ISSN 1392 – 124X INFORMATION TECHNOLOGY AND CONTROL, 2006, Vol. 35, No. 3 A

COMPONENT-BASED MODELLING AT THE SYSTEM DESIGN STAGE

Saulius Gudas[1,2], Edvinas Pakalnickas[2]

[1]*Vilnius University, Kaunas Faculty of Humanities*
Muitines 8, 44280 Kaunas, Lithuania

[2]*Kaunas University of Technology, Information Systems Department*
Studentu 50, 51368 Kaunas, Lithuania

Abstract. This paper presents an approach to applications development based on the principles of the model-driven architecture and using the component-based system model (CBSM). The CBSM helps to refine main components and interfaces of the application at the design stage. This model integrates all steps of IS development life cycle. The information system's architecture is structured considering a business system as a set of different domains with definite types of components, and with interfaces between the components of different types.

1. Introduction

The traditional approaches to engineering of information systems focus on identifying business requirements [11] and delivering the specific functionality required to automate some activities. Not enough attention is being attached to how the created system will interact with the rest of the business. As a result, there is often a gap between the business requirements and the systems implemented to support them. To bridge this gap, many organizations are developing enterprise architecture to provide a holistic vision of how systems will support their business [6].

Model-driven architecture (MDA) focuses on modelling activities in software development process and shifts the software development process from the writing code to the modelling activities. MDA separates the business level from the technological platform which implements information system. The key feature of the model-driven architecture is its ability to transform automatically the platform-independent model (PIM) into the platform-specific model (PSM). MDA uses modelling languages as programming languages

Using interface-based programming is the evolution of the object-oriented programming and design. The interfaces have made the software design more adaptable to the rapid changes of the business environment. While using interfaces, software systems achieve reusability, extensibility and maintainability [4,10].

The idea of the interface-based design and programming is used mainly in the service-oriented architecture (SOA) and the component-based development (CBD) approaches [14] that are realized in the technological approaches such as the Microsoft .NET platform and the Java 2 Enterprise Edition.

The CBD approaches use the interface-based design idea and therefore have advantages such as more effective management of complexity, a greater degree of reuse and a wider range of usability [8]

The CBD requires a systematic approach to focus on the component aspects of software development [1]. The traditional software engineering approaches must be adjusted to the component-based approach. The software reuse and rapid IS development are obtained building systems from components.

The building of the information systems from components requires methodologies and processes not only in relation to the aspect of development, but also to the entire life cycle of system [9].

This paper deals with the strengths and the limitations of temporary IS development approaches and depicts the IS development approach based on the component-based system model (CBSM) and the MDA principles.

2. Semantic gap at IS development process

Figure 1 depicts the IS engineering approaches in the IS development life cycle. On the horizontal axis there is the information system development life cycle, on the vertical axis there is the abstraction level of the IS development activities. The IS development approaches such as the enterprise architecture (EA), the business process models (BPM), the object-oriented analysis and design (OOAD) are used at different levels of abstraction in the IS development process. A very common route in the process of the IS

development is from BPM to OOAD or from EA to OOAD. For this reason, there arise some problems because the practices of BPM, EA and OOAD development are often disintegrated (Figure 1) and therefore a semantic gap emerges between the business's vision and structure of the high level and the systems implemented to support them.

