Component Based Development Methods - comparison

Dan Laurențiu Jişa

***Abstract:** This paper realizes a comparison among three of the best known component based development methods, emphazing on the earlier phases of the development process (domain modeling, requirements modeling, analysis) and on the modality used to identify business requirements and to encapsulates them in software components. In the first part of the paper it tries to establish comparison criteria, used in the second part in order to compare the methods.*

***Keywords:** UML, Component Based Development, Business Components, Object-Oriented Analysis and Design.*

INTRODUCTION

The evolution of the software applications development paradigms was significantly influenced by the continuous growing of the demand on the software applications market and by the increasing of applications complexity. Development based on components opens the possibility to assemble applications from predefined building blocks. The main advantages are:

- better management of the applications complexity;
- decrease of the development effort;
- increased flexibility;
- increased quality (there are used solutions whose correctness was proved in earlier projects).

Although the components based development promises significant benefits, there are, however, some technical problems that limit their use. Some of these problems are the following:

- how must be captured and refined the business requirements, based on a process that leads to the development of a component based system;
- how the components must be put together and deployed using the latest technologies;
- how could be retrieved in a library the components that are closest to the developers' needs.

This paper tackles the problems related to the first of the three aspects above mentioned. There are several approaches in the component based development, proposed by the research centers and IT-business industry: KobrA, Castek CBD methodology etc. Three of the best-known methodologies, discussed in this paper, are: the unified software development process [3], Catalysis [4] and Select Perspective [1].

A software component can be defined as a strong and flexible modality used to implement reusable services. These services are available to the component's users through the interfaces implemented by the component. In the last years there have been accomplished considerable progresses concerning the middleware technologies (CORBA, Enterprise JavaBeans, .Net platform).

According to [5], components based architecture implies the existence of the following elements:

- User interface layer;
- Business layer – this includes process components (provides local business functionality), business domain components (provides functionality across different business processes), business infrastructure components (provides functionality across different business domains);
- Technical Infrastructure Layer.

As regards the development of components belonging to the business layer, these still represent a challenge for companies in the software industry. A business component can be defined as a software implementation of an autonomous business concept or process ([7]).

1. A framework for the comparison of the component based development methods

The component fabrication consists of various phases [8]: domain analysis, component identification, component design, implementation, acceptance and roll out and deployement. The paper discusses the first three of the above mentioned phases.

Table 1 presents the stages and the outcomes for each stage, in business components building process.

Table 1

Domain analysis	Domain model
Components identification	Requirements model
	Analysis model
	Architectural model
Components design	Detailed design model
	Deployment model

The goal of the component fabrication process is to design business components that can be reused within the same domain or may be reused across domains. As a consequence it is important how the CDB methods drive their activities towards the identification of domain specific processes and to the encapsulation of these processes in software components. Related to this problem, this paper tries to establish some criteria in order to compare the CBD methods.

Returning to the stages mentioned in table 1, their goal in the discussed problem is as follows:

1) Domain Analysis – it achieves a detailed domain model. This model will be independent of the application that must be built. The analysis begins with the study of the processes specific to the organization for which the application is built. As a result of decomposition, the domain specific processes (that can be reused within the domain) can be identified and, in the next phases of development, encapsulated in business components.

From this point of view the following comparison criteria are proposed:

a) Concepts and notations used to build the business model;
b) Building up a glossary of terms;
c) Domain objects identification and the relationships among them;
d) Dynamic modeling (building up business model);
e) Business rules modelling (utilization of invariants, pre and postconditions for the business rules specification);
f) Activities for business model realization.

2) Component identification - the business components identification process has to begin in the analysis phase of the developed system. The classes that collaborate in order to achieve a business process are clustered in analysis packages (in accordance with the unified process terminology). The classes must be clustered so that to obtain high cohesion values for the classes belonging to the same package and values as low as possible for the coupling among classes in different packages.

There are proposed the following comparison criteria:

a) Requirements model realization;
b) Activities specification for identification of reusable functionalities, based on the business model;

c) Role based identification of the components' interfaces;
d) Building up of an application architecture;
e) Components interaction modeling;
f) Trasability between business model – requirements model – analysis model – high level design model;
g) Guiding the activity in order to group classes into components.

3) Components design – once the business processes identified and the classes which collaborate in order to realize the business processes, grouped in packages (modules), the detailed design phase for each of the packages identified can begin. In this phase the packages may evolve into software components.

