual, Logical and Execution viewpointsas proposed by Bredemeyer et al. [37]. We work on the conceptual high level of design, as a basis for later interface



specification and representation done at the logical level, and the validation at runtime level of design.

We analyze the system mostly from the interaction-related viewpoint, decomposing the system into key constructs related to components, relationships and interactions, without zooming into interface specification details. At this level we aim to capture the architectural vision, and understand high-level relevant system functionality. To this end, we identify three first level abstractions for designing simulation-centric interactive biomedical applications: *Simulation*, *Visualization* and *Interaction*. In this work we concentrate mostly on the *Interaction* system, as mentioned before, and treat the *Simulation* (Biomedical Solver) and *Visualization* (Visualization Kernel) systems as black boxes: the simulation subsystem is composed of the simulation application, which is loosely-coupled with the other components, and the visualization component, realized as a visualization kernel running somewhere on the Grid infrastructure that receives the simulation's output. We elaborate on these issues in the integration discussion.

Based on the previous actor analysis, we dissect the interaction component functionality as containing three main components: *Biomedical Data Handler*, *Grid Portal* and *Job Manager*:

- the *biomedical data handler* is a component used by scientists to apply pre-processing on the datasets used by the simulation component. Once such data preparation has taken place, it submits the simulation job via the *Grid portal* component. Also, it is used for rendering of the output from the *visualization* component,
- the *Grid portal* component provides secure data access and job monitoring via its *data management*, *Grid security infrastructure*, *job submission* and *application monitor and benchmarker* services,
- the *job manager* component is composed of *job management* and *infrastructure monitoring* services, for the submission of computational jobs and resource monitoring.

We identify the high-level flow of information between system components by means of abstract message interactions within the running system. For instance, the sequence diagram in Figure 4.9 shows typical dynamic interactions among components in the following process flow.

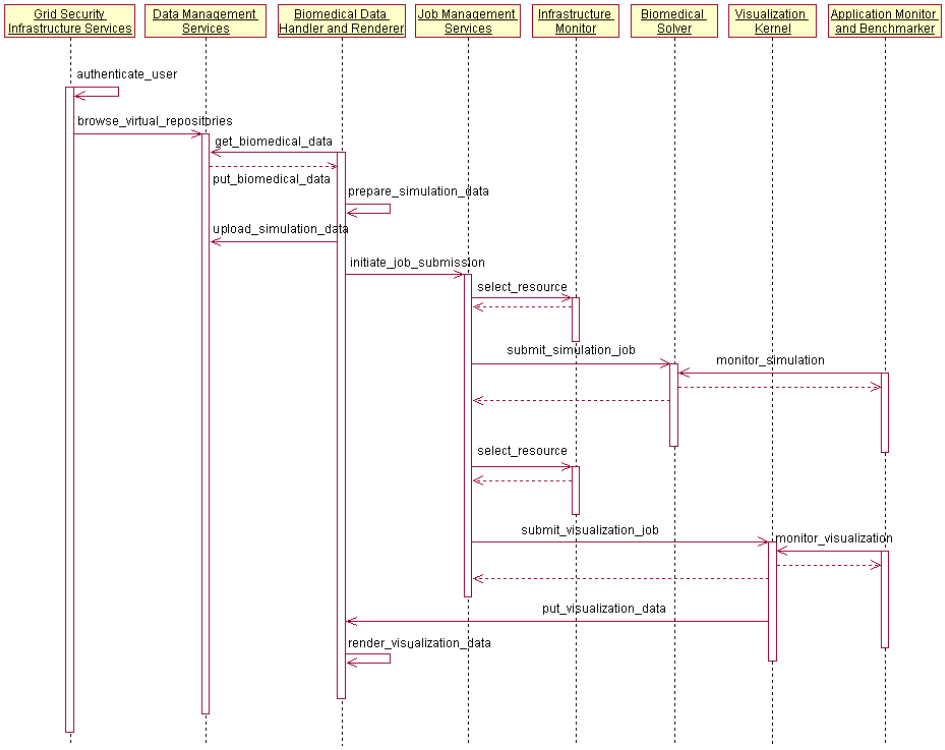
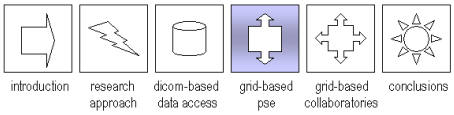


Figure 4.9: Conceptual interaction diagram: detailed communication among abstract components, with the workflow initiated by the *Grid Security Infrastructure* component and finalized by the *Biomedical Data Handler* component

Component Instantiation

We mapped the actor-based architectural design to the specific VRE biomedical application, and the Grid infrastructure offered by CrossGrid middleware and interactive services (Figure 4.10). The VRE application for blood flow simulation is used, which is loosely coupled with the CrossGrid Grid Visualization Kernel (GVK) [103] and the Grid portal components, via the portal's backend roaming access service component. We use the VRE rendering facilities, the DesktopVRE and Personal Space Station (PSS) [122] loosely coupled with the GVK, RB and Globus and Datagrid data management components such as GridFTP and Replica Locator Service (RLS).



The portal allows the use of Public Key Infrastructure (PKI) security, based on Globus GSI and other components such as Virtual Organization servers (VOServer). Both application monitoring and benchmarking via CrossGrid's Grid Performance Monitoring (GPM) and GridBench applications, as well as other infrastructure services. For further details on integration and validation, we refer the reader to [154].

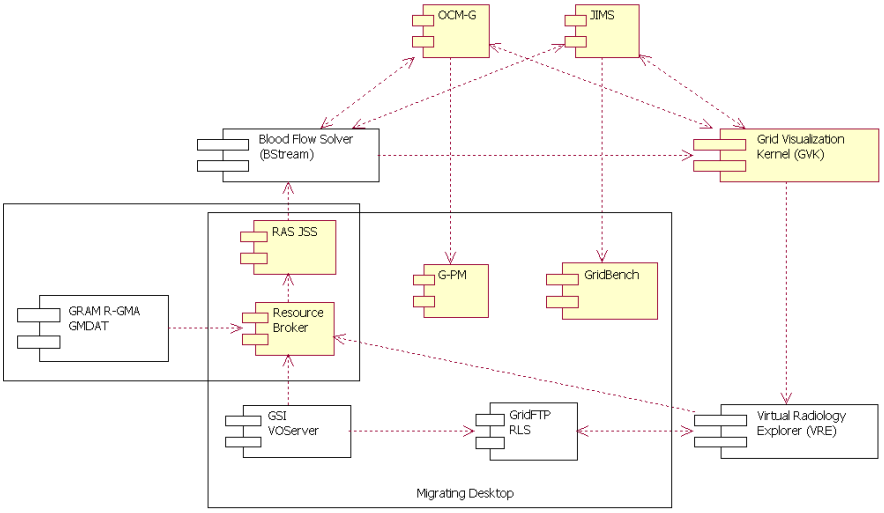


Figure 4.10: Mapping of the Grid architecture to the VRE application and CrossGrid infrastructure. The CrossGrid Migrating Desktop portal offers a number of components such as a virtual organization server, Globus GridFTP, EDG Replica Location Service and Resource Broker, as well as back-end monitoring and benchmarking services. Our simulation solver is loosely integrated with the portal's Roaming Access Server and Grid Visualization Kernel

4.5 Grid-based Prototype

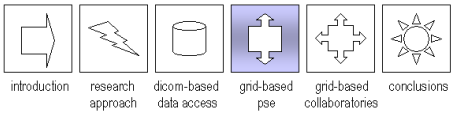
Once we designed and specified the Grid architecture, we consider the following scenario, as described in Figure 4.11: a patient walks into a medical center scanning room in Leiden to get his blood flow measured, the technician scans the abdominal aorta area, and the resulting image is stored in the

radiology information system repository or local PACS, to be pre-examined and segmented. Later, a physician (user) in Amsterdam logs into the Grid portal using his Grid certificate and private key. The user checks if there are segmented medical data sets ready for analysis, and securely transfers a few to Grid virtual repositories. The user then starts the desktop version of the VRE from within the portal, loads the segmented medical data, selects a region of interest, crops image, adds a bypass, and creates a computational mesh. The user submits the data for simulation on the Grid, to the nearest/most adequate computing element in the computational infrastructure using a replica manager service. The user may then check status of the job submitted via the portal. After the job has been completed, the relevant parameters (e.g., velocities, pressure, shear stress) are transferred to the local storage element or directly to the appropriate visualization service to be rendered and reviewed by the user. All processes are transparent to the user, and components such as the visualization service are loosely-coupled transparently into the process flow [152]. We next elaborate on some of the steps in this scenario, and how we prototyped the architecture.

4.5.1 Medical Image Segmentation and Access

Once medical images are acquired, e.g., by magnetic resonance angiography (MRA), the data is stored in a medical image repository for further analysis. Next, advanced image segmentation techniques are applied: the accurate assessment of the presence and extent of vascular disease requires the determination of vessel dimensions. For this, a method for automatically determining the trajectory of the vessel of interest, the luminal boundaries, and subsequent the vessel dimensions has been developed by the Department of Radiology, Leiden University Medical Center (LUMC) [110]. Relevant 3D structures such as arteries are extracted from the raw data.

The Grid portal enables access to the Grid resources from roaming machines like stand-alone personal computers, notebooks or desktop workstations. It allows running applications, managing data files, and storing personal settings independently of the localization or the terminal type. Users may handle Grid and local resources, run applications, manage data files, and store personal settings. The portal provides a front-end framework for embedding some of the application mechanisms and interfaces, and facilitates the user virtual access to Grid resources from other computational nodes. Access to the testbed is based on globus Grid security infrastructure (GSI). GSI uses public key encryption, X.509 certificates, and the secure sockets layer (SSL)



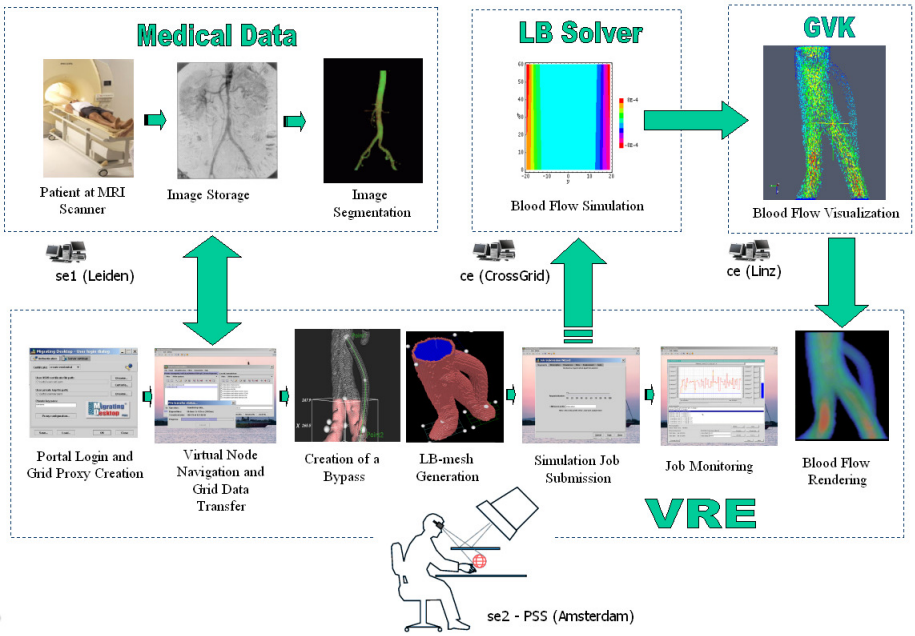


Figure 4.11: Data and process flow in the VRE environment, where a Grid-based virtual simulated environment is used to access medical image data, visualize it, and explore patient vascular condition; here the Grid-based process flow allows for natural mapping to the user process flow

communications protocol. Extensions to these standards have been added for single sign-on and delegation. The GSI provides a delegation capability, with an extension of the standard SSL protocol to reduce the number of times the user must enter his pass phrase, as done withiin the CrossGrid portal. If a Grid computation requires several Grid resources, or if there is a need to have agents requesting services on behalf of a user, the need to re-enter the user's pass phrase is avoided by creating a proxy.

The segmentation step is connected to the DesktopVRE-based reconstruction of a 3D model of an artery. Furthermore, geometrical modeling tools allow the interactive manipulation of medical images for pre-processing such as clipping, editing of LBM mesh, handling of problematic areas and interactive placement of a bypass (Figure 4.12).

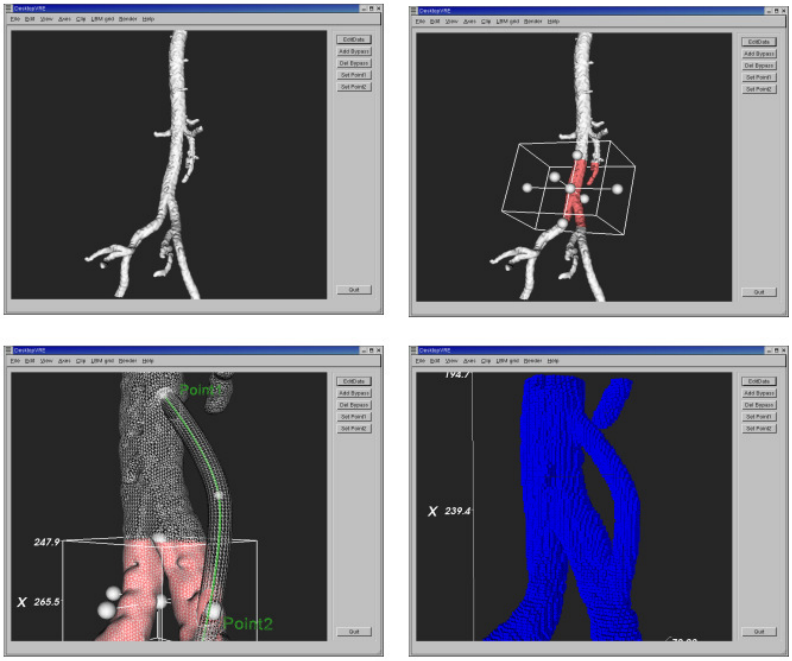
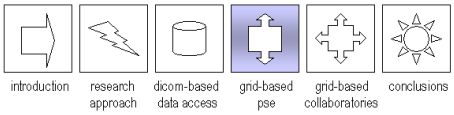


Figure 4.12: Segmented data rendering, bypass creation, and Lattice-Boltzmann mesh creation in the 2D Desktop Virtual Radiology Explorer; the LBM mesh editing also allows indicating boundary conditions

4.5.2 Virtual Browsing and Data Access

The portal uses the roaming access service, a set of back-end portal services that are used by the portal front-end. The access service is also responsible for communicating with other modules, and offers a set of services such as a lightweight directory access protocol manager service responsible for storing the user profiles, a job submission service that provides an interface that makes accessible the submission mechanisms, an scheduling agent, and a session manager service responsible for managing application-user session. Data transfer in the testbed is based on Globus GridFTP [6], a common data transfer and access protocol that provides secure, efficient data movement in Grid environments.



4.5.3 Job Submission and Infrastructure Monitoring

Within the portal, application-specific information can be described using XML schema. In order to integrate visualization libraries into the computational Grid testbed, we created and posted application XML schemata for job submission, to be dynamically linked to the portal. Then XML style sheet transformations (XSLT) were used in order to transform the schemas into appropriate XHTML. The portal sends the job request to the access service, which then is sent to a job submission service, which then submits the job to a resource broker (RB) and logs all operations. The RB starts the job on the target computing element. Before the job is started, though, a job submission script downloads all necessary files for simulation from a virtual node. Within the portal, we used the EDG replica manager, which allows one to copy files into Grid storage, register files, replicate files between SEs, delete individual replicas, and delete all replicas of a particular file.

Grid monitoring is included as services for applications as well as for instruments and infrastructure. Application monitoring is different from monitoring infrastructure, so separate approaches were available in CrossGrid, with application monitoring aimed at observing a particular execution of an application. The collected data is useful for tools for application development support, which were used to detect bugs, bottlenecks or just visualize the application's behavior, in the context of a particular execution. For our purposes, we use the main Grid portal and CrossGrid light-weight portal capabilities for monitoring job submission (Figure 4.13).

We also worked on the integration of more advanced application monitoring and performance prediction tools offered by the CrossGrid, as well as fine-grain infrastructure monitoring to allow for more interactive usage of collected information. The application monitoring infrastructure developed in the CrossGrid is the Grid-based OMIS-compliant Monitoring (OCM-G) [17], a distributed descentralized, autonomous system that runs as a permanent Grid service. It provides monitoring services accesible via a standardized interface, to be used by visual tools such as Grid-Performance Monitoring (GPM) [41]. We used GPM extensively to define our own performance metrics, to access a set of fixed available ones, and to handle the process of measuring the performance properties. We also used GridBench [156], a tool developed to manage benchmarking experiments, publish their results, and produce graphical representations of their results.

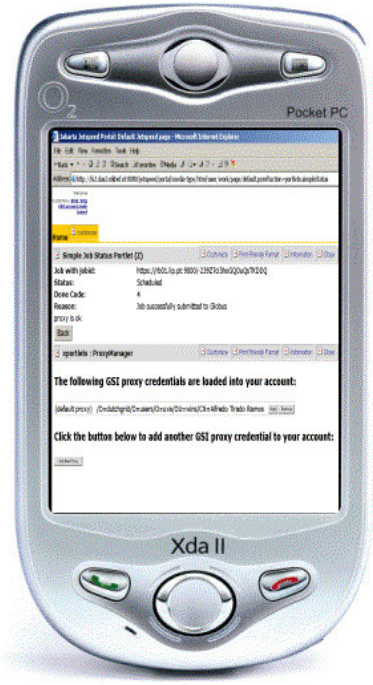
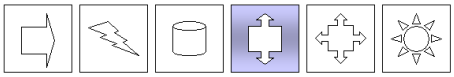


Figure 4.13: Simulation monitoring via the CrossGrid light-weight portal, based on an Enterprise Information Portal, using Java and XML, and interfacing with a light-weight personal digital assistant (PDA)

4.5.4 Blood Flow Visualization and Rendering

After jobs are completed, the output is redirected to Grid visualization tools either to anywhere in the network or to a specific CE where the simulation has run, to avoid large data transfers. We address the combination of Grid applications and corresponding visualization clients on the Grid. While Grids offer a means to process large amounts of data across distant resources, visualization aids in understanding the meaning of data. For this reason, visualization capabilities use Globus services, thereby providing Grid visualization services via dedicated interfaces and protocols while at the same time exploiting the performance of the Grid for visualization purposes.

A resource intensive module of the visualization pipeline is instantiated on a high-performance computer. Then, the visualization pipeline on a graphics workstation connects (via re-direction through the portal service) to this mod-



ule, uses the power of the high-performance computer to generate the visual results, and downloads them to the visualization device. We created links within the Grid portal for initialization of the visualization client application, and experimented with rendering the flow both remotely and locally in the access storage element. This way, remote visualization and local rendering were fully linked via the portal, for final rendering in 2D or 3D rendering work stations.

4.6 Results

In this section we present results after prototyping the Grid architecture for an interactive biomedical application and deploying it on the CrossGrid testbed infrastructure. We begin the discussion with a *Computational Grid* analysis using the application benchmarking and online monitoring experiments from the viewpoint of resource selection, followed by a *Data Grid* analysis using data access experiments from the viewpoint of integration with Grid services like visualization.

4.6.1 Grid Interoperability

We study biomedical data transfer and job submission among these infrastructures, focusing on biomedical dataset transfer times, CPU usage overhead, as well as job submission using both intra and inter cluster computational runs. We discuss Grid architectural interoperability in the context of the Open Grid Service Architecture (OGSA), and service semantics issues. Even though we focus mostly on Grid infrastructures and services based on the LCG model in this thesis, we experiment with OGSA infrastructure interoperability in the context of future extensions to the results. Our experimental Grid infrastructure is based on and supported by the CrossGrid project's testbed, including resources spread across Europe, which range from relatively small computing facilities in universities to large research computing centers.

Framework Interoperability

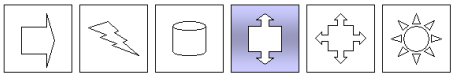
The CrossGrid testbed largely inherits from the European DataGrid (EDG) [89] experience on setup and it is fully based on LCG middleware distributions for services. The basic security requirements are covered by the Grid Security Infrastructure (GSI) and its public key services, so the priority is to

make sure that biomedical data access from Storage Elements (SEs) remains reliable, and that the data management services do not overwhelm the resources. The focus is on allowing users transparent data access for analysis and computational job submission, acccording to a basic usage scenario. The user of the application loads the biomedical medical data, selects a region of interest, and submits the analyzed data for simulation. After the job has been completed, the results are transferred to local machines for visualization and rendering. This scenario assumes a number of data transfers, both input segmented data for the simulation application, as well as simulation output to be visualized. In the LCG-based infrastructure we use Globus GridFTP and DataGrid Replica Locator Service (RLS) to transfer data between Grid nodes. We set out to replicate some characteristics of this behaviour between the testbed and a few newly added GT4 nodes. The recent addition of new resources to our Grid infrastructure provides us with increased computational capacity. However, evolution in Grid middleware raises issues of structural interoperability, specially when migrating from established architectural paradigms to new ones.