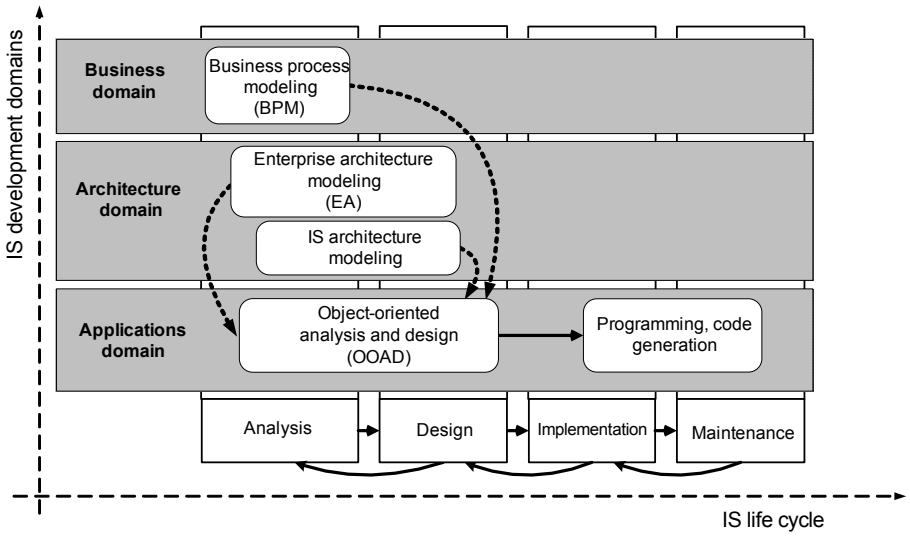

Figure 1. IS development activities

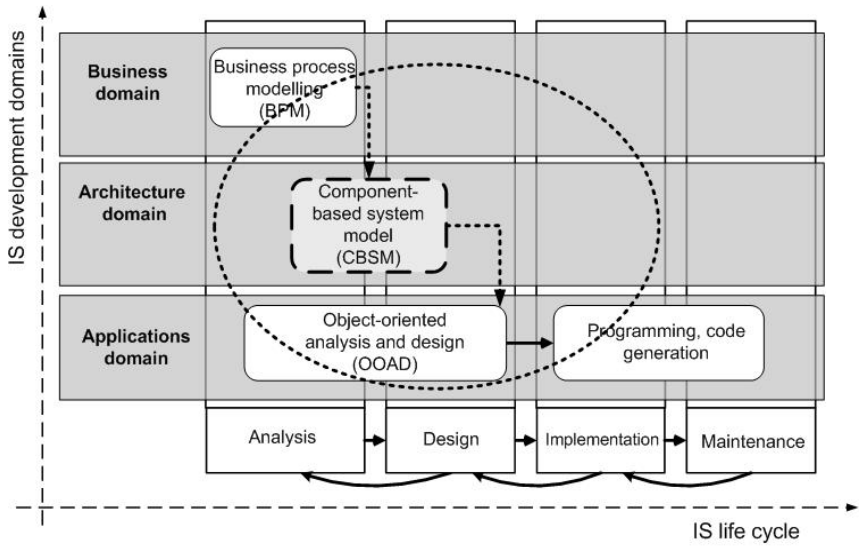

Figure 2. The place of the component-based system model in IS engineering process

2.1. Semantic gap at the IS development process

To bridge the gap between the activities of the IS development, it is reasonable to create a component-based approach (the dotted oval depicts proposed method application area), which is used to integrate elements from different modelling techniques (Figure 2) and the component-based system model, which integrates the enterprise architecture with the information system architecture (the dotted square with the rounded corners).

3. The main features of contemporary IS development approaches

The **Business process modelling** describes the business domain without any IT view. The business processes help to describe value-creating activities within enterprise and between organizations. The business processes describe also the interactions between any participants.

The processes exchange information with other processes. So we have a system of interacting processes and the interaction is performed by communication. The communication is used for exchanging information among the entities. The communication could be realized using interfaces. These interfaces will be described in the component-based system model (Figure 1).

Workflow model reflects the traditional view of the implementation of the processes in the enterprise. In the workflow model, the modelling of the business processes are sequences of activities that may be split

into the parallel sequences and partitioned into subsequences or elementary activities.

BPM approaches provide an end-to-end view on the functional units of the system, but typically they do not reach into the architecture and implementation domain.

The enterprise architecture frameworks such as DoDAF, TOGAF, Zachman are used in architecture modelling for IS development. The frameworks provide structured and systematic data and knowledge for designing of the information systems.

The definition of 'architecture' used in ANSI/IEEE Std 1471-2000 is: "the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution" [3].

An architecture framework helps to improve the understanding of the problem domain, but some aspects of the architecture remain not clear. There are some ambiguities [2]:

- Should the scope of the architecture encompass software components only or include other aspects of information system development?
- Architecture activities involve design and modelling, but which level of detail belongs to the architecture and when do the detailed design activities start?
- What is the relationship between the enterprise architecture and the information system architecture?

