AN INTEGRATED COMPONENT-BASED APPROACH TO ENTERPRISE SYSTEM SPECIFICATION AND DEVELOPMENT

Zoran Stojanovic, Ajantha Dahanayake
Faculty of Information Technology and Systems, Delft University of Technology, Zuidplantsoen 4, Delft, The Netherlands
E-mail: Z.Stojanovic@its.tudelft.nl, A.N.W.Dahanayake@its.tudelft.nl

Henk Sol
Faculty of Technology, Policy and Management, Delft University of Technology, Jaffalaan 5, Delft, The Netherlands
E-mail: h.g.sol@tbm.tudelft.nl

Keywords: Component-Based Development, Enterprise Systems, Open Distributed Processing.

Abstract: Component-Based Development (CBD) represents an advanced system development approach, capable for managing complexity and ever-changing demands in the business and IT environment. While many of the component technology solutions have been already settled in practice, of equal importance to their success are the methods and techniques closely aligned with CBD principles. Current methods do not offer a systematic and complete support for component-based way of thinking. This paper presents a new approach to CBD, integrating the component concept consistently into all phases and aspects of the enterprise system development. The approach combines the CBD paradigm and ISO Reference Model for Open Distributed Processing (RM-ODP), providing a comprehensive component-based specification and development framework for building enterprise systems of nowadays.

1. INTRODUCTION

Effective use of advanced information and communication technologies is nowadays considered as an integral part of modern enterprises' business strategy, significantly impacting the way they perform business services and compete on the market. The main challenge enterprises face today is how to manage the complexity inherent in the systems they are deploying, while at the same time being able to rapidly adapt to changes in technology and business environment. The solution by many lies in growing interest in the research community and industry over component-based development (CBD) [Szyperski, 1998; Brown and Wallnau, 2000]. CBD provides organisations with a method for building complex enterprise-scale solutions that are flexible and able to accommodate the ever-changing demands in the environment, in a cost-effective, timely manner. By using the CBD approach, a system development becomes the selection, reconfiguration, adaptation, assembling and deployment of encapsulated, replaceable and reusable, functional elements called components, rather than building the whole system from scratch.

To date the move toward CBD paradigm has impacted mainly the implementation and technology level, providing pieces of system's functionality to be placed across the network nodes. Several distributed infrastructure technologies based on Microsoft's Component Object Model (COM) [Microsoft, 2001], Object Management Group's (OMG) Common Object Request Broker Architecture (CORBA) [Siegel, 2000] or Java-based tools [Sun Microsystems, 2001] have been introduced and already widely used. While the technology is a necessary element of any solution, it is not sufficient by its own. Of equal importance to the success of the CBD paradigm are the methods, approaches and techniques for developing component-oriented applications, targeting that technology at the final phase. Current CBD methods and approaches to enterprise system development tend to consider business and technology issues separately, which leads to unsatisfactory results from both viewpoints. They use component concept in rather an inconsistent manner handling mainly a component as a binary or source-code software

package. Therefore, a new component-based approach to system development is required, covering all aspects of enterprise-scale, data-intensive applications in integrated and consistent manner. Such integrated solution should provide a total system specification, from concept to deployment, through business, information, application and technology issues, using component-based principles. This paper presents an integrated and systematic approach providing basic concepts, and guidelines for enterprise-scale component-based system development.

2. COMPONENT-BASED DEVELOPMENT

Although CBD has been introduced by many as a new silver bullet for complex, enterprise-scale system development in the Internet age, it is rather evolutionary than revolutionary approach. CBD inherits many concepts and ideas from the earlier encapsulation and modularisation, "divide-and-conquer" initiatives in the information technology (IT), such as module programming and object-orientation.]. Higher productivity, flexibility, and quality, through reusability, replaceability, maintainability, scalability, legacy compatibility and the parallel-work ability are among the CBD claimed benefits [Brown and Wallnau, 1998; D'Souza and Wills, 1999; Szyperski, 1998].

While many of the technology solutions have been already settled in practice, of equal importance to their success are the methods and techniques for developing components and component-oriented applications effectively targeting that technology. Conventional methods and tools do not offer a full support for CBD, so that new approaches that are much closely aligned with CBD principles are required. IT community has just recently started exploring and inventing the methods and best practices for developing enterprise-scale, web-based systems from components. The most prominent of these are:

- Rational's Unified Process [Jacobson et al., 1999].
- Select Perspective [Allen and Frost, 1998]
- Catalysis [D'Souza and Wills, 1999]

Rational Unified Process (RUP) is development and management process, providing an effective way for system development, using breaking a process into phases that can be performed parallelly. The support for CBD paradigm in the RUP is rather declarative, offered through the use of UML notations in system development and limited by it. There is not a strong emphasis on the component concept, still seen here as an unit of software code.

