THE VEGA PLATFORM
IT for the Virtual Enterprise

RICHARD JUNGE
Faculty for Architecture, Professorship for CAAD,
Technical University Munich

MANFRED KÖTHE
KARSTEN SCHULZ
European Applied Research Center
Digital Equipment, Karlsruhe

ALAIN ZARLI
Centre Scientifique et Technique du Batiment
Sophia Antipolis

WIM BAKKEREN
TNO Building and Construction Research
Delft

Abstract

One of todays manny buzzwords is ‘virtual enterprise’. The objectives of the EPRIT project VEGA are the developement of an IT platform enabling such enterprises. Virtual enterprise means a number of people or smaller companies grouped together for a destinct contract, which none of them alone could or would able to get and to undertake. Modern decentralized, distributed IT solutions typically could support such virtual enterprises in their competituion against those who are big or strong enough to to carry out such contracts with their internal resources alone. VEGA gathers together the necessecary components as technically available and extends their cababilities as needed for a platform enabling collaboration in an flexible and distributed environment.

Introduction

The domain of architectural and engineering design and erection of a building is surely one of the most typical the term ‘virtual enterprises’ applies to. Always this has not been the domain if single centralized large companies as e.g. in car industry. Rather

these the task in the process of design and erection of a building are undertaken by relatively to really small entities forming teams (to use this more commonly used term instead of 'virtual enterprise') mostly on a project to project basis.

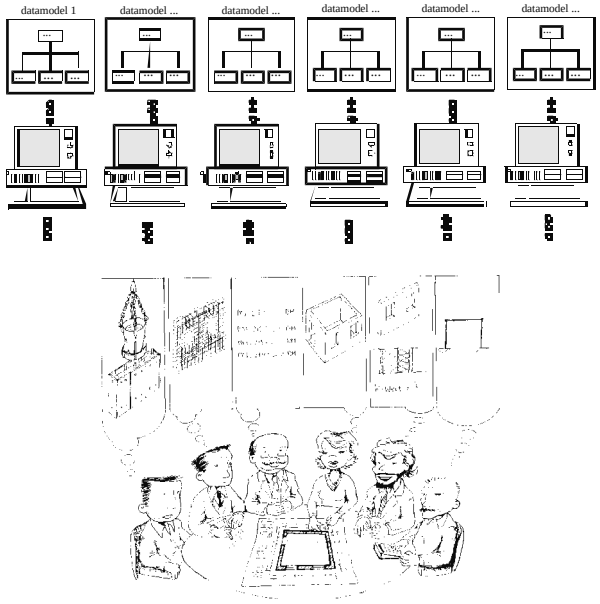

Figure 1 "Collaboration today and tomorrow

There are at least three scenarios encountered for the collaborative work in future design offices and teams:

- Parallel work of architects within a single office on a larger project
- Collaboration and parallel work of architects across scattered places
- Collaboration between architects and various design teams

Architects and/or engineers within a design team usually work in parallel on their specific part of the building project. This team specific part, mostly identical with one specific view type, should be represented by one common data model within the chosen AEC software application. This common data model covers all project objects, the team is working on. Each team member should have direct access to a common data model and be enabled to modify his or her part of it. All other team members should be informed, when one of these changes will affect their part of responsibility. This common data model, by no means, is a central model.

Current technology gives only little, or no support at all, to the parallel work even within one AEC application. In particular, the file locking mechanisms and in CAD the layer structures provided by current technology, forces designers to split their work topologically, e.g., into building sections or storeys. This mostly is in contradiction

with the division of work that is more task oriented. An example is e.g. that one designer is responsible for the structural part, the other for the space layout and preparing the finish, and the third for the curtain wall. These tasks can not be separated topologically. In consequence, work has to be done separately and in sequencially.

The result of todays technology is, that the software applications do not provide enough support to ensure consistency between the different parts of the data model. Other current technologies, such as referencing other drawings (often called xref-technology) as background layers have their shortcomings as well, since conflicts have to be detected by the users only on a purely visual basis.

Figure 1 shows how different users/clients of an application could work in parallel within a networked solution of applications that are supported by a chosen distributed data management. Then there is no further need to structure the projects necessarily according to the requirements of data technology. The team members can now work in parallel with one common data model, and the software application is responsible for the automatic update of the data model and its forwarding to the application.

A second end user need concerns the whole process and interaction between the different design teams. Since each design team works on its own design task and with its own view on design, the communication of informtion or data sharing has to incorporate a way that allows for the translation between the different views. On the other hand the various design teams also work independently for a certain period of time (off line), usually until a next team meeting or deadline. Within this time they work on local versions, that might contradict with local versions of other teams or team members. These local versions are based on the last agreed or approved versions of the building project, but the work can proceed in different directions within the various teams.

It is, indeed, the task of collaborative work to plan for the entire interactions of the actors of the teams. This again incorporates advantages since each team can search for an optimal solution by leaving side effects out of scope for a moment and disadvantages since there is a risk, that some of the inconsistencies might not be detected at the right time. Aiming at the totally integrated and consistent working environment for the whole project would therefore mean a restriction on the freedom of design teams and facing them with unwanted efforts to check consistency for every design action.

On the other hand intelligent workflow management together with a model based exchange of partial information would prevent the high risk of inconsistency, as it would allow to update and check the projects' data base at certain times during the project. Especially during the co-ordination phase the incremental exchange of data by collaborative action needs IT support. The scenario of a second vision therefore

includes that the designers are enabled to exchange distinct bits of data they are concerned about. The support of collaborative work between teams will include the update of the project data models on demand. It will help to support the flow of information needed for collaboration and guide the collaborative actions, such as approval or documentation according to the understanding of the building process.

The vision is, that whenever a design member needs actual data from another team to check its current version, the new technology allows him or her to retrieve this particular data set. He or she can also update the database of the partners after negotiation in order to provide the newest version at their disposal. At certain times, given by the workflow management, support is given to the update of the whole projects' data base, which includes that all relevant teams are being informed and that the results can be approved and documented as a certain reached design step.

The VEGA project aims at specifying and demonstrating a software architecture suited for such virtual enterprises. One of the key problems of the virtual enterprise is the sharing of information between the different companies. This problem includes the need for neutral specification of information, distribution of information, control of information flow, and security of information. To address these aspects of information sharing in the virtual enterprise, VEGA makes use of three different technologies:

- product-data technology for the specification of meaningful project information;
- middleware technology for the distribution of project information;
- workflow management technology for the control of the flow of information and work in the virtual enterprise.

Because the virtual enterprise requires an open solution that allows any company to use its own favourite applications, VEGA builds on existing open standards for the three technologies mentioned:

- STEP and especially EXPRESS for the neutral specification of product model data;
- OMG CORBA for the distribution of objects over networks and for information security;
- WfMC interface specifications for the common definition of workflows, the interaction between workflow management system components and for workflow system interoperability.

In addition to these standards VEGA also makes use of standards like SGML, HTML and VRML for specific areas such as exchange of administrative messages, document handling and presentation of information.

