Bibliography


[Dodd04] E. J. Dodd. “Visualization and Collaboration for the On Demand Upstream Petroleum Enterprise”. IBM technical reports, Toronto,


[GR99] S. Greenberg, M. Roseman. “Groupware toolkits for synchronous

[GridSAM] GridSAM - Grid Job Submission and Monitoring WebService
http://www.omii.ac.uk/wiki/GridSAM


Lightweight Directory Access Protocol - LDAP


[SCA] Service Component Architecture
http://tuscany.apache.org/

[Schlumb] Schlumberger Inside Reality website.
http://www.slb.com/content/services/software/virtual/


In this appendix we provide some technical detailed about some of the technologies used by CEE. The objective here is to give a brief overview of those technologies in order to make clear how are they combined in the CEE SOA architecture.


SOA is an approach to delivering business solutions through services (capabilities) that are linked together by business logic – this approach reflects how a business actually operates compared to conventional applications development methods. As a result, the relationship between IT and the business is transformed from consumer/supplier to collaborating partners. SOA is an architectural style that creates new business applications through the intelligent “orchestration” of discrete, reusable business functions called “services” (figure X), each of which performing a single and well-defined task.

SOA will help to react much more quickly and cost effectively to new market opportunities, changes in business climate, and new regulation. When the business wants to introduce, change or improve a process, often one can simply adapt, reconfigure and resequence the existing services. When there is a need to bring new Software, this can be taken off-the-shelf, making it more cost-effective and faster to implement. In this way SOA offers an exciting opportunity in a world where companies need to adapt quickly and costs are a constant challenge.

Business solutions, in this new paradigm, are “composite applications” consisting of standard services linked together with business logic and standard service connections. Unlike traditional monolithic software applications, which reflect current (even outdated) process, a suite of component services can be rapidly rearranged and/or extended to reflect new business strategies and evolving market conditions.

In a conceptual model of a SOA, users of a composite business application average a common interface layer, which provides access to standard business
process modeling and orchestration tools, a common set of generic SOA functions (including security, management and governance of services), and a repository of specific business services they can work with – including component services provided by external vendors, and legacy internal applications “wrapped” with a standard interface to look and act like any other service.

Once a global organization has a sufficient library of services available, almost any business process can be orchestrated without having to write new code. Besides that, new and better services can be swapped out for old ones without causing a ripple in the business workflow.

8.2. Service Oriented Architecture

Service-Oriented Architecture (SOA) is a style of architecting software systems by packaging functionalities as services that can be invoked by any service requester [HKG+05], [Ort05]. An SOA typically implies a loose coupling between modules. Wrapping a well-defined service invocation interface around a functional module hides the details of the module implementation from other service requesters. This enables software reuse and also means that changes to a module’s implementation are localized and do not affect other modules as long as the service interface is unchanged. Once services in SOA are loosely coupled,
applications that use these services tend to scale easily because there are few dependencies between the requesting application and the services it uses.

The adoption of an SOA will produce a dramatic reduction of technology development costs by leveraging functions already built into legacy systems, by reusing services developed for other process, and by simplifying maintenance and support through elimination of redundant, siloed applications. Indeed SOA architectures are becoming a popular and useful means of leveraging Internet technologies to improve business processes in the oil&gas industry nowadays [GFF+05], [SBO+06]

In service-oriented design a service is generally implemented as a coarse-grained, discoverable software entity that exists as a single instance and interacts with applications and other services through a loosely-coupled, message-based communication model. The following definitions comprise important service-oriented terminology:

- **Services**: logical entities, with contracts defined by one or more published interfaces.
- **Service provider**: network-addressable software entity that implements a service specification. Accepts and executes requests from consumers. It publishes its services and interface contract to the service registry so that service consumer can discover and access.
- **Service consumer (or requestor)**: an application, a software module or another service that requires a service from a service provider. It initiates the enquiry of the service in the registry, binds

![Figure 8.2: Service-oriented terminology. (IBM RedBooks)](image)
to the service over a transport, and executes the service function. The service consumer executes the service according to the interface contract.

- **Service locator:** a specific kind of service provider that acts as a registry and allows for the lookup of service provider interfaces and service locations.
- **Service broker:** a specific kind of service provider that can pass on service requests to one or more additional service providers.
- **Service registry:** the enabler for service discovery. It contains a repository of available services and allows for the lookup of service provider interfaces to interested service consumers.