Select Perspective provides the ability to leverage object-oriented technology while integrating it with other widespread and proven approaches, such as business modelling and data modelling, using the best of breed tools and techniques. In handling the component concept, the method uses the concept of package, defined in UML as "a general purpose mechanism for organising elements into groups" [UML, 1999]. Two basic stereotypes of the package are distinguished: service package used in business-oriented component modelling and component package used in component and system implementation. A service package contains classes that have a high level of interdependency, and serve a common purpose by delivering a consistent set of services. A component package represents an executable component, i.e. actual code. The Select method does not provide a systematic, formal way for building component-based systems. It rather suggests using the best of breed tools and techniques.

Catalysis represents the most comprehensive and complete CBD method among the presented ones. The first impression about Catalysis is its complexity, although it is based on a small number of underlying concepts, namely type, collaboration, refinement and framework. Catalysis has an object-oriented foundation; it is primarily dedicated to object-oriented development, which impacts in a great deal its way of handling CBD concepts. The Catalysis process is not a rigorous methodology, formally directing a system development. It is rather a semi-structured set of design principles, advices, guidelines and patterns that should be practiced throughout the system life cycle. Therefore, a systematic "roadmap" of the Catalysis way is lacking. The whole method tends to be fuzzy, with possible difficulties of applying it in practice.

All presented methods do not go far enough in their support of a component concept. It results in the fact that components are treated mainly as implementation and deployment artifacts, instead of being central point throughout the complete system life cycle. A systematic component way of thinking in the development process is missing as well as proper support for component concepts in requirements analysis, modelling and specification of a component-based system. The conclusion is that a more formal and systematic approach for component-based development is needed. It must incorporate CBD concepts into each phase of the life cycle, efficiently targeting existing CBD technology infrastructure at the end. Integration of various enterprise system aspects, not only technology issue than business, information, and application issues as

well, must be provided. Having that in mind, we have proposed a new systematic and integrated approach to component-based development of complex enterprise-scale systems, providing comprehensive, theoretical and practical support for CBD paradigm.

3. AN INTEGRATED CBD APPROACH

In order to gain full benefit of CBD in enterprise system development, a component concept must be the integral part of the whole system life cycle, from business requirements to implementation and deployment. The sooner we can identify components and integrate them first at the business requirements level and then into an appropriate model of the enterprise system, the faster and more accurately we can build the target system, through component-based analysis, design and implementation.

3.1 Integrated Component Concept

Among the other important characteristics and benefits of the CBD approach claimed so far, in our opinion the component concept represents an excellent solution for providing a meeting point between technology and business worlds. By defining a component as an encapsulated concept, clearly delimited from the environment, with specific roles and behaviour in the domain, with hidden interior and exposed functionality through services and interfaces, it can be easily understood by both worlds. Such concept gives business analysts and managers greater ability to model business processes and requirements at a higher-level, in a domain-specific, but implementation-independent way. On the other hand, the application developers retain control over how their models are turned into complete applications using advanced component-based technology infrastructure.

Object-orientation cannot be effectively used for that purposes. Objects can be too small in size and technologically oriented to be considered as basic units of a development process by business side people. The logic is usually too trivial to justify the expense of modelling, building, documenting and reusing a single object interface. On the other hand, business processes are often too fuzzy and complex to be uniformly and easily understood by IT side people. Furthermore, components as service-based concepts represent more natural approach for describing complex business processes than objects as entity-based structures. Thus, a component can represent a *lingua franca* for business and IT worlds, i.e. a means for integrating instead of separating them. Component hierarchy considering granularity and functionality issues is shown in the Figure 1.

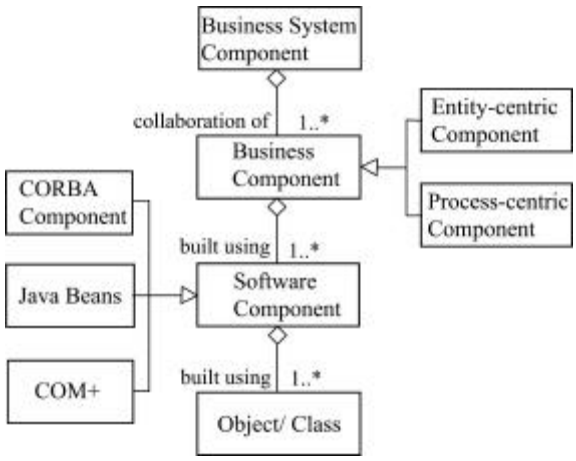

Figure 1. Component hierarchy.