Fundamentals: Standards in Information Technologies

The VEGA project structure can be depicted as in figure 2. The envisioned VEGA Platform stands on for columns.

1. Product Data Modeling (PDM)
2. Technologies enabling communication between applications in an distributed product model environment (COAST)
3. Distributed information services (DIS)
4. Work- and information management and control (WFL)

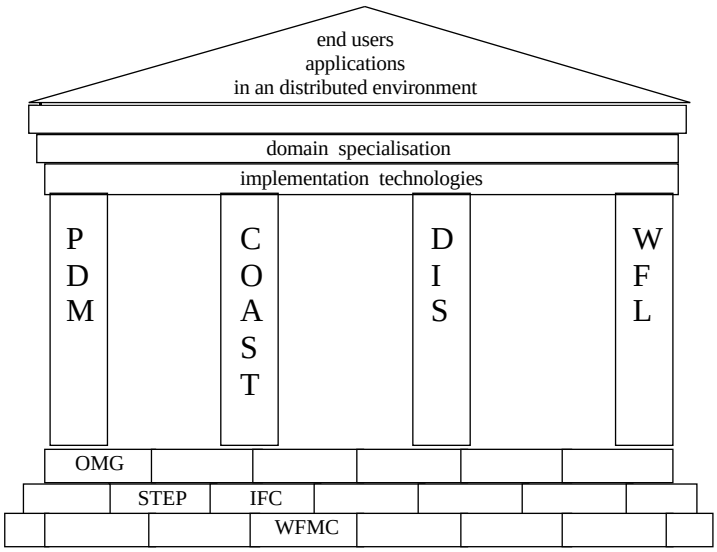

Figure 2 The structure of the VEGA project

These collumns can also be regarded as four fundamental standards: STEP, OMG and CORBA, Internet technology, Work Flow Management Coalition's Interfaces.

MODELING METHODOLOGY

The need of breaking down the islands of information in several industries - from aerospace to the car industry - has lead to dramatic research efforts to achieve effective product data exchanges. These efforts have become necessary due to the implication in the industrial processes of an increasing number of different actors on different sites using different systems and software. Consequently, wide enterprises have lead to the standardisation of methodologies, languages and technologies, especially in the context of STEP, the goal of which is a to allow uniform and complete representation of

industrial product data, with general mechanisms for the exchange, the archiving of the data and the integration into any kind of application.

STEP is an emerging International Standard for the representation and exchange of product data, which is developed in ISO TC 184/SC 4 (Industrial data and global manufacturing programming languages). For a more comprehensive description, the reader could refer to [Fowler, J. 1995]. The main objective of STEP is to provide information sharing through neutral mechanisms, independent from any particular system or software, and allowing the description of product data throughout the life-cycle of a product. The nature of this description makes it suitable not only for neutral file exchange, but also as a basis for implementing and sharing product databases.

To reach its objectives, STEP has been designed as a quite modular standard, composed of different components including a data modelling language for product modelling (EXPRESS), an encoding format (STEP Physical File Format - SPFF) for the expression of product data under an ASCII form, a set of generic models (called Integrated Resources), and a set of user-driven procedures (Application protocols) to adapt the Integrated Resources to application tasks most of the time in specific industrial sectors. A STEP data file is built up according a set of rules and procedures, and on the ground of specific standardised languages or formats. These data files could then be read directly by STEP processors, STEP compatible applications or through STEP access interfaces, allowing a better integration of applications through standardisation of information interchange between different enterprises using different kinds of methods and systems.

One of the best reasons of the success of STEP is the standardised EXPRESS language [Industrial automation systems and integration 1994a], which allows a complete and unambiguous description of information related to the different conceptual components of a product. EXPRESS allows the designer to describe the data structures of the objects of his application, especially through the concept of entity describing a class of objects ; the corresponding instances are described in the SPF Format, which is a STEP standard for a neutral format of all the actual data in an application [Industrial automation systems and integration 1994b]. EXPRESS and SPFF are a first medium for the exchange of data, through ASCII files. STEP also offers the specification of an interface to STEP databases, the SDAI - "Standard Data Access Interface" - [Industrial automation systems and integration1995a], which is a functional Application Programming Interface (API). The SDAI allows the access and the manipulation of information contained in STEP databases in a quite transparent manner, i.e. independently of the proprietary archives methods of these databases, this separation being of course a fundamental key-point for industry needs.

Of course, all the STEP standards, including EXPRESS, SPFF and the SDAI, are defined in a complete independent way and with no direct connection with a given software language and implementation. But, to allow the use on a computer, some

compilers for EXPRESS and SPFF are needed, and the SDAI must be "translated" in an implementation language : the specification of such an implementation is called a "Binding". Several bindings have yet been defined, especially for C [Industrial automation systems and integration 1995b] and C++ [Industrial automation systems and integration 1996] programming languages, which enables SDAI compliant calls thanks to functions written in C and C++ programs.

Besides STEP, another major effort is currently undertaken with the IAI (International Alliance for Interoperability), which is a non-profit alliance of the building industry including: architects and engineers, building project manufacturers, software vendors, research laboratories, and so on, divided in several national chapters (North America, Germany, United Kingdom, France, Singapore, Nordic Countries, and Japan). The IAI objective is to integrate the AEC/FM industry by specifying the IFC (Industry Foundation Classes) as a universal language to be a basis for collaborative work in the building industry and consequently to improve communication, productivity, delivery time, cost, and quality throughout the design, construction, operation and maintenance life cycle of buildings.

STEP and IAI/IFC share the same goals, which are applications interoperability, data exchange and actors co-operation. Nevertheless, they differ in their respective processes: the IAI promote a bottom-up approach, with an iterative and incremental development for quick bring in play, STEP is rather a long-term project, with a top-down approach and is concerned with broad standardisation. The IAI has partially adopted the STEP technology, mainly through an EXPRESS version of the IFC. In the future, an integration of the IFC within STEP is planned.

THE WEB INFRASTRUCTURE AN DISTRIBUTED ENVIRONMENT

The growth of world-wide data networks have generated a growth of methodologies, technologies, tools and interfaces for Intranet/Internet distributed solutions. The most well-known infrastructure integrating such tools and interfaces is the Internet/WEB. The Internet basically offers networks interoperability, and a multimedia-based exchange and access level. This is a different concept of the one of «exchange» or interoperability between applications, as it essentially concerns interoperability through networks connections. Technologies like CORBA are dedicated to distributed systems and environments, clearly allowing applications to «inter-work» and interact each other, invoking remote application resources in a dynamic way, and so on. The level of Wan, and Internet/WEB, is rather dedicated to information infrastructure which implies of course to allow the access to all the components of the infrastructure, but not real interoperability between applications: in fact, in that context, applications interoperability is a possible consequence of WAN/Internet/Intranet/WEB set of technologies as they provide network hardware and software support for that, on the other hand applications interoperability is the main objective of CORBA technologies. Even if the current developments on JAVA methodologies concern more integration

and co-operation between applications around the WEB, they are at the moment far from the already huge set of specifications of the OMG with respect to distribution and interoperability of applications.

