REQUIREMENTS AND TECHNOLOGY INTEGRATION FOR IT-BASED BUSINESS-ORIENTED FRAMEWORKS IN BUILDING AND CONSTRUCTION

RECEIVED: June 1999
REVISED: November 1999
PUBLISHED: December 1999 at http://itcon.org/1999/4/
EDITOR: Z. Turk

Alain Zarli
Centre Scientifique et Technique du Bâtiment, Sophia Antipolis cedex, France
email: zarli@cstb.fr

Olivier Richaud
Centre Scientifique et Technique du Bâtiment, Sophia Antipolis cedex, France
email: richaud@cstb.fr

SUMMARY: *Key challenges nowadays facing industry are increased competition, increased complexity and wider market reach, and in such a context, the information infrastructure of an enterprise determines its strengths and weaknesses. As a consequence, this infrastructure has become vital to enterprise competitiveness, though the diversity of enterprise databases and the heterogeneity of strategic applications are still a barrier to industrial exploitation of the opportunities offered by this infrastructure. Modern enterprise information systems must interoperate in Inter/Intranets and with the WEB in a quite interactive, reliable and secure way, and have to be flexible enough in order to quickly adapt to today's fast moving business environment. This paper first gives a synthesised investigation of the requirements for a standardised open infrastructure, relying on now available distributed objects systems, and integrating in a flexible way the enterprise business model through the emerging concept of Business Objects (BOs), that allows systems designers to put the stress on the business they model and no more the data they exploit. It then focuses on a specific part of the work undertaken in the context of the Esprit project, relying on the OMG Business Object Component Architecture (BOCA) proposed by the BODTF. This was finally not accepted by the OMG board. Nevertheless, at the time we started our work, this proposition was the only one meeting requirements for distributed business objects, and especially the Component Description Language (CDL) and its concepts: a short presentation is given of a CDL compiler that produces IDL, according to the BODTF recommendations for CORBA-based implementations, and Java code, on the basis of a framework we developed. The originality of this approach resides in the fact that it takes into account most of the needs when developing BOs and gives an automated implementation whenever possible. Hence, we automated the generation of factories, event typed dispatching, and relationship handling. This approach lets the BO developers concentrate on the business and relies on improved solution backed by design patterns. This research is regarded as a solid foundation for designers to set up information systems that are a better fit to business user requirements, and expected to be a major step towards the forecast delivery of WEB-oriented software components for the Building Construction and other sectors as well.*

KEYWORDS: *distributed client/server architectures, communication middleware, business objects, CORBA, CDL, open and standardised industrial business objects frameworks.*

1. INTRODUCTION AND CONTEXT

Corporations are today becoming largely distributed, and deeply founded on networking technology allowing to share and access information in different locations. Meanwhile, computer-based information systems have become the spinal chord of modern enterprises, and new appropriate information tools satisfying fast reactive business requirements and offering a strategic corporate advantage are occurring. However, the so-called Virtual Enterprise (VE), which binds A fragmented and geographically spread set of partners collaborating together, today still needs the elaboration of new powerful frameworks to support business models, concurrent enterprising, access to large corporate data sources and multimedia information management, within Intranets, Extranets and even the Internet. The vital information for the future business of companies must be easily

accessed and manipulated in a safe and comprehensive way by multiple actor-oriented applications, thereby satisfying the need for improved customer service, on-time delivery, quality management and project co-ordination.

After an identification of the current major needs and requirements for computer-based support of business processes with a focus on the Building and Construction (B&C) sector, this paper emphasises on the possible use and integration of some new advanced standard based technology for the design and development of standardised, flexible, and upgradable information systems, embodying networking and object-oriented (OO) distributed systems technology that are today recognised as the foundations for concurrent engineering in all sectors, including the construction industry.

Despite the fact that advanced computer technology, including Client/Server and distributed-object computing, and Internet/WEB technology, provides reliable and relevant mechanisms and tools for Product Data Management in the large, companies still deal with intricate and non flexible corporate information systems. Indeed, Information Technologies (IT) imply an increasingly complexity in software architecture, development and use. To mask this complexity, we promote the concept of Business Objects (BO), which are related to software components encapsulating business rules and aiming at providing secure sophisticated access to diverse electronic content and software applications. BOs are defined as components of the information system representing the enterprise business model, and are to be promising enablers to build information systems meeting end users and customers requirements, thus revealing critical to the success of the enterprise. Of course, those BOs are supposed to rely on the so-called N-tier architectures based on transactional distributed objects systems as the CORBA (Mowbray and Zahavi, 1996, Siegel, 1996, Orfali et al, 1996) architecture supported by the Object Management Group OMG (http://www.omg.org), leading to some high level glue between clients and enterprise data.

Thus, component-based development is the next step in the evolution of object systems, and a major improvement: uniting all the components needed to run the entire IT system will enable to reach with low fixed costs new categories within the market, making enterprise-level functionality available to the broadest universe of users. An integration of such concepts have been specified in the context of the WONDA project. WONDA delivered the specification of an open framework, based on a scalable tier-less component-oriented architecture and relying on standards, leading to a comprehensive solution for future enterprise information systems, electronic commerce, and electronic publishing. The combination of new concepts such as BOs and technologies based on standards (either *de jure* ones like official norms, or *de facto* ones in their large industrial use but still leading to openness), is supposed to open the door for substantially enhanced integration of value-added networks, high level application development, and distributed object middleware. This combination will radically simplify and reduce time for the deployment and management of fashionable distributed applications providing corporations with a competitive advantage, especially on the Internet market.

This paper describes enabling technologies and key components, specified within the WONDA project, that constitute a new-fashioned way of designing and deploying secure and open BO-oriented framework. It is structured as follows: it first tries to give an overview of current industrial trends of the Virtual Enterprise, that is a consequence of a world-wide increased competition and mutation. Technologies issues are then investigated, showing that there is a technology and market gap, especially with respect to secure and improved information systems for the SMEs (Small and Medium sized Enterprises). We then introduce and detail the BOs and components concept, as an enabling technology of industrial importance with respect to user needs and practice. A focus is placed on componentware state-of-the-art, definitions and standardisation tendencies for BOs, integration of BOs in a 3-Tiered architecture, and emerging implemented models as potential candidates for a BO framework. The last part before the conclusion focuses on the presentation of a CDL compiler automatically producing IDL specifications and Java code as well, as specified in WONDA, thus facilitating the elaboration of a BO-based information system for distributed companies.

2. REQUIREMENTS AND TECHNOLOGY GAP

2.1 Virtual Enterprises

Facing an increasing complexity of product development along with an intensifying market competition, the Virtual Enterprise (VE) (Hardwick et al, 1996, Hardwick et al, 1997) appears nowadays as a necessity within

nearly all the industrial fields, when even large enterprises are no more be able to design and produce all the different parts of a product, due to time constraints and the lack of some widely required specific expertise inside the enterprise. The necessity for VEs can be illustrated by the results of increased competition, change and complexity:

- **Best of Breed:** Firms have to provide Best of Breed solutions which means that they now have to concentrate on their own core competences and outsource for the right quality components/services for the right project. Effective outsourcing can necessitate partnering with suppliers within a VE.
- **Time to Market:** The race in time-to-market necessitates shorter development times which is enabled by re-use of out-sourced components. Again VEs are vital.
- **Shorter Product Cycles:** Increased change and competition necessitate increased agility which can only be achieved by a flexible organisation. Such an organisation concentrates on its core competence and re-invents its offerings by re-configuring its VE per project.

Though often considered as a traditional industry, the construction industry is quite a good example of such a situation. Like many other engineering sectors, the end product of construction is a value-added arrangement of standard component parts, designed and constructed by non co-located teams of separate firms who come together for a specific project. The building must be designed for functionality, safety, longevity, and aesthetics. Designers consult regulatory, best practice, pricing, aesthetics and proprietary product information, and design services. The main difference with other engineered products is that a building is most of the time a unique prototype. The design team comprises up to 7 disciplines: property developers, project managers, civil engineers, architects, surveyors, building services engineers, and contractors. All these factors, among others, participate to the fact there is a crucial demand in construction industry for solutions in the VE enabling to manage software incompatible applications running on heterogeneous platforms and systems, data exchange and interoperability mechanisms between applications managing different types of information with different levels of performance and functionality, together with powerful means of communication between most of the time distant applications. This would lead to more agile manufacturing in an industry characterised especially by a large number of SMEs, with a large group of very small businesses.