![Collaboration in SOA](image)

Figure 8.3 : Collaboration in SOA  (IBM RedBooks)

SOA constitutes an approach for building distributed systems that deliver application functionality as services to either end-user applications or other services. The collaborations in SOA follow the “find, bind and invoke” paradigm [EAA+04], where a service consumer performs dynamic service location by querying the registry for a service that matches its criteria. If the service exists, the registry provides the consumer with the interface contract and the endpoint address for the service.

The “find, bind and invoke” paradigm presents some drawbacks. First, the point-to-point nature of interaction between services means that service consumers often need to be modified whenever the service provider interface changes. This is often not a problem on a small scale, but in large enterprises it could mean changes to many client applications. It can also become increasingly difficult to make such changes to legacy clients. Second, it can lead to a fragile
and inflexible architecture when a large number of service consumers and providers communicate using point-to-point “spaghetti” style connections. Last, every new deployed service requires that each service consumer has a suitable protocol adapter for that new service provider. Having to deploy multiple protocol adapters across many client applications adds to cost and maintainability issues.

8.2.1. Enterprise Service Bus

An Enterprise Service Bus (ESB) is a pattern of middleware that unifies and connects services, applications and resources within a business [EAA+04]. ESB is a platform built on the principles of SOA and other open standards to help applications integrate seamlessly. Put another way, it is the framework within which the capabilities of a business’ application are made available for reuse by other applications throughout the organization and beyond. The ESB is not a new software product, it’s just a new way of looking at how to integrate applications, coordinate distributed resources and manipulate information. Unlike previous approaches for connecting distributed applications, such as RPC or distributed objects, the ESB pattern enables the connection of software running in parallel on different platforms, written in different languages and using different programming models.

![ESB Conceptual model (IBM RedBooks)](image-url)
A basic ESB provides a messaging infrastructure along with basic transformations and routing. It mainly uses open standards like web services enabling application to talk. ESB is a centralized, scalable, fault-tolerant, service-messaging framework that:

- Provides a transparent means for communicating with heterogeneous services over a diverse set of message protocols.
- Provides a shared messaging layer by which enterprise engineering applications, services, and components can connect and communicate.
- Can transmit messages synchronously or asynchronously to service endpoints and intelligently transform and secure the message content to meet the requirements of each service endpoint.
- Provides sophisticated error recovery, allowing for failed message delivery, scalability problems, duplicate messages, network failure, etc.

The main aim of the Enterprise Service Bus is to provide virtualization of the enterprise resources, allowing the business logic of the enterprise to be developed and managed independently of the infrastructure, network, and provision of those business services. Resources in the ESB are modeled as services that offer one or more business operations. Implementing an Enterprise Service Bus requires an integrated set of middleware services that support the following architecture styles:

- **Service-oriented architectures**, where distributed applications are composed of granular re-usable services with well-defined, published and standards-compliant interfaces.
- **Message-driven architectures**, where applications send messages through the ESB to receiving applications.
- **Event-driven architectures**, where applications generate and consume messages independently of one another

### 8.2.2. Web Services

Web services form an attractive basis for implementing service-oriented architectures for distributed systems. Web services rely on open, platform-
independent protocols and standards, and allow software modules be accessible over the internet. Web services and service-oriented architectures are becoming a popular and useful means of leveraging Internet technologies to improve business processes in the oil&gas industry as we showed in the Chapter 2.

8.3. Workflow Management System

Ellis [Ellis99] presents Workflow Management Systems (WfMS) as a tool to assist in the specification, modeling, and enactment of structured work process within organizations. These systems are a special type of collaboration technology which can be described as “organizationally aware groupware” [EN96]. According to the Workflow Management Coalition (WfMC), a WfMS is “the computerized facilitation or automation of a business process, in whole or in part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules” [WfMC95].

A WfMS contains two basic components:

- **Workflow modeling component**, which enables administrators and analysts to define processes (or procedures) and activities, analyze and simulate them, and assign them to people, agents or processes. This component is sometimes called “specification module” or “build time system”.

- **Workflow execution component (or enactment)**, sometimes also called the “run-time system”. It consists of the execution interface seen by end-users and the “workflow engine”, an execution environment which assists in coordinating and performing the processes and activities. It enables the units of work to flow from one user’s workstation to another as the steps of a procedure are completed. Some of these steps may be executed in parallel; some executed automatically by the computer system.

There are different types of workflows, which suit different organizational problems:

- **Production workflow** – the key goal is to manage large numbers of similar tasks, and to optimize productivity.
- **Administrative workflow** – its most important feature is the ease to define the process. Flexibility is more important than productivity, and these systems handle one or two orders of magnitude lower numbers of instances per hour than Production Workflow Systems.