The Internet can be considered as a network of networks and offers a physical interconnection between computers spread all over the world: any end-user, should it be any public administration, private company or individual end-user, can connect to it through any kind of computer (PC, Apple Macintosh, Unix mainframe, and so on). This idea relies on the fact that the connection is done through a set of de facto standards, the main of them being TCP/IP (Transmission Control Protocol/Internet Protocol), for the different kinds of information in order to provide a better interoperability and opening to a large set of different platforms and systems. Various communications are ensured through a set of standard tools: the electronic mail (using SMTP: Single Mail Transport Protocol), the NEWS servers (using NNTP: Network News Transport Protocol), remote transfer of any types of file usinf FTP (File Transfer Protocol), connection to remote computers through Telnet mechanisms, etc. Thus Internet allows to access any kind of information, provided that this information is available under one of the standardized formats as defined within Internet.

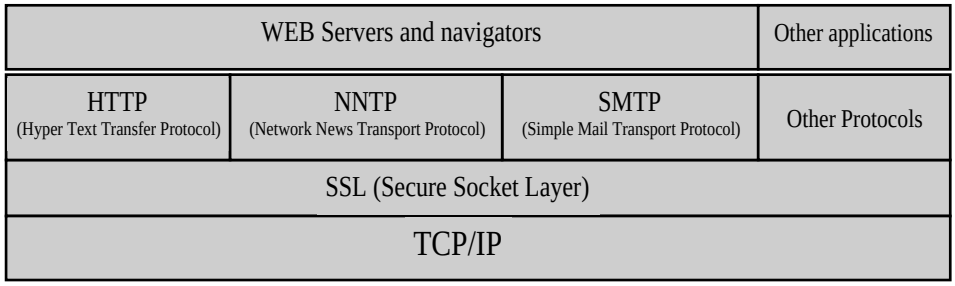

Figure 3 Components of the WEB

The WEB is a powerful interface for Internet users, introducing two new protocols. Indeed, it associates two technologies which are Internet and the hypertext, this second one consisting in defining electronic links between the various documents of Internet, so that it is no more necessary to know the exact site where stand all the information as any information is located according to other information. It offers facilities for the organisation and the access to any type of manufacturing information: this is achieved by means of an encapsulation and distribution of the information through a generic transfer protocol which is http (HyperText Transfer Protocol), and a standardized representation for hypertext documents which is html (HyperText Mark-up Language).
All the Internet standard protocols are managed by all the WEB navigator tools (the two most famous ones are Netscape *Navigator* and Microsoft *Internet Explorer*), and this is quite transparent to the end user connected to the network, as shown in figure 3. Thus, the WEB nowadays federates almost all the Internet standards through a common friendly user interface. Even if it offers at the moment mainly electronic

management of hypermedia documents, the Internet/WEB is a world-wide support for communication, as anyone can connect to the network and access to any tool or service through http (whatever the hardware and software are).

The WEB can nowadays be considered as a basic support for multimedia communication as well as group-communication protocols, and one of the interests of the WEB is linked to the concept of groupware. Groupware is more than distribution or client/server computing: it is a collection of technologies that allow to represent and manage complex processes in the context of collaborative human activities. Groupware is built on several foundation technologies like remote actors and distributed processes, multimedia document management, workflow, electronic mail and (audio or video) conferencing, and so on. It is not the role of client/server or distributed systems technologies, for instance, to give a global answer to all the features involved in groupware, whilst the Internet/WEB offers effective fulfilment for some of the basic services of groupware like electronic mail, multimedia or the access to global wide public WEB databanks.

On the other hand, the Internet/WEB essentially collects highly unstructured data, most of the time grouped in WEB documents including text and images, under http-based formats. At the moment, the basic unit of exchange under the WEB is the document, which can be stored, viewed and even retrieved and replicated, and the WEB can be mainly considered as a multimedia document management architecture. Moreover, due to their intrinsic nature, the WEB protocols are too generic to vehicle the deep semantics of the different applications, and these protocols are not to be compared with powerful modeling languages as found in PDT. Another weakness of the WEB is the administration and mainly the access to all the local corporate information: no precise control is effective on what is available to other network users and what is kept private, and no access control or security considerations are really managed. Indeed, as the Internet/WEB offers a quite open and free world, this means many risks for private companies and their internal knowledge and databases, and also the need for powerful protection mechanisms in the context of commercial processes. Finally, it can be mentionned that the WEB essentially implements a pure client/server approach: the relationships between clients and servers are not bi-directional, while technologies like CORBA (see next section) specify a more general framework where clients may be servers too.

Thus, the integration of modeling methodologies, distributed technologies and the Internet/WEB suggests a new strategy for integrating legacy environments, providing more powerful mechanisms and services and offering a common interface for the access to any type of information, and the effective exploitation of an infrastructure based on PDT and distribution to deliver Value-added Web-based solutions. All these technologies should complement each other for the development of large scale information systems.

DISTRIBUTED ARCHITECTURES: CORBA

Standards like STEP or IFCs allows applications interoperability at a conceptual level, in fact through communication based on data model integration. They allow the exchange of information on the base of the same understanding and representation of the information semantics. But in practice and in a real computer-aided industrial and business processes context, there is a added need for operational exchange between applications, and applications software have to physically interoperate. This means that each application must be able to invoke, most of the time remotely, resources of the other applications. This can be achieved through powerful computer based mechanisms like client/server architectures or more recently object-oriented distributed systems. Among the various recently emerging technologies in this area, the OMG CORBA standard is one of the most important current developments and is becoming one of the most popular for applications interoperability. The following paragraphes hereafter shortly introduce to the major concepts of client/server and distributed systems, and then give an overview of the CORBA specification.

A standard for distribution as CORBA offers powerful means to make applications inter-operate each other on the base of a networked environment, but no specific prerequisite is given with respect to the network support, to all the various types of managed information or to the different accessible services. Thus, the applications may for instance work together all linked through some private net work or Local Area Network (LAN). However, in the context of the today increasing global competition, corporations have more and more to envisage partnerships over a large geographic and organisational spread, and there is a need for a general information infrastructure (information highways) able to support world-wide communication and interactions.

Client/server and distributed architectures

A distributed system can be roughly defined as two or more pieces of software sharing information with each other, and relying on:

- a set of (heterogeneous) hardware, software, and data components, connected by a network,
- providing a uniform set of basic services as naming, user registration and access control, remote process execution, management, security, and so on.

The first point mentions that a distributed system is first a networked system, with multiple heterogeneous and independent computers having their own CPUs, their own software and their own databases. The second point introduces the fundamental concept of services. These are components of the distributed environment providing generic features or behaviours quite independently of the various applications connected to the network and allow applications to really inter-operate. These services must be globally and uniformly accessible. They ensure the coherence of the distributed system as a whole rather than as a simple aggregate of interconnected computers and

tools, thus contributing to provide a standard way to manage all the relevant generic-purpose information to ensure the smooth running of the distributed system.