In the current Grid infrastructure, which relies on the CrossGrid middleware, we make use of both MPICH-P4 and MPICH-G2 MPI devices to run the application. Support for MPICH-P4 is available for more recent Grid production middleware releases; MPICH-G2 support is usually known to be unstable, at best. In order to explore the support for both MPICH-P4 and MPICH-G2, we proceed to execute and measure the performance of the application on three different platforms. These three platforms include the current infrastructure (LCG2.0 for production and LCG 2.3 for development versions), the LCG 2.6 production middleware release of the EGEE project, and the Globus Toolkit 4. We use globus-job-run and globusrun-ws, where available, for the MPICH-P4 runs, while MPICH-G2 runs were initiated directly by Globus Resource Specification Language.

We transfer a number of representative compressed input datasets between new GT4 and CrossGrid LCG SEs spread out across Europe, measuring transfer times for comparing performance of the GridFTP implementations. Once we successfully transfered the input geometry data files between GT4, LCG2.0, LCG2.3 and LCG 2.6 resources, we then made a number of transfers using larger simulation output files, studying CPU usage to get an idea of the overhead caused by the transfers.

In those experiments, the LCG2.0 machine performed significantly faster due to superior hardware specifications. The total iteration time of the LCG 2.6 machine was slightly lower than that of the GT4 machine, but the total



execution time tended to be equal or slightly higher.

4.6.2 Grid Data Access

We next describe data access experiments between the CrossGrid portal and the Grid Visualization Kernel (GVK) [82] for blood flow visualization. We focus on the initial integration into the interactive, simulation-centric biomedical application running on the Grid. GVK is a middleware developed at GUP Linz within the European CrossGrid project, which aims to enable the use of visualization services within computational Grids. The idea of GVK is to encapsulate modules of an arbitrary visualization pipeline and distribute these modules to a number of computing elements across the Grid. By interconnecting these modules, the visualization environment takes advantage of the benefits offered by the Grid, such as location independence and efficient network transportation. We test virtual node creation, job submission and flow visualization via Grid technologies based on Web Services, and experiment with secured file transfer and flow visualization.

The GVK addresses the combination of Grid applications and corresponding visualization clients. The visualization capabilities of GVK are implemented using Globus services, thereby providing flexible services via dedicated interfaces and protocols while at the same time exploiting the performance of the Grid for visualization purposes. We extend the work on medical data simulation and visualization to the Grid via the CrossGrid Grid portal.

The Grid portal is an application that offers a seamless interface which is independent of software and hardware environments, on which applications and resources may be highly distributed. The portal rests on a set of back-end portal services which are used by the portal front-end. These access services are also responsible for communicating with other modules, and offer a set of services such as an LDAP manager service responsible for storing the user profiles, a job submission service, an scheduling agent, and a session manager service responsible for managing application-user session. We integrated local visualization workstations and mesh creation applications with the Grid visualization service, configuring all relevant libraries, and used sites registered within the VO, such as Amsterdam and Leiden, for testing secure GridFTP [6] data transfer between storage and computing elements.

It is important that, when prototyping the system, the latency shown by the Grid transfer of datasets was comparable with common data transfer. We set out to perform a set of experiments for the transfer of segmented medical datasets, ranging from 24 KB up to approximately 5 MB loads. When com-

paring the transfer times of the medical data, we found that average transfer times, once taking into account the Globus caching mechanism (GASS manipulation of local file caching), did not vary much above 400ms for the smaller size files and no more than 850ms for the larger size files, which is comparable to regular FTP functionality (Figure 4.14).

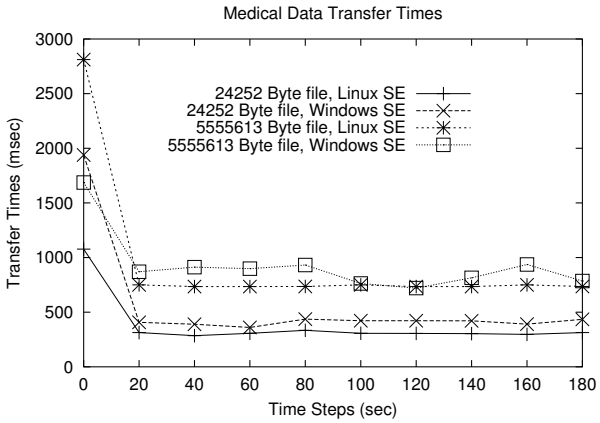


Figure 4.14: GridFTP transfer times to roaming access nodes; transfer times of the data, at time steps of 20 seconds, showing average transfer times to roaming storage elements running nodes, once taking into account the Globus caching mechanism, not varing much above 200 milliseconds for the smaller size files and no more than 350 miliseconds for the larger size files

We found during the initial efforts that integration of the testbed and visualization libraries are not a trivial task, but the added functionality and security infrastructure offered by Grid technologies came at a minimal performance cost.

4.6.3 Grid Application Benchmarking

The fact that jobs fail frequently on the Grid has has been widely documented [32, 161]. There are many reasons for this, like having high levels of resource diversity with varying performance at runtime [119]. One of the main reasons, though, is the fact that resource broker components, which are part of job submission services, are normally implemented to take a rather naive aproach to resource selection, affecting the selection process and therefore the ratio of

submitted against aborted jobs, as shown in Figure 4.15. There is interesting work on extensions to the classic benchmarks [72], low-level probes [51], as well as Relational GMA [52], an extension to the widely used Grid monitoring architecture accepted as a standard for Grid monitoring systems.

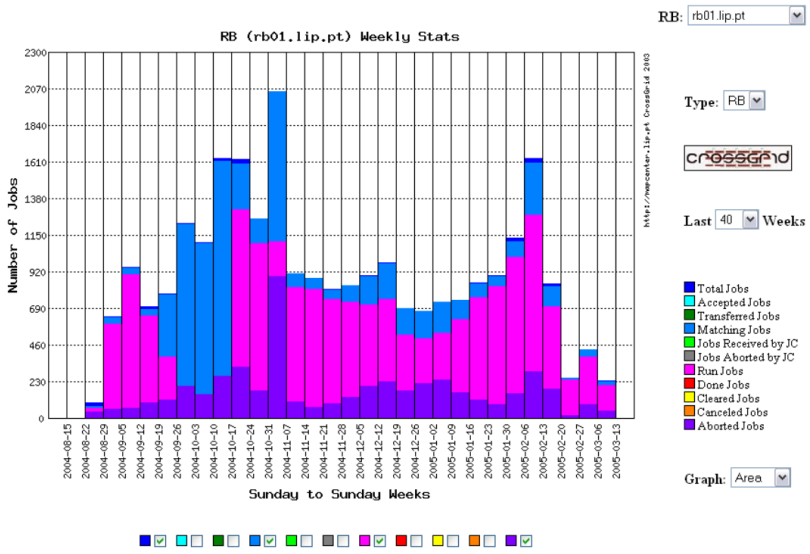
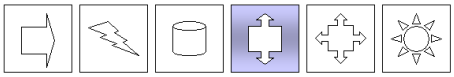


Figure 4.15: Statistical information about Grid Computational and Storage Elements, from the CrossGrid Resorce Broker, showing CEs' SpecInt2000 values, published by the CrossGrid Information Index, showing total number of jobs submitted, matched and run jobs by the CEs, and jobs aborted; the rank is computed by taking the average rank of the clusters, weighted by the number of free CPUs they contribute

Objectives

Resource selection in Grid environments is a crucial problem. Regardless of who performs the resource selection, be it users or automated systems, the decision makers are faced with the difficult task of matching/mapping jobs to resources. Previous work on the specification of resources and services in complex heterogeneous computing systems and metacomputing environments in general [39] and particularly in Grid environments, has led to a better understanding of the issues.



The impact of the evolution and wide acceptance of Grid architectures underlines the need for addressing and validating the application-specific characterization of available resources. For instance, Grid application benchmarking, or the characterization of Grid computational resources for improving resource selection, can be used to help improving the performance of computationally intensive parallel applications by enhacing the resource selection process. Since the application is often run in multiple instances using parameterised runs, it would be desirable to have acces to information that would help to better schedule these jobs.

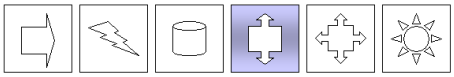
We investigate the levels of performance offered by hardware resources distributed across the CrossGrid European computational Grid for the biomedical application and show how to rank resources based on a benchmark derived from the blood-flow simulation. For the experiments, we use a Grid benchmarking framework developed within the CrossGrid project, the GridBench tool.

The GridBench Tool

GridBench [156] consists of a framework containing a set of components that aim to facilitate the characterization of Grid nodes or collections of Grid resources. The framework has two main objectives: to generate metrics that characterize the performance capacity of resources belonging to a Virtual Organization (VO), and to provide a tool for researchers that wish to investigate various aspects of Grid performance using well-understood applications that are representative of more complex applications deployed on the Grid. In order to perform benchmarking measurements in an organized and flexible way, GridBench provides a means for running benchmarks on Grid environments as well as collecting, archiving, and publishing the results. The framework allows for convenient integration of new and existing benchmarks into the suite, as well as the customization of existing benchmarks through parameters. We have used the tool extensively to perform the biomedical application benchmarking experiments.

Resource Comparison

The VRE biomedical application involves heavy processing of 3D data, which makes it computationally expensive. Figures 417 - 419 show the computationally intensive part of the application, which uses the Message Passing Interface (MPI) for parallelization. This code is instrumented to measure elapsed



time for each iteration as well as the time spent on MPI communication, and integrated into GridBench. As a dataset we used different sample files that represent normal workload, from simple tube-like artery structures to aorta segments containing bifurcartions.

For the experiments, we have set the biomedical application to run for 800 iterations, so it can be seen that right before the end of each run (at around 760 to 780 iterations) a jump in performance of about 30% larger time per iteration values is experienced in all nodes (Figure 4.16). This is mainly due to the design of the current version of the application, where the first processor that started running gathers data from all other processors before producing the final output. Nevertheless, iteration times remain relatively invariant regardless of the number of iterations. For this reason it is reasonable to assume that short run-time experiments (using a small number of iterations) are representative of real-life experiments, in which we use larger iteration counts. Figure 4.16 also shows the performance of the application at a set of sites using 4, 8 and 12 CPUs respectively. Generally we observed a downward trend indicating that the code is somewhat scalable, i.e. using a larger number of CPUs at a given site yielded a faster run-time (Figure 4.17).

Communication Measurements

The biomedical application used MPI for inter-proces communication, which we compiled using the MPICH4 device. The code is highly coupled and we expected that the performance of the interconnect, i.e., the LAN connecting the cluster nodes had a considerable impact on the performance of the application. To investigate this, the biomedical application code is instrumented to measure the time spent in communication*.

To isolate the effect of the network we ran the code using just two CPUs on a dual-CPU Worker Node (1x2)†, using two CPUs on two different (identical) Worker Nodes (2x1). This is shown in Figure 4.18. There, we show a similar experiment using 4 CPUs, whereas the other figure shows that there is considerable difference in communication performance, which also impacts the time per iteration. On the other hand, in the first plot we observe no significant difference when running in either mode, since the network is used is both cases (both in 2x2 and in 4x1).

*The impact of the instrumentation was measured and found to be insignificant.

†The MPI library used was not optimized for SMP, and communication still went through the TCP/IP stack.

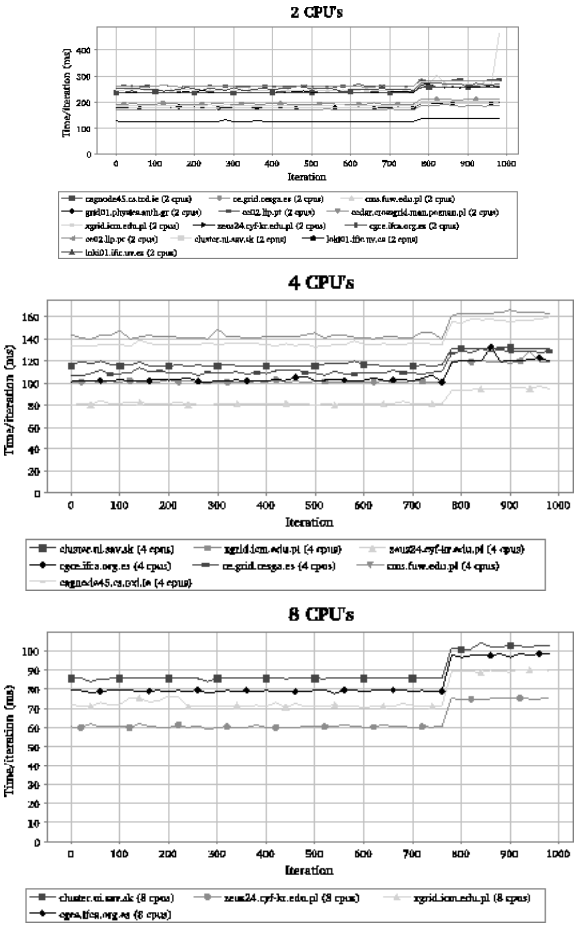


Figure 4.16: Performance of the biomedical application, running for 1000 iterations, at a set of distributed sites using 2, 4 and 8 CPUs. In each case, the same workload is applied by using identical input data and parameters. It shows the results obtained by using 2 CPUs in each measurement. Also, for 2 CPU measurements, using 2 CPUs on the same Worker Node is preferred over using two CPUs on two different Worker Nodes. This is important, since it is found that this would seriously impact performance of this application, as seen in Figure 4.18. Resources *cluster.ui.sav.sk* and *loki01.ific.uv.es* employ single-CPU nodes while the majority of site employ dual-CPU Worker Nodes

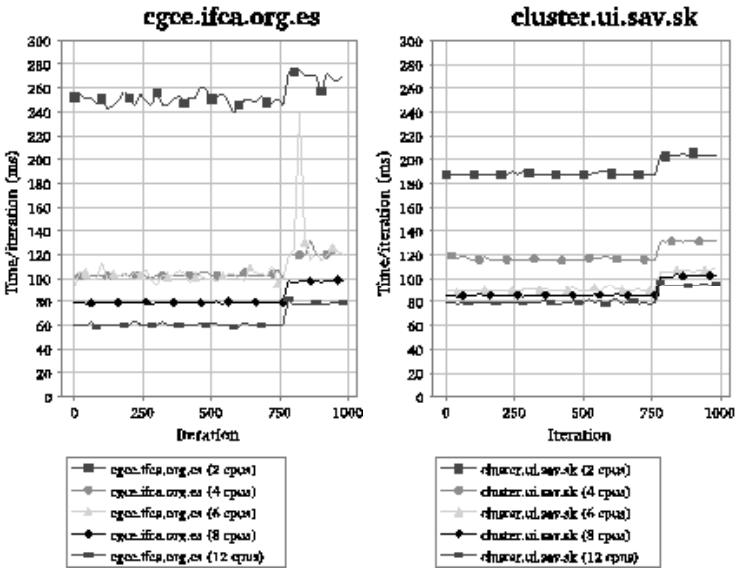


Figure 4.17: Benchmarking the biomedical application at two sites: *cgce.ifca.org.es* (up to 12 CPUs) and *cluster.ui.sav.sk* (up to 12 CPUs). It is quite interesting to observe that the different sites display a different scalability. For example, the *cgce.ifca.org.es* site shows that the runtime is reduced to less than 30% when going from 2 CPUs to 8 CPUs, while *cluster.ui.sav.sk* shows that the improvement is only just under 50%. Similarly, while in *cgce.ifca.org.es* there is approximately a 25% improvement in runtime when going from 8 CPUs to 12 CPUs, in *cluster.ui.sav.sk* there is only marginal improvement. Scalability *at each resource* needs to be taken into consideration for efficient resource selection

Decision-making from Benchmarking Results

Figure 4.19 conveys usefull information since it provides a ranking of run times on all of the resources available. This ranking could be used directly in resource selection *specially* in cases where the relative CPU, Memory and network speeds at each resource (site) are not known. For example, it appears that it is better to run the code at *zeus24.cyf-kr.edu.pl* using 4 CPUs than at *cluster.ui.sav.sk* using 8 CPUs.

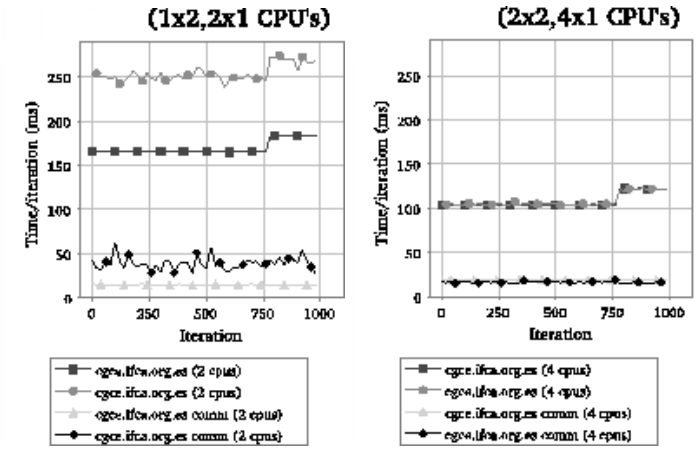
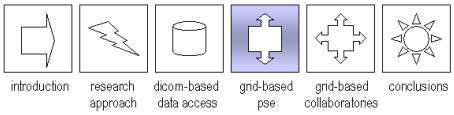


Figure 4.18: Effect of MPI communication on runtime; shows iteration and communication times using 2 CPUs on the same (dual) Worker Node (1x2), and 1 CPU on each of 2 Worker Nodes; the second plot shows iteration and communication times using 2 CPUs on each of 2 (dual) Worker Nodes (2x2), and 1 CPU on each of 4 Worker Nodes (4x1)

4.6.4 Online Performance Monitoring

Distributed computational resources should provide the required performance for large-scale simulations, complex visualization, and collaborative environments which are expected to become of major importance to different areas of medicine [38]. The evolution of Grid architectures has underlined the need for addressing application specific, on-line monitoring of the resources available to scientists. On-line performance monitoring in Grid environments is a very useful technique [40, 81], particularly when dealing with interactive applications that include a human expert in the loop. Regardless of who performs the application optimization, users can greatly benefit from up-to-date knowledge of Grid resource performance (as opposed to post-mortem analysis) in order to make the difficult task of matching/mapping jobs to resources a more efficient one, or to identify application performance bottlenecks.

Connectivity between distant locations, interoperability between different kinds of systems and resources, and high levels of computational performance are some of the most promising characteristics that the Grid offers for biomedical applications. We instrumented the VRE biomedical application to work seamlessly with the CrossGrid online monitoring infrastructure. It is



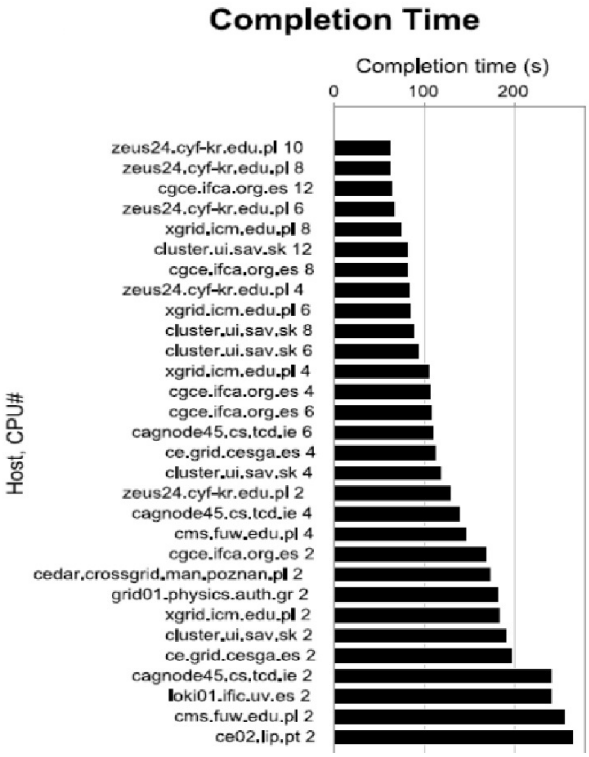


Figure 4.19: Completion times of the biomedical application using different numbers of CPUs on several resources

an important requirement for us that the initialization of the online monitoring tool we use does not interfere with normal job submission to the testbed, and the Grid Performance Monitoring tool (G-PM) [41] fills this requirement flawlessly by virtue of its loosely-coupled integration with the testbed.