2.2 Technology issues

Simply considering the product information modelling and exchange area, the complexity and large scope of the problem has involved numerous industrial actors in establishing new standards for improved communication between applications used by different project partners, in order to get better productivity. This has lead to boot major changes in the last few years in industrial enterprises, especially those devoted to the construction of large-scale engineered products, for example in aerospace, shipbuilding and automotive industrial areas. A typical example is STEP (Björk, 1996, Fowler, 1995, ISO, 1995), a nowadays well-known standard for real world product information modelling and interpretation, data exchange and actor co-operation. STEP (STandard for the Exchange of Product data) is an International Standard for the representation and exchange of product data, developed in ISO TC 184/SC 4 (Industrial data and global manufacturing programming languages). It allows the expression in a uniform and complete way of the whole information required for a product during its life-cycle through the EXPRESS language, together with means for exchanging data physical files through STEP Physical Files and sharing product databases through data and application independent mechanisms (SDAI: Standard Data Access Interface). Another example in the specific construction domain is the Industry Foundation Classes (IFC) (IAI, 1999), a universal model 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. As a leading industry driven initiative in the Architecture, Engineering in Construction (AEC) sector, the International Alliance for Interoperability (IAI), which is a non-profit alliance of the building industry including: architects and engineers, building clients, software vendors, and so on, is pushing the IFC as a *de facto* standard in Building industry in a very near future.

Thus, the VEs of tomorrow are characterised by decentralisation and numerous location of actors to be networked, project - rather than enterprise, department or even legacy data - centred activities, tighter partnering and connection in Intranet and with suppliers, customers and banks, and complex interdependencies like outsourcing, logistics, cash flow, etc. In order to both simplify and improve co-ordination and relationships between activities, there is an inquiry for software product components that are seamlessly interoperable across

many boundaries, and for a framework managing the interoperation of these components across and within different application domains. The enterprise information systems are now focused on arming all the VE actors with the whole information needed to assess the above factors, leading to new requirements for those enterprise information systems, which are:

- Seamless access to any enterprise-wide information source, on the base of end-user oriented views on the physical data sources. Such an access must be totally transparent and independent of the server, as today, a VE is mostly composed of a set of pre-existing databases supporting different models and architectures: those databases have been inherited over the past years and reflect the internal changes that occurred in the enterprise, and they are heterogeneous in the sense that they operate in different environments and support unrelated data models such as relational or object-oriented models, data definition and data manipulation facilities.
- Communication and interoperability which are supported through mechanisms like messages passing, data shipping, etc., have been brought at the extreme, as clients and servers in these new VE frameworks are far from being homogeneous and compatible. This means that any new information and/or application can be directly deployed across the VE boundaries. Furthermore, granularity for interoperability must be no more at level of applications , but at level of the application objects themselves, allowing better leveraging of the whole enterprise information systems.
- Flexibility of the infrastructure to accommodate any particular business logic, including dynamic building and adaptation to the business environment. Considering one step ahead, business rules and information knowledge can be the grounds of more value-added high level features like decision-makers facility, data mining, rule-based capabilities and managing functions, on the basis of advanced methods or reasoning applied on data.
- Component-based development: components must offer the opportunity to be used individually to meet specific application requirements, and to be assembled as mission-critical business components at demand, in the context of a distributed architecture and in order to provide a full integrated framework for business solutions.
- Openness, dealing with information modelled through well-known and standardised formats, as well as the capability to seamlessly integrate other components or libraries potentially built on other technology in the global enterprise information framework.
- Scalability, with full functionality and performance regardless the number of users, processes, transactions and data access.
- Security, both in terms of information access and circulation - identification, authentication, access control and message encryption - and information integrity through transactional behaviour of the system, and non repudiation.

2.3 The technology gap

With respect to the business and technological issues as previously raised, and especially considering the construction sector where the drive towards VEs is patently characterised by partners who are smaller, more numerous, changeable, distributed, and with heterogeneous data and applications, current technology solutions have one or more of the following characteristics:

- **Homogeneity:** solutions are fixed and not open, with a lack of support for legacy, as well as new, upcoming systems in terms of hardware, software, databases, and networks.
- **High Entry Level:** solutions are often too much expensive to buy for SMEs. There have to be more entry levels, e.g. from cheap personal to costly enterprise editions.
- **Lack of Scalability:** limited growth path in terms of hardware and software.
- **Application Centric:** need to organise the enterprise around the application.

- **Fixed Infrastructure:** need for leased lines between partners, restricting location independence and requiring long term relationships.
- **Lack of Support for Business Processes***:* limited security and transactional support.

To enable VEs by providing the ability to transact business processes seamlessly and to capture, access and assess the state of the business/project., industry needs a combination of:

- Low Entry Level,
- Scalability,
- Open Infrastructure & Location independent access,
- Enterprise Information (i.e. seamless capturing of the state of business from distributed legacy data),
- Support for Business Processes,
- Security and Transactional Support.

As attempting to provide an open, distributed, and secure framework for electronic publishing and commerce and BOs supporting brokerage and construction, the WONDA project aims to satisfy these needs.

3. THE WONDA PROJECT OBJECTIVES

The information infrastructure is recognised as vital to Europe's competitiveness. Yet the diversity of enterprise databases, applications and components is a barrier to industries exploitation of the opportunities offered by this infrastructure. It is particularly difficult for industry's diverse enterprise databases to inter-operate with the WEB with full interactivity, reliability and security, and this is a barrier to the take-up of Enterprise Information Systems (EIS) and e-commerce.

The main aim of WONDA has been to specify an open and secure framework for BOs and electronic publishing and commerce. In this context, BOs are software components which encapsulate business rules and procedures and which can run anywhere on the network. They aim at providing customers with secure and sophisticated access to diverse electronic content and software components. BOs are defined as components in the information system representing the enterprise model and promise to be the building blocks of information systems meeting end users requirements critical to the success of the enterprise.

With respect to the global objectives of WONDA, the following technical characteristics to meet the needs of VEs have been investigated:

	Low Entry Level	Scalability	Security & Transactional support	Open Infrastructure	Support for Business Processes	Enterprise Information
Business Objects		*			*	*
Distribution Middleware			*	*		*
WEB Infrastructure	*	*		*		
Modular Security	*	*	*			
Database Federation		*				*
Standards	*	*				

Targeting openness has been one of the main WONDA's differentiating features especially in terms of:

- extendability (both functional and schema oriented),
- standards (which in turn enable database, platform and applications independence),
- scalability (both architectural and considering performance).

Extensibility can be enabled by schema federation and configurability. Standards can be aligned to World-Wide-Web Consortium (http://www.w3.org) and OMG. Scalability can be enabled by an open, modular and configurable framework, which can be viewed as the architectural hub of WONDA. It is called the WebMapper and is an enabler for BOs, electronic publishing and security solutions. These features can be achieved through a symbiosis of:

- federated database technology,
- generic mapping technology between the data from the database federation to the Inter/Intranet environments,
- security components,
- domain specific environments offering the WEB-interface tailored to the end-user requirements and the domain business logic.

The main components of the infrastructure designed in the WONDA project are depicted in the section 4.5.6 of this paper.

4. BUSINESS OBJECTS: CONCEPTS AND INTEGRATION IN FLEXIBLE ENTERPRISE INFORMATION SYSTEMS

4.1 Introduction and technology state-of-the-art

Due to the proliferation of the Internet and its standards (HTML, VRML, XML, etc.), the emergence of specifications for distributed object computing and architectures like CORBA from the OMG, or the Microsoft OLE/DCOM model, and large efforts around these middleware technologies like JAVA/RMI and JavaBeans, Active X controls or server-side components, etc., large corporations are on the way to move towards new object-based Inter/Intranet distributed computing architecture since few years, in order to build the next generation of enterprise-wide applications.

But these new architectures will involve various languages, systems and protocols, sometimes of low-level, while the end users expect to deal with high level application objects. Previous developments have been done in the past with respect to the issues and requirements identified herein above. With respect to linking and accessing information as contained in large enterprise databases and world-wide data banks and the WEB, one can mention several attempts like for instance:

- Connection of the WEB with relational or object-oriented DBS to generate and manage HTML pages, in particular dangling links (e.g. Oracle WebServer), but those developments have been most of the times customised for specific data stores.
- Enterprise object frameworks coupling multiple relational DBS and generating HTML pages with database information. An interesting example of such a framework are the WebObjects (Information about WebObjects can be found at the following URL: http://www.apple.com/webobjects), but they federate relational databases only, and don't provide language independence nor security.

Regarding components issues, the WEB has led to lot of developments too. For instance, the WebBots (Information about WebBots can be found at the following URLs: http://www.internetbay.com/bots.htm, http://www.netjammer.com/TRAINING/HPD/engines.htm, http://websupport.net/fpguide/fp97webbots.html) components are dynamic objects evaluated when saving or browsing WEB pages: they allow the query of WEB servers, the updating of WEB pages each time contents change, the inclusion of other pages or images on a page, and adds full text-searching capability. They also grant the insertion of advanced components through the use of scripts (VB script, Javascript) or Java applets. WebBots embody valuable features, but they only deal with client

side interfaces, and make transactions with WEB servers, not enterprise servers. Another open solution are the JavaBeans (Orfali et al, 1998), as they rely on the Java language and platform, supposed to be portable on any hardware and operating systems, but even JavaBeans focus on graphical components and environments for end-user interfaces.