The nowadays most popular distributed systems are client/server constructions. With this model, there are two major types of software: client software requesting the information to servers, and server software which accomplishes the requests. Despite some advantages, like the common location within the server of some basic services (access rights, transactions management, etc.), this model does not consider all the applications as similar independent entities: clients must be aware of distribution handling to find the location of the server on the network, clients and servers are not symmetric as clients are pro-active and servers reactive.

Distributed architectures as defined in CORBA are a step further than client/server system in their ability to allow in a seamless way the interoperability of applications and the sharing of information and resources between geographically spread corporations. They allow the sharing of resources, the use of any type of hardware and software on the base of a broker-service, and the parcelling of computing workload among many different machines. Distributed Systems authorise a great deal of autonomy through separated components which moreover can keep their own specific data close to their processing (and access other information on a remote server in the network only when necessary). They are naturally extensible and thus can grow in small increments over a large range of sizes, and maybe the most important property is the fact that in case one component crashes it does not necessarily imply that all the (remote) components cannot keep on running.

The CORBA specification

CORBA [Mowbray, T. J., Zahavi, R.][Otte, R., Patrick, P. Sr., Roy, M. 1995][Siegel, J., et al.1996] is the central component of the Object Management Architecture (OMA) defined by the OMG. In the context of a client/server architecture, its main goal is to manage requests from a client for specific operations on objects to the most appropriate object implementation for execution, on the base of the specification of a set of interfaces to objects. CORBA objects are effective means for hiding the characteristics of the underlying hardware and software implementations behind a high-level portable interface. They can invoke operations on each other even when implemented in different programming languages and when running on incompatible operating systems.

CORBA allows a flexible client/server relationship. In contrast to traditional client/server implementations, CORBA allows a piece of software that acts like a client for one request to act as a server for the next request. This flexibility is the result of CORBA's object-oriented approach defining an object model. Thus CORBA views all the applications in the CORBA-System as nothing else but a set of objects and operations associated to them. The location of the underlying application on the

network and the operating system is transparent to the client. The CORBA specification is mainly composed of the following parts:

- The Interface Definition Language (IDL), a common neutral language to specify the object boundaries and its interfaces with potential clients. It is the way promoted in CORBA to separate interface from implementation and thus to provide language-neutral data types that make it possible to access the object with no matter of language the object is written in and system supporting it.
- The Object Request Broker (ORB), a software bus that establishes the client-server relationships between objects and seamlessly interconnects multiple object systems.
- The CORBA object services: a common collection of system-level distributed middleware services, complementing the functionality of the ORB. These services are packaged as components with IDL-specified interfaces: because IDL provides operating system and programming language independent interfaces to all the services and components that reside on a CORBA bus, this is true for the specified CORBA services as well as for specific business client applications.
- The inter-operation architecture, defined for the communication between different ORB implementations, which consequently can be interconnected using CORBA 2.0 inter-ORB services.

The ORB (see figure 4) plays the role of a broker between a client and a server. It is the middleware spinal chord, allowing to access objects independently of their locations on the network. Thus, when a client invokes a service attached to a given object, he does not need to know who (and where) is the server on the network implementing this service, and similarly, the server has no idea of the client invoking him. Everything is managed by the ORB acting as a link between the client and the server, essentially through message (request and reply) passing. Such a notion is a true conceptual improvement with respect to the underlying notions of classical client/server architectures. As the ORB is in charge of finding the appropriate server for a client request, it manages a specific repository in order to realise the task: the Implementation repository. The Implementation repository is a runtime repository of information that allows the ORB core to locate and activate implementations of objects. It is also the common place to store additional information associated with implementations of objects. Finally, the ORB also must have knowledge of the services offered by servers through the objects interfaces, with the help of the Interface repository. It is a runtime database containing persistent objects that represent the IDL information in a form available at execution, i.e. a description of all the IDL interfaces under the form of metadata usable by the ORB along with an API allowing components to dynamically access, store and update metadata information.

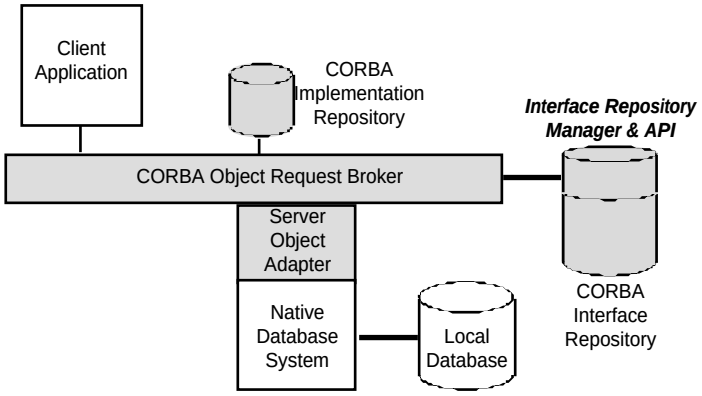

Figure 4 CORBA System Architecture

On the other side, the CORBA services [Object Management Group, 1995] are an added interesting feature with respect to classical client/server architecture: application developers can elaborate their components without any concern for middleware services, and then can mix the components with any combination of the CORBA services through IDL specifications (customising the application objects via sub-classing and multiple inheritance).

Introduction to Workflow Management

What workflow management comprises is probably best explained with the following quotation from Coleman and Khanna [Coleman, D. and R. Khanna. 1995]: 'Workflow is often explained with the analogy of the factory floor. In America, manufacturing made great strides in productivity during the late '80s and early '90s, most due to automation. Now, visionaries want to take the automated processes of the factory floor and apply them to the office.' Initially, there were some exciting success stories of applying workflow tools and techniques. Automating production-oriented processes like processing claims forms or loan applications has yielded dramatically lower cycle times for processing this paperwork and improved customer service. This has translated into a competitive advantage for organizations using these tools. Additionally, competitors are frantically trying to catch up and surpass early workflow tools users.

The strategic issue of workflow systems is to bridge the gap between the world of processes and the world of data modelling, as Gawlick et. al. [Gawlick, D., M. Hsu, and R. Obermarck. 1994] state. Figure 5 shows this. While business process engineering focuses on process management, computer support for operations focuses on data modelling.

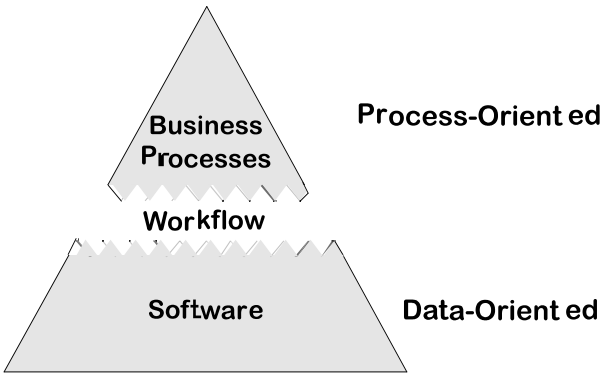

Figure 5 Workflow management bridges the gap between business processes and available software support.