The proposed component-based modelling approach seeks to eliminate these ambiguities.

The Object-Oriented Analysis and Design on the applications' level enables efficient design and development of applications.

The main problems are that the OOAD approach is used on the applications level mainly, but not on the BPM level, and OOAD design level of granularity is at the class level, which is too low at the level of abstraction for the modelling of the business processes. A strong association such as inheritance creates a tight coupling and dependency between the OOAD entities, therefore it is difficult to maintain or extend system without breaking the code of the existing IS [10].

The OOAD models are used to model the software systems, a technical specification of software. BPM and OOAD are related, the business processes model can be a specification of the OOAD model, which is a specification of the information system.

The OOAD is a very valuable approach for design of the underlying class structure within a developed IS.

3.1. Model driven architecture (MDA).

Figure 3 depicts the MDA process, which starts with the creation of a computation-independent model (CIM). This model depicts the system for the business domain project members. The requirements are captured in the CIM. The CIM facilitates the communication between the domain experts and the system designers.

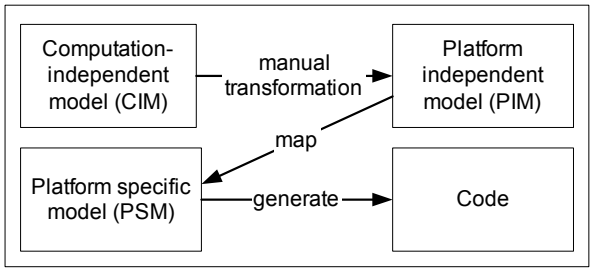

Figure 3. MDA process

After the CIM model is created, system designers build object-oriented models. These models are represented by the platform-independent model (PIM) where system requirements are represented by the UML diagrams. The PIM should capture the domain's semantics. The platform-independent models are mapped to the PSM models. A mapping between the models is assumed to take one or more models as its input and produce one output model. The rules for the mapping are described within a mapping function. The mapping functions enable the construction of the target models that are synchronized with their source models. The mapping functions alone are not always sufficient to transform a source model completely and additional inputs may be needed to perform the mapping, therefore marks are used. The marks are extensions to the models that capture the information required for the model transformation.

The key feature of the model-driven architecture is its ability to transform automatically the platform-independent model into the platform-specific model (PSM). Designers create the transformation rules, which automatically convert the model into the code. The UML modelling gives a high-level representation of the software system. The generated code creates a skeleton of the software system to be implemented [13].

4. The Component-based system model

Referring to the issues of the contemporary IS development approaches, there was developed a component-based system development approach, which enables to apply different elements from temporary IS development methods in order to accelerate and to simplify the process of IS engineering. It is generally acknowledged that the IT architecture should be aligned with the business processes. This approach allows aligning the IT architecture with the business processes.

The component-based view is used in all IS development phases. This lets assemble the information system from the distributed components or services

and to use the service-oriented view in the information system development process [3,5].

The developed approach uses ideas from the component and model-driven architecture and integrates the business requirements and the information system implemented to support the business processes.

Figure 4 depicts proposed component-based system model (CBSM) is a graphical notation for the identification of the information system components and their inter-relationships.