For the components design stage, there are proposed the following criteria:
a) Trasability with early stage packages;
b) Use of design patterns for components design;
c) Activities description for integration of legacy applications.

This latest aspect is important because there are situations when in an organization there already exist applications that implement some of the processes that must be included into the system under development.

Taking into consideration the detailed design is not independent of the implementation environment, in addition to the above criteria, the following can be considered too:
d) Realization of a physical architecture. Concepts and notations.
e) Realization of a deployment model. Concepts and notations.

2. A comparison among Select Perspective, Catalysis and Unified Process

Farther on, the three selected methods will be compared related to the criteria from the first two categories (domain analysis and components identification) discussed in the previous section.

1) Domain analysis

As regards the construction of a domain model, all three methodologies build up a domain model, but the used concepts and notations are different.

Unified Process – there are used stereotyped UML notations contained in the *UML Profile for Business Modeling*: business actor, business entity, business worker, business use case etc.

The diagrams used in business modeling are as follows: business use-case diagrams, business objects diagrams, interaction diagrams and activity diagrams (in order to describe the workflow within a business use-case).

Select Perspective – uses a notation adapted after CATALYST methodology (CSC propriety). A business process is decomposed to the elementary business processes level (EPB). For each EPB the steps that compose it will be identified.

Catalysis – the method use the same concepts and notations all through the development lifecycle. In addition to the standard UML concepts and notations there are two Catalysis specific concepts: type of objects (it is used instead of the class concept in the early phases of development) and collaboration (in order to model the interactions among objects).

As regarding the activities for business model specification, the situation is:

- *Unified Process* – there are not very clear activities described for the business model construction;
- *Select Perspective* – although the authors dedicate a whole chapter to the business model specification, it cannot be alleged that the method describes clear activities;
- *Catalysis* – as well as Select Perspective, the building of business model is detailed, but in contrast with the other two methods, Catalysis proposes a series of process patterns to be followed in the development process.

The next table shows the degree of covering criteria presented in chapter 2.

Table 2

Criteria	Unified Process	Select Perspective	Catalysis
Concepts and notations used to build the business model	+++	+	++
Building up a glossary of terms	+++	-	+++
Domain objects identification and the relationships among them	+++	-	+++
Dynamic modeling (building up business model)	+++	+	+++
Business rules modelling	+++	-	+++
Activities for business model realization	-	+	++

Table 3

Legend	
+++	Very well covered by the CBD method
++	The problem is partially covered
+	The problem is summarily discussed
-	The problem is not covered at all

<u>2) Components identification</u>

The business components identification have to begin with the identification of the processes specific to the domain. These may be common to manny applications developed for the same domain. The business model represents a base for these processes identification.

<u>Unified Process</u> – the unified process describes activities (not very detailed) used to identify requirements, starting from the business model.

A generation process of requirements model, based on business model, is also described in the paper "Towards Use Case and Conceptual Models through Business Modelling". The authors consider that the activities whithin process diagrams (represented by UML activity diagrams) are at a suitable granularity level to be associated with a single use case. In this way there is created a use case for each activity from process diagrams. The role that performs the activity will be the main actor for the use case. The business rules will be captured under the form of pre and post-conditions.

<u>Select Perspective</u> – in Select Perspective the business process is considered as being structured on activities levels. Each level uses the services provided by the lower level. Through the *service* concept the method brings in the notion of business component as early as the system requirements phase, from this point of view the definition of a business component is: a cohesive collection of related functionalities, accessed through an interface and encapsulated at implementation. The development of use-case model starts from the EPBs.

<u>Catalysis</u> – the building up of the requirements model according to the business model is not very clearly described in Catalysis (although there are presented o series of process patterns that can be applied to the requirements model construction).

All three methods are architecture centric:

<u>Unified process</u> – one of the main characteristics of the process is that it is *architecture centric*. During the development lifecycle an equilibrium between user requirements and application architecture will be reach.

The analysis model can be decomposed in analysis packages, both in a *top-down* and *bottom-up* manner:

- top-down – the packages are initially identified, as a modality to divide the analysis model;
- bottom-up – the analysis classes are identified, and as the model evolve in larger structures, it will be decomposed in packages (classes will be clustered in packages).

The unified process gives the following recommendations for use-case allocation to the packages:

- the use-cases that realize a certain business process;
- the use-cases that support a certain actor.