The Workflow Management Coalition (WfMC), founded in 1993, is a non-profit, international organisation of workflow vendors, users and analysts. The WfMC's mission is to promote the use of workflow through the establishment of standards for software terminology, interoperability and connectivity between workflow products. Up to now, the WfMC has specified a Workflow Reference Model [WfMC. 1994], a glossary [WfMC. 1996 a] for workflow management, the interfaces 1, 2, 4 and 5 of the WfMC's reference model as shown in Figure 6 and a common Workflow Process Definition Language (WPDL) [10 WfMC. 1996 b]. Furthermore, the WfMC has shown demonstrations for the interoperability of Workflow Management Systems (WfMS).

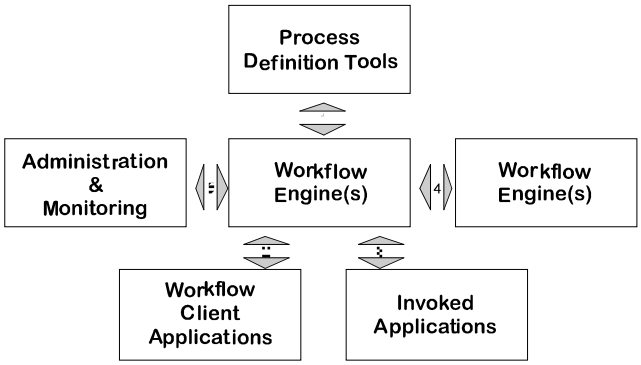

Figure 6 The Workflow Management Coalition Reference Model

The functionality of a WfMS is achieved by the interaction of several parts:
- process definition tools for the definition (i.e., the modelling) of workflows;
- workflow engines as the proactive computer systems that execute and control the processes;
- invoked applications, which are mostly legacy applications, e.g. word processors;
- workflow client applications, which act as the interface for an agent (i.e., a human being or another computer) with the WfMS;
- tools for administration and monitoring of the workflow process.

COAST: A global infrastructur for EXPRESS models

This section describes a more technical presentation of the COAST infrastructure [OMG. 1996], first by showing how COAST is embedded into CORBA, followed by an Introduction of its various components, and then giving a functional description on the basis of these components. The COAST is specified to offer powerful dynamic mechanisms as it is intended in a late binding approach.

ARCHITECTURAL CONTEXT

As the name suggests, the baseline technology for the COAST system is CORBA. A set of services is grouped around this middleware platform, which implement the special services of COAST. These services are partially standard OMG services, and partially native COAST services. The classification of COAST can be best described in the context of OMG's Object Management Architecture:

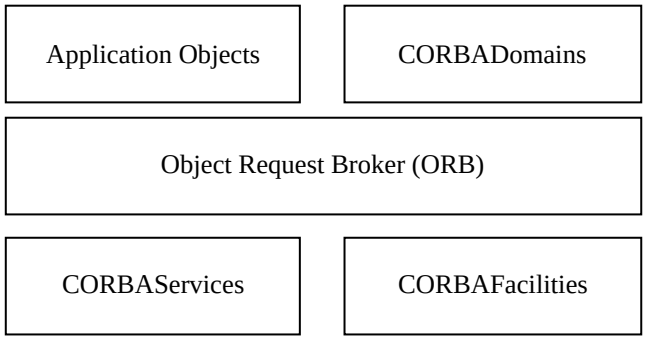

Figure 7 Object Management Architecture

The Object Management Architecture Guide (OMAG) describes OMG's technical objectives and terminology and provides the conceptual infrastructure upon which supporting specifications are based. The guide includes the OMG Object Model, which defines common semantics for specifying the externally visible characteristics of

objects in a standard implementation-independent way, and the OMA Reference Model. [OMG. 1996]

The Reference Model identifies and characterizes the components, interfaces, and protocols that compose the OMA. This includes the Object Request Broker (ORB) component that enables clients and objects to communicate in a distributed environment, and four categories of object interfaces:

- CORBA Services are interfaces for general services that are likely to be used in any program based on distributed objects.
- CORBA Facilities are interfaces for horizontal end-user-oriented facilities applicable to most application domains.
- CORBA Domains are application domain-specific interfaces.
- Application Interfaces are non-standardized application-specific interfaces.

COAST can be considered as a new vertical service which is settled within the CORBA Domains section. To accomplish its work, COAST uses the Object Request Broker, CORBA Services and CORBA Facilities. In addition, COAST supplies additional services and entities as described below.

SYSTEM OVERVIEW

COAST is a true object-oriented access method, which supports by default distributed and heterogeneous environments. However, following the object-oriented paradigm, all details about distribution, heterogeneity and storage schema details are completely hidden from the application using COAST. All activities carried out by the COAST are transactional, thus supporting save information sharing.

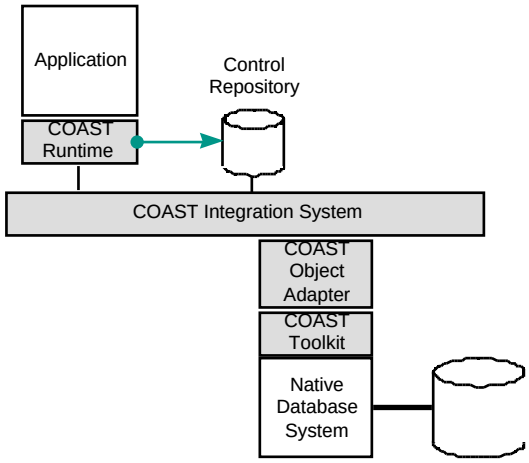

Figure 8: COAST System Architecture

Another classification can be done into client components, server components, and core components, complemented by some independent management components. The following list categorizes the various components and gives a brief description of each of them.

COAST Runtime System [client] This component implements the semantical functionality of the client part of the COAST system. It provides to the application an interface with late binding information access semantics to data models described by an EXPRESS schema. These models may be located at any storage server known to the COAST system. The distribution and explicit location of these servers are hidden to the application. Usually, the runtime system will be called by the Language Binding Interfaces, which provide the COAST API to the application conforming to a specific programming language.

COAST Control Repository [core] This repository stores information about available information models, available schemata, existing storage servers, etc. Furthermore, the control repository holds required information for the Object Relationship Service, the Object Transaction Service, and other CORBA services. The control repository is automatically self-replicating within a COAST domain.

COAST Object Adapter [core] It is the CORBA conform connection facility, linking the server methods with the CORBA core system. This specialized object adapter cooperates with the control repository to resolve invocations on distributed and multi-branch models linked by the Object Relationship Service. It also provides an extended object identification facility.

COAST Server Toolkit [server] This is a toolkit implementing the storage system independent server functionality. It will also provide assisting methods to bridge the gap to the native storage system functionality.