- Business System Component represents a business process built up of functional pieces (components) that cooperate to deliver the cohesive set of functionality required by a specific business need i.e. to provide a solution to a specific business problem, such as Invoice Management or Order Management. It can be a constituent component of a larger business system.
- Business Component represents a single autonomous real-world business concept in a service-based business architecture. It encapsulates everything about that concept including name, purpose, knowledge, behavior, role, objective, etc. Examples include: Customer, Product, Order, Inventory, Pricing, Credit Check, and Billing. At the specification level, service-based business component is completely independent on particular implementation. It can be developed on any platform using any technique and tool, depending on available technical environment.
- Software Component is a physical building block used in the assembly of applications. As independent piece of software, it encapsulates data and operations, and fulfills specific service through well-defined interface. This type of component corresponds to the usual concept of component in the software industry, i.e. it may be represented as an Java Bean, CORBA or COM+ component. It is normally, but not necessarily, built up using object-oriented paradigm.

Business component has two facets, specification and implementation, and represents a basic unit of an enterprise-scale, web-based system. Business Specification Components represent the results of applying componentisation concepts and principles in the business domain. They model the enterprise, and thus enable future enterprise applications to more completely satisfy the business needs. They provide service-based, behaviour-partitioned view of the business domain and consequently an excellent way to define and scope the solution space, as a good basis for further development of an enterprise application.

On the other hand Business Implementation Components handles the aspects and realities of the technical environment in order to offer a component-based implementation of the certain business concept. They provide an excellent way of applying "divide-and-conquer" principle at the implementation level in a way that ties application to the business domain. They provide the capability to package software realisation into more meaningful and manageable artifacts, inheriting the benefits of CBD approach. Business Implementation Components smoothly evolve from specification ones, simply by taking into account the reality of technical environment. In that way, they provide a mapping between particular business concepts and their technical representation.

Business Components can be either entity-centric or process-centric. Business Entity Components have an emphasis on representing significant entities and concepts, as well as related information, inside the particular business domain. Business Process Components represent existing business processes, workflows and functionality in the domain. This categorisation cannot be quite strict, because most business concepts are a blend of information and functionality. However, this separation of concern can be helpful in analysis and design to better understand the nature of business concepts, but especially in the implementation phase, where concepts like permanent data storage, as well as functions and function calls, are arising.

3.2 RM-ODP

RM-ODP is a joint standardisation effort of the ISO (International Standardisation Organisation) and ITU-T (International Telecommunication Union) [ODP, 1996/1-4]. It originally consists of four parts, namely Overview, Foundations, Architecture and Architecture Semantics. RM-ODP defines a reference model to integrate a wide range of future ODP standards for distributed processing, such as Trading function, Type Repository function, Naming Framework, and Quality of Service, as well as to maintain consistency among them.

RM-ODP defines a framework for describing architectures for distribution, interoperability and portability of applications based on object-oriented technology. It is widely recognised as offering the most complete and internally consistent specification framework that crosses organisational and technological boundaries. RM-ODP specification of a system consists of five different specifications, corresponding to five different, but related and consistent viewpoints, Figure 2.

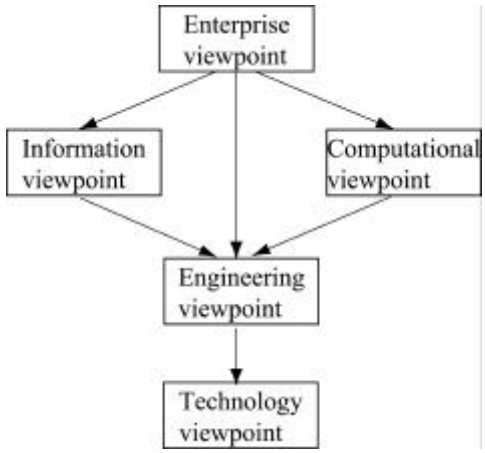

Figure 2. RM-ODP Viewpoints.

- Enterprise viewpoint – focuses on the purpose, scope and policies governing the activities of the specified enterprise system.
- Information viewpoint – focuses on the kinds and semantics of the information handled by the system, as well as information processing and constraints on it.
- Computational viewpoint – focuses on the functional decomposition of the system, enabling system distribution.
- Engineering viewpoint – focuses on the mechanisms and functions of the infrastructure for distributed processing support.
- Technology viewpoint – focuses on the choice of technology to support system distribution.