Grid Monitoring

The monitoring infrastructure developed in the CrossGrid is the Grid-based OMIS Compliant Monitoring (OCM-G) infrastructure [18], a distributed decentralized, autonomous system that runs as a permanent Grid service. It provides monitoring services accessible via a standardized interface, to be used by visual tools such as G-PM. We use G-PM extensively to access a set of fixed,

predefined performance metrics, and to handle the process of measuring the performance properties. G-PM allows us to measure application and Grid resource response times and their breakdown, detailing performance data for specific interactions while the application is still running. This is very useful for us to determine performance bottlenecks and computational Grid resource responses to specific loads, allowing us to immediately correlate performance data with the application's behavioral patterns at specific Grid resources.

Integration

We instrumented the application to OCM-G, which meant linking the application with instrumented versions of runtime libraries containing communication and I/O routines. Also we added a few lines of code for more application-specific information to be taken into account for the performance analysis. Once the G-PM graphical interface is running, we can interactively define measurements and select proper visualization modes for the results.

As mentioned before, the blood flow simulation application involves processing of 3D data, which makes it computationally expensive. For the computationally intensive part of the solver, MPI is used for parallelization, via the MPICH-P4 device.

Monitoring Results

We run a set of experiments on a number of computational resources on the CrossGrid production testbed, monitoring computing time, node load, communication delay per iteration, and iteration time at different resources, with different loads. For result visualization, we used an integral mode, aggregating results when relevant, with multicurve graphs. We used linear scale partition, in real time mode, and with variable scales. From experience working with the CrossGrid's Resource Broker (RB), resource selection left to the RB can give us a success rate of roughly 60% (we run around 100 job submission tests for this experiments, with diverse configurations). Nevertheless, once we run performance monitoring tests and ranked the potential target resources to use with the help of G-PM, we achieved near 90% success rate. We rated a number of computational resources according to success rate of job completion, communication delay per iteration, and iteration time, among other metrics (Figures 4.20 and 4.21).

For comparison with RB success rates we submitted simulation jobs manually to those resources that, e.g., offered a higher success rate or less com-

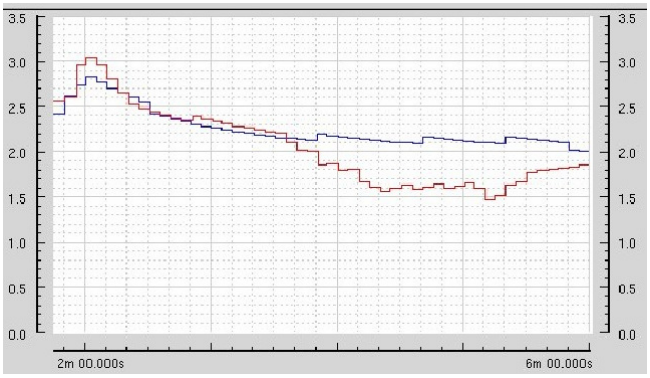


Figure 4.20: Node load, or UNIX load value showing exponential mean of run queue length in the last minute. Here the number of ready processes in 2 clusters (cgce.ifca.org.es, cluster.ui.sav.sk) are pretty much synchronized except for a brief period between t=3m50s and t=5m50s, which gives us similar completion times, useful for comparison of blood flow results with different Reynolds numbers

Figure 4.21: Simulation monitoring of the blood flow simulation in the CrossGrid, via OCM-G/G-PM: graph showing communication delay per iteration or percentage of time which the solver spends in communication routines in one iteration, at timescales of simulation within 7 minutes of running time (x-axis) and delay (y-axis). The figure identifies a two-fold jump in amount of time in communication at t=5m32s, and points out to a possible bottleneck on application performance

pounded delay. We used the tool's defaults aggregation modes for performance visualization, resulting in a single measurement result, which is the aggregate of the results for all selected objects. For instance, in communication delay per iteration, since the processes of the solver are tightly synchronized, it is sufficient to measure the iteration time in one process because it is the same in all processes.

4.7 Conclusions

The prototype of the system's architecture, running within a large Grid virtual computer, was analyzed and a set of benchmarking experiments for specific computation versus communication metrics was performed. We found that ranking resources based on the performance of a stripped-down and instrumented version of an application can give us realistic resource rankings that reflect the performance of the application itself. We have shown how the results obtained using GridBench can be used for ranking of resources and how they can help in resource selection.

In regards to Grid interoperability, we were pleasantly surprised by our relatively smooth experience running the prototype with the MPICH-G2 device simultaneously on both the LCG 2.6 and GT4 middleware, with runs performed on both platforms and providing a competitive total execution time, regardless of the location where the run was initiated. To the best of our knowledge, this is the first work to report successful runs of a real computationally intensive application using multiple nodes featuring different middleware in a production-type test bed.

Moreover, integration of visualization libraries to the computational Grid testbed proved quite smooth; we created and posted application XML schema for job submission, to be dynamically linked to the Grid portal via a job sumbmission wizard and created links within the portal to initialization of both the Grid visualization client and server startup applications, and experimented with rendering the flow both remotely and locally in the access storage element. This way, Grid remote visualization and local rendering were fully linked via the Grid portal.

4.8 Summary

In this chapter we developed an architecture for distributed problem solving environments on the Grid, modeled it, and prototyped it for conducting computer simulation experiments in pre-operative planning of vascular reconstruction with a physician in the experimental loop. This prototype of a problem solving environment offers an integrative approach for constructing and running complex interactive systems. Grid resources are used for access to medical image repositories, segmentation services, simulation of blood flow, and visualization in virtual environments of the simulated results together with medical data obtained from MRI/CT scanners. This complete framework for virtual exploration environment permits a user to explore interactively the visualized results of a simulation, and manipulate the simulation parameters in near real-time. We introduced the generic architectural requirements, defined a generic component-based software architecture and its abtract interactions, and identified a specific use case for validation of the architecture, allowing us to experiment with issues related to data transfer interoperability and performance (Table 4.1).

Table 4.1: Extending the Virtual Radiology Explorer with Grid computing infrastructure and resource virtualization, which enabled unified access to virtual resources, a scalable service-based architecture based on a resource-centric model, near real-time interactivity and the potentially concurrent access to distributed resources within a virtual organization

State of the art	**Contribution**
Problem Solving Environment	*Highly-distributed Problem Solving Environment*
Tightly-coupled client/server model	VO-based loosely-coupled implementation
Component-based architecture	Service-based architecture
Real-time interactivity	Near real-time interactivity
Centralized security model	PKI-based access to virtual resources (data, services, computational, instruments)
Data-centric model	Resource-centric model

5

Virtual Collaboratories*

5.1 Introduction

The challenges discovered when studying humans as complex systems, from a biomedical viewpoint (from cells to interacting individuals), cover the whole spectrum from genome to health and cross temporal and spatial scales [141]. This includes studying biomedical issues using multiscale and multiscience models and techniques all the way from genomics to the macroscopic medical scale (Figure 5.1). This is also complicated by the continuous increase in the amount of digital data produced by modern high-throughput biomedical detection and analysis systems. New experiments in science and engineer-

*The results presented in this chapter formed the basis for the following three papers:

- A. Tirado-Ramos, P.M.A. Sloot, M. Bubak, "Grid-based Interactive Decision Support in BioMedicine", *Grids for Bioinformatics and Computational Biology*, T. El-Ghazali and A. Zomaya Editors, John Wiley and Sons, USA, in press.
- P.M.A. Sloot, A. Tirado-Ramos, I. Altintas, M. Bubak, C.A. Boucher, "From Molecule to Man: Decision Support in Individualized E-Health", *IEEE Computer*, November 2006, vol. 39, no. 11, pp. 40-46.
- P.M.A. Sloot, A.V. Boukhanovsky, W. Keulen, A. Tirado-Ramos, C.A. Boucher, "A Grid-based HIV Expert System", *Journal of Clinical Monitoring and Computing*, October 2005, vol. 19, nr. 4-5, pp. 263-278.

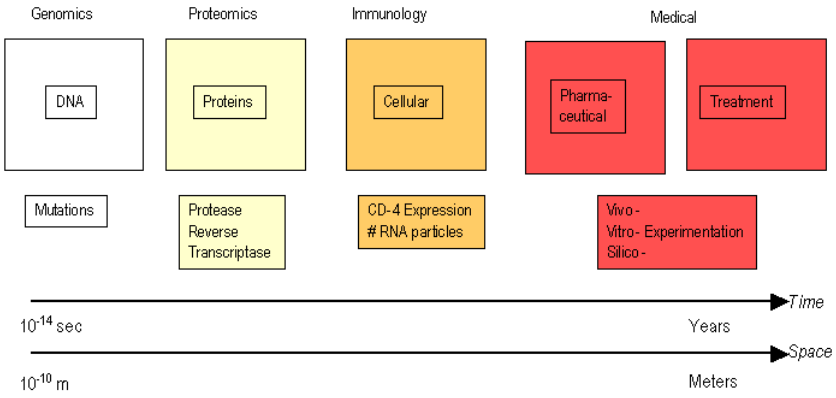


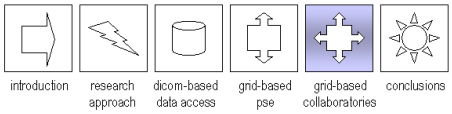
Figure 5.1: Time and space: studying drug response in infectious diseases requires multiscale, multiscience models and techniques to cover the huge spatial and temporal scales [142]

ing will cover the whole spectrum, from the simulation of complete biological systems, to cutting-edge research in bioinformatics.

In this chapter, we analyze the design and early work on a virtual laboratory that enables Grid-based access to highly distributed resources for processing and analyzing virological, immunological, clinical and experimental data. We extend our discussion on the challenges found when addressing software architectures to support multiscience complex systems, from a biomedical informatics viewpoint. For this, we elaborate on a Grid-based framework for large scale collaborative e-Science, the ViroLab, that builds on the lessons learned from the previous highly-distributed system analyses.

5.1.1 Problem Statement

As we have seen in *chapters 3* and *4*, research efforts in biomedical informatics at the macroscopic scale are gradually pushing the boundaries of the state of the art, moving from monolitic software architectures to building more generic components. Such efforts normally leverage object-oriented and distributed component architectures to encapsulate or wrap legacy data in order to improve application interoperability and scalability [117, 151]. This allows



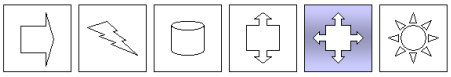
for enhanced data and process flow at the macroscopic level, where models such as DICOM provide support for data acces from work stations to archiving and communications systems and back to hospitals' information systems.

Current distributed computing technologies address communication between tightly-coupled systems very well, though may fail when addressing loosely-coupled resources. Such resources may belong to sites within large distributed virtual organizations that use distributed computing models like CORBA. These technologies allow seamless and secure data access *within a single organization*. Large amounts of data can be distributed across domains, with distributed applications forming federations that may be scaled but that assume architecturally invariant systems. Relational data representation and access, modular component and object oriented models have clearly advanced the state of the art.

New sets of conditions and requirements for software architectures for biomedical applications are emerging in the field, like in systems biology, where genetic information is becoming increasingly significant. This stems from the recent and anticipated achievements in the use of genomics for the understanding of the role of genetic factors in human health and disease. One representative example of applications of genetic information in biomedical informatics is the case of decision support systems for researching drug resistance.

5.1.2 Chapter Organization

We describe an interactive decision support environment, from the perspective of bioinformatics, for a system-level approach to distributed collaborative laboratories. We focus on a complex system centered around a decision support engine for drug ranking in Human Immunodeficiency Virus (HIV) drug-resistance. Our reasons to use this bioinformatics application as main case study in this chapter are twofold: HIV drug resistance is becoming an increasing problem worldwide, with combination therapy with antiretroviral drugs failing to completely suppress the virus in a considerable number of HIV-infected patients. On the other hand, HIV drug resistance is one of the few areas in medicine where genetic information is widely available and has been used for many years. This has resulted in large numbers of data available, not only on complex genetic sequences but on all levels, up to populations. The sheer complexity of the disease, the distribution of the data, the required automatic updates to the knowledge base, and the efficient use and integration of advanced statistical and numerical techniques necessary to as-



sist the physician motivate our research. We elaborate here on some of the possibilities for individualized e-Science that can be supported by virtual collaborative environments based on Grid technology.

5.2 Individualized e-Science

Computer system architectures reflect the same laws and organizing principles used to build individualized biomedical systems, which can account for variations in physiology, treatment, and drug response. Closing the computational gap in systems biology requires constructing, integrating, and managing a plethora of models. A bottom-up, data-driven approach will not work for this: integrating often incompatible applications and tools for data acquisition, registration, storage, provenance, organization, analysis, and presentation can be greatly enhanced by innovative and scalable aproaches like Web and Grid services. Once the computational and integration challenges are adressed, we need a system-level approach to close the collaboration and interaction gap. Such an approach would involve sharing processes, data, information, and knowledge across geographic and organizational boundaries within the context of distributed, multidisciplinary, and multi-organizational collaboration teams, or virtual organizations. These methods dynamically streamline and, most importantly, *individualize* scientific data flow processes depending on their availability, reliability, and the specific interests of medical doctors, surgeons, clinical experts, researchers, and other end users. We call this a *molecule to man* approach [138] (Figure 5.2).

5.3 The ViroLab Collaboratory

During the past decade, researchers have made significant progress in treating patients with viral diseases. Effective antiretroviral therapy has lead to sustained HIV viral suppression and immunological recovery in patients who have been infected with the virus. Long-term therapy can lead to metabolic complications. Other treatment options are now available, with the recent introduction to clinical practice of fusion inhibitors, second-generation non-nucleoside reverse transcriptase inhibitors, and nucleotide reverse transcriptase inhibitors. However, in order to completely suppress the virus, patients must take a combination of at least two of the four different classes of antiretroviral drugs [61].

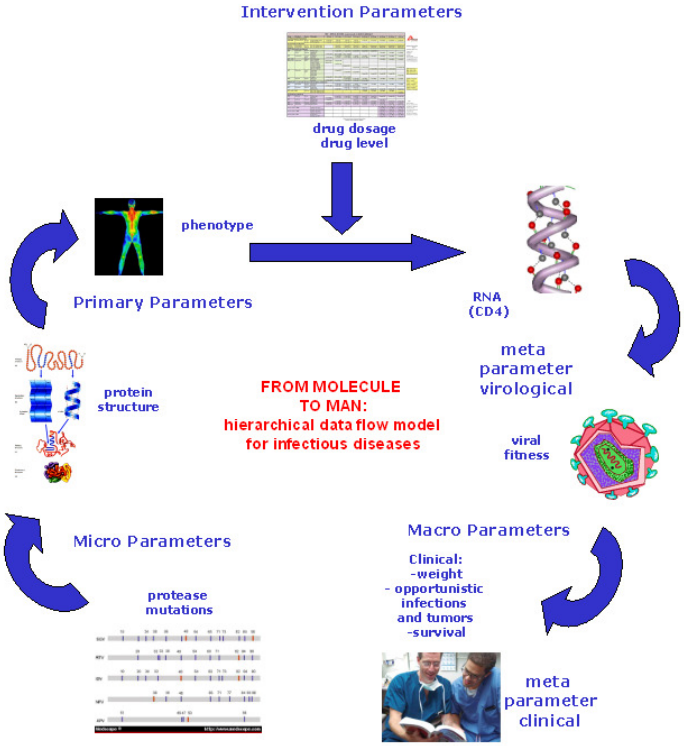
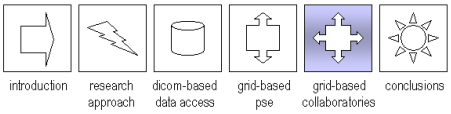


Figure 5.2: Data flow model in ViroLab, showing the model for prediction of the temporal behaviour of the immune system to drug therapy aims to qualitatively correspond to clinical data. The multiscale approach from micro parameters such as protease mutations to macro results at the clinical level go through primary, interventional and meta virological parameters, as supported by the virtual laboratory; see *www.virolab.org*

Nevertheless, in a significant proportion of patients the drugs fail to completely suppress the viral disease, resulting in the rapid selection of drug-resistant viruses and loss of drug effectiveness. This complicates the clinician's decision process, since clinical interpretation is based on data sets relating mutations to changes in drug sensitivity and relating mutations present in the virus to clinical responses to specific treatment regimens. In recent years, though, researchers have developed several genotypic resistance and interpretation tools that help clinicians and virologists choose effective ther-



apeutic alternatives to address, e.g., genotypic resistance interpretation. In this context, applying artificial intelligence and computational techniques to biomedicine has resulted in the development of specialized computer-based Decision Support Systems (DSSs) for biomedicine.

Recent developments in distributed computing further allow the virtualization of the data, computational, and software resources required by complex e-Science [83]. ViroLab is an international collaboration whose goal is to provide such virtual laboratory where researchers and medical doctors have easy access to distributed simulations and can share, process, and analyze virological, immunological, clinical, and experimental infectious disease data [139]. Currently, virologists browse journals, select results, compile them for discussion, and derive rules for ranking and making decisions. ViroLab's Grid-based DSS for infectious diseases consists of modules for individualized drug ranking in human immunodeficiency disease. It offers clinicians a distributed virtual laboratory securely accessible from their hospitals and institutes throughout Europe.

5.3.1 System Requirements

ViroLab's research goal is to investigate novel computational methods and techniques that support the development of a secure and user-friendly integrated decision support system for physicians. We use emerging Grid technology to combine data discovery, data mining, statistical analyses, numerical simulation and results presentation. Some of the main technical requirements for building such a system include:

- efficient data management,
- integration and analysis,
- error detection,
- recovery from failures,
- monitoring and logging for process flows,
- distributed execution of data- and compute-intensive tasks,
- visualization and image processing on the data through the analysis steps,
- metadata-based data access, authentication, and authorization.

For the system to support Grid-based distributed data access and computation, virtualization of its components is important (Figure 5.3). ViroLab includes advanced tools for biostatistical analysis, visualization, modeling, and

simulation that enable prediction of the temporal virological and immunological response of viruses with complex mutation patterns for drug therapy [134, 140].

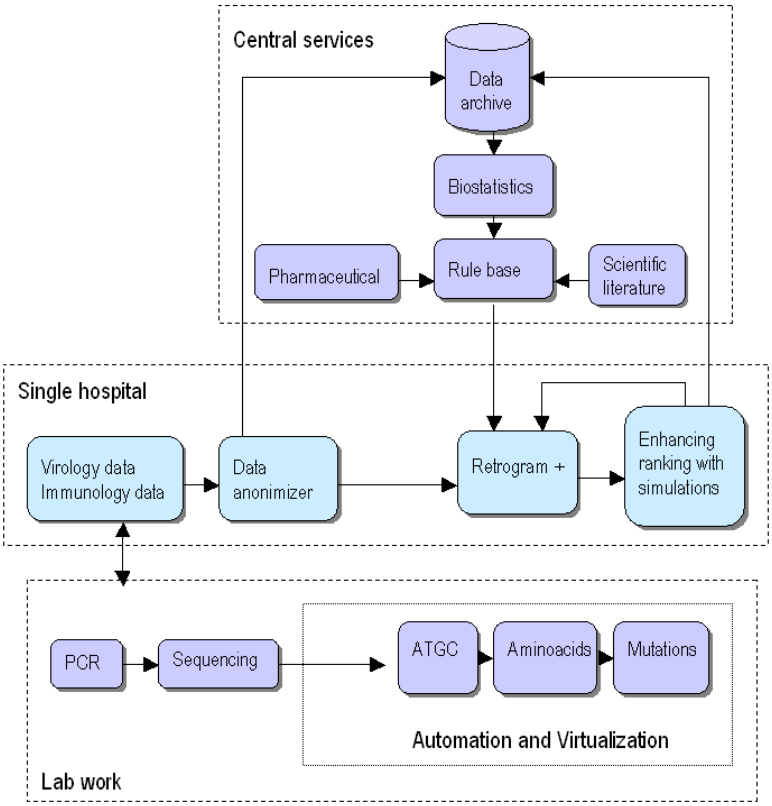
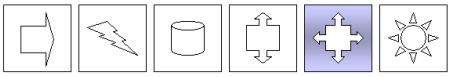


Figure 5.3: ViroLab data flow schematic; manual wet lab is automated and virtualized, and the resulting data is fed to anonymizing components, as well as directly to the Decision Support System, in order to be ranked. Simulation components enhance output rankings, which are applied to rule-based algorithms and then fed back for improved prediction of the virus's drug sensitivity



Virtual Organization

As elaborated in *chapter 3*, Grid computing is based on the central concept of distributed *Virtual Organizations* that span multiple trust domains. Trust in Grids is commonly established via a Public Key Infrastructure (PKI): every entity in the system is issued with a "certificate" that links an identifier to a piece of unique cryptographic data.