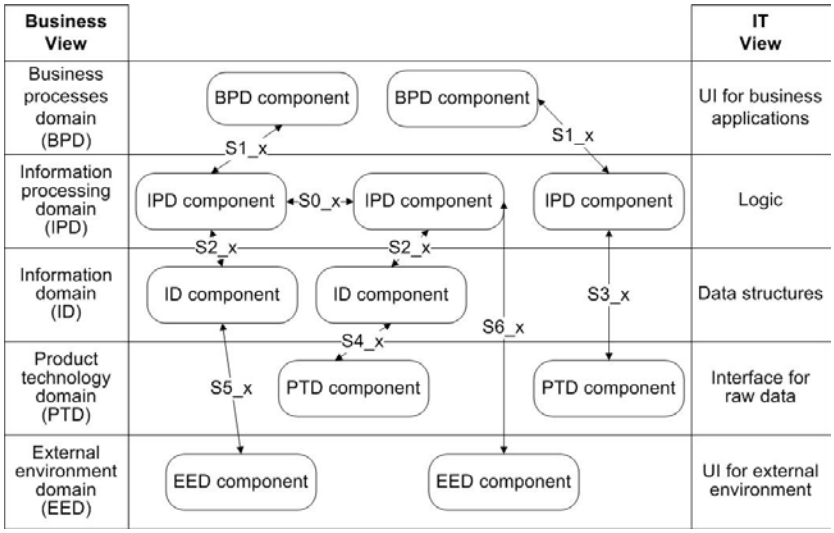

Figure 4. Component-based system model

The CBSM helps to break an overall IS into a number of related areas (domains) and to relate the enterprise architecture with the IS architecture [7]. This model describes IS as a set of interacting components or services and includes a clear definition of the collaboration between these components or services. The CBSM enables to create an interfaces model, which specifies for each component, what the component expects from the environment and what provides to the environment.

The CBSM lets decide how the components can be grouped, how they interact with the systems and between each other, how the components are shared across IS.

The components in the CBSM are displayed as a rounded rectangular, the interfaces are displayed using arrows marked with type of component's interface.

The meanings of the tracks of the component-based system model from the business viewpoint are as follows [12]:

- **business processes domain (BPD)** – includes the business processes, critical to the enterprise's functionality and development, for marketing, operation strategy, manufacturing planning, and human resources management;
- **information processing domain (IPD)** – identifies the major information processing activities that the enterprise performs to produce business driven decisions and products;
- **information domain (ID)** – includes the activities aimed to organize data and knowledge, necessary for the enterprise management and product development; for example quality control standards, products and process definitions, inventory files and etc.;
- **product technology domain (PTD)** – includes the technological processes and facilities for the development of the enterprise products and services; for example product design, materials processing and handling;
- **external environment domain (EED)** – includes the activities aimed to organize the processes with the enterprise suppliers and clients.

The meanings of the CBSM tracks from the information technology viewpoint are as follows:

- **user interface for business applications –** includes the components, which the enterprise users will use for the interaction in the system. For example windows, screens and menus;
- **logic –** includes the components of the application logic. These components embody the business rules;
- **data structures –** includes the data structures;
- **interface for raw data –** includes the data from the technological processes or control systems;
- **interface for external users –** includes the interface for the external users or the information systems of the clients and the suppliers.

The business domains interact between each other. The purpose of the interfaces is to integrate the domains' interaction. Interfaces are grouped in such types:

- **S0_x** – an interface between the components within the same domain;

- **S1_x** – an interface between the business process domain and the information processing domain;
- **S2_x** – an interface between the information processing domain and the information domain;
- **S3_x** – an interface between the information processing domain and the product technology domain;
- **S4_x** – an interface between the product technology domain and the information domain;
- **S5_x** – an interface between the external environment domain and the information domain;
- **S6_x** – an interface between the external environment domain and the information processing domain.

The index "x" shows the number of the interface's instance.

In order to use knowledge gathered in component-based system model, CBSM UML profile was created. This profile contains five stereotypes - <<BPD>>, <<IPD>>, <<ID>>, <<PTD>>, and <<EED>>. Stereotypes are used for components marking. When component is placed in particular CBSM track, it is automatically marked with track's stereotype. For example, for component from Business domain will be applied stereotype BPD.

5. Information system design - a case study

Figure 5 depicts the main roles and activities in the component-based development approach.