The *COAST Integration System* is the middleware level, relying on an ORB implementation and including an implementation of some of the OMG CORBA Services like the Object Relationship Service (ORS) to link distributed and multi-branch models and schemata together, the Object Transaction Service (OTS) for transactional behavior, and the Object Persistence Service (OPS) for objects long-term existence and management of object storage through a single common interface. It also connects the following components (see figure 8):

COAST Metadata Management System [core] A special object-oriented repository maintaining an internal representation of any schema following EXPRESS semantics. This component will intensively cooperate with the COAST Control Repository, and play a role similar to the Interface Repository in standard CORBA implementations, as a dynamic Metadata Repository for the COAST ORB.

COAST EXPRESS Compiler [management] This is a specialized EXPRESS compiler, which translates an EXPRESS schema into an object network conforming to the internal representation in the COAST Metadata Management System.

COAST Schema Loader Toolkit [management] A collection of routines to build a storage system specific schema loader as backed to the Metadata Management System.

COAST Management Tool [management] An interactive tool to fulfill all kinds of system management activities related to COAST. It allows the user to monitor and influence the operation of the Integration Services during runtime, assists application programmers, and provides fault isolation tools.

APPLICATION SUPPORT

Applications are not burdened with the need to know the actual distribution situation of stored information, nor is there any need to navigate to a particular storage server. Any request issued by an application is entered into the COAST system via a Language Binding routine which calls the appropriate Runtime System functions which submit the request to the COAST ORB Core. By using the Language Binding routine, the application only needs to specify the name of the model it intends to access, and the COAST runtime system locates the correct storage server and establishes the access path to the requested model. This is possible due to the additional context information from the session object and the navigational assistance from the control repository.

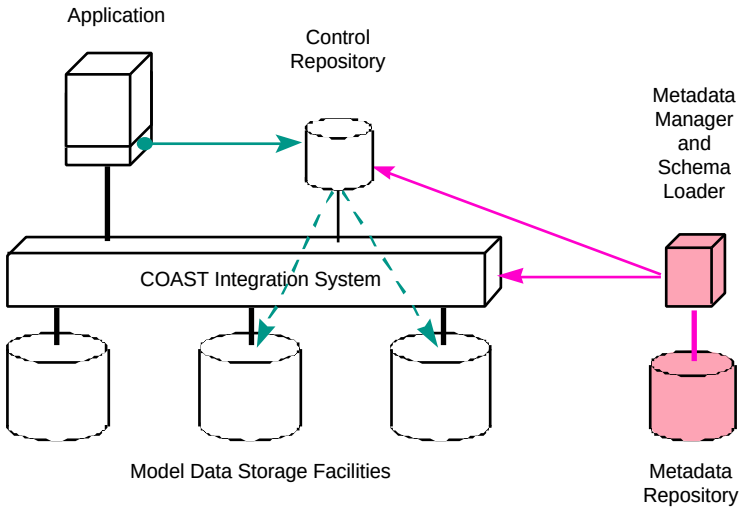

Figure 9: COAST Functional Overview

After an access path to a particular model has been established, the COAST runtime system will continue using this path until it is explicitly closed or an unrecoverable failure occurs.

If the request is a schema information request, the Control Repository implementation will directly satisfy the request with assistance from the Metadata Management System which holds the information of available schemata. If it is a data access request, the object adapter of the located storage server will activate the corresponding method routine. This routine will fulfill the request by accessing the storage system through storage system-specific access code based on the Server Toolkit. Any exception is handled finally by the top level method routine.

STORAGE SYSTEM AND SCHEMA SETUP

Any new application schema to be handled by the COAST system has first to be loaded into the Metadata Management System. This is done with the COAST EXPRESS Compiler. From the internal form, a database-specific schema has to be generated with an appropriate schema loader backend. The schema can then be loaded into the storage system using the normal CORBA transport mechanism (provided the storage system allows run-time schema loading). The top-level set-up of the storage structure is done with the interactive management tool. To ensure maximum information integrity, the COAST architecture relies on transactional technology. No access to stored information is permitted outside the context of a valid transaction. Therefore „access“ and „open“ operations are distinguished in several places. The „access“ operation only establishes the communication path to the requested object and registers the access intend if applicable. However, to actually reach the stored information, a transaction has to be started and the object must be opened for the intended access mode.

RUNNING APPLICATIONS WITH THE COAST

As mentioned previously, two approaches can be envisaged (and possibly mixed) in CORBA, a static one based on IDL stubs and skeletons, and a more dynamic one based on ORB requests construction «on the fly». STEP-based applications having to inter-operate in a LSE project have to be plugged to the COAST either through IDL interfaces or using a COAST Application Programming Interface (API). This allows applications to either use a complete and natural integration or the COAST-Access on a rather wrapper-like behavior. The last approach allows the use of COAST without rebuilding a legacy application but assumes, the application has some proprietary interfaces which could be wrapped.

Workflow Management

The application of workflow management has to date been limited primarily to the control of administrative, document-based processes in individual organisations. The aim of the VEGA project is to apply workflow management to both the business and engineering activities across a virtual enterprise. This introduces several new requirements for workflow management in two different categories:

- requirements from managing processes in the LSE industry;
- requirements from managing processes across company boundaries.

The processes in the LSE industry include not only administrative but also technical activities, such as design and engineering, fabrication of elements, assembly and erection of the product, maintenance and demolition. Moreover, LSE projects are characterised by the large number of participants who are collaboratively and concurrently developing the LSE product. Consequently, there are many parallel processes that need to share information and need to be co-ordinated on a regular basis. In other words, unlike administrative processes in the traditional application domain of workflow management, LSE processes are complex processes that require flexible workflow management systems that can deal with frequent changes in activity sequences, parallel processes, co-ordination of work and information. In addition, because of the inherent complexity of LSE processes, it should be possible to control the workflow process on different levels of detail, ranging from the level of project milestones to the level of individual activities.

Another requirement, in addition to the support for complex processes, follows from the VEGA vision. The vision is a software architecture that enables project partners from different companies to work together in a virtual enterprise and share and access information without any problem. One of the three components of the VEGA solution is the use of product-data technology to describe and store meaningful product information. The VEGA vision requires an integration of product-data technology and workflow management. Document-based workflow management systems, as they can often be found in the traditional application domain of workflow management systems, are therefore not suited. The VEGA vision requires a tight coupling between the models describing the workflow process and product data models containing the information processed in the various activities in the workflow.
To summarise, workflow management for virtual LSE enterprises has to be able to deal with the complexity of collaborative and concurrent engineering activities and has to be able to link the relevant product data models to the workflow process.

CROSSING COMPANY BONDARIES

Workflow management in a virtual enterprise requires interoperability of the different WfMSs in the companies participating in the virtual enterprise. Because the communication between the different organisation will use a public network, firewalls

are required to ensure security of project information, organisation specific information and organisation specific working procedures (see also [Amar, V. et al. 1997]). The workflow management systems (or rather the workflow enactment services in WfMC terminology) must be able to interoperate across the firewalls between the organisations, as shown in Figure.