In Virolab we develop a distributed virtual organisation that binds the various components of the distributed VO. This binding layer spans a number of geographically separated physical research institutions across Europe, including five hospitals. ViroLab's VO-based security infrastructure is based on Grid middleware. This provides a number of interfaces for user-friendly and transparent access to the ViroLab applications and resources, such as Grid portals. Security is, naturally, an important concern. The sensitive nature of clinical patient data, together with concerns that data and resources be made available in a timely fashion to just those who are authorized to access them, is supportd by Grid authentication and authorization components which span all aspects of the infrastructure. It is important to note, though, that in ViroLab Grid security policy definition is left to the local owner's trust policy. VO members with access to the VOs resources can therefore use and share distributed resources securely, leveraging single-sign on.

Decision Support System

A DSS and data analysis tools are at the center of the ViroLab distributed collaboratory. Such tools may estimate the sensitivity for available drugs by interpreting a patient's genotype using mutational algorithms that experts developed based on scientific literature, taking into account the published data relating genotype to phenotype. This way, rankings are also based on data from clinical studies of the relationship between the presence of particular mutations and the clinical or virological outcome.

A number of bioinformatics software programs have been developed in the last few years to support bioinformatics decision making in clinical environments. A couple of examples of such systems are the Virtual Phenotype (developed by Virco NV), and Retrogram [†](developed by Virology Networks BV in collaboration with parts of our research team). The output of these programs consists of a prediction of the drug sensitivity of the virus, generated by comparing the viral genotype to a relational database containing a large

[†]U.S. Patent no. EPA 1176539, 2006

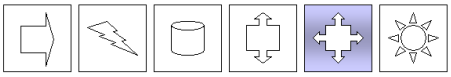
number of phenotype-genotype pairs. The Retrogram decision software, in particular, interprets the genotype of a patient by using rules developed by experts on the basis of the literature, taking into account the relationship of the genotype and phenotype. In addition, it is based on (limited) available data from clinical studies and on the relationship between the presence of genotype directly to clinical outcome. It is important to note, however, that these systems focus on biological relationships and are limited to the role of resistance. The next step is to use clinical databases and investigate the relationships between the viral resistance profile (mutational profile and/or phenotypic data) and therapy outcome measures such as amount of virus (HIV-RNA) and CD4+cells.

In DSSs like the Retrogram, the primary goal of the data analysis is to identify patterns of mutations (or naturally occurring polymorphisms) associated with resistance to antiviral drugs and to predict the degree of in-vitro or in-vivo sensitivity to available drugs from an HIV genetic sequence. The statistical challenges in doing such analyses arise from the high dimensionality of these data. A variety of approaches have been developed to handle this type of data, including clustering, recursive partitioning, and neural informatics. Neural informatics is used for synthesis of heuristic models received by methods of knowledge engineering, and results of the formal multivariate statistical analysis in uniform systems. Clustering methods have been used to group sequences that are near each other according to some measure of genetic distance: once clusters have been identified, recursive partitioning can be used to determine the important predictors of drug resistance, as measured by in-vitro assays or by patient response to antiviral drugs.

Interactivity and Process Flow

The availability of Grid infrastructures and tools for interactive applications presents an important research opportunity, as we showed in *chapter 3* in the context of simulation-based medical informatics in the European CrossGrid project. It consists of a unified approach for running interactive distributed applications on the Grid by providing solutions to the following issues:

- automatic porting of applications to Grid environments,
- user interaction services for interactive startup of applications, online output control, parameter study, and runtime steering,
- advanced user interfaces that enable easy plug-in of applications and



tools, like interactive performance analysis combined with online monitoring,

- scheduling of distributed interactive applications,
- benchmarking and performance prediction,
- optimization of data access to different storage systems.

In ViroLab, an important issue is for users to be able to register and publish derived data and processes and to keep track of the provenance of information flowing through the generated pipelines, as well as accessing existing (patient and scientific literature) data and acquiring new data from scientific instruments. These domain-independent features can then be customized by adding domain-specific components and semantic annotation of the components and data being used. In order to automate the construction of process flow applications, the system needs to generate ontological descriptions of services, system components, and their infrastructure [42, 113]. Semantic data is usually stored as a registry that contains Web Ontology Language (OWL) descriptions of service class functionality, instance properties, and performance records. The user provides a set of initial requirements about the process flow, then the system builds an abstract process flow using the knowledge about services' functionality that service providers have supplied to the registry. Subsequently, the system must apply semantic information on service properties, which results from analyzing the monitoring data of services and resources, to steer running process flows that still have multiple possibilities of concrete Web service operations. The system can select the preferable service class by comparing semantic descriptions of the available services classes and matching the classes' features to the actual requirements. ViroLab users can therefore verify and identify the data's origin and rerun experiments when required. ViroLab extends this feature by categorizing the level of information, including the data and process flows. The collected data-provenance information is archived in ViroLab's portal and accessible through search and discovery methods.

5.3.2 Actor Analysis

We next present an actor-based representation of the main architectural components for this bioinformatics use case. For this, we follow the modified actor model presented in *chapter 2.3* and used in *chapters 3* and *4*. We identify two levels of abstraction, and start our analysis at the first one.

Actor-based Grid Representation

The main components in this *stage* are a set of five actors, *portal*, *virtual lab manager*, *security manager*, *application manager* and *resource provider*, as seen in Figure 5.4, where:

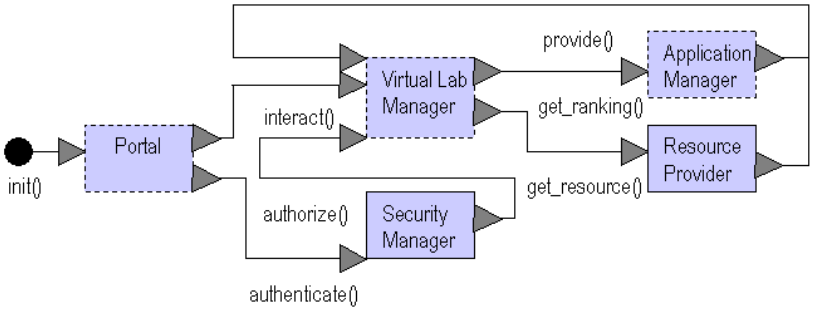


Figure 5.4: ViroLab actor model, first level. Three composite and two simple actors are contained, with *portal* as the interface between the systems and the user

- *portal* offers an external input interface to the system's user, and two output interfaces: one for interaction with the *virtual lab manager*, and one to authenticate the user's Grid credentials via the *security manager*,
- the composite *virtual lab manager* offers three input interfaces: one to *portal*, one for authorization from the *security manager*, and one from *application manager* providing the requested application. It also offers two output interfaces to *application manager* and *resource provider* to provide rankings and data, software or computational resources,
- *application manager* and *resource provider* provide their services to *virtual lab manager* via their output interfaces.

We next focus on the composite *portal*, *virtual lab manager* and *application manager* actor internals, which compose the second level of containment.

The first level's *portal* actor (Figure 5.5) is composed of three components, *browser*, *web server*, and *servlet container*:

- actor *browser* offers an interface for external input (the user input from the first level), and an output interface for calling *web server*,

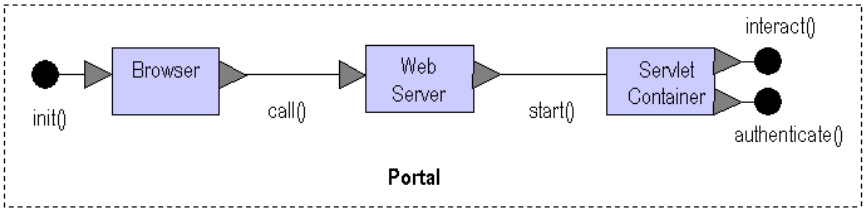


Figure 5.5: ViroLab actor model, *portal* composite actor. The *browser* and *servlet container* actors interface with the upper level of containment

- *web server* provides an output interface to *servlet container* for starting interaction, in this case with the upper level's *virtual lab manager* and *security manager*.

The first level's *virtual lab manager* (Figure 5.6) is composed by five components: the *session manager*, *provenance server*, *collaboration manager*, *runtime engine* and *data handler* actors:

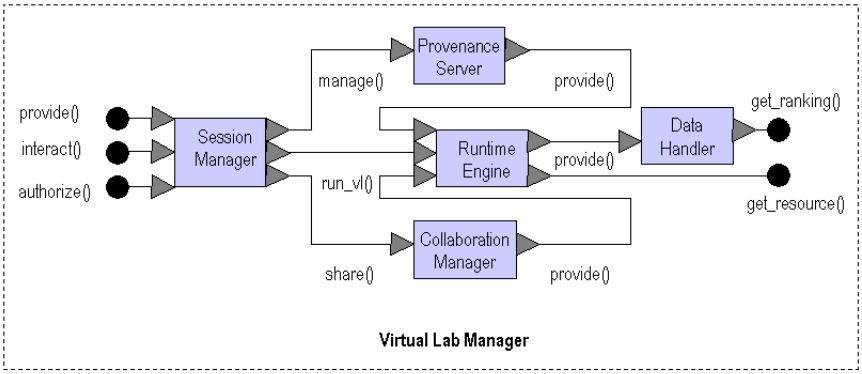


Figure 5.6: ViroLab actor model, *vl manager* composite actor. This actor provides the basic virtualization services required by the virtual laboratory, via a *session manager* and *runtime engine* that interact with *collaboration manager*, *data handler* and *provenance server*

- *session manager* provides interfaces from the upper level to the rest of the

internal *virtual lab manager* actors, with three external input interfaces; it also provides three output interfaces to *provenance server* for managing data provenance, *collaboration manager* for sharing session data, and *runtime engine* for running the virtual lab facilities,

- *runtime engine* offers an input interface from *collaboration manager* for data provision, *provenance server* and collaboration manager offer interfaces to provide their data to *runtime engine*, which provides interfaces to the upper level for resource sharing and to *data handler*,
- *data handler* provide an output interface to the upper level for new ranking provision.

Finally, the *application manager* actor (Figure 5.7) is composed by the *scientific tool* actor, which offers an input interface to the *virtual lab manager* to get new or current drug rankings, and an output interface to *rule-based system* for current or improved rules for the ranking.

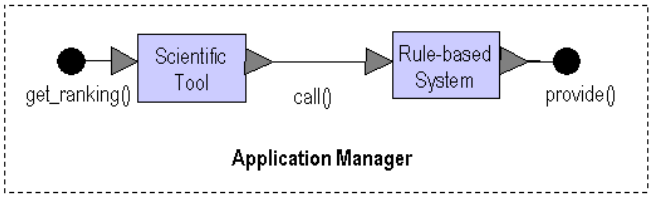
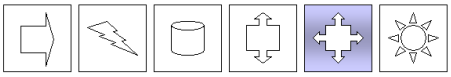


Figure 5.7: ViroLab actor model, *application manager* composite actor. The DSS application is represented by the interaction between *scientific tool* and *rule-based system* actors

Actor Signatures

As in previous chapters, the actors' port signatures are defined in XML Schema actor interface definition described in *chapter 2*, mainly for matching of composite actor signatures. A code fragment is shown in Figure 5.8. Here, a *stage* is defined, containing actors *virtual lab manager*, *security manager*, *application manager*, *resource provider* and *portal*. The composite *portal* is shown containing actors *browser*, *web server* and *servlet container*. The figure shows *portal*'s and *browser*'s matching signatures.

As stated previously, we compare the collaboratory actor patterns found in this chapter with the ones found in *chapters 3* and *4*, and discussed in the thesis *conclusions*.




```
<?xml version="1.0" encoding="UTF-8"?>
<stage xmlns="http://staff.science.uva.nl/~alfredo/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://staff.science.uva.nl/~alfredo/actor1.xsd">
  <actor type="composite" name="portal">
    <actorSignature>
      <inPorts>
        <inPort name="init">
      </inPorts>
      <outPorts>
        <outPort name="interact">
        <outPort name="authenticate">
      </outPorts>
    </actorSignature>
    <actor type="simple" name="browser">
      <actorSignature>
        <inPorts>
          <inPort name="init">
        </inPorts>
        <outPorts>
          <outPort name="call">
        </outPorts>
      </actorSignature>
    </actor>
    <actor type="simple" name="web server">
    <actor type="simple" name="servlet container">
  </actor>
  <actor type="composite" name="virtual lab manager">
  <actor type="simple" name="security manager">
  <actor type="composite" name="application manager">
  <actor type="simple" name="resource provider">
</stage>
```

Figure 5.8: ViroLab actor port definitions: code fragment of the actor's port signatures for collaborative actor architecture described in *Figures 5.6-5.9*, showing matching *portal*'s and *browser*'s *init()* inPort abstract interfaces

5.3.3 Grid-based Prototype Architecture

We instantiate the previous actor model into a system design that guarantees the interaction between a user and running applications, similar to methods used in real experiments, so the user can change a selected set of input data or parameters at runtime. For instance, under a typical usage scenario in ViroLab:

- A scientist from a clinical and epidemiological virology laboratory in Utrecht, Netherlands securely accesses virus sequence, amino acid, or mutations data from a hospital AIDS lab in Rome using Grid technology components running in Stuttgart, Germany,
- the scientist applies quality indicators needed for data-provenance tracking using provenance-server components running in Krakow, Poland,
- researchers use this data as input to (molecular dynamics) simulations and immune system simulations running on Grid nodes that reside at

University College London and the University of Amsterdam,

- the virtualized DSS automatically derives meta rules,
- intelligent system components from Amsterdam use first-order logic to clean rules, identify conflicts and redundancy, and check logical consistency,
- the scientist validates new rules that the system automatically uploads into the virtualized DSS,
- the system presents a new ranking.

We next elaborate on ViroLab's architecture design, in terms of the system's virtual laboratory, presentation and virtualization viewpoints (Figure 5.9).

Our system's architecture is based on the Grid concept of distributed virtual organizations (VOs). Here, a number of components is virtualized and distributed among VO members, supported by a *Middleware* service layer that provides basic services such as security. We distinguish a base *Grid Resources* layer, where computational (computing elements within hospitals or research centers) and data resources (storage elements where individual patient data, medical knowledge data, intermediate experimental data, and so forth) is archived. On top of this layer, a *Virtual Laboratory* layer encapsulates the runtime system that interacts with the collaboration and data access components via a session manager that interfaces to the provenance components. Finally, *Application* and *Presentation* layers contain the user interfaces and individual application interfaces to the core rule-based system used for initial decision support and ranking, or to the scientific tools for the enhancement of such rankings.

5.4 Discussion

5.4.1 Virtual Laboratory

In order to cover the temporal and spatial scales required to infer information from a molecular level up to patient medical data, application-based multiscale methods are applied in ViroLab, where distributed simulation, statistical analysis and data mining are combined and used to enhance the base rule-based decision process. In this scenario, resources are widely distributed, and the data processing requirements are highly variable, both in the type of resources required and computational processing demands. We reuse Grid

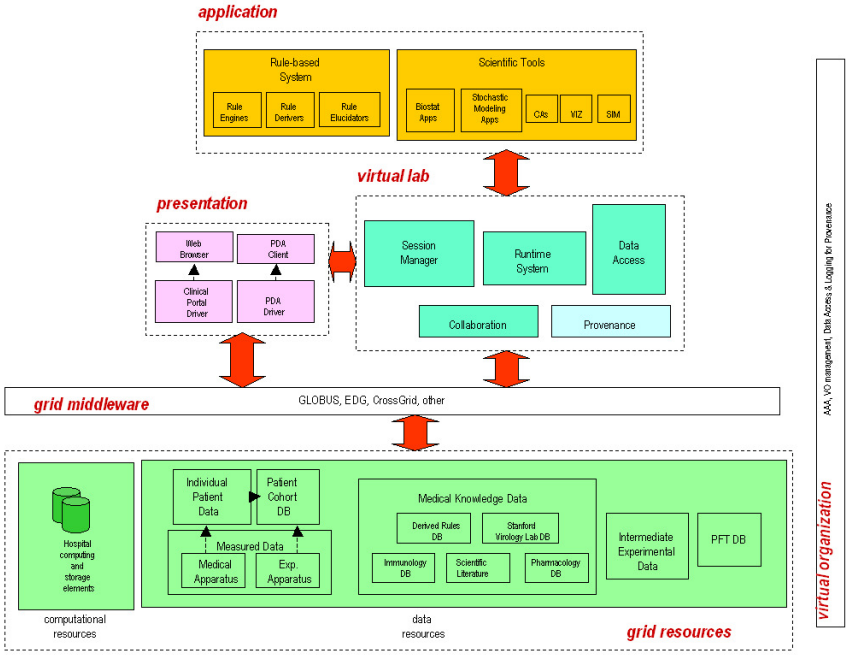


Figure 5.9: Layered representation of the ViroLab system architecture; distributed resources (computing elements, data, and storage) that the biomedical applications use are coordinated with the Grid middleware and a virtualized runtime system. Resources are automated and virtualized, and the resulting data is fed to anonymizing components, as well as directly to the Decision Support System, see http://www.virolab.org:8080/virolab/documents/description-of-work/

middleware from successful European projects to provide basic Grid services for data management, resource management, and information services on top of Globus middleware.

In order to support such a distributed infrastructure, we start with VO support for a DSS-centered prototype. Here, users are assisted by rules developed by experts on the basis of available literature, taking into account the relationship between relevant genotype and phenotype data. This way, we extend a monolithic base DSS by virtualizing its basic building blocks, and distributing the relevant components and data from clinical studies across the VO.

5.4.2 Presentation

In the initial design phases we aim to maintain the same level of usability and readability as the original Web version of Retrogram's interface. This is accomplished by maintaining the same structure, but with some modifications (Figure 5.10).

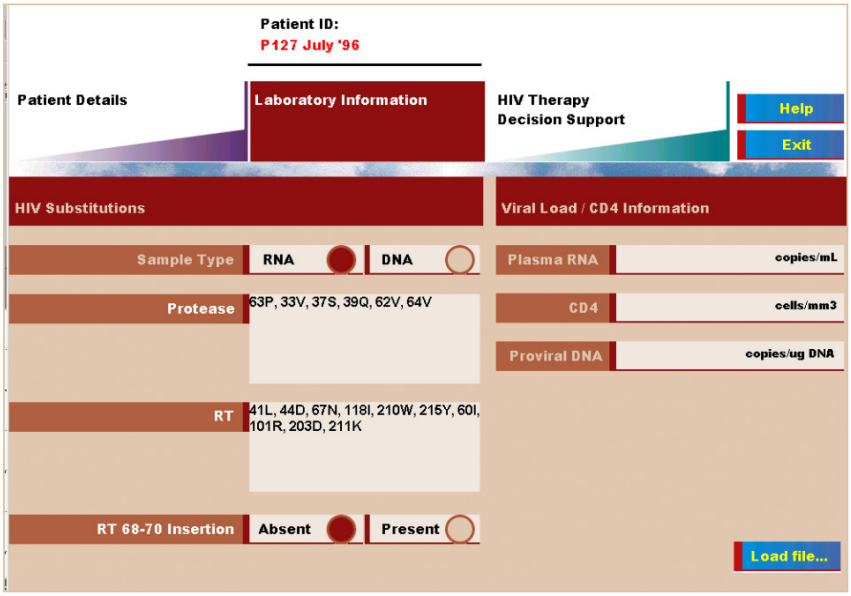
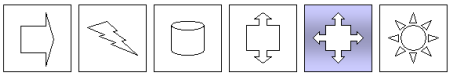


Figure 5.10: Web Retrogram interface; user enters Protease or Reverse Transcriptase substitutions in order to get resistance interpretation rankings by login into the interface, and accessing patient detail and laboratory information [22]

A Proxy method is implemented for accessing the web-based software from mobile devices as well, where the Proxy server acts between the remote server (the DSS) and a mobile device. Here, a navigation script in the Proxy is responsible for the following:

- Take the patient data from the mobile user (i.e. patient detail, laboratory information),
- create an HTTP communication with the remote server,



- submit data to the remote server,
- take the result from the remote server,
- parse HTML code and retrieve only relevant information (i.e. drug ranking, error messages, drug references etc.),
- send the wireless pages to the mobile device.

For the distributed and virtualized version used in ViroLab, an extra layer of Grid services is implemented in order to allow access to both applications and resources via a Grid portal. The portal serves as the central access point where users are authenticated using single sign-on, and provides direct access to the virtual laboratory infrastructure, runtime system and collaboration support. Our aim is that the portal is based on standard portlet technologies, using a set of portlet web applications that collaborate within the framework and support standard Grid security. We initially extend the support for Grid integration of the GridSphere portal framework [124]. In GridSphere, a collection of Grid portlets provided as add-on modules form a cohesive end-user environment for managing users, groups and supporting remote job execution, file staging and providing access to information services. GridSphere provides two portlet implementations; one is the JSR 168 de facto portlet API standard and the other is based upon the IBM WebSphere Portlet API. GridSphere supports the development of re-usable portlets and portlet services. It includes a set of core portlets and portlet services that provide the basic infrastructure required for developing and administering Web portals.