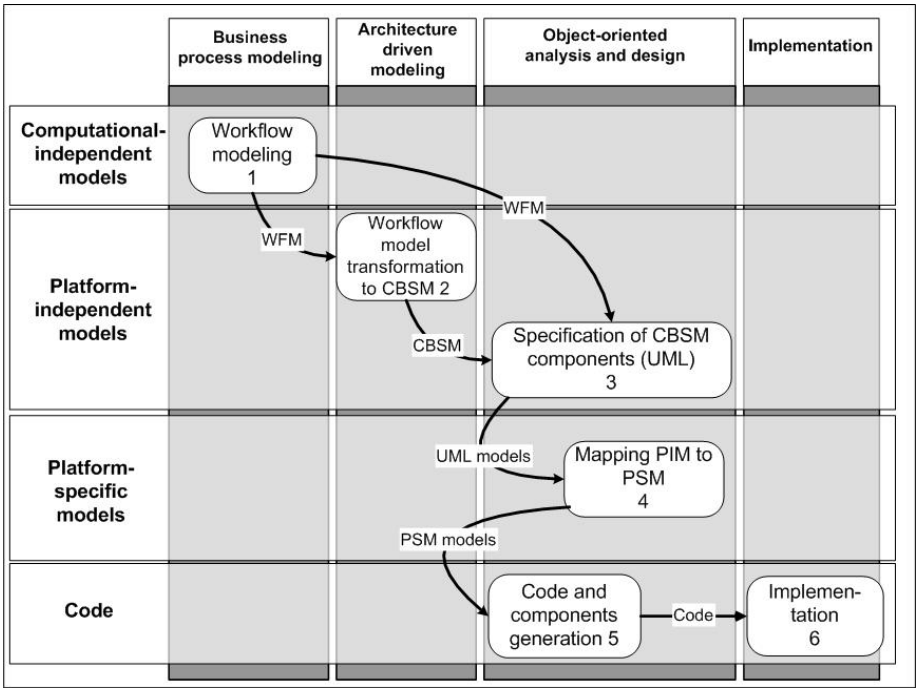

Figure 5. IS development process with CBSM

The first step is to perform the business process modelling (Figure 6) in order to capture the business requirements. For the business modelling it is used the MagicDraw tool and the Business Process Modelling Notation (BPMN) [4]. The BPMN is the standard for the modelling business processes. The BPMN is the most recognized standard used by the business users for the end-to-end business process modelling. The BPMN specifies a business process diagram (BPD). The BPD is easily understandable by non-IT users. The model is a description of how the business operates. From the MDA viewpoint, the BPD is a computation-independent model.

The second step is to perform the mapping from the computation-independent model to the platform-independent model. The mapping rules are applied when the designer performs the component-based analysis – analysing the BPD in order to create the component-based system model. From the MDA viewpoint this model, is a platform-independent model. The designer applies the following mapping rules:

1. the computational processes are transformed into the components of the information processing domain;
2. the management processes and the gateways are transformed into the components of the business processes domain;
3. the information flows connecting processes in the workflow model are transformed into the information domain components;
4. the material processes are transformed into the components of the technological processes domain;
5. the processes from or to an external business environment are transformed into the components of the external environment domain.