- **Collaborative Workflow** – focuses on teams working together towards common goals. Groups can vary from small, project-oriented teams, to widely dispersed people with interests in common. Effective use of collaborative workflow to support team working is now considered a vital element in the success of enterprises of all kinds. Throughput is not an important consideration, and Process Definitions are not rigid and can be amended frequently.

- **Ad-hoc Workflow** – allows users to create and amend Process Definitions very quickly and easily to meet circumstances as they arise. So it is possible to have almost as many Process Definitions as there are instances of the definitions. It maximizes flexibility in areas where throughput and security are not major concerns. Whereas in Production Workflow, clearly the organization owns the process, Ad-Hoc Workflow users own their own processes.

These are workflows that enable the coordination of different types of exception, dynamic change problem and possibilities of late modeling and local adaptation of particular workflow instances [vdAalst99]. Adaptive workflows aim at providing process support like normal workflow systems do, but in such a way that the system is able to deal with certain changes. These changes may range from simple changes to ad hoc changes towards the redesign of a workflow process, as usually happens when an organization finishes a review on its business process.

The support for managing partial workflows present in an “adaptive workflow” is very attractive for Large Engineering Projects because processes in engineering domains have a very dynamic nature which means that they cannot be planned completely in advance and are under change during execution. Furthermore, in contrast to well-structured business processes, they are characterized by more cooperative forms of work whose concrete process steps cannot be prescribed.

Typically, a workflow system is implemented as a server machine which has and interprets a representation of the steps of the procedures and their
precedence; along with client workstations, one per end-user, which assists the user in performing process steps. This is typically combined with a network and messaging system (or communication mechanism) to allow the server to control and/or to interact with end-user workstations. Also included is a database that stores the process representation, attributes of end-users, and other pertinent workflow information. Many of the workflow products are combined with imaging and/or Document Management Systems (DMS).

8.3.1. Workflow Components

To achieve workflow interoperability, the Workflow Management Coalition (WfMC) created The Workflow Reference Model that describes FIVE Interface definitions [WfMC95].

- **Interface 1 (Process Definition)** - deals with passing Process Definitions from external tools to the workflow engine where they are enacted. This is the link between the so-called “Process Definition Tools” and the “Enactment Service”.

- **Workflow APIs (Interfaces 2 & 3)** - these interfaces have been combined and cover the WfAPIs (Workflow API’s). The support of these interfaces in workflow management products allows the implementation of front-end applications that need to access workflow management engine functions (workflow services). Such implementations might be written by workflow management exploiters or workflow systems integrators (WfSI). Integration between workflow and other desktop tasks (calendar, mail, reminders, etc) is often a common target and the workflow APIs allow workflow task integration into a common desktop.
- **Inter-Engine Workflow (Interface 4)** - defines the mechanisms that workflow product vendors are required to implement in order that one workflow engine may make requests of another workflow engine to effect the selection, instantiation, and enactment of known process definitions by that other engine. The requesting workflow engine is also able to pass context data (workflow relevant or application data) and receive back status information and the results of the enactment of the process definition. As far as possible, this is done in a way that is “transparent” to the user. This interface is intended for the use of WfSIs, and not users. As a side effect of facilitating communication between workflow engines, there is a requirement for audit data to be produced.

- **Audit and Monitoring (Interface 5)** - the support of this specification in workflow products allows analysis of consistent audit data across heterogeneous workflow products. During the initialization and execution of a process instance, multiple events occur which are of interest to a business, including WfAPI events, internal workflow management engine operations and other system and application functions. With this information, a business can determine what has occurred in the business operations managed by
8.3.2. Process Definition Language

The WfMC defines a Process Definition as “the representation of a business process in a form which supports automated manipulation, such as modeling, or enactment by a workflow management system. The Process Definition consists of a network of activities and their relationships, criteria to indicate the start and termination of the process, and information about the individual activities, such as participants, associated IT applications and data, etc.” [WfMC95]. This reveals the necessity for a Process Definition interchange mechanism. First, within the context of a single workflow management system there has to be a connection between the design tool and the execution/run-time environment. Second, there may be the desire to use another design tool. Third, for analysis purposes it may be desirable to link the design tool to analysis software such as simulation and verification tools. Fourth, the use of repositories with workflow processes requires a standardized language. Fifth, there may be the need to transfer a definition interchange from one engine to another.

```
<WorkflowProcess Id="Sequence">
  <ProcessHeader DurationUnit="Y"/>
  <Activities>
    <Activity Id="A">
      ...
    </Activity>
    <Activity Id="B">
      ...
    </Activity>
  </Activities>
  <Transitions>
    <Transition Id="AB" From="A" To="B"/>
  </Transitions>
</WorkflowProcess>
```

Figure 8.6: Workflow pattern Sequence in XPDL.