Tackling the more general concept of intelligent agents, most of the today agent-based systems are custom architecture or proprietary frameworks with internal models, legacy script languages, connection to legacy databases, leading to less interoperability with other applications, poor integration in large enterprise information systems, and above all, less ability to extend - e.g. by direct integration of other pre-built components not based on the same agent network - and to follow the market evolution, even if some of them are business-oriented. Thus, despite notable advances, software technology still looks for improvements with respect to flexible and reusable components, and for reactive and adaptable corporate information systems. Actually, the enterprise information system is oftentimes a complex and inflexible mix of old fashioned solutions. Technological state-of-the-art does not reveal today systems or frameworks providing at the same time the following features:

- uniform access to the federation of all enterprise data;
- language neutrality;
- clear separation of data content and appearance;
- secure access to enterprise data;
- business oriented access to information;
- all models and components based on world-wide recognised and adopted standards.

A new solution coming to help with these issues expanded from OMG with Business Objects and the Business Application Architecture that consists of a specification of a standard framework for business applications. Though the initial OMG standardisation efforts have not been successful in terms of the emergence of a new standard, the main concepts of this work are still valid and fundamental from our point of view. The intention was to promote a standard framework for business application residing on top of CORBA. BOs are the «glue» between client applications and enterprise data contained in large data stores. They are expected to facilitate communication, design and modelling between implementers and business domain experts through the sharing of the same concepts. They have to model the real world so that people focus on main characteristics and relationships among BOs**.**

BOs are means to give high-level views on enterprise product data, related to various representations both throughout the lifecycle of products and for the various actors involved in the design/development/use of products. They can be associated to software components, and therefore can be assembled into *frameworks* to support high-level industrial products design and developments. When considering *distributed architectures*, they are the ultimate solution for the interoperation of business components in heterogeneous user views on product and computer implementation of the product data. Thus, while product data models correspond to a conceptual structural approach of data, the BOs conform to a more functional and process view on information, though relying of course on data structures. BOs are concerned with the definition of methods and operations available for objects, and possible queries on these objects. They equally have to manage information about objects, identifying constraints and relationships according to some process or business context. They can be related to ways of retrieving information like object browsers, query languages, keywords-based technologies, etc..

4.2 Definition and standard

A *BO* is an abstract representation of a concrete and active thing in a specific business domain in the real world, and uniquely identified by its name in this business domain. A possible definition is given below as stated by the OMG. *«A business object is defined as a representation of a thing active in the business domain, including at least its business name and definition, attributes, behaviour, relationships, rules, policies and constraint. A business object may represent, for example, a person, place, event, business process, or concept. Typical examples of BOs are: employee, product, invoice and payment.»*

Thus, a BO may act as a participant in a business process and try to mimic the way things are. Conceptually, BOs add value over other representation because they give a higher level view and package the main concepts of the business model they represent. Consequently, they ease the realisation of business applications while masking the complexity of the underlying communication and implementation means. Software components of the information system implementing those BOs manage the enterprise business model by encapsulating business rules, thus enabling a greater focus on business logic and application development.

As mentioned in X3H7 and RM-ODP (Kilov and Simmonds, 1997), a business rule consists of a set of things that are meaningful for the depicted business. As stated in X3H7: *"A business rule is a proposition about business things, relationships between them and operations applied to them, from the business enterprise viewpoint.»*

The major problem is certainly today the absence of standards for BOs, and thus no consensus with respect to the interfaces to provide for BOs. Standards like STEP or the IFC normalise product data entities at an appropriate level for data exchange and sharing, but not for interoperable product components, nor they specify a framework supporting component interoperability. In that context, current efforts, as undertaken in the OMG or by Sun with the EJB: (section *4.4.1*) are of primary importance, both for the standardisation of Common BOs (CBOs) and domain specific BOs as well. These standards have to focus on the interfaces of the Application Programming Interface (API) which define the communication layer, whilst the communication mechanisms ensuring transactional behaviour will be implemented by a tier middleware. The choice for OMG is naturally CORBA, with its services (OMG, 1995a).

In response to an initial Request For Proposal (RFP) issued by the OMG for Common Business Objects and Business Object Facility (OMG Document CF/96-01-04), the Combined Business Object Facility Proposal introduced a general architecture for providing a coherent and standardised solution for BOs. Elements that are part of this solution are:

- The Business Object Architecture (BOA), the underlying layer which would enable BOs. It integrates a meta-model which describes constructs and types as well.
- The Component Description Language (CDL), an advanced language to describe the BOs interfaces and specifications semantics following an OO manner, and dedicated to writing down BOs in a textual form. CDL helps to standardise BOs, thus providing an unambiguous way of specifying properties, behaviours, business rules and so on.
- The mapping from CDL to IDL (Interface Definition Language) promoted by the OMG would encompass the necessity of interoperability. A given specification using CDL would produce IDL interfaces which would take into account the various specified BOs and the underlying framework.

As a consequence, BOs expressed through CDL are CORBA objects. Thus, the BOA is built firmly upon the Object Management Architecture (OMA), because all BOs must be network accessible through an Object Request Broker (ORB). A CORBA reference will uniquely refer to a BO. Nevertheless, CDL simply express the BOs' semantics and interfaces. In order to represent an entity or a process in a specific business domain, and on the base of an underlying object technology, a BO is a set of:

- **Attributes**, that stand for associated data elements which provide information about the BO They are transient or persistent data the BO holds and which compose its internal state. Existence of attributes depends on the BO that refers to.
- **Operations**, that is its behaviour.
- **Constraints**, that is not only integrity constraints, but also triggered business rules it must ensure in order to fulfil the requirement expressed by the end user.
- **Relationships**, that is the set of objects it is in relation with. Relationships connect types, and help to make components interact. Without any relationships, components would be isolated islands of knowledge. They allow bi-directional traversal between instances of different types. Moreover, they permit loose coupling of co-operative distributed components and a way to easy assembly components originated from different vendors. Several kinds of relations may be supported by a BOA implementation, because each type of relationships covers some specific semantics.

- **Events** that externalise state changes when operations are invoked on BOs. An event model is intended to avoid tight coupling while integrating several BOs in a specific system. By doing so, changes are easier to manage.
- **Business rules** that focus on the semantics of the component in a specific domain. An appliance rule is plugged into a container on which the rule applies. The BO's behaviour is partly defined by this sort of sub-component. For example, invariant appliances define rules that must always hold for the component.
- **States** that symbolise the mutually exclusive conditions a component may be in. Moreover, some states may be reached only when pre-defined conditions yield.

4.3 Integration of BOs in Three-Tiered infrastructures

In order to take plentiful advantages of them, BOs are to be considered in the context of a full distributed architecture based on middleware technology i.e. on a software bus in charge of communication between objects, like an ORB or a DCOM bus. Indeed, BOs must support distribution to be easily integrated in client/server architectures, and they promote the nowadays well-known 3-tier architecture. They aim at being distributed on remote sites for specific use and event, while at the same time being accessible in a seamless and transparent way. They constitute the fundamental bricks of the «business-centred» middle tier of today emerging architectures.

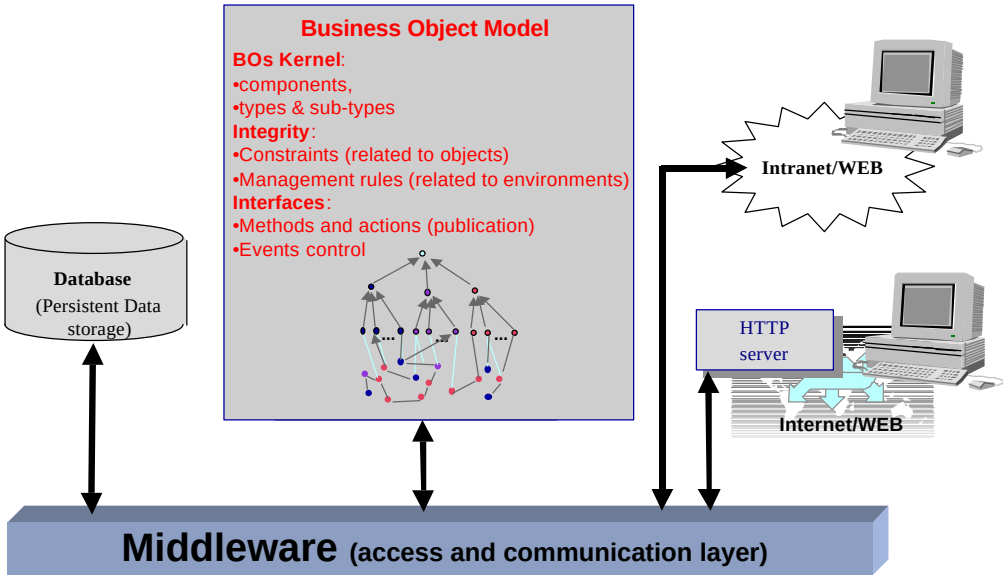

Figure 1: The general 3-Tier architecture.