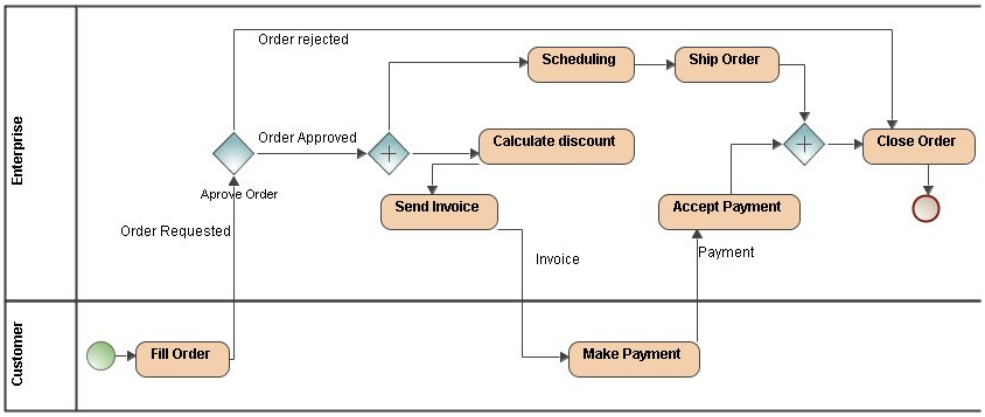

Figure 6. Ordering business process diagram

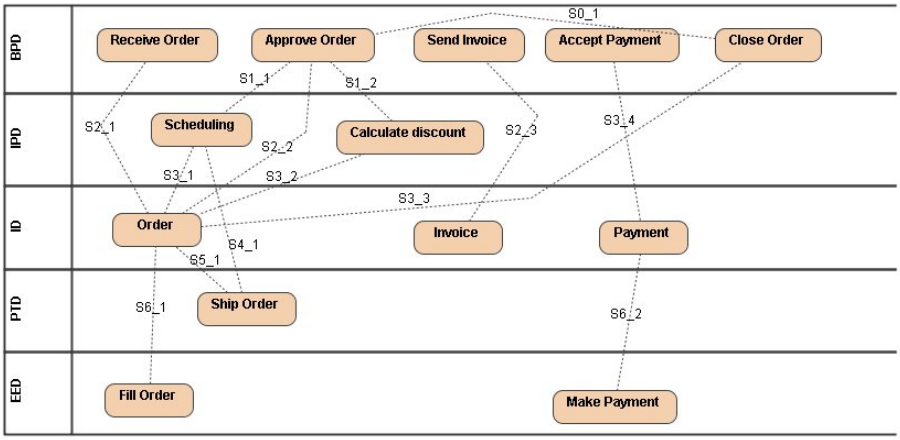

Figure 7. Component-based system model of the ordering process

The discovered components are placed in the CBSM (Figure 7). The interaction relationships between the components are considered as components' interfaces. The designer draws the interfaces as arrows between the components. There are several types of interfaces. The type of the interface depends on the fact, from which domains are the interconnected components. There are also constraints imposed on the interconnection of the components:

- the components of the business process domain can only talk to the components of the information processing and the information domains;
- the components of the information processing domain can talk to the business processes, the information domain components and the components of the product technology domain;
- the components of the product technology domain can talk to the information domain components and to the components of the information processing domain;
- the external environment components can talk to the information domain components and to the components of the information processing domain.

The major result of the CBSM development is the refinement of the main components (Table 1) of the IS and the identification of their interfaces (interactions) (Table 2).

Table 1. Components of ordering process

Domain	Components
BPD	Receive order, Approve order, Send invoice, Accept Payment
IPD	Scheduling, Calculate discount
ID	Order, Invoice, Payment
PTD	Ship Order
EED	Fill order, Make Payment

Table 2. Interfaces between components

Interface	Component	Component
S0_1	Approve Order	Close Order
S1_1	Approve Order	Scheduling
S2_1	Receive Order	Order
S2_2	Approve Order	Order
S2_3	Send Invoice	Invoice
S3_1	Scheduling	Order
S3_2	Calculate discount	Order
S3_3	Order	Close Order
S4_1	Scheduling	Ship Order
S5_1	Order	Ship Order
S6_1	Order	Fill Order
S6_2	Payment	Make Payment

The information system specification, developed using the CBSM, identifies the IS components' interfaces, the components and their relationships, bridging with the definite business model (i.e. with BPD in this case).

The third step of the component-based application development approach is detailed IS design – a specification of the components of all CBSM domains (tracks) using the object-oriented approach and the UML. In this step a sequence diagram of the ordering process is created. This diagram helps to specify the components' interactions. Figure 8 depicts part of the sequence diagram, derived from the component-based system model. For components specified in CBSM, are marked with stereotypes. Components in the BPD domain became GUI forms. For example component **Approve Order** became **Approve Order Form.** Similar component **Fill Order** became **Fill Order WebService.**