Each viewpoint is an abstraction of the whole system focusing on a specific area of concern, and in combination with others provides the complete specification of the whole enterprise system. Five viewpoints have been chosen to be both simple and complete, covering all the domains of architectural design. For each viewpoint there is an associated viewpoint language, which can be used to express a specification of the system from that viewpoint.

3.3 ODP-Based CBD Approach

By using two such powerful concepts in enterprise distributed processing, CBD and RM-ODP, in a meaningful and seamless combination, we intend to define a comprehensive, systematic, and flexible approach to enterprise system development, Figure 3. It is designed to ensure a systematic path from business to technology in the CBD manner, integrate the semantic layers on that path and provide as smooth a transition as possible between them. In that way business inspired component-based development and/or CBD directed business solutions are effectively achieved.

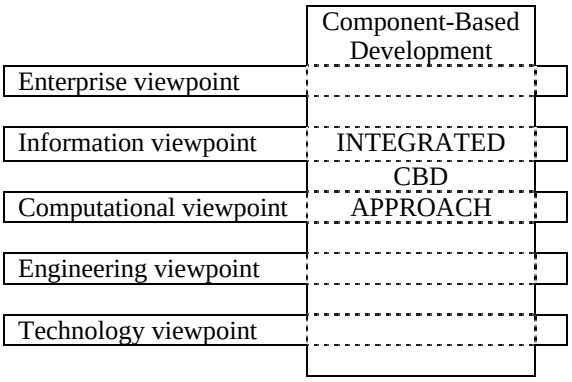

Figure 3. Integrated CBD approach.

Since the component middleware infrastructures, like OMG/CORBA and COM, are already widely accepted and used, the ODP Engineering and Technology Viewpoint Specifications can be based on them. In the case of CORBA, these specifications can be developed using CORBA ORBs and IDL, CORBA services and CORBA common facilities [Siegel, 2000]. This makes the semantically-lower part of the ODP Specification component-oriented. Our aim is to incorporate CBD concepts and principles into the other ODP viewpoints, first to Enterprise Viewpoint, and then to Information and Computational Viewpoints. In that way, the component concept, seen from different perspectives, is the common conformance point and unified factor across all ODP viewpoints.

Enterprise Viewpoint defining purpose, scope and policies of an ODP system, naturally fit into behaviour-focused, interface-driven component specification and modelling. The main concepts of this viewpoint, namely community, contract, role, behaviour, enterprise object and policy perfectly meet the component concept theory, and can be represented using the concept of business components. Community is the key enterprise concept in RM-ODP, defined as a configuration of objects formed to meet an objective, and can be represented as a Business System Component, formed by a collaboration of enterprise objects, i.e. business components of a lower granularity. Roles, contracts and policies can be defined through specifying interfaces and behaviours of enterprise objects, as well as constraints, pre- and post-conditions on these interfaces.

Information viewpoint specifies the significant information stored and processed by the system, as well as invariants, constraints, and rules regarding the transformation of this information. Three types of information schemas are used for this purpose in RM-ODP, namely static (rules about state and structure of information at some point in time), invariant (information independent of behaviour) and dynamic (evolving of information through time). Elements of these schemas can be defined as Business Entity Components, representing elements in the ODP system, which carry necessary information. And again, information transformation and processing can be specified by component interfaces, with corresponding pre-, post- and invariant conditions representing constraints and rules on that information management.

Computational viewpoint is used to specify the functionality of an ODP system in still a distribution-transparent manner. It specifies functionality and process flows in the system by defining services offered by service-provider objects and used by service-consumer-objects. This functional linkage between objects is used to define complex collaboration and interaction in the ODP system. Object in the Computational viewpoint can be represented as Business Process Components, while interaction between them can be specified through their provided and required interfaces. Some aspects of an implementation issue must be taken into account, since the placing of component services and function flows in a multi-tier system architecture should be considered as well.

Typically, an Enterprise object in the Enterprise viewpoint as a Business Component of a higher-level abstraction and granularity can map into one or many objects from Information and Computational Viewpoints. Roles, behaviours and policies of an enterprise object are represented by conditions and rules related to information object(s), along with functions and corresponding constraints specified by computational object(s).