For each main tier of this infrastructure, the following comments can be made:

- **Information servers (DBMS) level**: this is the data storage and server level. Access to basic data within DBMS can be realised through gateways - native gateway for direct connection to legacy systems, co-operative gateway using the network interface of the accessed database, or procedural gateways for access to the database through inside procedures or TP monitors - or through a middleware layer in a distributed environment (CORBA/ORB, DCOM, etc.).
- **Client applications level**: this level is mainly concerned with Graphical User Interface (GUI). WEB de facto standards (HTML, etc.) offer standardised interfaces to the client, this dealing with Intranet as well as Extranet/Internet users. Depending on the fact that client computers are simple PC or NC computer (Network Centric computer), or on the other side powerful workstations, specific treatments can be localised on the client side.
- **Middleware level**: this communication level has to manage generic services as information search, object or server identification, communication between objects through messages transport,

marshalling and unmarshalling, etc., at level of the distribution layer. Eventually, global rules which are not related to business objects, but rather to their inter-working in the context of the full enterprise model, can be handled at that level too, like rules defining actions when a given program execution has failed, actions to undertake in some specific context defined through global variables, transaction rules, etc.

- **Central tier (business) level**: it contains the embedded business logic, with one or more business objects model(s). From a client point of view, this business model expresses the business domain and provides the end-user with some unified view on data hiding the underlying data sources. From the architecture viewpoint, it implements the business rules and processes, dealing with complex operations on the business model such as object mappings or links, views filtering, model conversion/transformation, fine-grained security issues, possible client interface and so on.

A flexible and evolving architecture must integrate models and rules so as to control treatments on engineering knowledge, both at level of managing and massaging the information and at offering various views and interfaces to client applications and end-users. This role is indeed devoted to the central tier component, acting as a specific applications server for fundamental services encapsulating objects and views with their procedures and rules in business objects as a modelling of the business enterprise.

4.4 Existing implemented model for BOs

The emerging technologies presented hereinafter can be considered as implemented solutions of a model and application architecture for BOs. Nevertheless, it is worth noticing that these developments have been undertaken regardless specific standardisation efforts of the OMG with respect to Business Objects Facility and Business Application Architecture. They are proprietary SUN (Enterprise JavaBeans - EJB) and Microsoft (Active X components) solutions to deal with business oriented framework and components. As based on Java, the Enterprise Java Beans, however, can be considered, at least at the moment, as the most open framework. Moreover, BOs are supposed to rely on distributing computing: the most currently achieved specification is the OMG CORBA (OMG, 1995b), but even other distributed-object systems, like OLE/DCOM for instance, are candidates to support BOs. In any case, the integration of an ORB-like backbone is a requirement for the real deployment of a BOs application infrastructure, allowing any application on the network to deal with BOs.

4.4.1 Enterprise JavaBeans

Java, including the language itself, the first basic Java libraries and the first Java oriented Integrated Development Environments (IDEs), initially provided only the opportunity to client applications for dynamic WEB user interfaces, with sometimes some ready-to-use simple client-side JavaBeans components. The Java Virtual Machines (JVMs), allowing to execute Java applets, were not designed to support enterprise applications servers, especially because of a lack for essential support of transactions. The on-going definition of the EJP (Enterprise Java Platform) is making the situation evolved.

The EJP can be viewed as a virtual application server specification, defining Java-based portable application servers on top of any OS (Unix-like, 9x/NT, proprietary mainframe OS), thus allowing to build enterprise applications servers independently of the underlying OS layer. The objective of EJP is to standardise the services and API required for object oriented distributed applications based on Java, i.e. transactional services, objects lifecycle, synchronous or asynchronous objects interoperability, security aspects, and so on. In a similar way JavaBeans describe an API for reusable graphical object components, which does not meet large enterprise systems requirements, EJP includes the Enterprise Java Beans (EJB) (Sun Microsystems, 1998), (Orfali et al, 1998), specified by Sun, in order to provide a component architecture for development and deployment of distributed, enterprise wide objects. Applications written using EJB should be deployed on all Java-enabled server systems.

The EJB model aims at offering distributed scalable, transactional and reusable components, and any Java enabled platform may run these components, provided they use a corresponding EJB component enabler. EJBs extend the JavaBeans component model to the server side, and aim at accommodating components for large transaction-oriented applications, tying legacy data to Intra/Internet clients, under the control of the business logic coded in the EJB on a middle tier. Thus, an EJB provider is an expert in a business domain and develops

specialised components. Typically, he will distribute his product which implements specific and standardised tasks related to a well-known business domain.

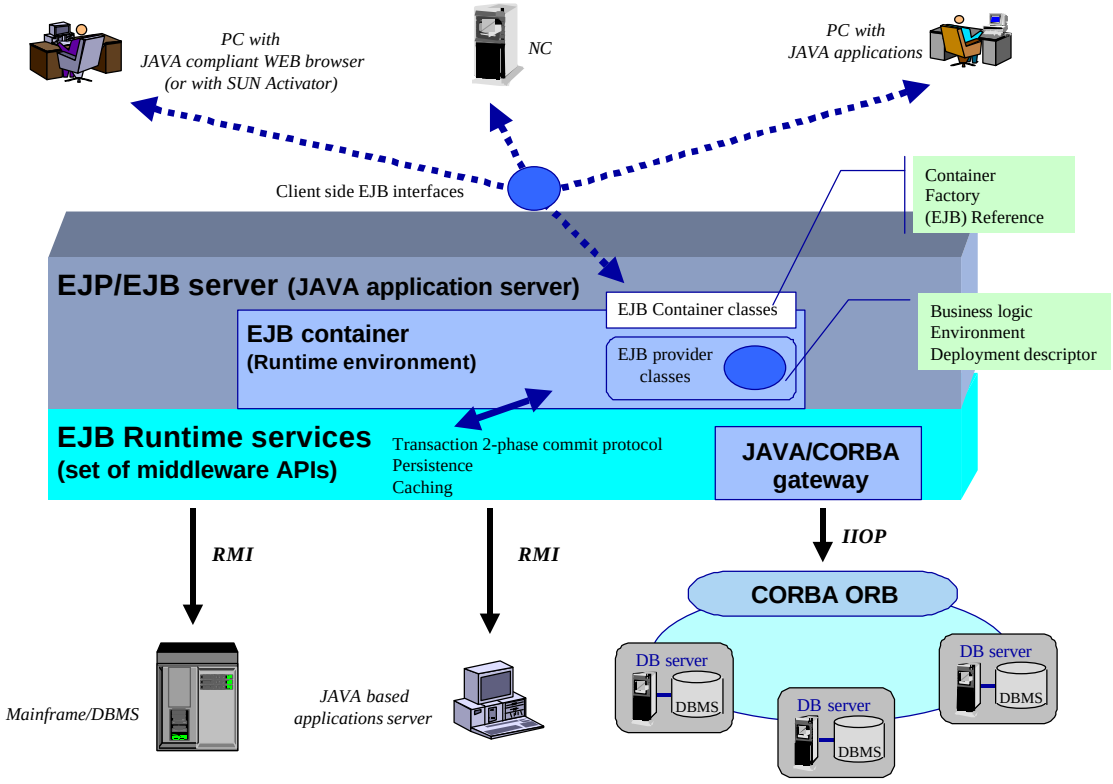

Figure 2: General architecture for an EJB platform.

It is worth noticing that, in its third release of CORBA (currently 3.0), due by the end of 1999, the OMG plans to incorporate the Java Beans model in order to deliver CORBA Beans. This will both enable rapid development of CORBA objects and integration using graphical tools. Moreover, CORBA objects may be easily and seamlessly combined with Microsoft's Active X components. With the emergence of EJB, a plentiful market for ready-to-use JavaBeans components for client and servers can now be deeply envisaged. In a very synthetic way, the major positive points for EJBs are:

- They suggest a unified model for distributed components.
- They encompass life cycle and naming.
- They rely on JTS (Java Transaction Service), which is a mapping of the Object Transaction Service (OTS) to Java.
- They shield developers from transactions management
- They form a specification for Java components and for CORBA components as well.

On the other hand, the current main shortcomings are the following:

- Current lack of proven implementations of EJB Servers.
- CORBA Beans are only planned at the moment.
- The model is not published by the OMG, but by Sun, thus cannot be considered so far as a standard specification.
- Transactions support is not fully defined and specified in the EJB draft specification (1.0).
- Naming and life cycle do not rely on corresponding OMG services.

4.4.2 Active X components

Besides *de facto* standards as those promulgated by the OMG or specific efforts of SUN/Javasoft for promoting Java and RMI, Microsoft (COM Home, 1998) is developing more proprietary solutions in the field of distributed architectures. Solutions as offered by Microsoft have similar characteristics than other distributed architectures, but initially don't provide platform independence. However, this is less and less the case, as a lot of efforts for porting Microsoft developments on other platform/OS are underway: especially the COM/DCOM model aims at running on multiple platforms, including its original Windows system as well as various implementations of Unix-based environments.

Windows DNA is a unified approach for building distributed, scalable, multi-tier applications that can be delivered over any network. Windows DNA is the first application architecture to integrate the Internet, client/server, and PC models of computing for a new class of distributed computing solutions.