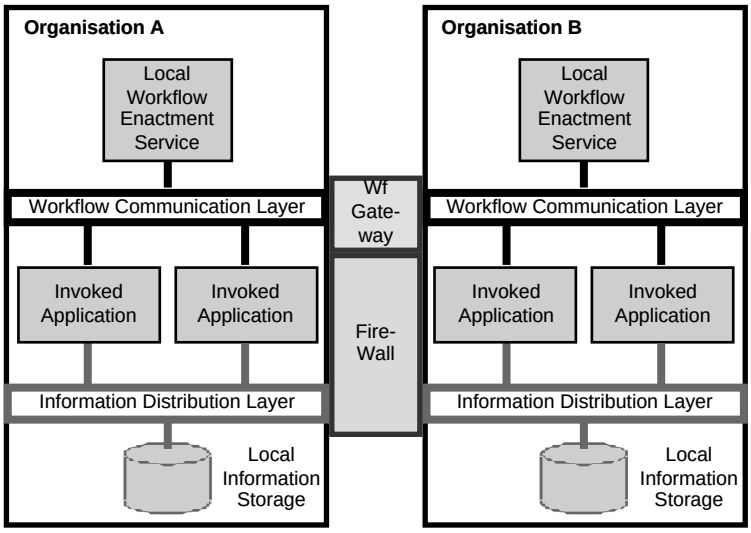

Figure 10 The WfMSs must be able to interoperate across the firewalls between organisations.

The interface between the workflow enactment services (interface 4) as described in the Workflow Reference Model of the WfMC (Figure 10) is of special interest here. As (sub)workflows are executed by different WfMSs, these systems need to interoperate and exchange information to achieve a systemwide workflow. A systemwide workflow is the sum of the distinct workflows in the respective participants of the virtual enterprise. The WfMC has demonstrated the interoperability of WfMSs based upon the interface 4 definition of the workflow reference model several times. The first demonstration of this kind took place only last year, in June 1996. Further demonstrations have been undertaken. Currently, the exchange of information is based upon email. This is a simple solution with many deficits. The VEGA platform will offer a more reliable solution.

As the different WfMSs have only a restricted view of the (sub)workflows they are involved in, a global monitoring service needs to be introduced to achieve monitoring of system wide workflows. Monitoring workflows means to analyse their runtime information. In addition, the objects which are created or used during this life-cycle are

of interest. The global monitoring service also requires a solution that can cross the firewalls between the companies in the virtual enterprise.

To summarise, workflow management across company boundaries requires WfMSs that interoperate across firewalls and a global monitoring service that is able to monitor across firewalls.

THE VEGA WORKFLOW MANAGEMENT SUPPORT

VEGA will meet the requirements described above by developing:

- a workflow process meta-model to define workflow processes in virtual enterprises and to link product model data to the workflow definition.
- a workflow management architecture to manage workflow across company boundaries.

These two components of the VEGA workflow management solution are discussed below.

THE VEGA WORKFLOW PROCESS META-MODEL

The objective of VEGA is to specify a workflow meta-model that meets the specific requirements of concurrent engineering processes in virtual LSE enterprises. The core of the VEGA workflow meta-model is the model behind the workflow process definition language (WPDL) as specified by the workflow management coalition (WfMC) [WfMC, *Interface*. 1996]. The WfMC aims at the specification of international and open standards for the interoperability in workflow management. This is the main reason why this language has been chosen as the core of the VEGA model. Currently, the WfMC specifications are up for international ISO standard. Moreover, the WfMC WPDL is a generic and extendible meta-model of workflow processes. Taking this specification as a baseline for the VEGA meta-model increases the chance that useful extensions can and will be adopted by the WfMC and vendors of workflow management systems. This conforms to the objective of the VEGA project: to specify and realise an integrated software architecture based on open standards.

The main entities in the WPDL are the workflow process, the workflow activity, the transition, the workflow participant, the workflow application and the workflow relevant data, as shown by the EXPRESS-G diagram of Figure 11. The WPDL specifies the attributes for these entities and the relationships between them.

The VEGA workflow meta-model extends the WfMC WPDL in several places. Modelling constructs missing in the WfMC-WPDL schema are:

- a way to model the participant responsible for an activity
- a way to assign multiple participants to an activity instead of only one participant
- the ability to model the position of a participant in an organisational unit (It is only possible to model the role of a participant in a workflow process. VEGA deliverable

D501a specifies the requirement for modelling positions, or organisational roles, too)

- a way to model arbitrary collections of participants as a group
- a way to model the relation between a workflow activity and the relevant product model schema, e.g., an EXPRESS schema.

The VEGA workflow meta-model adds several constructs to address these shortcomings.

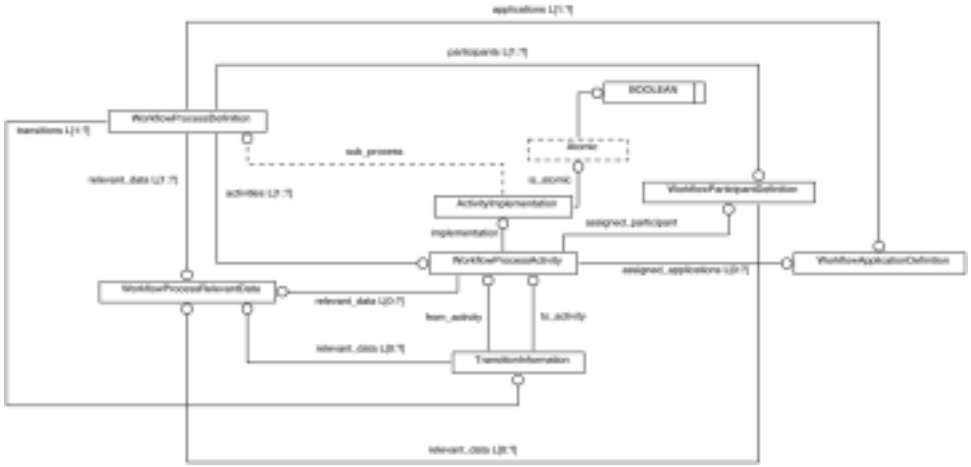

Figure 11 EXPRESS-G diagram of the minimal meta-model of the WfMC workflow process definition language.

THE VEGA ARCHITECTURE FOR WORKFLOW MANAGEMENT

The VEGA workflow architecture must enable WfMS interoperability and workflow monitoring across company boundaries.

Interoperability will be realised through a CORBA-binding as already demonstrated by Digital Equipment Corporation in the DeTeBerkom-project VorTel [VorTel. 1996] This is a more reliable solution than using Email. To achieve the interoperability across organisation boundaries, VEGA introduces workflow gateways, as shown in Figure. These are intended to connect the workflow engines of the respective WfMSs and to work on top of the organisation's firewalls to assure that workflow-related information can cross company borders. The workflow gateways assure, that only that workflow information is exchanged that is intended for partner companies. Any other workflow-information will be kept secure within an organisation's network.