5.4.3 Virtualization and Collaboration

ViroLab infrastructure provides virologists with an advanced environment to study trends on an individual, population, and epidemiological level. That is, by virtualizing the hardware, compute infrastructure, and databases, the virtual laboratory offers a user-friendly environment, with tailored process flow templates to harness and automate such diverse tasks as data archiving, integration, mining, and analysis; modeling and simulation; and integrating biomedical information from viruses (proteins and mutations), patients (viral load), and the literature (drug-resistance experiments). In ViroLab we need access to different types of data resources via Grids. In order to achieve this goal, we provide a way of querying, updating, transforming and delivering data via web services in a consistent, independent way. In order to automate archiving, integration, mining, as well as transparent access to applications, we work with metadata and the data resources in which this data is stored,

accessed via web services that can be combined to provide higher-level services that support Grid-based data federation and distributed query processing. We approach virtualization by allowing data and application resources, to be accessed via web services. That is, a web service allows data to be queried, updated, transformed and delivered, while integrating the data via services to clients. In ViroLab we use the OGSA-DAI web services model, which can be deployed within a Grid environment. This enables us to Grid-enable our distributed data resources.

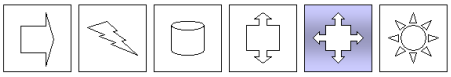
Collaboration technology aims to enhance the productivity and effectiveness of multi-disciplinary biomedical research. To this effect, Grid technology offers the possibility of leveraging computing tools into distributed collaborative environments, or collaboratories.

The basic collaboration functionality for ViroLab users is offered by basic portlet functionality, such as the one provided by the GridSphere installation. This installation is composed of the GridSphere and the Google Web Toolkit (AJAX development in Java), supporting a series of basic portlet templates. This initial setup provides a series of services, such as portlet interface. That is, a flexible XML-based portal presentation description can be easily modified to create customized portal layouts, built-in support for Role Based Access Control (RBAC), separating users into guests, users, administrators and super users, and integrated JUnit/Cactus unit tests for complete server side testing of portlet services including the generation of test reports. The basic collaboration features are seamlessly integrated in the ViroLab Grid portal, offering functionality such as interactive chat, file sharing and whiteboard applets.

5.5 Summary

In this chapter we proposed and developed parts of an integrative approach to biomedical e-Science in general and to infectious diseases in particular, based on a decision support system which compares patients' viral genotype to a distributed relational database containing a large number of phenotype-genotype pairs. The decision software interprets a patient's genotype by using rules developed by experts on the basis of the literature, taking into account the relationship of the genotype and phenotype. In addition, the output is based on available data from clinical studies and on the relationship between the presence of genotype and the clinical outcome.

With the increasing availability of genetic information and extensive patient records, researchers can now study diseases from the DNA level all the



way up to medical responses. Resolving the long-standing challenges of targeted treatments is coming within reach. It is necessary to provide integrating technology to the medical doctors and researchers bridging the gaps in multiscale models, data fusion, and cross-disciplinary collaboration (Table 5.1).

Table 5.1: Snapshot of the integrative approach used for distributed collaborative decision support in ViroLab. Data collection, virtualized application, data and computational resources are supported by a Grid infrastructure and a secure Grid virtual organization

State of the art	**Contribution**
Web-based monolitic decision support system	*Grid-based virtual collaboratory*
Tightly-coupled implementation	VO-based loosely-coupled implementation
Object-oriented architecture	Service-based architecture
Simple Web-based interaction	Real-time collaboration and provenance support
SSL-based access to data and software	PKI-based access to virtual resources (data, services, computational, instruments)
Monolitic software architecture	System-level aproach to e-Science

Although the ViroLab research is still preliminary, our initial work indicates that our personalized drug ranking prototype is viable and extensible. The system remains under development, with new functionalities being added from usability studies in a network of European hospitals.

6

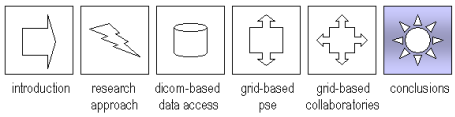
Conclusions

Virtual Organizations have the potential to change dramatically the way we use computers to solve problems, much as the Web has changed how we exchange information

—Ian Foster et al.
The Anatomy of the Grid: Enabling Scalable Virtual Organizations,
Int. J. High Perf. Comp. App., 2001. 15(3): pp. 200-222

6.1 Actor Comparison

The use of dynamic virtual organizations (VOs) offers users transparent and secure concurrent access to large numbers of resources, which can be accessed on the fly by large, distributed virtual machines. We argue in this thesis that reasoning about architectures based on Grid virtual organizations requires approaches that reflect the resource dynamics natural to Grid virtual machines. In *chapters 3, 4* and *5* we approach the problem by the use of a mesoscopic model of computation based on the classical actor model, as described in *chapter 2*, based on a model which was originally designed for workflow support. In this chapter we build on that design and prototyping experience with a comparison of the actor components found in those chapters, in order to bet-



ter understand component evolution and invariability via those actors' hierarchical composition and localized component interaction.

Actor models can be used to separate functionality from component interaction, using hierarchy refinement and component-based compositionality to divide a model into nested sub-models. We used this approach in the gradually more complex case studies shown before for the identification of abstract architectural components.

Figure 6.1 shows a number of high and low-level components which we found in our work dealing with: a) *component and object-based distributed data access*, b) *highly-distributed problem solving environment* and c) *distributed collaboratory for biomedicine*. In this comparison we identify, for instance, abstract component mappings such as:

- System access mappings between: a) *interface* (user interface, session manager, retriever, and patient info manager), b) *interactor* (grid portal, data handler, and job manager), c) *portal* (browser, web server, and servlet container),

- service mappings between: a) *server* (service provider, object broker, object adaptor, and object creator) and *broker*, b) *job manager* (job submitter and infrastructure monitor), *data handler*, and *grid portal* (data manager, security provider, and application monitor), c) *virtual lab manager* (session manager, runtime engine, data handler, collaboration manager, and provenance server), *resource provider*, and *security manager*,

- application mappings between: a) *client*, b) *simulator* and *visualizer*, c) *application manager* (rule-based system and scientific tool).

We find that composite higher-level actors in more complex systems such as the c) *distributed collaboratory for biomedicine* abstracts components that are presented in less complex systems. That is, e.g., a c) *virtual lab manager* abstracts b) *job manager*, *data handler*, and *grid portal* functionalities via its session manager, runtime engine, and data handler, and extends them via collaboration manager, and provenance server components.

The actor model of computation for the representation of concurrent behaviour is quite useful for us to understand such patterns of distributed component behaviour and how such components evolve and interact across levels of complexity. For instance, in a)*component and object-based distributed data access* analyzed in *chapter 3*, we find a simple architecture where actor *interface* (responsible for the interfacing, data transfer and session management) may

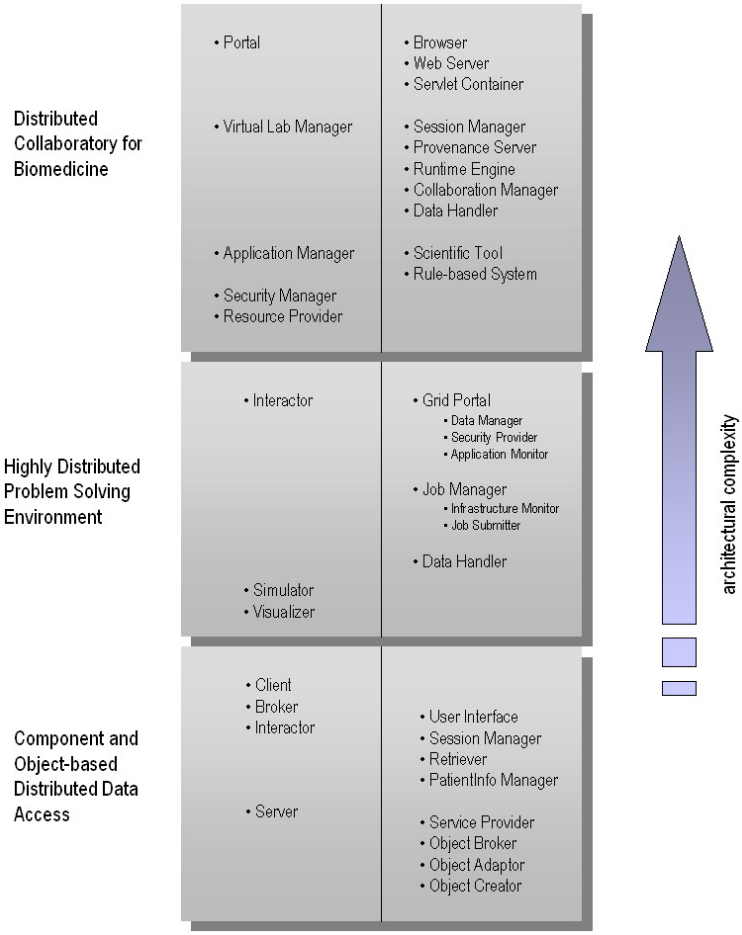
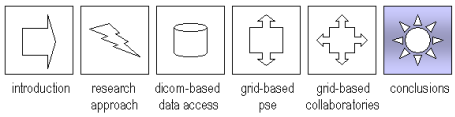


Figure 6.1: Abstract architectural actor models, from simpler to more abstract system components as system complexity and functionality evolves

be accessed concurrently by a number of *client* actors that initialize interaction or by *server* actors requesting data. The point in case is clearer in b)*highly-distributed problem solving environment* architecture from *chapter 4*, where it is important to note that the *visualizer* actor might be accessed concurrently by the *interactor* and *simulator* actors, in the case where users may be getting interactive feedback from the simulation kernel and need to visualize this partial



feedback for interacting with a number of running jobs, such as in the case of parameter sweeping.

6.2 Discussion

Current work in the literature on how to extend system modeling to highly distributed systems such as Grid architectures is by no means exhaustive. On one hand, popular communications models such as *message-passing* offer a powerful approach for modeling concurrent behaviour in Grids. Nevertheless, even though there is an important amount of work done to understand message-passing, such as Petri nets [127], process calculi [128] and the *actor model* [3], much work still is under way on how to apply them to Grid architecture modeling.

Triggered by advances in the state of the art in biomedical database, simulation, and visualization technologies, new architectures and infrastructures have to deal nowadays with issues of:

- Secure data access across organizational domains,
- scalability capabilities to support services dynamically being added to the changing infrastructure,
- on-line collaborative environments that allow biomedical scientists transparent information exchange,
- support for simulation and visualization services,
- complex workflows that require interactive services with man-in-the-loop steering of computation, visualization, and virtual data navigation.

In this work we proposed and validated a roadmap for the analysis of information models and architectures. We described a formal approach, based on an actor model of computation, which was used for analyzing biomedical informatics case studies. We described the DICOM model and elaborated on some of its most notable shortcomings by extending its functionality using object-oriented and component models for interoperability of data access. We presented our experiments on complex biomedical problem solving environment within interactive Grid infrastructures, where we use Grids for addressing our requirements for interactive biomedical applications in highly distributed environments. Finally, we analyzed a distributed bioinformatics support-decision system in which collaborative discourse among biomedical researchers was a main area of concern.

Our base requirements lead us to a set of research issues, which we approached by addressing a set of specific research objectives, as set forward in the introduction to this thesis in *chapter 1*:

- Objective 1: *To design software architecture models that support distributed virtual laboratories for interactive collaboration (i.e., collaboratories), building on the state of the art, which allow integration of distributed biomedical applications (simulation, visualization, decision support, and so forth).*
- Objective 2: *To identify software components for biomedical informatics that should remain invariant against next generations of technology, and once they are identified, to map them to distributed systems.*
- Objective 3: *To prototype such systems and reason about them, using runtime components that enable the virtualization of distributed resources, as well as their integration and secure access, leveraging advanced tools for the support of dynamic trust domains, collaborative analysis, modeling, simulation, visualization, data management, and process flow.*

We achieved successful completion of all three stated initial objectives by developing a methodology for the study of Grid architectures for biomedicine, and using it for identifying and prototyping such systems. We used such a methodology for the analysis and validation of distributed collaboratories, by analyzing incrementally complex biomedical problem-solving environments.

For the first use cases, the proof of concept prototypes showed that when modeling DICOM-based services using a component model, relatively small time delays are found in the data inquiry/transmission processes, even though the new components ran on top of open services. This allowed for transactional, secure and persistent objects being accessible by applications that have no information on where to find the needed object implementations to invoke specific methods in them. That is, by extending standard relational facilities like the ones used in the DICOM information model to object models and implementing them in XML, we achieved not only transparent but also efficient transparent data access to PACS deployed across networks of distributed hospitals.

Since infrastructure interoperability is a big issue when dealing with highly-distributed applications, we experimented with Grid infrastructure interoperability. We found that, given the relatively immaturity of the core transfer utilities in Globus, job submission and data management services running on the SEs used for our experiments were quite stable, even though the resources are shared across different middleware architectures. Our experiments showed

fairly good interoperability for data access, except when using a GT4 client accessing a LCG2.0 gsiftp server, for reasons discussed earlier.

We experimented with a concise set of benchmarking results for the characterization of computational Grid resources in terms of the performance of CPU, main memory and interconnects, for a biomedical application for the simulation of blood flow. In order to improve our resource selection process, we improved the benchmarks with on the run performance monitoring. We ran a number of tests where we found out issues related to, e.g., load balancing problems in the simulation application. In some cases the output could be quite slow and consume a significant fraction of the total computing time. This way, the performance results obtained from application benchmarking (coupled with real-time resource monitoring) proved quite more useful for Grid resource selection than the regular resource information provided by standard Grid information services to resource brokers. This offered good potential for optimizing our biomedical application for more efficient runs. Also, configuration issues with the Grid tools are quite complex and critical. For instance, the functionality between gridmapfiles seemed to be highly prone to multiple synchronization issues. Also, on the Grid information system side, we noted that when the information system is queried often, as is the case with production-type of runss, noticeable information loss is experienced. This seems to be a known problem, and one of the reasons for active research in the field.

We have shown how in the understanding of processes from bioionformatics to health informatics, from molecule to man, distributed computing in general and Grid technology in particular can play a crucial role. We found that in order to cover the huge time and spatial scales required to infer information from a molecular (genomic) level up to patient medical data, we need to apply multi-scale methods where simulation, statistical analysis, datamining is combined in an efficient way. Moreover, such required integrative approach requires distributed data collection (e.g. HIV mutation databases, patient data, literature reports, etc.) and a virtual organization (physicians, hospital administration, computational resources, etc.) to support it. The access to and use of large-scale computation (both high performance as well as distributed) is essential since many of the computations involved require near real-time response, and are too complex to run on a personal computer or personal digital assistant. Furthermore, data presentation is crucial in order to lower the barrier of actual usage by the physicians, here the Grid technology (server-client approach) can play an important role.

In this thesis we found that the actor components identified by our innovative ap-

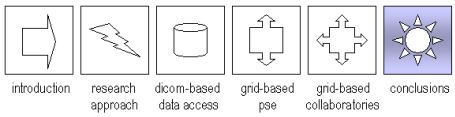
proach represent a mesoscopic level of architectural representation, capturing relevant microscopic component details, while still representing the macroscopic overview of the system architecture. Our solution to the stated problem provided a reusable approach that included a formal description, a set of design primitives that includes a visual syntax and interface language description that we can use for representing not only the high-level macroscopic compositionality, of UML, but also the well-defined semantics concurrency support of low-level microscopic approaches like the ones offered by Petri Nets and process calculi. Furthermore, this approach showed us a gradual evolution in biomedical informatics modeling constructs from simpler monolitic to more abstract components.

6.3 Future Work

The future of biomedical informatics depends in large part on integrated software problem solving environments that combine distributed resources (hardware, software, data, instruments) belonging to multiple stakeholders. The key enablement is the capability of bringing together scientists from diverse fields to aggregate their skills and bring new system-level approaches to a new kind of science. Creating new kinds of computational environments requires major breakthroughs in software architectures, such as seamless interoperability among architectural components, easy and secure access to new toolboxes such as Grid technologies for researchers via user friendly and scalable systems, and more powerful implementations of computational algorithms that leverage the distributed and concurrent characteristics of Grid technologies.

In Grids, as opposed to classic distributed computing modeling, codesign is quite essential, and that modeling the flow of data is just as important as modeling the state. For future work, we believe that investigating Hamiltonian cost functions to improve efficency of flow processes, the evaluation of design patterns for Grids, and the effects of unbounded indeterminacy are of great importance to the field. We find that recent approaches to measure Grid application performance [102, 159] and resource selection [93] are good first steps in the way, but complex challenges still exist.

There is of course room for enhancements to the model and current experimental testbed, e.g., scientific collaboration support required to process the variety of data and information generated from a number of ViroLab applications as well as data providers and hospitals. For instance, in addition to the basic requirements of voice and video support among scientists, we will work on scientific collaboration support for the sharing of drug rankings (current



rankings and new rankings resulting from the new applications), collaborative validation of drug rankings (once validation of a new ranking has been performed, users may want to discuss and share their findings with relevant stakeholders), and feedback from experts via links to the workflow engine (collaboration tools may allow the direct and instant communication with experts during and at all steps of scientific workflow execution).

Furthermore, important work related to large-scale biomedical e-Science such as the initial results of the Physiome Project *, coupled with the complementary work by biomedical Grid initiatives such as the HealthGrid † initiative's SHARE action plan for a European e-Health area and the Dutch National Research Initiative for Computational e-Science [141] provide a fertile and huge field for research and exploration.

*http://www.physiome.org
†http://www.healthgrid.org

Bibliography

[1] R. Acharya, R. Wasserman, J. Stevens, and C. Hinojosa. Biomedical imaging modalities: a tutorial. *Computerized Medical Imaging and Graphics*, vol. 19(1), pp. 3–25, 1995.

[2] H. Afsarmanesh, R. Belleman, A. Belloum, A. Benabdelkader, J. van den Brand, G. Eijkel, A. Frenkel, C. Garita, D. Groep, R. Heeren, Z. Hendrikse, L. Hertzberger, J. Kaandorp, E. Kaletas, V. Korkhov, C. de Laat, P. M. A. Sloot, D. Vasunin, A. Visser, and H. Yakali. VLAM-G: A Grid-based virtual laboratory. *Scientific Programming (Special issue on Grid Computing)*, vol. 10(2), pp. 173–181, 2002.

[3] G. Agha. *Actors: a model of concurrent computation in distributed systems*. MIT Press, MA, USA, 1986.

[4] G. A. Agha, I. A. Mason, S. F. Smith, and C. L. Talcott. A foundation for actor computation. *Journal of Functional Programming*, vol. 7, pp. 1–72, 1997.

[5] M. Aguilar and J. New. Fusion of Multi-Modality Volumetric Medical Imagery. *Proc. of the Fifth International Conference on Information Fusion*, 2002.

[6] W. Allcock, J. Bester, J. Bresnahan, A. Chervenak, S. Liming, S. Meder, and S. Tuecke. GridFTP Protocol Specification, September 2002. GGF GridFTP Working Group Document.

[7] G. Allen, W. Benger, T. Goodale, H. C. Hege, G. Lanfermann, A. Merzky, T. Radke, E. Seidel, and J. Shalf. The Cactus Code: a problem solving environment for the grid. *Proceedings Ninth International Symposium on High-Performance Distributed Computing*, pp. 253–260, 2000.

[8] R. Altman. The Interactions between Clinical Informatics and Bioinformatics: A Case Study. *J. Am. Med.Inform. Assoc.*, vol. 7, pp. 439–443, 2007.

[9] S. Altschul, W. Gish, W. Miller, E. Myers, and D. Lipman. Basic local alignment search tool. *J. Mol. Biol.*, vol. 215, pp. 403–410, 1990.

[10] E. Ammenwerth and N. de Keizer. An inventory of evaluation studies of information technology in health care: Trends in evaluation research 1982 - 2002. *Methods Inf Med*, vol. 44, pp. 44–56, 2005.

[11] A. M. Artoli, A. G. Hoekstra, and P. M. A. Sloot. 3D Pulsatile flow with the Lattice Boltzmann BGK method. *Int. J. Mod. Phys.*, vol. 13(8), 2002.

[12] A. M. Artoli, A. G. Hoekstra, and P. M. A. Sloot. Simulation of a systolic cycle in a realistic artery with the Lattice Boltzmann BGK method. *Int. J. Mod. Phys.*, vol. 17(1-2), pp. 95–98, 2003.

[13] P. Arzberger and T. Finholt. *Data and Collaboratories in the Biomedical Community*. 2002.