OLE (Object Linking and Embedding) is a set of libraries and applications for storage, data exchange and integration of elements within compound documents. **COM** (the Component Object Model) is the Microsoft component software model and underlying software architecture for applications to be built from binary software components, leading to higher-level software services like those provided by OLE for various aspects of commonly needed system functionality, including compound documents, custom controls, data transfer, and so on. The COM specification contains the standard APIs supported by the COM library, the standard suites of interfaces supported or used by software written in a COM environment, along with the network protocols used by DCOM in support of distributed computing.

DCOM (Distributed Component Object Model) complements COM and OLE as being a model allowing to get benefits from a component-based approach across a broader scale of multi-user applications in a distributed component architecture. It is mainly a protocol enabling software components to communicate directly with each other **across networks**, including the Internet and intranets, the various objects involved being associated to different processes on remote machines. DCOM introduces to new interfaces and APIs for distributed objects. This specification is still evolving, and thus subject to change.

Eventually, what is called **Active X** is an extension of OLE/COM for the specific context of the WEB and the Internet. It is the Microsoft infrastructure for WEB-centric communication between clients and servers of distributed objects, and the integration of objects within WEB pages for Internet and Intranet applications. Indeed, Active X is a set of integration technologies for distributed client/server applications, dealing both with client and server sides and including:

- The Active X controls, which are pre-fabricated components packaged by programmers, and which can be manipulated at design time by GUI-based development tools. ActiveX controls are among the many types of components that use COM technologies to provide interoperability with other types of COM components and services. At Internet/Intranet client side, they can be conceptually compared to JavaBeans.
- The Active X documents, which can be considered as an extension of OLE documents for the WEB, and which are composite documents (i.e. with non-HTML parts as well) containing Active X objects to be possibly manipulated with a scripting language, like JavaScript, Perl, and ECMA Script.
- Active X components on the server side for multi-tier client/server applications: these components and the Microsoft WEB server Internet Information Server (IIS) together offer an architecture (Active X Server Framework) which allows to use Active X components, documents and scripts on the WEB server in order to build applications further accessed by Internet Explorer. Especially, the Active Server Page(ASP) technology on the server side enables to combine HTML with various scripts to build presentation interfaces, manage the business logic and call COM-based components.
- Microsoft Transaction Server (MTS), which can be viewed as the Microsoft Runtime execution environment for server components, extending the COM architecture with transaction semantics, providing concurrency, multi-threading, queuing, atomicity, etc., and finally making easier to

encapsulate business logic in a middle-tier, as MTS can be the basis of a framework for linking this middle-tier with enterprise data stores.

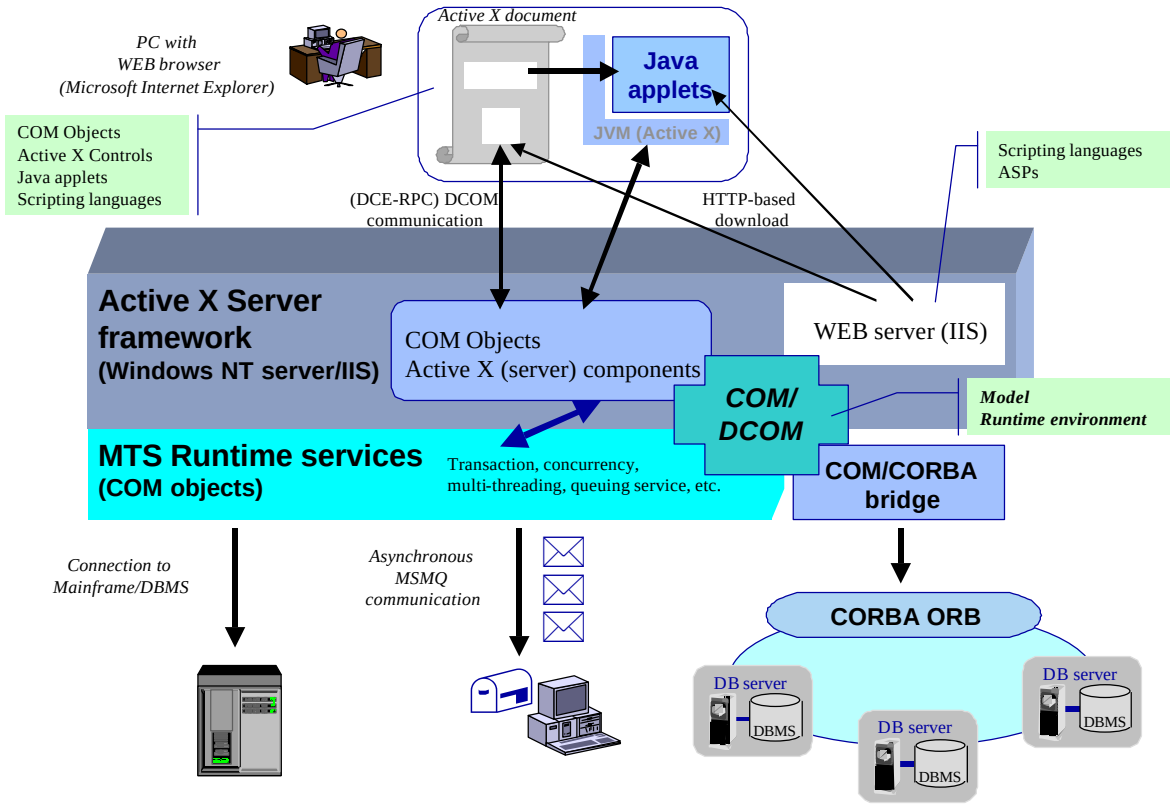

Figure 3: General framework for COM/Active X components.

All these features make the Active X technology a potential answer as a component-based framework for Internet and Intranet business applications, at least from an end user point of view. Of course, CORBA can be identified as a more general specification than DCOM, and in a similar way, the EJBs can be viewed as a more general transaction-oriented platform specification than Active X. Anyway, Active X offers the advantage of being practically well integrated and performing, and is available today, while EJBs implementations are still emerging, and interoperability between EJBs coming from various vendors still have to be proved. Moreover, it is the intention of Microsoft to keep on integrating the various aspects of COM and DCOM, and its services inside the Operating System, leading to a future integration of the COM objects model and the Internet-oriented distribution model in order to unify the access to any kind of document anywhere that is, for local files as well as remote WEB documents. On the other hand, issues related to standardisation aspects for components and executable specification languages for components are not grasped in the DCOM/Active X universe. Moreover, and this is also the case for EJBs, both sides are seen as an easier way to create distributed transaction-based applications, but both are still in the early stages of development.

4.5 BOs in the WONDA specification

4.5.1 Introduction

As already stated, BOs are distributed (CORBA) objects. But whereas CORBA provides distributed objects with a rich framework, a more elaborated framework to support BOs is required, to manage the business logic and processes as well. In order to effectively, unambiguously and rapidly define BOs, the WONDA specification especially built upon CDL for a textual computer-oriented definition of BOs. From that CDL specification, both IDL specifications and Java code are generated.

The choice of Java and CORBA is to be explained through their attractive features. They offer a quite interesting complementarity. Java provides portability of code and platform independence. With CORBA, you add location transparency and an enterprise level object model that allows your application to inter-operate with a multitude of existing languages and integrated or legacy systems. From a technological point of view, CORBA is a general infrastructure specification for applications interoperability, based on OO technology, and allowing heterogeneous clients and servers connected by a network to live as individual entities able to access to the information they need in a seamless transparent way. It includes a middleware that lets technical objects and business components find each other in a network environment and invoke their respective services regardless of the underlying platforms or languages on which the applications are running. It also ensures ORB's interoperability through the Internet Inter-ORB Protocol (IIOP) protocol. From a business-oriented point of view, the CORBA standard is getting more and more acceptance within a lot of industries, now appearing as a potential solution for production of business environments. There are several reasons for this driving force promoting CORBA: its complementary with Internet/WEB technology and its association with Java, and the need for component-based interoperability relying on middleware technology and standard communication mechanisms between applications, arising as true solutions to build cross-platform component-based applications to be deployed over the WEB.

We hereafter detail concepts related to BOs together with an example, and then how they are related to a CORBA framework.

4.5.2 Business Objects kinds and modelling

The OMG Business Object Domain Task Force (BODTF) (BODTF, 1998) identifies several types of BOs:

- **Business Entities**: the entity concept is closely related to the one found in Entity/Relation modelling. Entities can unambiguously stand for a concept usually found in the business model, and could represent a person, an item, etc..
- **Business Processes**: in Eeles and Sims (1998), processes are referenced as encapsulated units of work that stand for sets of modifications to apply. Actually, a process is an *activity* like purchasing, for instance, and represents a flow of work and information needed to accomplish a business.
- **Business Events**: events are persistent records of changes that occur in the modelled domain at a specific time, e.g. *»Contractor achieved the building»*. Events can be classified by their sources of generations and occurrences, and four major groups can be identified: *user interaction events* for human activity, *temporal events* triggered at specific pre-defined times, *error notification events* to report failures in maintaining business rules, *change of state notifications* when a process terminates.