Figure 8.7: WfMC reference model.

The XML Process Definition Language (XPDL) is a format standardized by the WfMC to interchange Business Process definitions between different workflow products like modeling tools and workflow engines. XPDL defines a XML schema for specifying the declarative part of workflow. This language is a low level language and it can be used to model higher level business languages.

A workflow pattern is a specialized form of a design pattern as defined in the area of software engineering. Workflow patterns refer specifically to recurrent
problems and proven solutions related to the development of workflow applications in particular, and more broadly, process-oriented applications. presents an example of Sequence pattern [VanderAlst03].

8.3.3. Workflow Integration with other technologies

In the literature there are a lot of proposals concerning integration of a WfMS and other technologies. [Joeris97] proposes the combination with a Document Management System. He suggests the creation of a new data-oriented perspective for the WfMS, centered on the documents and data produced during the execution of tasks, in order to improve the coordination and cooperation support for engineering processes.

Weske [WVM+98] proposes the junction with a Geographic Information System to combine a data-oriented view with a process-oriented view aiming to support the complex cycle of process and data modeling in environmental-related geoprocessing applications.

8.3.3.1. VCS and WfMS

The integration of VCS into a WfMS is not new. Weber et al. [WPS97] proposed the integration of a VC tool into a WfMS in order to furnish a synchronous collaboration work. To allow the coordination of the conference by the WfMS he suggests the creation of new entity in the workflow model, called “conference activity”. Another important aspect is the time dimension. Conferences that are already planned at the time of the creation of the workflow are called pre-scheduled, while an ad-hoc conference is the one that was not foreseeable at the time when the workflow model is specified. This implies that in the former case some of the steps can be formally prescribed in the WfMS providing a tighter control of the results and documents generated during the conference section by the workflow engine, while in the later the results of the section should be updated by the users in the system.

The combination of VCS and WfMS can support problems which cannot be well supported by each one of them isolated. Embedding synchronous teamwork as part of the workflow produces a complementary way of conducting project activities. Such integration would enable a continuous stream of tasks and activities in which fast, informal, ad hoc, and direct actions can be taken through conferences within the usual formal workflow. The use of a coordination tool,
WFMS, and a communication tool, VCS, constitute a good combination which improves the collaborative capabilities of the CEE [Dus00].

Another important aspect is the time dimension. Conferences that are already planned at the time of the creation of the workflow are called pre-scheduled, while an ad-hoc conference is the one that was not foreseeable at the time when the workflow model is specified. This implies that in the former case some of the steps can be formally prescribed in the WFMS providing a tighter control of the results and documents generated during the conference section by the workflow engine, while in the later the results of the section should be updated by the users in the system.

8.4. Scientific Workflow Management Systems

Scientific Grid computing environments are increasingly adopting the Open Grid Services Architecture (OGSA) [Ort05], which is a service oriented architecture for Grids.

OGSA was developed by the Globus Alliance and based on standard XML-based web services technology. With the proliferation of OGSA, Grids effectively consist of a collection of Grid services, web services with certain extensions providing additional support for state and life cycle management. Hence, the need arises for some means of composing these basic services into larger workflows in order to, for example, express a scientific experiment.

The OASIS standards organization has defined the Business Process Execution Language (BPEL) as a standard-based way of orchestrating a business process composed of services. WS-BPEL 2.0 was ratified as a standard in 2007. As an execution language, WS-BPEL defines how to represent the activities in a business process, along with flow control logic, data, message correlation, exception handling, and more.

BPEL is emerging as the standard XML-based workflow language for defining and executing business processes using XML Web services. Without this standardization, the environment of the commercial systems would be not unlike the current Grid workflow engine landscape.

BPEL enables the composition, orchestration and coordination of web services. A business process described in BPEL can itself be treated as an XML web service. BPEL converged from two other workflow description languages –
Microsoft’s XLANG [12] and IBM’s WSFL [13]. BPEL provides constructs for invoking a web service and exchanging messages with a web service, both synchronously and asynchronously. It also has other primitive constructs which include constructs for manipulating data variables, indicating faults and exceptions, terminating a process and, waiting for some time. It also supports compensation blocks for exception handling. BPEL also has control constructs, such as looping, if-then-else and switch-case activities. BPEL supports both sequential and parallel execution of activities. Since BPEL is XML-based, it is extensible, which means that we can add our own constructs and also provide our own implementation of these extensions.