Since the Engineering and Technology Viewpoints are already component-based by using CORBA (or COM) component model, introducing the component concept into remain viewpoints makes the whole RM-ODP component-oriented. In that way the component concept can serve as a conformance and consistent point among the ODP

viewpoints. Since RM-ODP originally represents an object-oriented framework for specifying open, flexible and distributed systems, UML as a standard notation for object-oriented and component software design, along using available extension mechanisms, can be used for specifying component-based ODP viewpoints. The Object Constraint Language (OCL) [Warmer and Kleppe, 1999] can be used to specify constraints, conditions and invariants on ODP concepts and elements. By combining RM-ODP as a standard framework, CBD as an advanced development paradigm, and UML as a standard notation, we intend to define a new, integrated CBD approach, covering the whole system life cycle in a component-oriented manner. Using RM-ODP framework as a standard base results in applying of CBD principles and practice, from concepts and requirements to implementation and deployment, in a rigorous and consistent manner. Only in that way, the full benefit of CBD paradigm will be achieved in complex enterprise system development.

4. CONCLUSION

As effective as possible way of using new Internet, even mobile, technologies are nowadays becoming a crucial point in organisations' business strategy, and main factor for being competitive on the market. New enterprise systems are becoming more and more complex, under the constant pressure of ever-changing business and IT requirements in the environment. One of the solutions for more efficient managing of complexity and changeability factors in the enterprise systems world is a new paradigm for system development, known as component-based development (CBD). CBD has mainly impacted the technology infrastructure level, where CORBA, COM+ and EJB represent de facto standards for component-based middleware. However, a set of methods and techniques are needed, starting from earlier phases of the system life cycle and targeting that technology. Current methods supporting CBD do not go far enough in their support of a component concept. It results in the fact that components are still handled mainly at the implementation and deployment phase, instead of being focal point through the complete system development process.

In this paper, we have proposed a new systematic and integrated approach to component-based development of complex enterprise-scale, information systems. By offering well-defined consistent CBD theory, through basic definitions, and categorisations, and by integrating them with the ISO RM-ODP standard framework for open distributed processing, we intend to provide a comprehensive method for applying CBD concepts and principles throughout all phases and from all viewpoints of the system development in an integrated, systematic and consistent manner. Our idea is that by combining and integrating two such powerful concepts CBD and RM-ODP, a complete set of principles, techniques and guidelines for development of complex enterprise systems using component way of thinking can be provided.

REFERENCES

Allen, P., Frost, S. (1998), "Component-Based Development for Enterprise Systems: Applying the Select Perspective", Cambridge University Press.

Booch, G., Rumbaugh, J., Jacobson, I. (1999), "The unified modeling language user guide", Add.-Wesley.

Brown, A.W., Wallnau, K.C. (1998), "The Current State of Component-Based Software Engineering", IEEE Software, September/October 1998.

D'Souza, D.F., Willis, A.C. (1999), "Objects, Components, and Frameworks with UML: the Catalysis Approach", Addison-Wesley.

Jacobson, I., Booch, G., Rumbaugh, J. (1999), "The unified software development process", Reading MA: Addison-Wesley.

Microsoft (2001) COM, DCOM, MTS, ActiveX, COM+, available at http://www.microsoft.com/com/

ODP (1996/1), International Standard Organisation (ISO), Information technology - Open Distributed Processing - Reference model: Overview, ISO/IEC JTC1/SC07, 10746-1, ITU-T X.901.

ODP (1996/2), International Standard Organisation (ISO), Information technology - Open Distributed Processing - Reference model, Part 2: Foundations, ISO/IEC JTC1/SC07, 10746-2, ITU-T X.902.

ODP (1996/3), International Standard Organisation (ISO), Information technology - Open Distributed Processing - Reference model, Part 3: Architecture, ISO/IEC JTC1/SC07, 10746-3, ITU-T X.903.

ODP (1996/4) International Standard Organisation (ISO), Information technology - Open Distributed Processing - Reference model, Part 4: Architecture semantics, ISO/IEC JTC1/SC07, 10746-4, ITU-T X.904.

OMG (1999a) Object Management Group: Unified Modeling Language 1.3 specification, document ad/99-06-09. Available at http://www.omg.org/uml/

Siegel, J. (2000), "CORBA 3: Fundamentals and Programming" OMG Press, John Wiley & Sons, Inc.

Sun Microsystems (2001), The source for Java™ technology at http://java.sun.com

Szyperski, C. (1998), "Component Software: Beyond Object-Oriented Programming", ACM Press, Addison-Wesley.

Warmer, J.B., Kleppe, A.G. (1999), "The Object Constraint Language: Precise Modeling with UML", Reading, Mass., USA: Addison-Wesley.