A business system domain is a set of business entities, processes and events, and to federate BOs that apply to a particular business domain, the business system domain stands for the smallest solution that allow for maintaining the business rules. Especially, this notion is meaningful when trying to interconnect different businesses, by connecting business system domains using semantics adapters. To describe the BOs of a given business domain, CDL can be used so as to write down in a textual form BOs specifications in an OO manner. An analogy can be drawn with IDL, and CDL can be considered as a superset of IDL, dedicated to producing BOs specifications, keeping in mind that BOs are in essence distributed objects. CDL definitely captures the semantics expressed with BOs and enable to model attributes, operations, business rules, events and bi-directional relationships. CDL should be compiled, according to the BODTF vision, to IDL interfaces to be further integrated in a CORBA framework. Moreover, those interfaces are implemented according to a given target framework as presented in the next section, and code can be automatically generated to free the developer from low-level and repetitive tasks.

4.5.3 Business Object Framework

The objective of a Business Object Framework (BOF) in (Eeles and Sims, 1998) and (Orfali et al, 1998) is to offer a full environment for developing and deploying BOs. Hence, a BOF must support BOs as plug-and-play components, ensure interoperability between them and shield developers from developing complex components. Moreover, a BOF can be implemented on top of CORBA and cope with:

1) **Business objects' naming:** BOs must be named in order to be retrieved and their reference accessed through the use of such a unique name in the considered business system domain. Such a service can be supplied by the CORBA Naming Service (OMG, 1995a).

2) **Life cycle management:** The BOF is responsible for managing the creation, deletion, activation and deactivation of BOs at runtime. A BO is created and deleted once in its lifetime, so the BOF must ensure that this rule of thumb is enforced. A BO may be activated and deactivated several times because the BOF has to efficiently manage its resources. When a BO is activated, it must acquire its state from persistent storage and vice versa, when deactivated, it must dump its state into persistent storage. Concerning life cycle management, BOs could rely on the CORBA Life Cycle service (OMG, 1995a).

3) **Persistency management:** Persistency addresses the need to access enterprise data (i.e. stored into legacy systems). This must be achieved in a safe manner that frees the BOs developers from most of the task.

4) **Event notification:** Clients need to be aware of changes in the BOs internal states because they maintain, for instance, graphical views. An event service (OMG, 1998a), based on the publish/subscribe pattern is a flexible and scalable architecture that allows clients to be asynchronously notified of changes and to receive meaningful information. Moreover, filtering features are added to sort out events that are meaningful.

5) **Transaction management and concurrency control:** Since BOs can be shared by different users that perform actions at any time in a distributed environment, the BOF must offer a way to serialise methods calls by handling distributed transactions. Still in the CORBA world, we can point out OTS (OMG, 1998b) designed to respond to that issue. Distributed transactions raise the question of whether a specified BO should be recoverable or not. Non Recoverable BOs (i.e. that cannot roll back, sometimes called stateless BOs) must be found among business processes whereas business entities should be recoverable since rolling back a transaction must leave the system in the previous state.

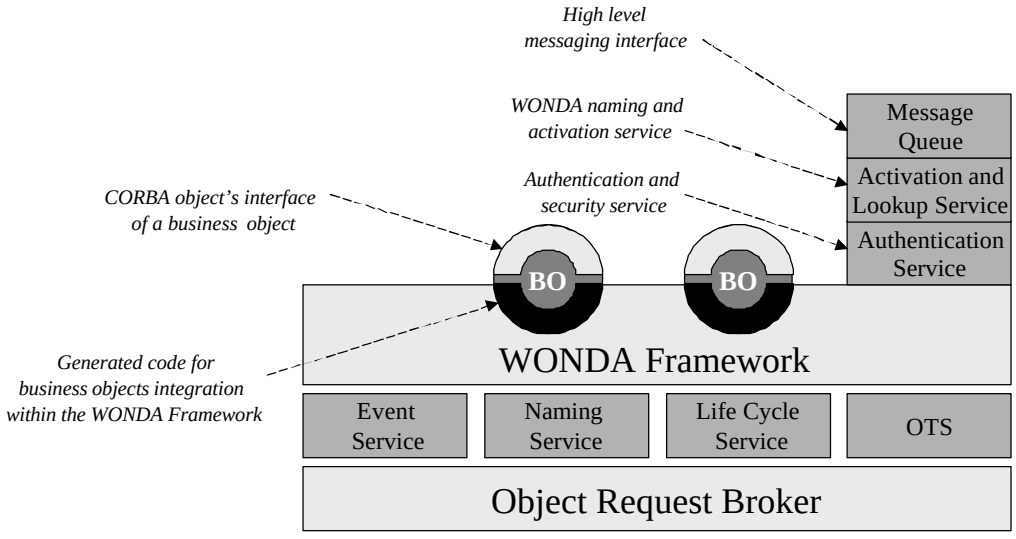

Figure 4: The WONDA framework for BOs.

4.5.4 From CDL to CORBA business objects

Starting from a typical development design process with a CDL textual specification, an objective then is to automate as much as possible the work involved in creating and deploying software component-based applications, wrapping existing systems and data for presentation as components, in order to let BO developers concentrate on the business rules. CDL specifications can be processed in order to produce:

- The corresponding IDL specification of BOs, towards some underlying framework for distributed BOs. The CDL to IDL mapping generates IDL so that attributes, relationships, operations that constitute entities, processes and events in CDL are directly mapped.
- Code in a target language (C++ and/or Java), still on the base of some target framework, to compensate the loss of semantics induced by the mapping from CDL to IDL. CDL has a richer semantics than IDL, and especially, rules and appliances that are part of CDL cannot be translated into IDL as IDL is a pure descriptive language for interface definition. Hence, some code must be produced to bridge the gap introduced by IDL. This is achieved through automated generation of code written in an OO programming language: what can easily be generated is code for maintaining bi-directional relationships, triggers that fire rules, invariant checking, and state transitions.

It is however worth mentioning that BOs implementation can be investigated against other new emerging models, especially the future multi-language CORBA component model, also referred to as CORBA Beans, supposed to encompass the EJB model and aiming to tightly integrate Java, EJB, and CORBA so that applications and objects can be used across more than EJB servers. Indeed, the CORBA model grants standard services layer for EJB to go against, and the complementary of these two models seem to be a true potential basis for future large-scale component-based business applications.

Side Bar: A simple CDL example:

The following entity represents a contractor related to zero or more other contractors considered as subcontractors.

```
entity Contractor {
     relationship subcontractors Many 0..* Contractor inverse contractor;
};
```

The process specified below represents the task of choosing a subcontractor among a list of possible contractors.

```
process HireContractor {
     relationship contractor Reference 1..1 Contractor;
     relationship possible_contractors Uses 1..* Contractor;

     void evaluate_propositions();
};
```

The following process deals with constructing a new building that can be on hold, being constructed or released to the new owner. Two signals start the two associated transition rules that make the state evolve.

```
process StartBuilding {
     state_set achievement { ON_HOLD, BUILDING, RELEASED };

     signal start_building();
     signal release_building();

     apply StateTransitionRule when_started {
       source = achievement::ON_HOLD;
       target = achievement::BUILDING;
       trigger = start_building;
     };

     apply StateTransitionRule when_built {
       source = achievement::BUILDING;
       target = achievement::RELEASED;
       trigger = release_building();
```

```
    };
};
```

The first step is to compile the CDL code to IDL by mapping CDL features and kinds of BOs:

- **Business objects (business entities, processes and events)**: clearly, the only way to distinguish a specific kind of BO is achieved through inheritance. Hence, an interface representing a BO (e.g. an entity) must at least be a sub-type of a specific interface that stands for that kind of BO (e.g. the topmost interface is an interface called *Entity*).
- **Attributes**: CDL attributes are mapped to a pair of operations based on attribute's name. The BOs designers can restrict access to an attribute by specifying that attribute as read only, or only visible in the business system domain. This will impact the way those operations are generated.
- **Relationships**: CDL offers different ways of declaring relationships. Depending on the parameters that configure the relationship in CDL, operations enabling insertion, removal, and iteration are generated. Additionally, an interface representing an iterator well suited for iterating collection in a distributed environment is produced by the CDL compiler.
- **Operations and Signals**: CDL and IDL have a very close semantics for operations. This leads to directly map a CDL operation to an IDL one with the same name. Signals are operations that are dedicated to triggering rules that are expressed using CDL. Nevertheless, IDL is unable to propose a better way to signals than operations.
- **Rule and appliances**: CDL helps in describing invariant, action/condition rules. Whereas this helps in describing components, IDL does not at all capture the semantics expressed in rules, so that no IDL code is generated.
- **During clauses**: when specifying BOs, it is sometimes useful to restrict access to some features when specific conditions are not reached. CDL offers that possibility by declaring anonymous or named *during* clauses. IDL is clearly unable to express this notion and a mapping is done by recursively concatenating the name of the enclosing during clauses, no concatenation being realised in the case of an anonymous clause.