There are a number of advantages from adopting BPEL for the orchestration of scientific workflows. There are industrial-strength enactment environments and middleware technologies available that exhibit a level of scalability and reliability that a research prototype could not match. The multitude of providers supporting BPEL creates a market, which means that it is a live standard with ongoing efforts to develop new features. Furthermore, BPEL could serve as a standard representation for scientific workflows and hence aid reproducibility. Finally, as a programming language that focuses on high-level state transitions, it could enable computational scientists to compose scientific workflows themselves, relieving them of a dependence on software engineers.

8.4.1. Scientific Workflows Tools

8.4.1.1. Kepler

Kepler [Kepler] is another extensible workflow system aimed at scientific workflows. The Kepler project is cross-project collaboration between SDM (Scientific Data Management) Center, SEEK (Science Environment for Ecological Knowledge), GEON (Cyber-infrastructure for the Geosciences) and RoadNet (Real-time Observatories, Applications, and Data Management Network). The aim of Kepler is to provide a framework for design, execution and deployment of scientific workflows. Kepler is built on top of Ptolemy II [PtolemyII]– an API for heterogeneous, concurrent modeling and design. Kepler currently provides the following major features [LAB+06]:

- Prototyping workflows: Kepler allows scientists to prototype scientific workflows before implementing the actual code needed for executions.
MoML – an internal XML language for specifying component-based models and composing actors into workflows

- Distributed execution (Web and Grid-Services): Kepler’s Web and Grid service actors allow scientists to utilize computational resources on the network in a distributed scientific workflow.
- Database access and querying: Kepler includes database interactions.
- Other execution environments: Support for foreign language interfaces via the Java Native Interface provides the flexibility to reuse existing analysis components and to target appropriate computational tools.

8.4.2.
Condor

Condor [Condor] is a specialized workload management system for compute-intensive engineering simulations. Condor provides a job queueing mechanism, scheduling policy, priority scheme, resource monitoring, and resource management. Condor is known to provide a High Throughput Computing (HTC) environment on a large size of distributed computing resources. It can manage a large size of machines and networks owned by different users. Besides controlling idle components, Condor can be configured to share resources. When a user submits a job to Condor it put it into a queue, selects when and where to run the job based on a policy, monitors the job, and informs the users about the status of the task upon completion. Condor-G is used to schedule and run jobs on heterogeneous grid resources. It uses Globus GRAM service, a uniform interface to heterogeneous batch systems. Condor-G creates an abstract view of the grid as local resource and allows the user to submit jobs to different batch systems (Condor, Load Leveler, etc.) and get updates regarding the status of the tasks.

8.4.3.
InfoGrid

InfoGrid [LMC+05], is a client/server system for grid environments which, in addition to the support for usage and management of distributed computational resources, offers facilities to integrate applications and manage data and users (Figure 11). InfoGrid presents to its users, through a web browser, a workspace
with all available applications and with the user’s data files organized by project. A user can extend the system adding new applications. InfoGrid also provides its users with some collaborative work facilities.

Applications which are executed in the client utilise available services of the InfoGrid to have access to and to manage distributed computational resources. One of these services is the remote execution of algorithms which are in computers linked to the InfoGrid. For InfoGrid, algorithms are defined as executable programs implemented in any language which accept input parameters, generate an output and do not have any type of interaction with the user during their execution. Many computers can be incorporated to the grid environment to serve as a platform for algorithms execution. New algorithms can be easily made available in the environment and the process to execute them is turned into a transparent task for the user.

Figure 8.8: InfoGrid architecture.
8.4.4. 
**Grid Job Submission and Monitoring System**

GridSAM is a Grid Job Submission and Monitoring Web service for submitting and monitoring jobs managed by a variety of Distributed Resource Managers (DRM). GridSAM implements the Job Submission Description Language (JSDL) defined by the Global Grid Forum (GGF) [LMN+04]. Transparency of the underlying Grid scheduler being used to execute jobs on a Grid is achieved by using GridSAM. Scientists only need to define the JSDL for their jobs once and not worry about which scheduler is used now or at any point in the future.