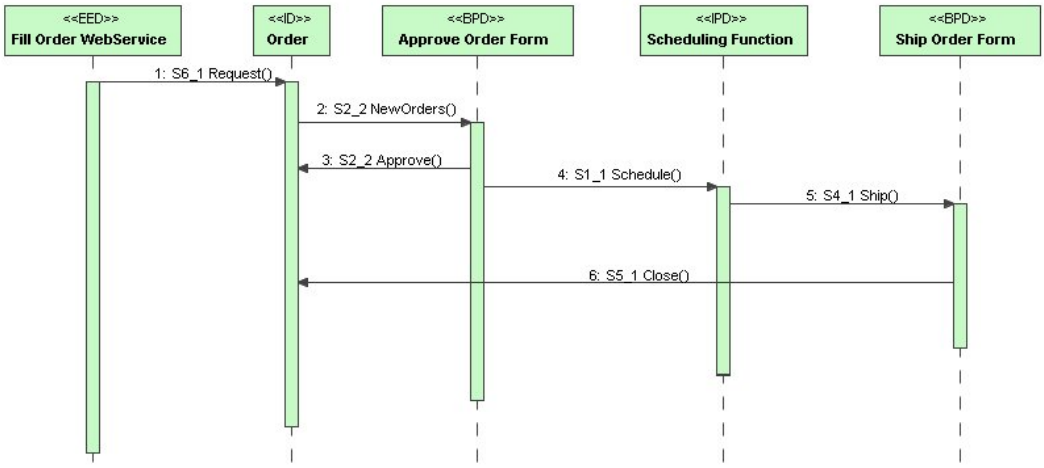

Figure 8. Part of sequence diagram

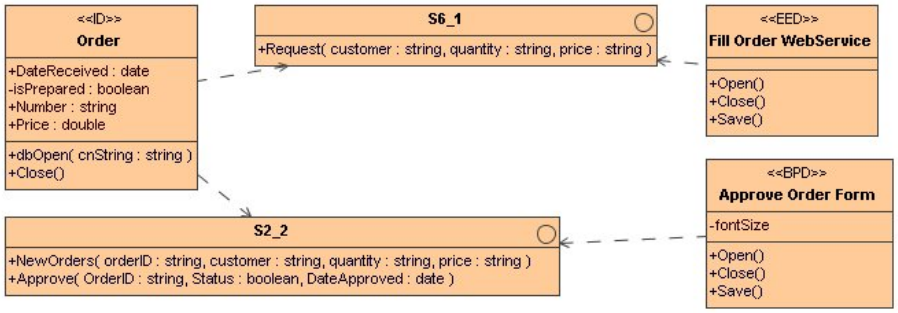

Figure 9. Part of UML class diagram

After the sequence diagram creation, will be created a class diagram for the components and their interfaces. Figure 9 depicts a part of detailed components' specification.

To exemplify the specification process of the components, detailed specification of components **FillOrder** and **Order** will be depicted. These components are interconnected by the interface S6_1. According to the CBSM (Figure 7) specification and the sequence diagram (Figure 8). The IS designer creates a detailed specification of the Order and FillOrder components (Figure 9). From the MDA viewpoint, in this step, the platform independent model (PIM) is created.

In the fourth step IS design UML models is converted to the platform specific models (Figure 5), applying a platform specific code patterns. For each component specification is applied a code pattern, which depends on which track component resides in the CBSM. The compiler "knows" which pattern to apply on component specification, because every component is marked with stereotype.

The fifth step is to perform code generation activities. In this step, information system component's program code is generated from the platform-specific specification of components. The following .NET program code (Table 3) is generated for the **Fill Order Webservice** and **Order** components.

Table 3 depicts the **Fill Order Webservice** and **Order** components' code. For **Fill Order Webservice** component program code generation, was applied external environment domain code pattern. That pattern modifies the external environment domain component design. If it is developed a .NET based application, the required properties for .NET WebServices are added, because information exchange with the clients and the suppliers will be performed through the WebServices. After the code generation, in the component signature automatically is added a code

required for the Webservices. The code, which depends on the component stereotype, is different for the components **Fill Order Webservice** and **Order**, because these components are from different CBSM domains – the **Fill Order Webservice** is from the external environment domain, whereas the **Order** component is from the information domain. The **Fill Order Webservice** component will be used in the Webservices implementation. The **Order** component will be used in the application server for the communication with the DBMS for storing data in the database.

Table 3 shows that both the **Fill Order Webservice** component and the **Order** component implement the same interface S6_1. This ensures the application code integrity, because all changes in the IS design transform automatically into the components' code.

Table 3. Code generation results

Generated Fill Order Webservice component code for WebServices	**Generated Order component code for Application server**

(CBSM domain specific code)
`Option Explicit On`
`Option Strict On`
`Imports System`
`Imports Systm.Web`
`Imports System.Web.Services`
	(CBSM domain specific code)
	`Option Explicit On`
	`Option Strict On`
	`Imports System`
	`Imports System.Data`
	`Imports System.Data.SqlClient`

(interface S6_1 code)
`Public Class Fill Order Webservice`
`  Implements S6_1`
`...`
`  Public Sub Request(ByRef customer As String, ByRef quantity As String, ByRef price As String) Implements S6_1.Request`
`  End Sub`
`...`
	(interface S6_1 code)
	`Public Class Order`
	`  Implements S6_1`
	`...`
	`  Public Sub Request(ByRef customer As String, ByRef quantity As String, ByRef price As String) Implements S6_1.Request`
	`  End Sub`
	`...`

After the code generation the .dll libraries is created (they could be used as components) using the VB.NET command line compiler vbc.exe.