[14] T. Attwood. The Babel of Bioinformatics. *Science*, 2000.

[15] M. Awad, J. Kuusela, and J. Ziegler. Object-oriented technology for real-time systems: a practical approach using OMT and Fusion, p. 1996.

[16] P. Baker, A. Brass, S. Bechhofer, C. Goble, N. Paton, and R. Stevens. TAMBIS: transparent access to multiple bioinformatics information sources. In *6th Internaional Conference on In telligent Systems for Molecular Biology*, 1998.

[17] B. Balis, M. Bubak, W. Funika, T. Szepieniec, and R. Wismueller. An infrastructure for grid application monitoring. In D. Kranzlmueller, P. Kacsuk, J. Dongarra, and J. Volker, editors, *Proceedings of the Ninth European PVM/MPI Users Group Meeting*, pp. 41–49, Linz, Austria, September/October 2002.

[18] B. Balis, M. Bubak, T. Szepieniec, R. Wismueller, and M. Radecki. OCM-G Grid Application Monitoring System: Towards the First Prototype. In *Proceedings Cracow Grid Workshop 2002*, Krakow, December 2002.

[19] G. Barnett, J. Cimino, and J. Hupp. DXplain: an evolving diagnostic decision, support system. *JAMIA*, vol. 258, pp. 67–74, 1987.

[20] T. Barsalou. An object-based architecture for biomedical expert database systems. *Comput. Methods Programs Biomed.*, vol. 30(2-3), pp. 157–68, 1989.

[21] G. W. Beeler. HL7 version 3–an object-oriented methodology for collaborative standards development. *Int. J. Med. Inform.*, vol. 48(1-3), pp. 151–161, 1998.

[22] A. Been-Tiktak, K. Korn, W. Keulen, E. Schwingel, H. Walter, B. Schmidt, H. Stirnadel, J. Schapiro, and C. Boucher. Evaluation of an Open Expert-Based Genotype Interpretation Program: RetroGram. *Abstr. Intersci. Conf. Antimicrob. Agents Chemother. Intersci. Conf. Antimicrob. Agents. Chemother.*, 2001.

[23] D. S. Bell, E. Pattison-Gordon, and R. A. Greenes. Experiments in concept modeling for radiographic image reports. *J. Am. Med. Inform. Assoc.*, vol. 1(3), pp. 249–62, 1994.

[24] R. Belleman, J. Kaandorp, D. Dijkman, and P. M. A. Sloot. GEOPROVE: geometric probes for virtual environments. *Lecture Notes in Computer Science*, vol. 1593, pp. 817–827, April 1999.

[25] R. Belleman, J. Kaandorp, and P. M. A. Sloot. A Virtual Environment for the Exploration of Diffusion and Flow Phenomena in Complex Geometries. *Future Generation Computer Systems*, vol. 14 (3-4), pp. 209–214, 1998.

[26] R. G. Belleman and P. M. A. Sloot. Simulated vascular reconstruction in a virtual operating theatre. *CARS 2001 Conference*, pp. 938–944, 2001.

[27] H. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. Bhat, H. Weissig, I. Shindyalov, and P. E. Bourne. The Protein Data Bank. *Nucleic Acids Research*, vol. 28, pp. 235–242, 2000.

[28] W. Bidgood and S. Horii. Introduction to the ACR-NEMA DICOM standard. *Radiographics*, vol. 12(2), pp. 345–55, 1992.

[29] W. D. Bidgood. Documenting the information content of images. *AMIA Annu. Fall Symp.*, pp. 424–8, 1997.

[30] W. D. Bidgood, B. Bray, and N. Brown. Image acquisition context: procedure description attributes for clinically relevant indexing and selective retrieval of biomedical images. *J. Am. Med. Inform. Assoc.*, vol. 6(1), pp. 61–75, 1999.

[31] B. Boeckmann, A. Bairoch, R. Apweiler, M. Blatter, A. Estreicher, E. Gasteiger, M. Martin, K. Michoud, C. O'Donovan, I. Phan, S. Pilbout, and M. Schneider. The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003. *Nucleic Acids Research*, vol. 31, pp. 365–370, 2003.

[32] F. Bonnassieux, R. Harakaly, and P. Primet. MapCenter: an Open GRID Status Visualization Tool. In *Proceedings of ISCA 15th International Conference on parallel and distributed computing systems*, Louisville, KY, USA, 2002.

[33] G. Booch. Object-oriented analysis and design with applications, 1993.

[34] G. Booch, I. Jacobson, and J. Rumbaugh. Unified Modeling Language. *Rational Software*, 1997.

[35] G. Booch, J. Rumbaugh, and I. Jacobson. *The Unified Modeling Language User Guide*. Addison-Wesley, 1999.

[36] A. Borgida. Description logics in data management. *IEEE Transactions on Knowledge and Data Engineering*, vol. 7, no. 5, pp. 671–682, 1995.

[37] D. Bredemeyer and R. Malan. The Role of the Architect, 2002. http://www.bredemeyer.com/papers.htm.

[38] V. Breton, M. Medina, and J. Montagnat. DataGrid, Prototype of a Biomedical Grid. *Methods of Information in Medicine*, vol. 42, no. 2, pp. 143–147, 2003.

[39] M. Brune, J. Gehring, A. Keller, B. Monien, F. Ramme, and A. Reinefeld. Specifying resources and services in metacomputing environments. *Parallel Computing*, vol. 24, no. 12, pp. 1751–1776, 1998.

[40] H. Brunst, H. Hoppe, W. Nagel, and M. Winkler. Performance optimization for large scale computing: The scalable VAMPIR approach. *Lecture notes in computer science*, vol. 2074, pp. 751–760, 2001.

[41] M. Bubak, W. Funika, R. Wismueller, T. Arod, and M. Kurdziel. The G-PM Tool for Grid-Oriented Performance Analysis. *Lecture Notes in Computer Science*, vol. 2970, pp. 240– 248, January 2004.

[42] M. Bubak, T. Gubala, M. Kapalka, M. Malawski, K. Rycerz, M. Bubak, T. Gubala, M. Kapalka, M. Malawski, and K. Rycerz. Workflow composer and service registry for grid applications. *Future Generation Computer Systems*, vol. 21, no. 1, pp. 79–86, January 2005.

[43] M. Bubak, M. Malawski, and K. Zajac. The CrossGrid Architecture: Applications, Tools, and Grid Services. *First European Across Grids Conference*, 2004.

[44] B. Buchanan and E. Shortliffe. *Rule-Based Expert System: The MYCIN Experiments of the Standford Heuristic Programming Projects*. Addison-Wesley, Reading, USA.

[45] K. Buetow. Cyberinfrastructure: Empowering a" Third Way" in Biomedical Research. *Science*, vol. 308, no. 5723, pp. 821– 824, 2005.

[46] C. Bult. From information to understanding: the role of model organism databases in comparative and functional genomics. *Anim. Genet.*, vol. 37(1), pp. 28–40, 2006.

[47] M. Cannataro, C. Comito, F. Schiavo, and P. Veltri. Proteus, a Grid based Problem Solving Environment for Bioinformatics: Architecture and Experiments. *IEEE Computational Intelligence Bulletin*, 2004.

[48] M. Carey, L. Hass, P. Schwarz, M. Aryo, W. Cody, R. Fagin, M. Flickner, A. Lahiewski, W. Niblack, D. Petkovic, J. Thomas, J. Williams, and E. Wimmers. Towards heterogeneous multimedia information systems: the garlic approach. *Fifth International Workshop on Research Issues in Data Engineering*, pp. 124–131, 1995.

[49] D. Channin. Integrating the Healthcare Enterprise: A Primer. *Radiographics*, vol. 21, pp. 1343–1350, 2001.

[50] C. Chui, J. Anderson, and K. Murphy. Training and pretreatment planning of interventional neuroradiology procedures, initial clinical validation. In *Medicine meets virtual reality*, pp. 96–102, 2002.

[51] G. Chun, H. Dail, H. Casanova, and A. Snavely. Benchmark Probes for Grid Assessment. *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, vol. 18, p. 276a, 2004.

[52] A. Cooke, A. Gray, L. Ma, W. Nutt, J. Magowan, M. Oevers, P. Taylor, R. Byrom, L. Field, S. Hicks, et al. R-GMA: An Information Integration System for Grid Monitoring. In *Proceedings of 11th International Conference on Cooperative Information Systems (CoopIS 2003)*, Sicily, Italy, November 2003.

[53] F. Corpet. Multiple sequence alignment with hierarchical clustering. *Nucleic Acids Research*, vol. 16 (22), pp. 10881–10890, 1988.

[54] R. Cote, D. Rothwell, J. Palotay, R. Beckett, and L. Brochu. *The Systematized Nomenclature of Human and Veterinary Medicine: SNOMED International*. College of American Pathologists, Northfield, USA, 1993.

[55] G. Coulson and S. Baichoo. Implementing the CORBA GIOP in a high-performance object request broker environment. *Distributed Computing*, vol. 14(2), p. 113, 2001.

[56] T. Critchlow, R. Musick, and T. Slezak. An Overview of Bioinformatics Research at Lawrence Livermore National Laboratory. *American Biotechnology Laboratory*, vol. 18(9), 2000.

[57] S. Davidson, O. Buneman, J. Crabtree, V. Tannen, G. Overton, and L. Wong. BioKleisli: integrating biomedical data and analysis packages. *Bioinformatics: Databases and Systems*, pp. 201–211, 1999.

[58] J. Day and H. Zimmermann. *The OSI reference model*. IEEE Computer Society Press, 1995.

[59] R. Dayhoff, P. Kuymak, and B. Shepard. Integrating medical images into hospital information systems. *J. Digital Imaging*, vol. 4, pp. 87–93, 1991.

[60] S. De, J. Kim, and M. Srinivasan. A meshless numerical technique for physically based real time medical simulations. *Stud. Health. Technol. Inform.*, 2001.

[61] S. Deeks. Treatment of Antiretroviral-Drug-Resistant HIV-1 Infection. *The Lancet*, vol. 362, pp. 2002–2011, December 2003.

[62] T. Delanoy. Toolkit for image mining: Usertrainable search tools. *Lincoln Laboratory Journal*, vol. 8(2), pp. 145–160, 1995.

[63] A. A. Eggert and K. A. Emmerich. Long-term data storage in a clinical laboratory information system. *Journal of Medical Systems*, vol. 13(6), pp. 347–354, 1989.

[64] M. D. Eisner and E. J. Topol. Cleveland Clinic Heart Book, 2000.

[65] D. W. Erwin and D. F. Snelling. UNICORE: A Grid Computing Environment. *Proccedings of Euro-Par*, pp. 825–834, 2001.

[66] B. Eyuboglu, Birgul, and Y. Ider. A dual modality system for high resolution true conductivity imaging. *Proc. 11th Int. Conf. Elec. Bioimpedance (ICEBI)*, pp. 409–13, 2001.

[67] A. Finkelstein, C. Gryce, and J. Lewis-Bowen. Relating Requirements and Architectures: A Study of Data-Grids. *Journal of Grid Computing*, vol. 2, no. 3, pp. 207–222, September 2004.

[68] I. Foster and C. Kesselman. Globus: A Toolkit-Based Grid Architecture. *The Grid: Blueprint for a New Computing Infrastructure*, pp. 259–278, 1999.

[69] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. *Globus Project*, 2002.

[70] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke. A security architecture for computational grids. *Proceedings of the 5th ACM conference on Computer and communications Security*, pp. 83 – 92, 1998.

[71] I. Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of High Performance Computing Applications*, vol. 15(3), pp. 200–222, 2001.

[72] M. Frumkin and R. F. V. der Wijngaart. NAS Grid Benchmarks: A Tool for Grid Space Exploration. *Cluster Computing*, vol. 5, no. 3, pp. 247–255, 2002.

[73] W. Fujibuchi, S. Goto, H. Migimatsu, I. Uchiyama, A. Ogiwara, Y. Akiyama, and M. Kanehisa. DBGET/LinkDB: an integrated database retrieval system. *Pacific Symp. Biocomput.*, vol. 3, pp. 638–694, 1998.

[74] D. Garlan, R. Monroe, and D. Wile. Acme: Architectural Description of Component-Based Systems. *Foundations of Component-Based Systems*, 2000.

[75] D. Garlan and M. Shaw. An introduction to software architecture. *Advances in Software Engineering and Knowledge Engineering*, pp. 1–39, 1993.

[76] GEANT. http://www.dante.net/server.php?show=nav.007.

[77] C. Goble, R. Stevens, G. Ng, S. Bechhofer, N. Paton, P. Baker, M. Peim, and A. Brass. Transparent access to multiple bioinformatics information sources. *IBM Systems Journal*, vol. 40, no. 2, pp. 532–551, 2001.

[78] J. S. Grethe, C. Baru, A. Gupta, M. James, B. Ludaescher, M. E. Martone, P. M. Papadopoulos, S. T. Peltier, A. Rajasekar, S. Santini, I. N. Zaslavsky, and M. H. Ellisman. BIRN: Building a National Collaboratory to Hasten the Derivation of New Understanding and Treatment of Disease. *From Grid to Healthgrid: Proceedings of Healthgrid 2005*, vol. 112, pp. 100–109, 2005.

[79] A. S. Grimshaw and W. A. Wulf. The Legion Vision of a Worldwide Virtual Computer. *Comm. of the ACM*, vol. 40(1), 1997.

[80] W. Gropp, E. Lusk, N. Doss, and A. Skjellum. A High-Performance, Portable Implementation of the MPI Message Passing Interface Standard. *Parallel Computing*, vol. 22, no. 6, pp. 789–828, 1996.

[81] D. Gunter. NetLogger: a toolkit for distributed system performance analysis. In *In Proceedings of the IEEE Mascots 2000 Conference, August, Lawrence Berkeley National Laboratory Report LBNL-46269*, 2000.

[82] P. Heinzlreiter and D. Kranzlmueller. Visualization Services on the Grid: The Grid Visualization Kernel. *Parallel Processing Letters*, vol. 13, no. 2, pp. 135–148, June 2003.

[83] L. Hertzberger. e-Science and the VL-e Approach. *Lecture Notes in Computer Science*, vol. 3939, pp. 58–67, 2006.

[84] C. Hewitt. Viewing control structures as patterns of passing messages. *Journal of aArtificial Intelligence*, vol. 8(3), pp. 323–364, 1977.

[85] C. Hewitt and R. Atkinson. Synchronization in actor systems. *ACM Symposium on Principles of Programming Languages*, pp. 267–280, 1977.

[86] T. Hey and A. E. Trefethen. The Data Deluge: An e-Science Perspective. *Grid Computing: Making the Global Infrastructure a Reality*, pp. 809–824, May 2003.

[87] D. Higgins and P. Sharp. CLUSTAL: a package for performing multiple sequence alignment on a microcomputer. *Gene.*, vol. 73(1), pp. 237–44, 1988.

[88] P. Hogeweg. Simulating the growth of cellular forms. *Simulation*, pp. 90–96, 1978.

[89] W. Hoschek, J. Jaen-Martinez, A. Samar, H. Stockinger, and K. Stockinger. Data Management in an International Data Grid Project. *IEEE/ACM International Workshop on Grid Computing*, 2000.

[90] E. Houstis, E. Gallopoulos, R. Bramley, and J. Rice. Problem-solving environments for computational science. *IEEE Computational Science and Engineering Mag.*, vol. 4(3), pp. 18–21, 1997.

[91] P. J. Hunter, W. W. Li, A. D. McCulloch, and D. Noble. Multiscale Modeling: Physiome Project Standards, Tools, and Databases. *IEEE Computer*, vol. 39(11), pp. 48–54, 2006.

[92] P. J. Huntera and T. K. Borg. Integration from Proteins To Organs: The Physiome Project. *Nature Reviews Molecular Cell Biology*, 2003.

[93] A. Iamnitchi and I. Foster. On Fully Decentralized Resource Discovery in Grid Environments. In *Proceedings of the Second International Workshop on Grid Computing*, pp. 51–62. Springer-Verlag London, UK, 2001.

[94] K. A. Iskra, R. G. Bellemanand, G. D. van Albada, J. Santoso, P. M. A. Sloot, H. E. Bal, H. J. W. Spoelder, and M. Bubak. The Polder Computing Environment, a system for interactive distributed simulation. *Concurrency and Computation: Practice and Experience, (Special Issue on Grid Computing Environments)*, vol. 14, pp. 1313–1335, 2002.

[95] C. E. Kahn. A generalized language for platform-independent structured reporting. *Methods Inf. Med.*, vol. 36(3), pp. 163–71, 1997.

[96] K. Karasavvas, R. Baldock, and A. Burger. A criticality-based framework for task composition in multi-agent bioinformatics integration systems. *Bioinformatics*, vol. 21(14), pp. 3155–3163, 2005.

[97] N. Karonis, B. Toonen, and I. Foster. MPICH-G2: A Grid-enabled implementation of the Message Passing Interface. *Journal of Parallel and Distributed Computing*, vol. 63, no. 5, pp. 551–563, 2003.

[98] P. Karp. A strategy for database interoperation. *J. Comput. Biol.*, vol. 2, pp. 573–586, 1998.

[99] M. Kay. XSLT Programmer's Reference. *Wrox Press Ltd*, 2000.

[100] G. Klebe. Recent developments in structure-based drug design. *J. Mol. Med.*, vol. 78, pp. 269–281, 2000.

[101] D. Kohn. Imaging world: how DICOM standard works. *Health Manag. Technol.*, 1995.

[102] V. Korkhov, V. Krzhizhanovskaya, and P. M. A. Sloot. A case study of parallel performance and adaptive load balancing. *JPDC*, 2007.

[103] D. Kranzlmueller, P. Heinzlreiter, and J. Volkert. Grid-enabled visualization with GVK. In *2003 Annual Crossgrid Project Workshop and First European Across Grids Conference*, 2003.

[104] M. Kupczyk. The Migrating Desktop, the General Entry Point to the Grid. *CUC Conference*, 2004.

[105] LCG. http://grid-deployment.web.cern.ch/Grid-deployment/cgi-bin/index.cgi?var=homepage.

[106] R. Ledley and L. Lusted. Reasoning foundations of medical diagnosis. *Science*, vol. 130, pp. 9–21, 1959.

[107] E. A. Lee. Model-driven development - from object-oriented design to actor oriented design. *Workshop on Software Engineering for Embedded Systems: From Requirements to Implementation (a.k.a. The Monterey Workshop)*, 2003.

[108] E. A. Lee. The Problem with Threads. *IEEE Computer*, vol. 39(5), pp. 33–42, 2006.

[109] S. Lefantzi, J. Ray, and H. N. Najm. Using the Common Component Architecture to design high performance scientific simulation codes. *International Parallel and Distributed Processing Symposium*, 2003.

[110] B. P. F. Lelieveldt, S. C. Mitchell, J. G. Bosch, R. J. van der Geest, M. Sonka, and J. H. C. Reiber. Time-Continuous Segmentation of Cardiac Image Sequences Using Active Appearance Motion Models. In *IPMI '01: Proceedings of the 17th International Conference on Information Processing in Medical Imaging*, pp. 446–452, London, UK, 2001. Springer-Verlag.

[111] J. Lewkowicz. *Complete Mumps: An Introduction and Reference Manual for the Mumps Programming Language*. Prentice Hall, Englewood Cliffs, USA, 1989.

[112] J. Liu, J. Eker, J. W. Janneck, X. Liu, and E. A. Lee. Actor-oriented control system design: a responsible framework perspective. *IEEE Transactions on Control Systems Technology*, vol. 12(2), pp. 250– 262, 2004.

[113] B. Ludascher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger-Frank, M. Jones, E. Lee, J. Tao, and Y. Zhao. Scientific Workflow Management and the Kepler System. *Concurrency and Computation: Practice & Experience*, vol. 18, no. 10, pp. 1039–1065, 2006.

[114] E. Lupu and M. Sloman. Conflicts in policy-based distributed systems management. *IEEE Transactions on Software Engineering*, vol. 25(6), pp. 852–869, 1999.

[115] V. Maojo, I. Iacovidis, F. Martin-Sanchez, and J. Crespo. Medical Informatics and Bioinformatics: European Efforts to Facilitate Synergy. *Journal of Biomedical Informatics*, vol. 34, pp. 423–42, 2001.

[116] F. Martin-Sanchez, V. Maojo, and G. Lopez-Campos. Integrating Genomics into health information systems. *Methods Inf. Med.*, vol. 41, pp. 25–30, 2002.

[117] R. Martinez, J. F. Cook, and A. Tirado-Ramos. Java CORBA DICOM Adapter Service for DICOM-compliant Datasets. *First Latin American Conference on Biomedical Engineering*, pp. 621–622, 1998.