The next side bar gives some hints according to the previous explanations.

Side Bar: A simple CDL example (continued)

The *Contractor* entity is mapped to the following IDL interfaces. First of all, the *Contractor* entity is mapped to an IDL interface with the same name that inherits from the WondaMetamodel::Entity which is the super type for all entities. By this way, the framework is automatically aware that this BO must be considered as an entity, especially when dealing with transaction at runtime. The bi-directional relationship labelled *subcontractors* is mapped to the 3 operations that are embodied into that interface. The first one enables to obtain an iterator for navigating the relationship from the beginning. This operation is defined only if the relationship is navigable. The two following operations are generated if the relationship is not read only, and allow to insert and remove BOs from the relationship. The developer does not have to implement theses methods since the framework can produce implementation classes in a language that supports a binding to IDL.

One can notice that there is also an additional support interface. This interface represents an iterator that can be produced by any relationship that is typed by a Contractor type. The first method returns the business that carries the relationship to which this iterator is related. The two following methods are accessors that help to obtain (for the first one, the second is a restriction) at most *count* Contractor entities. The get_by_key operation allows to retrieve in the relationship the BO that holds a key equal to the one being passed. The copy and destroy methods are life cycle management operations. It is obvious that the implementation of the interface can be automatically generated by the framework so that the developer can directly use instances to traverse relationships.

```
typedef sequence<Contractor> ContractorSeq;
interface Contractor : WondaMetamodel::Entity {
     ContractorIter get_subcontractors();
     void insert_in_subcontractors(in Contractor o);
```

```
     void remove_from_contractors(in Contractor o);
};
interface ContractorIter {
     Contractor get_origin();
     boolean next_n(out ContractorSeq values, in long count);
     boolean next_one(out Contractor c);
     Contractor get_by_key(in WondaMetamodel::Key key);
     ContractorIter copy();
     void destroy();
};
```

The process *HireContractor* is generated the same way as *Contractor* entity, except that it has one operation mapped to a standard IDL operation and it inherits from the interface WondaMetamodel::Process.

The process *StartBuilding* is very different from the two previous business entities. A nested enum is generated for the values that can take the achievement state. For that state, since it is not read only, two methods are generated (for setting and getting). Signals are mapped to standard operations, but with our framework no implementation is required since we can deduce that when a signal method is called we must check the state's value and accomplish a state transition if needed. Nevertheless, this is only achieved by producing some additional code in a language that support IDL mapping (C++ and Java are well-suited).

```
interface StartBuilding : WondaMetamodel::Process {
     enum achievement_kind { ON_HOLD, BUILDING, RELEASED };
     void set_achievement(in achievement_kind a);
     achievement_kind get_achievement();

     void start_building();
     void release_buildin();
};
```

4.5.5 Implementation

In order to rapidly produce BOs from a CDL specification, and besides the generation of IDL interfaces, an object implementation must be provided, which has been done in our developments in providing the generation of a pre-implementation of our BOs, indeed directly implementing them in Java within a BOF. This CDL to Java additional mapping endeavours both the issues of generating Java interfaces, and the real implementation of Java-based BO skeletons. Producing interfaces addresses the notion of relying on well-known interfaces on the client side. With respect to delivering BOs implementations, the issue had been tackled in a similar way the OMG did for CORBA. Hence, we produce our skeletons that connect the developer's implementations to our BOs framework. We do not get into details of this generation, for more information, the reader should refer to (Richaud and Zarli, 2000).

A prototype system has been built in the CIC (Computer Integrated Construction - http://cic.cstb.fr/) division at CSTB. The CDL compiler has been developed on Windows NT 4 with Visual C++ 6.0. We also developed a CDL model viewer that allows to visualise in a graphical manner any CDL specification and to graphically ask IDL or Java code generation. For now on, this is the only platform on which we support developments. The underlying Java/CORBA framework is the one supplied with the JDK 1.2. It comprises an IDL to Java compiler and an implementation of the Naming Service.

4.5.6 Integration of BOs: the main components of the WONDA specification

As shown in Figure 5, WONDA specified a 3-Tier architecture based on a set of existing and emerging international *de jure* or industrial *de facto* standards. Building upon standards further enables easy integration into existing enterprise IT environments based on these standards. The three WONDA layers are identified as follows:

- A federated database level, in order to search and access information within various heterogeneous databases throughout a unified interface (OMG/ODMG'93, STEP/SDAI). Database system federation and middleware, coupling in a transparent way distributed heterogeneous DBS, should be a major key factor in a very near future. Thus, any application will be independent of the used

DBS and of specific database system vendors. The figure shows those databases directly accessed by the BOs CORBA servers, but they can be distributed over the whole system, and accessible as CORBA servers as well.

- The WebMapper, which is a schema mapping layer enabling electronic knowledge bases to be seamless interconnected, and thus dealing with availability and interoperability issues. This mapper ensures the liaison between the federated database level and any Internet/Intranet application in a model, language and application independent way. In the WONDA specification, the WebMapper is also in charge of the management of media information, that is basically GUI and documents, along with a distributed model (Comprehensive Object Model for the Management Of multTImedia informatiON - COMMOTION -) and a Media Repository.
- A third level defining BOs and «plug-an-play» software components for domain specific applications, for intuitive access to WEB information by end users using their usual applications. A component-based approach is the key to lower cost, more reliable and flexible systems, and to customise the user needs and different application segments, e.g. construction, banking, mechanical SME's, etc.. The enterprise business logic layer for construction specific application via a BOs level was called *WeBuild* in WONDA.

A last specific WONDA feature is the introduction of security components for data protection and confidentiality levels configuration, so that applications can be securely linked, thus enabling WONDA to be exploited for real industrial and commercial applications. This is achieved through a set of components, named SILK, to be integrated at the client side, the BOs layer, and as a firewall component with the filtering of IIOP requests.

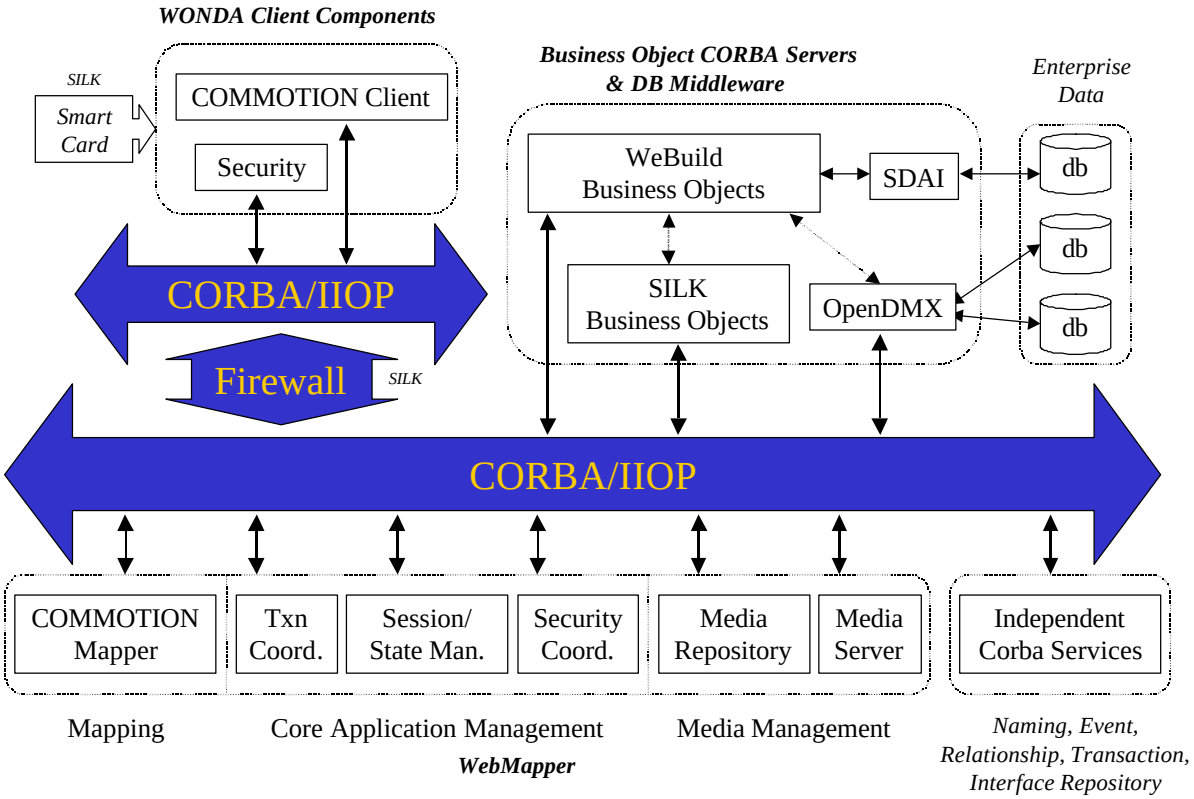

Figure 5: the WONDA global architecture.