The unified process provides a good trasability between business processes (specified by the use-case), analysis packages and subsystems (which are the concepts used by unified process for components at design level).

In the earlier phases of development (analysis and high level design) application architecture is represented by means of the package diagrams. Later, in the implementation stages, the physical components are specified and the component diagrams are built.

Select Perspective – the method uses a bottom-up approach in order to build application architecture. Classes that collaborate to the use-cases realization are grouped in components. For a component interfaces specification it is used the concept of *service class*.

Catalysis – in the chapter "How to Specify a Component" there are described a series of process patterns which, if applied in the development process, lead to the development of a component based architecture.

Catalysis uses a *top-down* approach for the components identification and architecture building. For the beginning it considers the system as being an objects type and, as a result of patterns application, the interfaces will be identified for each role played by the system. By applying "Recursive decomposition – divide and conquer" pattern, the system will be decomposed in subcomponents, the process will be repeated for each of them.

The following table shows the degree of covering criteria established for components identification.

Table 4

Criteria	Unified Process	Select Perspective	Catalysis
Requirements model realization	+++	+++	+++
Activities specification for identification of reusable functionalities, based on the business model	++	+++	+
Role based identification of the components' interfaces	-	-	+++
Build an application architecture	+++	+++	+++
Components interaction modeling	+	-	+++
Trasability	+++	-	+++
Guiding the activity in order to group classes into components	+	-	-

CONCLUSIONS

As regarding domain analysis, the following conclusions can be asserted:

- the notations and concepts used by the unified process and Catalysis are more comprehensive, but the unified process has the advantage that it uses standard UML notations, so that there is a better UML Case tools support;
- both Catalysis and the unified process allow business rules specification as invariants, pre and post-conditions;
- the description of business modelling process is better in Catalysis than in the other two methods.

For the components identification stage, it can be concluded:

- system requirements identification based on the business model is well described in Unified Process and Select Perspective;
- all three methods are architecture centric; the notation used for component is the notation used in UML for package;
- the unified process ensures very well the trasability between system requirements (specified as use cases), analysis model, design model and implementation;
- interfaces identification is better specified in Catalysis and it involves specification of component context and of the component roles (for each role it will be assigned an interface).

REFERENCES

[1] Allen, P., S. Frost. Component-Based Development for Enterprise Systems - Cambridge University Press, 1998.

[2] Booch G., J. Raumbaugh, I. Jacobson. The Unified Modelling Language. User Guide - Addison-Wesley, 1999.

[3] Booch G., J. Raumbaugh, I. Jacobson. The Unified Software Development Process - Addison-Wesley, 1999.

[4] D'Souza, D.F., A.C. Wils. Objects, Components and Frameworks with UML – The Catalysis Approach - Addison-Wesley, 1999.

[5] Eweka-Esiet, P., D. Mehandjiska-Stavreva. Why a Component-based Architecture for E-business? - http://www.it.bond.edu.au/publications/02TR/02-10.pdf.

[6] Fettke, P., I. Întorsureanu, P. Loos. Komponentenorientierte Vorgehensmodelle im Vergleich - 4. Workshop Komponentenorientierte betriebliche Anwendungssysteme, K. Turowski (editor), Augsburg 2002, pp. 19-43.

[7] Herzum, P., O. Sims. Business Component Factory - Johm Willey & Sons, 1999.

[8] Jain, H., N. Chalimeda, N. Ivaturi, B. Reddy. BusinessComponent Identification – A Formal Approach - Fifth IEEE International, Enterprise Distributed Object Computing Conference, 2001.

[9] Molina, J.G., M.J. Ortin, B. Moros, J. Nicolas, A. Toval. Towards Use Case and Conceptual Models through Business Modeling - Conceptual Modeling - ER 2000: 19th International Conference on Conceptual Modeling: Salt Lake City, Utah, USA, October 2000, pp. 281-284.

[10] Stojanovic, Z., A. Dahanayake, H. Sol. A Methodology Framework for Component-Based System Development Support - www.betade.tudelft.nl/publications/ Stojanovic_EMMSAD2001.pdf.

ABOUT THE AUTHOR

Dan Laurențiu Jişa, PhD student, Academy of Economic Studies Bucharest, Romania, Phone: +40 21 721 214 592, E-mail: dan.jisa@estwest.ro.