[118] K. McNeil and R. Martinez. Evaluation of the ACR-NEMA Standard for Communications in Digital Radiology. *IEEE Transactions of Medical Imaging*, vol. 9(3), pp. 281–289, 1990.

[119] R. Medeiros, W. Cirne, F. Brasileiro, and J. Sauvé. Faults in Grids: Why are they so bad and What can be done about it? In *GRID '03: Proceedings of the Fourth International Workshop on Grid Computing*, p. 18, Washington, DC, USA, 2003. IEEE Computer Society.

[120] P. Miller. Opportunities at the Intersection of Bioinformatics and Health Informatics: A Case Study. *J. Am. Med. Inform. Assoc.*, vol. 7, pp. 431–438, 2007.

[121] R. Milner. *Communicating and Mobile Systems: the Pi-Calculus*. Springer Verlag, 1999.

[122] J. Mulder and R. van Liere. The personal space station: Bringing interaction within reach. In S. Richer, P. Richard, and B. Taravel, editors, *Proceedings of the Virtual Reality International Conference, VRIC 2002*, pp. 73–81, 2002.

[123] Z. Nemeth and V. Sunderam. Characterizing Grids: Attributes, Definitions, and Formalisms. *Journal of Grid Computing*, vol. 1, pp. 9–23, 2003.

[124] J. Novotny, M. Russell, and O. Wehrens. GridSphere: a portal framework for building collaborations: Research Articles. *Concurrency and Computation: Practice & Experience*, vol. 16, no. 5, pp. 503–513, 2004.

[125] A. Ouksel and A. Sheth. "Editorial, Special Section on Semantic Interoperability in Global Information Systems". *ACM SIGMOD Record*, vol. 28, no. 1, pp. 5–12, 1999.

[126] V. Patel, J. Arocha, M. Diermeier, R. Greenes, and E. Shortliffe. Methods of Cognitive Analysis to Support the Design and Evaluation of Biomedical Systems: The Case of Clinical Practice Guidelines. *Journal of Biomedical Informatics*, vol. 34(1), pp. 52–66, 2001.

[127] C. Petri. *Kommunikation mit Automaten*. Institut fuer instrumentelle Mathematik, Bonn, 1962.

[128] B. Pierce and D. Turner. Concurrent Objects in a Process Calculus. *Theory and Practice of Parallel Programming*, vol. 907, pp. 186–215, 1994.

[129] A. Pitarch, C. Nombela, and C. Gil. Contributions of proteomics to diagnosis, treatment, and prevention of candidiasis. *Methods Biochem. Anal.*, vol. 49, pp. 331–61, 2006.

[130] A. Proppe and G. Wagner. Fur die Zuverlasigkeit medizinischer Dokumente und Befunde. *Med. Sachverst.*, vol. 52, pp. 121–127, 1956.

[131] M. Rambadt and P. Wieder. "UNICORE - Globus Interoperability: Getting the Best of Both Worlds". In *Proceedings of the 11 thIEEE International Symposium on High Performance Distributed Computing HPDC-11 20002 (HPDC'02)*. IEEE Computer Society Washington, DC, USA, 2002.

[132] W. Ruh, T. Herron, and P. Klinker. IIOP complete: understanding CORBA and middleware interoperability, 1998.

[133] F. Sanger, G. Air, B. Barrell, N. Brown, and A. Coulson. Nucleotide sequence of bacteriophage phi X174 DNA. *Nature*, vol. 265, pp. 687 – 695, 1977.

[134] T. Scheetz, N. Trivedi, K. Pedretti, T. Braun, and T. Casavant. Gene transcript clustering: a comparison of parallel approaches. *Future Generation Computer Systems*, vol. 21, no. 5, pp. 731–735, May 2005.

[135] I. Schultz, K. Wester, H. Straatman, L. Kiemeney, M. Babjuk, J. Mares, J. Willems, D. Swinkels, J. Witjes, P. Malmstrom, and J. de Kok. Gene Expression Analysis for the Prediction of Recurrence in Patients with Primary Ta Urothelial Cell Carcinoma. *Eur. Urol.*, 2006.

[136] J. Sensmeier. Advancing the state of data integration in healthcare. *J. Healthc. Inf. Manag.*, vol. 17, pp. 58–61, 2003.

[137] T. Slidel. Life Sciences Research Task Force. *Objects in Bioinformatics*, 1998.

[138] P. M. A. Sloot, C. Boucher, I. Kiryukhin, K. Saskov, and A. Boukhanovsky. A grid-based problem-solving environment for biomedicine. *Proceedings of the First European HealthGrid Conference*, pp. 300 – 323, 2003.

[139] P. M. A. Sloot, A. V. Boukhanovsky, W. Keulen, and A. Tirado-Ramos. A Grid-Based HIV Expert System. *Journal of Clinical Monitoring and Computing*, vol. 19, no. 4-5, pp. 263–278, October 2005.

[140] P. M. A. Sloot, F. Chen, and C. Boucher. Cellular Automata Model of Drug Therapy for HIV Infection. In S. Bandini, B. Chopard, and M. Tomassini, editors, *Proceedings of the 5th International Conference on*

Cellular Automata for Research and Industry. Springer-Verlag London, UK, 2002.

[141] P. M. A. Sloot, D. Frenkel, H. V. der Vorst, A. van Kampen, H. Bal, P. Klint, R. Mattheij, J. van Wijk, J. Schaye, H. Langevelde, R. Bisseling, B. Smit, E. Valenteyn, H. Sips, J. Roerdink, and K. Langedoen. Computational e-Science: Studying complex systems in silico. *Dutch National Research Initiative*, 2006.

[142] P. M. A. Sloot, A. Tirado-Ramos, I. Altintas, M. Bubak, and C. Boucher. From Molecule to Man: Decision Support in Individualized E-Health. *IEEE Computer*, vol. 39(11), pp. 40–46, 2006.

[143] D. Smith, N. Gray, J. Nourse, and C. Crandell. The Dendral Project: Recent Advances in Computer-Assisted Structure Elucidation. *Anal. Chim. Acta.*, 1981.

[144] W. Stevens. *TCP/IP Illustrated, Volume I: The Protocols*. Addison-Wesley, 1994.

[145] B. Stewart and S. Langer. Medical Image Databases and Informatics. *International Conference on Image Processing (ICIP)*, vol. 2, pp. 29–33, 1998.

[146] G. Stoesser, M. Tuli, R. Lopez, and P. Sterk. The EMBL Nucleotide Sequence Database. *Nucleic Acids Research*, vol. 27, pp. 18–24, 1999.

[147] S. Succi. The Lattice Boltzmann Equation for Fluid Dynamics and beyond, 2001.

[148] G. Sutton, O. White, M. Adams, and A. Kerlavage. TIGR Assembler: A new tool for assembling large shotgun sequencing projects. *Genome Sequence Technol.*, vol. 1, pp. 9–19, 1995.

[149] H. Tagare, C. Jaffe, and J. Duncan. Medical Image Databases, A Content-based Retrieval Approach. *J. Am. Med. Inform. Assoc.*, vol. 4(3), pp. 184–198, 1997.

[150] M. Thompson, A. Essiari, and S. Mudumbai. Certificate-based authorization policy in a PKI environment. *ACM Transactions on Information and System Security*, 2003.

[151] A. Tirado-Ramos, J. Hu, and K. Lee. Information Object Definition-based Unified Modeling Language Representation of DICOM Structured Reporting: A Case Study of Transcoding DICOM to XML. *Journal of the American Medical Informatics Association*, vol. 9, pp. 63–72, 2002.

[152] A. Tirado-Ramos, H. Ragas, D. Shamonin, H. Rosmanith, and D. Kranzlmueller. Integration of Blood Flow Biomedical Application Grid Visualization Kernel via a Grid Portal. In *Second European Across Grids Conference, Nicosia, Cyprus*, pp. 28–30, Nicosia, Cyprus, January 2004.

[153] A. Tirado-Ramos, P. M. A. Sloot, A. Hoekstra, and M. Bubak. An Integrative Approach to High-Performance Biomedical Problem Solving Environments on the Grid. *Parallel Computing, special issue on High-Performance Parallel Bio-computing*, vol. 30, pp. 1037–1055, 2004.

[154] A. Tirado-Ramos, P. M. A. Sloot, A. G. Hoekstra, and M. Bubak. An integrative approach to high-performance biomedical problem solving environments on the Grid. *Parallel Computing*, vol. 30, no. 9-10, pp. 1037–1055, 2004.

[155] O. Trelles. *Briefings in Bioinformatics*, vol. 2(2), pp. 181–194, 2001.

[156] G. Tsouloupas and M. Dikaiakos. GridBench: A Tool for Benchmarking Grids. In *Proceedings of the Fourth International Workshop on Grid Computing*, pp. 60–67. IEEE Computer Society Washington, DC, USA, 2003.

[157] M. Vannier and J. Marsh. Three-dimensional imaging, surgical planning, and image-guided therapy. *Radiol. Clin. North. Am.*, vol. 34(3), pp. 545–63, 1996.

[158] J. von Neumann and O. Morgenstern. *The Theory of Games and Economic Behavior*. Princeton Univ. Press, Princeton, 1944.

[159] F. Vraalsen, R. Aydt, C. Mendes, and D. Reed. Performance Contracts: Predicting and Monitoring Grid Application Behavior. *GRID*, pp. 154–165, 2001.

[160] D. Walker, O. F. Rana, M. Li, M. S. Shields, and Y. Huang. The Software Architecture of a Distributed Problem-Solving Environment. *Concurrency: Practice and Experience*, vol. 12(15), 2000.

[161] R. Wolski, L. J. Miller, G. Obertelli, and M. Swany. Performance information services for computational Grids. *Grid resource management: state of the art and future trends*, pp. 193–213, 2004.

[162] R. J. Zauhar, G. Moyna, L. Tian, Z. Li, , and W. J. Welsh. Shape Signatures: A New Approach to Computer-Aided Ligand- and Receptor-Based Drug Design. *J. Med. Chem.*, vol. 46, pp. 5674–5690, 2003.

[163] Z. Zhao. An agent based architecture for constructing Interactive Simulation Systems. *PhD thesis*, 2004.

[164] E. V. Zudilova and P. M. A. Sloot. Bringing Combined Interaction to a Problem Solving Environment for Vascular Reconstruction. *Int. J. Future Generation Computer Systems*, vol. 21(7), pp. 1167–1176, 2005.

English Summary

The future of biomedical informatics depends in large part on integrated software problem solving environments that combine distributed resources (hardware, software, data, instruments) belonging to multiple stakeholders. The key enablement being the capability of bringing together scientists from diverse fields to aggregate their skills and bring new system-level approaches to a new kind of science.

Creating new kinds of computational environments requires major breakthroughs in software architectures, such as seamless interoperability among architectural components, easy and secure access to new toolboxes such as Grid technologies for researchers via user friendly and scalable systems and more powerful implementations of computational algorithms that leverage the distributed and concurrent characteristics of Grid technologies.

During this work we have learned to create a novel software architecture for biomedical informativs research. We have addressed the requirements put forward by our biomedical scientists, and made advances to the state of the art in interactive Problem Solving Environments ultimately aimed at researchers and clinicians. Our modeling approach to Grid architectures allows for a clear representation of diverse roles taken by actors in accordance with their interactions with other actors in the system, as well as actor composition, which can be easily decomposed in sub-actors to any level of atomicity. Most importantly, this model allows to represent the concurrent processes that occur in interactive applictaions on the Grid (such as the concurrent access to one resource by a number of VO members, or the concurrent interaction of a user

and simulator with a visualizer) that are normally represented as sequential or just parallel processes. We investigate issues related to dynamic sharing of digital resources such as computational power, data, applications and instruments for supporting the acquisition of biomedical information. *We propose that computer science in general, and the Grid computing paradigm in particular, provide the language and tools needed to study and understand modern complex biomedical systems*. We identify a set of observations about the current state of the art in technology for the support of biomedical informatics research. We proposed and validated a roadmap for the analysis of information models and architectures. We described a formal approach, based on an actor model of computation, which was used for analyzing biomedical informatics case studies. We described the DICOM model and elaborate on some of its most notable shortcomings, by extending its functionality using object-oriented and component models for interoperability of data access, and presented our experiments on complex biomedical problem solving environment within interactive Grid infrastructures, where we use Grids for addressing our requirements for interactive biomedical applications in highly distributed environments. Finally, we analyzed a distributed bioinformatics support-decision system in which collaborative discourse among biomedical researchers was a main area of concern.

Complex interactivity using distributed biomedical data clearly requires new approaches. Not only interconnectivity, but also interoperability of components and tools are, in our experience, non-trivial issues. We require creative approaches that identify environments, architectural constructs and tools for reasoning about the system. Towards this goal, we identify a methodology for the analysis and validation of distributed, concurrent component-based collaboratories.

In *chapter 3* we started with a relational-based model of medical image representation, extended the architecture, prototyped with an object-oriented approach based on XML open technologies, and interfaced DICOM information services for enterprise interoperability. There, issues raised by, e.g., the concept of complex recursion within DICOM SR and the existence of artificial DICOM constructs such as *Macro* were important when the model was prototyped. For a more complex case study, we started with a distributed framework and extended the relational data access model with component and object technologies for distributed object-oriented transparent data access. The handling of the medical image information took place by defining the methods for the objects that use some existing functionality, and extending them with pure CORBA services and hybrid Java and CORBA bindings.

In order to provide richer DICOM functionality, the acquisition of study hierarchy and information was enhanced with the implementation of libraries for DICOM-file field parsing and mapping to CORBA objects, to allow a more dynamic flow of the case and image information. In *chapter 4* we move on to the simulation-centric Virtual Radiology Explorer application to find better solutions for treatment of vascular diseases. We extended the original VRE's virtual simulated environment by first drafting a high-level architecture, which was then analyzed using our actor model for prime components and or actors mapping of their abstract interfaces. The prototyped system included the virtualized VRE components, as well as a VO-based infrastructure that allow the scalable and transparent access to image segmentation services, simulation and visualization software, hardware resources for the performance-intensive simulation and visualization routines, and most importantly, a set of Grid services that handled access, security, file transfer, monitoring and many more infrastructure-related support. Finally, in *chapter 5*, we presented the Grid-based architecture of a virtualized decision support system that compares patients' viral genotype to a distributed relational database containing a large number of phenotype/genotype pairs. The decision software interprets a patient's genotype by using rules developed by experts on the basis of the literature, taking into account the relationship of the genotype and phenotype. In addition, the output is based on available data from clinical studies and on the relationship between the presence of genotype and the clinical outcome. We showed how in the understanding of processes from bioionformatics to health informatics, from molecule to man, distributed computing in general and Grid technology in particular can play a crucial role. We found that in order to cover the time and spatial scales required to infer information from a molecular (genomic) level up to patient medical data, we need to apply multi-scale methods where simulation, statistical analysis, data mining is combined in an efficient way. Moreover, such required integrative approach requires distributed data collection (e.g. HIV mutation databases, patient data, literature reports, etc.) and a virtual organization (physicians, hospital administration, computational resources, etc.) to support it. The access to and use of large scale computation (both high performance as well as distributed) is essential since many of the computations involved require near real time response and are to complex to run on a personal computer or personal digital assistant.

Our modeling approach to Grid architectures allows for a clear representation of diverse roles taken by actors in accordance with their interactions with other actors in the system, as well as actor composition, which can be easily decomposed in sub-actors to any level of atomicity. Most importantly, this

model allows to represent the concurrent processes that occur in interactive applications on the Grid (such as the concurrent access to one resource by a number of VO members, or the concurrent interaction of a user and simulator with a visualizer) that are normally represented as sequential or just parallel processes. In Grids, as opposed to classic distributed computing modeling, codesign is quite essential; modeling the flow of data is just as important as modeling the system state. For future work, we believe that investigating hamiltonian/cost functions to improve efficency of Grid flow processes, the evaluation of design patterns for Grids, and the effects of unbounded indeterminacy are of great interest to the field. We find that recent approaches to measure Grid application performance and resource selection are good first steps in the way, but complex challenges still exist. Also, important work related to large-scale biomedical s-Science such as the initial results of the Physiome Project, coupled with the complementary work by biomedical Grid projects such as the HealthGrid initiative's SHARE action plan for a European e-Health area provide a fertile and huge field for research and exploration.

Nederlandse Samenvatting

De toekomst van de biomedische informatica is in grote mate afhankelijk van geïntegreerde probleemoplossende omgevingen ('Problem Solving Environments' of 'PSE') die verschillende hulpbronnen (hardware, software, data, instrumenten), behorend aan verschillende belanghebbenden, samenbrengen. Het grote belang hiervan is met name dat het mogelijk maakt dat wetenschappers met verschillende achtergronden hun kennis en vaardigheden combineren en op deze manier een geheel nieuwe benadering scheppen voor een nieuwe wetenschap.
Het creëren van nieuwe klassen van computationele omgevingen vereist doorbraken in software architecturen, die naadloze interoperabiliteit tussen architecturale componenten mogelijk maken, evenals gemakkelijke en veilige toegang via gebruikersvriendelijke en schaalbare systemen tot nieuwe gereedschappen zoals Grid technologieën, en krachtiger implementaties van computationele algoritmen die de gedistribueerde en concurrente eigenschappen van het Grid benutten.

In de loop van dit werk hebben we geleerd om een nieuwe software architectuur voor biomedisch informatica onderzoek te maken. We hebben de vereisten, zoals naar voren gebracht door onze biomedische wetenschappers, in acht genomen en vooruitgang geboekt in de ontwikkeling van interactieve probleemoplossende omgevingen, die uiteindelijk bedoeld zijn voor onderzoekers en clinici. We hebben een model ontwikkeld voor Grid-gebaseerde applicaties gebaseerd op zogeheten "actoren". Dit maakt het mogelijk de verschillende rollen van deze actoren, en hun interacties met de andere actoren in

het systeem te beschrijven. Ook kunnen actoren worden ontbonden in "subactoren" tot elk gewenst niveau van detail. Het belang van dit model is dat het het mogelijk maakt de concurrente processen die plaats vinden in interactieve applicaties op het Grid correct weer te geven. Voorbeelden hiervan zijn de gelijktijdige toegang tot een hulpbron door verschillende leden van een virtuele organisatie (VO) en de gelijktijdige interactie van een gebruiker met een simulatie- en een visualisatiegereedschap. We onderzoeken de problemen bij het dynamisch delen van digitale hulpbronnen, zoals rekentijd, data, applicaties en van instrumenten die het verkrijgen van biomedische informatie kunnen ondersteunen. *Wij stellen dat informatica in het algemeen, en het Grid computing paradigma in het bijzonder, de benodigde talen en gereedschappen leveren om moderne complexe biomedische systemen te bestuderen en te begrijpen.*

We doen een aantal observaties over de huidige staat van technologische ontwikkelingen met betrekking tot de ondersteuning van biomedisch informatica onderzoek. We presenteren en valideren een routekaart voor de analyse van informatie modellen en architecturen. We geven een beschrijving van het DICOM model en lichten enkele van de meest opvallende tekortkomingen toe door de functionaliteit uit te breiden, gebruik makend van object-georiënteerde component modellen voor interoperabiliteit en data toegang. In onze experimenten over complexe biomedische probleemoplossende omgevingen binnen interactieve Grid infrastructuren, gebruiken we Grids om aan onze vereisten voor interactieve biomedische applicaties in zeer gedistribueerde omgevingen te voldoen.

Tenslotte analyseren we een gedistribueerd bio-informatisch beslissingsondersteunend systeem waar samenwerking tussen biomedische onderzoekers de focus is.
Complexe interactiviteit, in combinatie met gedistribueerde multidimensionale biomedische gegevens vereist evident nieuwe benaderingen. Niet alleen interconnectiviteit, maar ook interoperabiliteit tussen componenten en gereedschappen zijn, naar onze ervaring, geen triviale zaken. Er is behoefte aan creatieve benaderingen die omgevingen herkennen, architecturale constructies en gereedschappen voor het redeneren over het systeem. Met dit doel voor ogen identificeren we een methode voor het analyseren en valideren van gedistribueerde, concurrente, component-gebaseerde collaboratoria.