In the sixth step the compiled components is used in the system implementation process.

6. Conclusions

The proposed approach enables to use the component-based view in the applications' development and helps to refine and to specify the main IS components and the interfaces between them.

The key benefit of this component-based approach is the integration of the model-driven architecture and the component-based development principles with the elements from the BMP, the EA and the OOAD approaches. The integration of the application's development activities narrows the gap between the business requirements and the system implemented to support the business processes.

There are three major features of the presented component-based system model aimed to improve the IS development quality. First, the graphical modelling makes the process of the application's specification easier to understand. Second, the CBSM displays the results in a suitable form that is easy to understand for the system developers. Third, the CBSM relates the business requirements with the information system architecture and enables the requirements' traceability from business to software artifacts.

Creating an environment that supports the CBSM will enable developers to capture benefits from the model-driven architecture and the component-based applications design.

References

[1] **I. Crnkovic, M. Larsson.** A Case Study: Demands on Component-Based Development. *Proceedings of* 22*nd conference Software Engineering, Limerick, Ireland, ACM Press*, 2000.

[2] **A.Tang, J. Han, P. Chen.** A comparative Analysis of ArchitectureFrameworks. *http://mercury.it.swin.edu.au/ctg/weekly/tang1511.pdf*.

[3] **M. Owen, J. Raj.** BPMN and Business Process Management. *http://www.bpmn.org/Documents/ 6AD5D16960.BPMN_and_BPM.pdf*.

[4] **R. Wuyts, S. Ducasse.** Composition Languages for BlackBox Components. *http://www.iam.unibe.ch/~scg /Archive/Papers/Wuyt01c.pdf*.

[5] **C. Szyperski.** Component Software.Beyond Object-Oriented Programming. *Addison-Wesley*, 1998

[6] **A. Macaulay.** Enterprise Architecture Design and the Integrated Architecture Framework. *http://www. itarchitect.co.uk/articles/display.asp?id=34*.

[7] **E. Pakalnickas, S. Gudas.** Informacijos sistemos projektavimas ir realizavimas komponentiniu metodu. *Informacijos mokslai*, t. 24, 2003, 59-68.

[8] **A.W. Brown.** Large-Scale Component-Based Development. *Prentice Hall PTR*, ISBN: 0-13-088720-X, 2000.

[9] **T. Takeshita.** Metrics and Risks of CBSE. *Proc. 5th Int. Symp. Software Tools and Technologies, Pittsburg, PA, IEEE Computer Society*, 1997.

[10] **E. Pakalnickas, S. Gudas.** Sąsajomis grįstas IS projektavimas. *Konferencijos "Informacinės technologijos* 2006" *pranešimų medžiaga*.

[11] **R. Butkienė, R. Butleris, T. Danikauskas.** The approach of consistency check-ing of functional requirements specification. *Proceedings of 6th World Multi-conference on Systemics, Cybernetics and Informatics, Vol.*XVIII, *Information Systems Development* III, *Orlando, USA*, 2002, 67-72.

[12] **S. Gudas.** The component-based information system requirements modeling. *Business operation and its legal environment: processes, tendencies and re-sults. Riga "Biznesa augstskola Turiba" SIA*, 2002, 98-102.

[13] **T.Meservy, K.D. Fenstermacher.** Transforming Software Development: An MDA Road Map. *http://www.compuware.com/dl/mdaroadmap.pdf*.

[14] **A. Brown, S. Johnston, K. Kelly.** Using Service-Oriented Architecture and Component-Based Development to Build Web Service Applications. Internet: *http://www-128.ibm.com/developerworks/rational /library/content/03July/2000/2169/2169.pdf*.

Received August 2006.