8.5. 
**Virtual Environments**

The terms Virtual Environment (VE) and Virtual Reality (VR) are often used synonymously to describe a computer-generated, artificial environment or reality that is presented to a user. A VE tries to evoke a strong sense of reality in the user. This is achieved by the generation of artificial input to the user’s visual, acoustic and haptic senses.

By interfacing some of the user’s articulations in the real world back into the VE, the user can consciously interact with the environment. Typically, interfaces to direct-manipulation devices are used, but nowadays more advanced interaction techniques like speech and gesture recognition have become a major research interest.

The generation of high-quality visual feedback from the virtual environment is often considered the most important aspect in generating a high degree of immersion. The desire to increase the degree of immersion led to the development of sophisticated image generators and display devices. Beginning with low-resolution monoscopic CRT displays used in early flight simulators and image generators that where capable of rendering only a few hundred polygons per second, the development progressed toward today’s high-resolution stereoscopic display systems like the CAVE [CS+92] and readily available graphic cards that render hundreds of millions of polygons per second.

Parallel to the development of new display devices, image generators and input devices, various toolkits and application frameworks are developed. They provide a basic software infrastructure for the development of VE applications.
The main goal of these efforts is the maximization of software reuse in order to minimize the necessary development resources for application development. Designed for different application domains, the only common nominator of most toolkits and frameworks is a scene-graph based object model. The provided API, the supported hardware and operating systems and the set of supported display and input devices vary greatly.

Collaborative Virtual Environments (CVEs) are a special case of Virtual Reality Environments [Tramberend99], where the emphasis is to provide distributed teams with a common virtual space where they can meet as if face-to-face, co-exist and collaborate while sharing and manipulating, in real-time, the virtual artifacts of interest [GLG03]. They can be seen as the result of a convergence of research interests within the Virtual Reality and Computer Supported Cooperative Work (CSCW) communities. CVEs are becoming increasingly used due to a significant increase in cost-effective computer power, advances in networking technology and protocols, as well as database, computer graphics and display technologies. They have been used mainly by automotive and aircraft manufacturers aiming to improve the overall product’s quality and also aiming to reduce project’s life cycle, cutting down costs and reducing the time-to-market of new products. Examples of applications are Visualization of real-time simulation of 3D Complex Phenomena, Collaborative Virtual Design and Product Development, Training and Edutainment, Telepresence and Telerobotics, Business meetings among others.

Studies of a cooperative work in real-world environments have highlighted the important role of physical space as a resource for negotiating social interaction, promoting peripheral awareness and sharing artifacts [BH+92]. The shared virtual spaces provided by CVEs may establish an equivalent resource for telecommunication. In teleimmersive environments (TE), a VCS is integrated with a CVE to provide collaborators at remote sites with a greater sense of presence in the shared space [LJB+99]. TEs may enable participants to discuss and manipulate shared 3D models and visualizations in such a way that each user can adopt their own viewpoint and can naturally indicate the others where they look and point. Scientific visualization has also been used in many application areas and has proven to be a powerful tool in understanding complex data [FB+99]. Those characteristics of TEs are very important for Virtual Prototyping as in projects of oil production units explained in section 2.

The development of CVE technology has been driven mainly by the challenge of overcoming technological problems such as photo realistic rendering
and supporting multiple users in CVEs. Once those users are geographically distributed over large networks like the Internet, and the number of users has been increasing continuously, scalability turns to be a key aspect to consider for real-time interactions [LMH02].

Other important aspects are composability and extensibility or dynamic reconfigurability for assembling applications and improving adaptability of system at runtime with component-based system design, plug-ins functionality and service discovery mechanisms. In order to support the execution of CVEs with large-scale virtual worlds over long periods of time, they must be based on technologies that allow them to adapt, scale and evolve continuously. VE applications offer an almost limitless number of opportunities for the inclusion of plug-in technology. Graphical plug-ins may generate 3D models on the fly; network plug-ins may provide support for new protocols and filtering schemes; plug-ins for physical simulation may introduce previously unknown forces that improves the reality of the simulation. Persistence and portability aspects have also to be considered in order to guarantee the ability of building reusable large virtual worlds commonly needed in engineering projects.
9 Appendix B

9.1. List of Publications

In what follows is a list of all papers related to this thesis, it can be downloaded in the following website:
http://www.tecgraf.puc-rio.br/~ismael/imk/publications/publications.html

**Periodicals**


**Book Chapters**


**Papers in Conferences**

**2009**

Brasil. Santos, I. H. F, Braganholo, V., Mattoso M., Jacob, B. P., Albrecht, C.


2008


2007

2006


2004 and before