In hoofdstuk 3 beginnen we met een relationeel model van medische beeldrepresentatie, breiden we de architectuur uit, en maken een prototype met een object georiënteerde benadering gebaseerd op XML open technologieën om interoperabiliteit van DICOM informatie diensten tussen ondernemingen te geven. Hierbij zijn problemen die het gevolg zijn van bijvoorbeeld het con-

cept van complexe recursie binnen DICOM SR en het bestaan van kunstmatige DICOM begrippen zoals Macro van belang bij het ontwikkelen van een prototype van het model. In een meer complexe case study, beginnen we met een gedistribueerd framework en breiden het relationele model voor data toegang uit met component en object technologieën voor gedistribueerde, transparante, object georiënteerde data toegang. De behandeling van de medische beeldinformatie vind plaats door het definiëren van de methoden voor de objecten die gebruik maken van bestaande functionaliteit, en deze uit te breiden met pure CORBA services en hybride Java en Corba verbindingen. Om een verrijkte DICOM functionaliteit te verschaffen, is de acquisitie van informatie over de gegevens en hun onderlinge afhankelijkheid uitgebreid met de implementatie van bibliotheken voor de parsing van velden in DICOM-bestanden en vertalingen naar CORBA objecten, om een meer dynamisch verloop voor de use case en beeld informatie toe te staan.
In hoofdstuk 4 gaan we verder met de simulatie-gerichte Virtual Radiology Explorer (VRE) applicatie die erop gericht is betere behandelingen voor vasculaire aandoeningen te vinden. We breiden de oorspronkelijke VRE virtuele gesimuleerde omgeving uit door eerst een high-level ontwerp te schetsen, dat we daarna analyseren met ons actor model voor primaire componenten en/of actor equivalenten van hun abstracte interfaces. Het prototype van het systeem bevat de gevirtualiseerde VRE componenten, met daarnaast een VO-gebaseerde infrastructuur. Deze infrastructuur verschaft schaalbare en transparante toegang tot diensten voor beeldsegmentatie, simulatie- en visualisatiesoftware, hardware bronnen voor de rekenintensieve simulatie- en visualisatieroutines, en vooral, tot een reeks van Grid services die toegang, beveiliging, bestandsoverdracht, monitoring en vele andere infrastructuurgerelateerde diensten leveren.

Tenslotte, presenteren we in hoofdstuk 5, een Grid-gebaseerde architectuur voor een gevirtualiseerd beslissingsondersteunend systeem voor HIV dat het virale genotype van een patient vergelijkt met een gedistribueerde relationele database van grote aantallen fenotype/genotype paren. De beslissingsondersteunende software interpreteert het genotype van een patient door regels te gebruiken, ontwikkeld door experts op basis van de literatuur, die de relatie tussen genotype en fenotype in acht nemen. Daarnaast is het resultaat gebaseerd op beschikbare gegevens van klinische studies en de relatie tussen de aanwezigheid van genotype en de klinische uitkomst.

We laten zien hoe in het doorgronden van processen van bioinformatica tot medische informatica, van molecuul tot mens, van gedistribueerde berekeningen in het algemeen en Grid technologie in het bijzonder van cru-

ciaal belang kunnen zijn. We vinden dat om de benodigde temporele en ruimtelijke schalen voor het abstraheren van informatie op een moleculair (genetisch) niveau tot het niveau van medische gegevens van patiënten af te dekken, multi-schaal methoden toegepast moeten worden waarbij simulatie, statistische analyse en "data-mining" efficiënt worden gecombineerd. Bovendien vereist zo'n integratieve benadering een gedistribueerde verzameling van gegevens (bijv. HIV mutatie databases, patiëntgevens, literatuurverslagen enz.) en de ondersteuning door een virtuele organisatie (artsen, ziekenhuis administratie, computationele bronnen enz.). De toegang tot, en het gebruik van, grootschalige rekencapaciteit (zowel snelle clusters als ook gedistribueerde rekencapaciteit) is essentieel omdat veel van de benodigde berekeningen een bijna real-time respons vereisen en te ingewikkeld zijn om op een PC of PDA te worden uitgevoerd. Onze benadering voor modelleren op Grid architecturen verschaft een duidelijke representatie van diverse rollen gekozen door actoren in overeenstemming met hun interacties met andere actoren in het systeem en de compositie van actoren, welk simpel kan worden onderverdeeld in sub-actoren tot elk gewenst niveau van detail. In Grids is het, meer nog dan bij klassieke gedistribueerde computer modellen, essentieel gegevensstromen en de toestanden van het systeem in samenhang te ontwerpen. Voor toekomstig werk zijn we van mening dat het onderzoeken van Hamiltonian/kost functies om de efficiëntie van de stroom processen te verbeteren, de evaluatie van ontwerp patronen voor Grids, en de effecten van onzekerheden van groot belang zijn voor het vakgebied. We vinden dat recent ontwikkelde benaderingen voor Grid-applicatie prestatiemetingen en bronselectie goede eerste stappen zijn, maar er zijn nog veel complexe uitdagingen. Daarnaast verschaft belangrijk werk gerelateerd aan grootschalige biomedische e-Science projecten, zoals de eerste resultaten van het Physiome Project, gekoppeld met het aanvullende werk door Grid initiatieven zoals het SHARE actieplan voor een Europees e-Health gebied van het HealthGrid initietief, een groot en vruchtbaar veld voor onderzoek en ontdekkingen.

Acknowledgements

La science est d'un goût amer à ses débuts, mais à la fin
elle est ausi douce que le miel

—Anonymous
Medieval Arabic plate, Musée du Louvre

This has been a long, but fun journey. My enthusiasm for biomedical informatics research started during my years at the University of Arizona, under the excellent supervision of Ralph Martinez, to whom I owe my deepest respect and gratitude. Continuing my research work in this path, from telemedicine in Tucson to interoperability in New York and finally Grid architectures in Amsterdam proved to be a good decision. Special thanks go, naturally, to my Ph.D. promotor at the University of Amsterdam, Peter Sloot, for the opportunity he gave me to come to The Netherlands and work on challenging and complex research projects, collaborations and papers with scientists from all over Europe. Thanks to you I have been fortunate enough to meet scientists like Carl Kesselman, Peter Hunter, Jack Dongarra and Vaidy Sunderam, and even got to work with the likes of Charles Boucher, Simon Portegies Zwart, Alexander Bogdanov, Michael Turala and Dieter Kranzlmüller. Thanks for the trust and support, and for listening to my rants and raves throughout the past few years. Marian Bubak deserves a special mention as mentor as well; always a very critical supervisor whose help was paramount, particularly at the end of my thesis writing.

It all probably started with my friends and colleagues in New York. It was

they who planted the seed of the idea of writing a thesis in my head; Yasser al-Safadi and Martin and Joanna Rosner in particular. In those days, the thought of returning to university was running through my mind, though it was a number of events in 2001 which cleared the way for coming to Amsterdam the year after.

I would like to acknowledge all kinds of help I have got from a lot of people during my work at the University of Amsterdam: Kamil "C3PO" Iskra, Zeger "Conan" Hendrikse, Mike "Micro" Sipior, Rob "Caveman" Belleman, Roman "Darth Vader" Shulakov; thank you for all your help with practical and scientific problems ranging from MPI to Grid computing to bootstrapping complex applications and demos. I should acknowledge as well the feedback, endless discussions and fights with my brilliant co-authors and collaborators, who always came to me with ideas and extra work to satisfy the referees: Dick van Albada, Alfons Hoekstra, Alessia Gualandris, Kasia Rycerz, Zhiming Zhao, Irina Shoshmina, Vladimir Korkhov, Valeria Krzhizhanovskaya, Ilkay Altintas, and many more. It has been great working with you all. I would like to also thank as well my project colleagues, particularly the large and multicultural CrossGrid and ViroLab teams: drinking a good glass of vodka in St. Petersburg or Krakow, having a delicious dinner in Santiago de Compostela or Nicosia, or a nice cold beer in Stuttgart or Dublin with you made the hard work more fun and the whole experience unforgettable.

A large number of people crossed my way during these years at the office, teaching assignments, conferences and meetings. Special mention goes to my students, particularly Tor Kvaloy, Derek Groen, Dennis Kaarsemaker, Abdullah Ozoy, and Anton Bossenbroek. Derek and Anton, you guys just keep amazing me; keep up the great work!

At the Informatics Institute, I have benefited quite a lot from my conversations with my everyday colleagues Elena Zudilova-Seinstra, Erik Hitipeuw and Breanndán Ó Nualláin. Particularly Abdel Artoli, Roeland Merks, Mike Sipior, Yves Fomekong Nanfack, Jordi Vidal and Thomas Bernard offered me their true friendship and were always in for a good conversation, coffee or life after work hours. I enjoyed my work, though I have to admit that living in baffling Amsterdam helped me cope when work got to me: Tom de Jong and the Purmerend gang always made me feel at home in this most strange of strangest places. Even though my heart is and will always be in México, special mention goes to Hermann and Liese Denzel: you made me fall in love with Germany and made me feel part of a family that is now a home away from home. Thank you for all your love and care.

This note would not be complete without acknowledging the enormous

influence that my mother Guadalupe Ramos Rivera in particular, my brother Sergio Tirado Ramos, and sisters Rocío Duncan and Patricia Tirado Ramos have had on me. Muchas gracias por la confianza, el apoyo incondicional y los ánimos que me han dado durante toda mi vida, durante duras y maduras. Los quiero mucho.

Finally, my loving thanks go to Heidi Denzel de Tirado, my friend, accomplice, intellectual sparring partner, teacher, and loving wife; without your encouragement and understanding it would have been impossible for me to finish this work. I have to say that writing our Ph.D. theses simultaneously proved to be a much better experience than I ever thought it would be... maybe we should try it again in the future.

Amsterdam, The Netherlands, 2007.

Publications

Journal Publications

1. A. Tirado-Ramos, J. Hu, K.P. Lee, "Information Object Definition-based Unified Modeling Language Representation of DICOM Structured Reporting: A Case Study of Transcoding DICOM to XML", *Journal of the American Medical Informatics Association*, 2002, vol. 9, pp. 63-72.

2. A. Tirado-Ramos, P.M.A. Sloot, A.G. Hoekstra, M. Bubak, "An Integrative Approach to High-Performance Biomedical Problem Solving Environments on the Grid", *Parallel Computing*, special issue on High-Performance Parallel Bio-computing, vol. 30, 2004, pp. 1037-1055.

3. P.M.A. Sloot, A.V. Boukhanovsky, W. Keulen, A. Tirado-Ramos, C.A. Boucher, "A Grid-based HIV Expert System", *Journal of Clinical Monitoring and Computing*, vol. 19, nr. 4-5, October, 2005, pp. 263-278.

4. P.M.A. Sloot, A. Tirado-Ramos, I. Altintas, M. Bubak, C.A. Boucher, "From Molecule to Man: Decision Support in Individualized E-Health", *IEEE Computer*, November 2006, vol. 39, no. 11, pp. 40-46.

5. K. Rycerz, A. Tirado-Ramos, A. Gualandris, S.F. Portegies Zwart, M. Bubak, P.M.A. Sloot, "N-Body simulations on the Grid: HLA versus MPI", *International Journal of High Performance Computing*, in press.

6. A. Gualandris, S. Portegies Zwart, A. Tirado-Ramos, "Performance analysis of direct N-body algorithms on highly distributed systems", *Parallel Computing*, in press.

Book Chapters

1. A. Tirado-Ramos, Y. alSafadi, "Integrating the Healthcare Enterprise (IHE) Modality Worklist Management", *Technical Framework Specification Year 1*, Radiological Society of North America, Chicago, Illinois, USA, November 1999, vol. 1, pp. 56-60.

2. A. Tirado-Ramos, D.P. Shamonin, R.M. Shulakov, D.J. Groen, A.G. Hoekstra, G.D. van Albada, E.V. Zudilova, P.M.A. Sloot, "Interactive Problem Solving Environments on the Grid for Image-based Computational Haemodynamics", *The CrossGrid Book, European CrossGrid Consortium*, 2005, in press.

3. A. Tirado-Ramos, P.M.A. Sloot, M. Bubak, "Grid-based Interactive Decision Support in BioMedicine", *Grids for Bioinformatics and Computational Biology*, T. El-Ghazali and A. Zomaya Editors, John Wiley and Sons, USA, in press.

Springer Lecture Notes on Computer Science

1. K. Zajac, A. Tirado-Ramos, Z. Zhao, P.M.A. Sloot, M. Bubak, "Grid Services for HLA-based Distributed Simulation Frameworks", *Grid Computing: Proceedings of the 1st European AcrossGrids Conference*, Santiago de Compostela, Spain, February 13-14, 2003, in series Lecture Notes in Computer Science, Springer-Verlag, vol. 2970, pp. 147-154.

2. A. Tirado-Ramos, K. Zajac, Z. Zhao, P.M.A. Sloot, G.D. van Albada, M. Bubak, "Experimental Grid Access for Dynamic Discovery and Data Transfer in Distributed Interactive Simulation Systems", *3rd International Conference in Computational Science (ICCS 2003)*, Melbourne, Australia and St. Petersburg, Russia, June 2-4, 2003, in series Lecture Notes in Computer Science, Springer-Verlag, vol. 2657, pp. 284-292.

3. Z. Zhao, G.D. van Albada, A. Tirado-Ramos, K. Zajac, P.M.A. Sloot, "ISS-Studio: a prototype for a user-friendly tool for designing interactive ex-

periments in Problem Solving Environments", *3rd International Conference in Computational Science (ICCS 2003)*, Melbourne, Australia and St. Petersburg, Russia, June 2-4, 2003, in series Lecture Notes in Computer Science, Springer-Verlag, vol. 2657, pp. 679-688.

4. A. Tirado-Ramos, J.M. Ragas, D.P. Shamonin, H. Rosmanith, D. Kranzlmueller, "Integration of Blood Flow Visualization on the Grid: the FlowFish/GVK Approach", *Grid Computing: Proceedings of the 2nd European AcrossGrids Conference*, Nicosia, Cyprus, January 28-30, 2004, in series Lecture Notes in Computer Science, vol. 3165, pp. 77-79.

5. J. Gomes, M. David J. Martins, L. Bernardo, A. Garcia, M. Hardt, H. Kornmayer, J. Marco, R. Marco, D. Rodriguez, I. Diaz, D. Cano, J. Salt, S. Gonzalez, J. Sanchez, F. Fassi, V. Lara, P. Nyczyk, P. Lason, A. Ozieblo, P. Wolniewicz, M. Bluj, K. Nawrocki, A. Padee, W. Wislicki, C. Fernandez, J. Fontan, Y. Cotronis, E. Floros, G. Tsouloupas, W. Xing, M.D. Dikaiakos, J. Astalos, B. Coghlan, E. Heymann, M. Senar, C. Kanellopoulos, A. Tirado-Ramos, D.J. Groen, "Experience with the International Testbed in the CrossGrid Project", *European Grid Conference 2005, series Lecture Notes in Computer Science*, vol. 3470, Springer-Verlag Berlin Heidelberg, February 14-16 2005, pp. 98-110.

6. A. Tirado-Ramos, A. Gualandris, S.F. Portegies-Zwart, "Performance of a Parallel Astrophysical N-body Solver on pan-European Computational Grids", *European Grid Conference 2005, series Lecture Notes in Computer Science*, vol. 3470, Springer-Verlag Berlin Heidelberg, February 14-16, 2005, pp. 237-244.

7. T.A. Kvaloy, E. Rongen, A. Tirado-Ramos, P.M.A. Sloot, "Automatic Composition and Selection of Semantic Web Services", *European Grid Conference 2005, series Lecture Notes in Computer Science*, vol. 3470, Springer-Verlag Berlin Heidelberg, February 14-16 2005, pp. 184-192.

8. A. Tirado-Ramos, G. Tsouloupas, M.D. Dikaiakos, P.M.A. Sloot, "Grid Resource Selection by Application Benchmarking: a Computational Hemodynamics Case Study", *International Conference of Computational Science 2005, series Lecture Notes in Computer Science*, vol. 3514, Springer-Verlag Berlin Heidelberg, Atlanta, USA, May, 2005, pp. 534-543.

Refereed International Conferences

1. R. Martinez, J.F. Cook, A. Tirado-Ramos, "Java CORBA DICOM Adapter Service for DICOM-compliant Datasets", *1st Latin American Conference on Biomedical Engineering*, Sociedad Mexicana de Ingeniería Biomédica and IEEE Engineering in Medicine and Biology Society, Mazatlán, Mexíco, November 11-14, 1998, pp. 621-622.

2. P.M.A. Sloot, A. Tirado-Ramos, A.G. Hoekstra, M. Bubak, "An Interactive Grid Environment for Non-Invasive Vascular Reconstruction", *2nd International Workshop on Biomedical Computations on the Grid (BioGrid'04)*, in conjunction with 4th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid2004), Chicago, Illinois, USA, April 19-22, 2004, pp. 309-319.

3. A. Tirado-Ramos, D.J. Groen, P.M.A. Sloot, "On-line Application Performance Monitoring of Blood Flow Simulation in Computational Grid Architectures", *18th IEEE Symposium on Computer-Based Medical Systems*, special track Grids for Biomedicine and Bioinformatics, Trinity College Dublin, Ireland, June 23-24, 2005, pp. 511-516.

4. A. Tirado-Ramos, P.M.A. Sloot, "A Conceptual Grid Architecture for Interactive Biomedical Applications", *19th IEEE Symposium on Computer-Based Medical Systems*, special track Grids for Biomedicine and Bioinformatics, Salt Lake City, USA, June 23-24, 2006, pp. 762-767.

5. V.V. Krzhizhanovskaya, V.V. Korkhov, A. Tirado-Ramos, D.J. Groen, I.V. Shoshmina, I.A. Valuev, I.V. Morozov, N.V. Malyshkin, Y.E. Gorbachev, P.M.A. Sloot, "Computational Engineering on the Grid: Crafting a Distributed Virtual Reactor", *2nd IEEE International Conference on e-Science and Grid Computing*, Amsterdam, The Netherlands, Dec. 4-6, 2006, in press.

Other Conferences

1. R. Martinez, J.F. Cook, A. Tirado-Ramos, "Seamless handling of DICOM-related medical information through object-oriented distributed services", *SPIE Medical Imaging '99*, The International Society for Optical Engineering, San Diego, California, February 1999.

2. M. Rosner, A. Tirado-Ramos, "Security Functions of the MIRACLE System: A Case in Security for Enterprise Computerized Patient Record", *3rd Conference and Workshop on Distributed Object Computing Security (DOCsec'99)*, The Object Management Group, Baltimore, Maryland, USA, July 1999.

3. J. Hu, K.P. Lee, A. Tirado-Ramos, D. Sluis, "XML Encoding of DICOM Structured Reports in Ultrasound Applications", *87th Radiological Society of North America Scientific Assembly and Annual Meeting (RSNA 2001)*, Radiological Society of North America, Chicago, Illinois, USA, November 2001.

4. J. Hu, K.P. Lee, A. Tirado-Ramos, D. Sluis, "UML Modeling and XML Representation for DICOM Structured Reporting", *87th Radiological Society of North America Scientific Assembly and Annual Meeting (RSNA 2001)*, Radiological Society of North America, Chicago, Illinois, USA, November 2001.

5. K. Zajac, M. Bubak, M. Malawski, P.M.A. Sloot, A. Tirado-Ramos, "A Proposal of Services For Managing Interactive Grid Applications", *Cracow Grid Workshop 2002*, Cracow, Poland, December 11-14, 2002, pp. 155-163.

6. J. Gomes, M. David, J. Martins, L. Bernardo, A. Garcia, M. Hardt, H. Kornmayer, J. Marco, R. Marco, D. Rodriguez, I. Diaz, D. Cano, J. Salt, S. Gonzalez, J. Saenz, F. Fassi, V. Lara, P. Nyczyk, P. Lason, A. Ozieblo, P. Wolniewicz, M. Bluj, K. Nawrocki, A. Padee, W. Wislicki, C. Fernandez, J. Fontan, Y. Cotronis, E. Floros, G. Tsouloupas, W. Xing, M.D. Dikaiakos, J. Astalos, B. Coghlan, E. Heymann, M. Senar, C. Kanellopoulos, D.J. Groen, A. Tirado-Ramos, "Organization of the International Testbed of the CrossGrid Project", *Cracow Grid Workshop 2005*, Cracow, Poland, November, 2005, pp. 21-28.

7. A. Tirado-Ramos, D.J. Groen, P.M.A. Sloot, "Exploring OGSA Interoperability with LCG-based Production Grids for Biomedical Applications", *Cracow Grid Workshop 2005*, Cracow, Poland, November, 2005, pp. 367-374.

8. P.M.A. Sloot, C.A. Boucher, M. Bubak, A.G. Hoekstra, P. Plaszczak, A. Posthumus, D. van de Vijver, S. Wesner, A. Tirado-Ramos, "VIROLAB - A Virtual Laboratory for Decision Support in Viral Diseases Treatment", *Cracow Grid Workshop 2005*, Cracow, Poland, November, 2005.

9. D. Groen, A. Tirado-Ramos, D. Groep, W. van Leeuwen, J. Templon, "Resource Behaviour Characterization in Production-level Computational Grids", *Cracow Grid Workshop 2006*, Cracow, Poland, October 15-18, 2006, in press.

10. A. Bossenbroek, A. Tirado-Ramos, D. Groen, D. van Albada, "On the use of Call Option Contracts on Grid Resources", *Cracow Grid Workshop 2006*, Cracow, Poland, October 15-18, 2006, in press.