In WONDA, a special focus has been put on end users manipulation and visualisation of BOs, through Media Objects (MO). While BOs manage the business process logic, MOs deal with the presentation logic. Typically, BOs can be associated with graphical views or be accessed by simple client/server solutions that serve as a front-end. BOs/MOs interactions can be realised through notifications from BOs to (COMMOTION) MOs achieved throughout a publish/subscribe technology loosely coupling MOs and BOs. Particularly, such a mechanism can

rely on the CORBA event service, as a critical service for applications intended to automate a complete business process, letting CORBA objects dynamically register their interest in specific events. The WebMapper is then in charge of mapping BO interfaces and multimedia content to WEB interfaces with the help of MOs, through access to BOs with well-known (IDL) interfaces and their multi-modal GUI-oriented representations. Since data are «conceptually» embodied within BOs, both the BOs layer and the WebMapper form the bridge between user interfaces and enterprise data. Eventually, the WebMapper allows for BOs and generic front-ends to be designed, implemented and deployed separately, thus inter-operating in a flexible, configurable and seamless way. For extended details about BOs/MOs connection, and more generally about the various components exhibited herein above, the reader should refer to the WONDA WEB site (http://www.bild.ie/wonda). WONDA User Requirements and a «System Analysis» document, showing functionality of BOs, are accessible as well.

Hence, WONDA focused on a framework for business solutions and open interchangeable software tools, specifying an infrastructure fully dedicated to the WEB interconnection, and based on major standards. Along with uniform access to any databases, the objective is to extend the openness of such platform towards WEB standards and applications via the WebMapper for information presentation and query, and at a generic level in terms of standardised models for presentation and not from an tool-oriented point of view. Thanks to an implementation of a infrastructure as specified in WONDA, coupling of WEB data with internal corporate information memory including data managed by Intranet applications will become possible, along with the development of sector specific software for Internet servers and browsers with a component based technology.

5. CONCLUSION

This paper tried to identify a set of requirements and suggested a possible technology integration for IT-based Business-oriented frameworks, targeting the specific Construction sector, though most of the technologies considered and developments presented above are quite generic. Our work has been undertaken in the context of the WONDA project, whose main objectives was to develop an architecture specification meeting requirements typical of the construction industry, i.e. IT solutions for VEs delivering low entry level, scalability, open infrastructure and location independent access, seamless enterprise information, transparent support for business processes, security and transactions.

As presented in this paper, the central enabling technology is BOs, which are defined as components in the information system representing the enterprise model and which promise to be the building blocks of information systems meeting requirements critical to the success of the enterprise. BOs are dedicated to rapidly coping with business changes and making them appear in the involved information systems. This objective is not correctly addressed through the current common concept of application, since it is obviously difficult to mix applications to obtain a new application that is the sum of those applications. Moreover, modifications cannot be carried out at a high conceptual level: this is especially true if the source code is not available and/or if those applications are bought from different vendors. Furthermore, whether or not source code is available, traditional applications are hard to maintain. Even if distributed objects have been a major evolution towards interoperability and remote access, they don’t address all the integration issues. The future consists of distributed objects that are easy to integrate, and can be viewed as distributed BOs in the sense of the OMG approach.

The main hypothesis in our work is that BOs address the notion of distribution because they are network accessible through an ORB as found in CORBA. Furthermore, BOs are design time as well as runtime artifacts because they are described using OO modelling tools and because they are «distributable» components as long as there exists a BOF managing BO instances. One of the main advantages with such a framework is that developers are exposed to a higher level API than the one supplied by CORBA. Hence, BOs encapsulate the storage (i.e. access to persistent data and how to manipulate them in a safe way with a high level interface), metadata (i.e. the ability to describe itself), concurrency and transaction (i.e. the ways to safely interact with BOs when clients, for instance, perform asynchronous operations that makes the BOs internal state change). As designed in the WONDA project (and reflected by this paper), the strengths of BOs for information systems design and implementation support can be summarised as follows:

- They will allow the BO developer to focus on expressing the business logic (at least part of the this logic to be related to the object), being no more burdened by low-level concerns like networking, distribution and remote access, transactions management, security, and so on. Thus, each BO developer can concentrate on his own specific expertise and knowledge area, without the need of

learning about a lot of other areas and technologies in client/server information systems (networking, middleware, databases, etc.).

- They will allow the application designer to benefit from being able to reuse existing BOs components with a well-known and standardised interface, and to more naturally and easily assemble those components. Easy exchange and adaptation of single components will be possible as well.
- They will provide to applications a standardised manner in which server-side objects can be accessed and interacted with (open API), moreover offering the opportunity to use and make inter-operate objects and components from various vendors.
- At the end of the day, BOs will considerably increase the ability to easily and quickly design scalable, distributed, sophisticated business applications, on the base of a common understanding for a large set of generic or domain-oriented services, and hierarchies of components and objects.

From a more user oriented point of view, BO can manage quite different end user views, but also realising the links between information related to engineering product data, financial data, process and workflow oriented information, and so on. At the same time, they are quite masking the internal representation of the various stored information, and encapsulating the rules and relationships/associations associated to the BOs. Considering the construction sector, and just as a building is a unique arrangement of standard products, a project is a unique arrangement of financial, logistic and technical product data, and the role of the BOs is to translate it in a Business model.

Experimentation is now underway to model business applications, so as to access distribution services and package other services through BOs, in order to get tangible and physical proof of the original hypothesis, and its relevance regarding business practices in the construction field. The WONDA project itself endeavoured the vision of a standardised open WEB-oriented framework for information access and electronic publishing and commerce for the Building Construction. The combination of new concepts and technologies based on standards, either *de jure* ones like official norms, or *de facto* ones in their large industrial use but still leading to openness, is supposed to open the door for substantially enhanced integration of value-added networks, high level application development, and distributed object systems. This combination will radically simplify and reduce time for the development, deployment and management of distributed applications providing corporations with a competitive advantage, especially on the Internet market, and moreover will be the foundations of intelligent systems for a business which requires decision-making, adaptive or learning systems, data-mining, etc.. At the end of the day, it will enable businesses to profit and expand by harnessing the capabilities and promise of truly global electronic commerce.

6. ACKNOWLEDGEMENTS

The authors wish to acknowledge the financial support of the European Commission in the context of the EP 25.741 project WONDA, as well as the participation of their partners, SNI/C-Lab, TU Delft and Dassault Electronique, in the elaboration of the whole WONDA framework.

7. REFERENCES

Björk, B.-C. (1996). Requirements and information structures for building product data models, Espoo Technical Research Centre of Finland (VTT), 1995, VTT Publications N° 245.

BODTF (1998). The Business Object Component Architecture, Revision 1.1, Business Object Domain Task Force, OMG Document 1998: bom/98-01-07

COM Home (1998), COM: Delivering on the Promises of Component Technology, http://www.microsoft.com/com/comintro.htm

Eeles, P., Sims, O. (1998). «Building Business Objects», Wiley Computer Publishing,1998.

Fowler, (1995). STEP for Data Management, Exchange and Sharing. Technology Appraisals 1995.

Hardwick, M., Spooner, D.L., Rando, T., Morris, K.C. (1996). Sharing Manufacturing Information in Virtual Enterprise, Communications of the ACM, February 96, Vol 39 N°2.

Hardwick, M., Spooner, D.L., Rando, T., Morris, K.C. (1997). Data Protocols for the Industrial Virtual Enterprise, 1997.

IAI (1999). International Alliance of Interoperability (1999). http://iaiweb.lbl.gov

ISO (1995). Industrial automation systems and integration - Product data representation and exchange Part 22. Standard Data Access Interface. 1995.

Kilov, H., Simmonds, I. (1997). Business rules: from business specification to design, ECCOP'97 Workshop on Precise Semantics for OO Modelling Techniques

Mowbray, T. J., Zahavi, R. (1996). The Essential CORBA - System Integration Using Distributed Objects. John Wiley and Sons.

OMG (1995a). CORBA Services: Common Objects Services Specification, Revised Edition, 95-3-31 ed, Object Management Group, March 1995.

OMG (1995b). The Common Object Request Broker Architecture and Specification (CORBA) Revision 2.0, Object Management Group, July 95. http://www.omg.org.

OMG (1998a). CORBA Event Service Specification, Object Management Group, 1998. http://www.omg.org/corba/sectrans.htm#event

OMG (1998b). CORBA Transaction Service Specification, Object Management Group, 1998. http://www.omg.org/corba/sectrans.htm#trans

Orfali R., Harkey, D. (1998). Client/Server Programming with Java and Corba, John Wiley and Sons.

Orfali, R., Harkey, D., Edward, J. (1996). The Essential Distributed Objects Survival Guide, John Wiley and Sons.

Richaud, O., Zarli, A. (2000). Developing a Framework for CORBA components, submitted to the 14th European Conference on Object-Oriented Programming - ECOOP 2000, 15 pages.

Siegel, J. (1996). CORBA Fundamentals and Programming, John Wiley and Sons, April 96.

Sun Microsystems (1998). Enterprise Java Beans, http://java.sun.com/products/ejb/