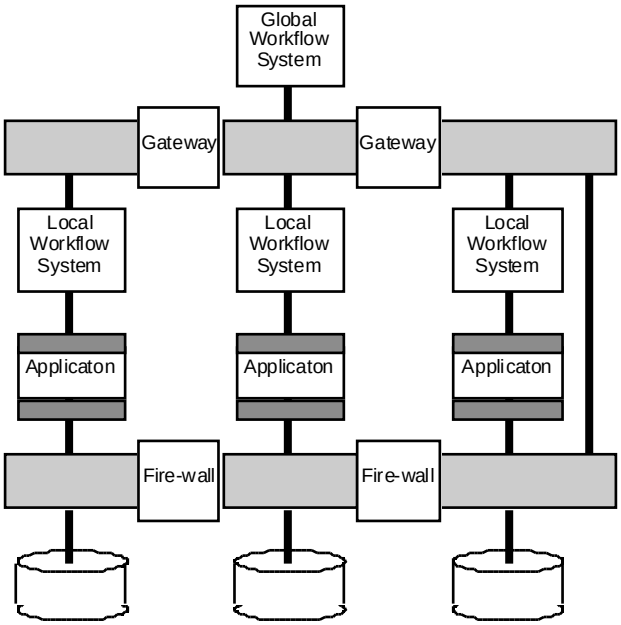

Figure 12 Workflow gateways to cross company boundaries.

For global workflow monitoring [Schulz, K. 1996]each local WfMS puts its information that is public within the virtual enterprise in a (logically) centralised database where all runtime-information of workflows is collected. An additional tool, a global workflow monitor, can access the database to show and analyse the information. Thus a workflow monitor is able to supply the following services:

- Global State Query: current state of workflows and their activities, which can be “Ready”, “Blocked”, “Started”, “Paused” or “Finished”.
- Improvement of capacities by redefining critical workflow-parts, i.e., activities or sub-workflows.
- Class-Improvement to apply workflows closer to reality.
- Object-Tracking for information about versions, actors, duration, etceteras of used objects within the workflow.

The WfMSs which interoperate in a virtual enterprise form a distributed workflow enactment service as described in Figure . To access workflow-related information like workflow process descriptions they access the CORBA Access to STEP information storage Architecture (COAST) to be developed in the VEGA-project. This architecture defines a new, third access method to stored information described by an EXPRESS schema. COAST is defined as a service layered on top of the Common Object Request Broker Architecture (CORBA), which has been defined by the Object Management Group (OMG). As a CORBA service, COAST conforms also to the OMG Object Management Architecture (OMA) [OMG, 1995][OMG, 1996].

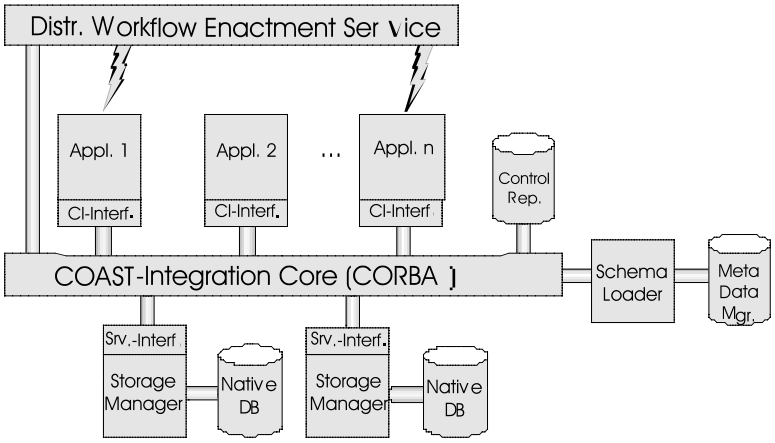

Figure 13 Workflow-COAST interaction within VEGA

COAST is intended to work independent from the WfMSs. The WfMSs form an additional COAST-Service. They access applications which participate as invoked application in a workflow via interface 3 of the WfMC Reference Model.

References

Amar, V., et al. [1997]
An open STEP-based distributed infrastructure: the COAST platform. in Concurrent Engineering in Construction. 1997. London, UK.

Coleman, D. and R. Khanna, [1995]
Groupware: Technologies and Applications. 1995: Prentice Hall.

Fowler J. [1995]
STEP for Data Management, Exchange and Sharing. Technology Appraisals 1995.

Gawlick, D., M. Hsu, and R. Obermarck, [1994]
Strategic Issues in Workflow Systems, ., Digital Equipment Corporation: Palo Alto, California, USA. 1994

Industrial automation systems and integration [1994a
Product data representation and exchange Part 11. Description methods: the EXPRESS language reference manual, 1994

Industrial automation systems and integration [1994b]
Product data representation and exchange Part 21 Implementation methods: Clear text encoding of the exchange structure, 1994

Industrial automation systems and integration [1995a].
Product data representation and exchange Part 22. Standard Data Access Interface,1995

Industrial automation systems and integration [1995b]
Product data representation and exchange Part 24. C Programming Language Binding to the Standard Data Access Interface Specification 1995.

Industrial automation systems and integration [1996]
Product data representation and exchange Part 23. C++ Programming Language Binding to the Standard Data Access Interface Specification 1996.

Köthe M., Schulz K., Amar V., Zarli A. [1997]
COAST Architecture: The CORBA Access to STEP Information Storage Architecture and Specification, Deliverable D301 in the ESPRIT-Project 20408 „VEGA“ - 1997.

Mowbray T. J. & Zahavi R.:
The Essential CORBA - System Integration Using Distributed Objects , John Wiley and Sons.

Object Management Group [1995]
CORBA Services: Common Objects Services Specification, Revised Edition, 95-3-31. 1995.

Object Management Group, [1995]
The Common Object Request Broker: Architecture and Specification Revision 2.0, . 1995, Object Management Group.

Object Management Group [1996]
CORBA services: Common Object Services Specification, . 1996,

Object Management Group [1997]
Product Data Management Enablers; Request For Proposal 1997

Object Management Group [1996]
Document: mfg/96-08-01, 1996

Otte R., Patrick P. Sr., Roy M. [1995]
Understanding CORBA The Common Object Request Broker Architecture 1995.

Schulz, Karsten, [1996]
Monitoring of Systemwide Workflows. Fachhochschule Furtwangen, Germany 1996

Siegel J.& Al [1996]
CORBA Fundamentals and Programming, John Wiley and Sons, 1996

WfMC, [1995]
Coalition Overview, Workflow Management Coalition: Brussels, Belgium.1995

WfMC, [1994]
The Workflow Reference Model, Workflow Management Coalition: Brussels, Belgium.1994

WfMC, [1996]
Terminology & Glossary, Workflow Management Coalition: Brussels, Belgium.1996

WfMC, [1996]
Interface 1: Process Definition Interchange, ,W.G. 1/B, Editor., Workflow Management Coalition.1996

Workflow-Teleservices. [1996]
Joint project, sponsored by the DeTeBercom GmbH, Berlin. Finalised June 1996. Participants: Digital Equipment ; FHG Dortmund, GMD Darmstadt, IBM, TU Dresden.