Development of industrial information systems on the Web using business components

Somjit Arch-int*, Dentcho N. Batanov

Computer Science and Information Management Program, School of Advanced Technologies, Asian Institute of Technology, P.O. Box 4, Klong Luang, Pathumthani 12120, Thailand

Abstract

Web technologies, as tools, are most popular and widely used today not only in the everyday life but in industry as well. Examples of such technologies are the browsers, search engines, related communication protocols and their implementations, mechanisms for creating dynamic and active Web sites, security mechanisms, etc. These technologies lead to requirements for substantial changes in the methodologies for development and deployment of all systems and subsystems, which constitute any business process. Changes and/or new decisions are necessary to make possible the modern Web-based technological tools to be used effectively for Web-based business processes. A methodology for developing business software components as basic building elements of industrial information systems, implemented and deployed in Web-based computing environment is proposed and discussed in this paper. The methodology is based on the idea of creating a unified framework for representing business activities as a basis for business process modeling. Respective models, methods and techniques ensure the business objects and components to be applied to currently most accepted three-tier Web application architecture using XML document as distributed object. For illustrative purposes only the proposed methodology is applied step by step, with respective comments, to developing a part of inventory subsystem of industrial information system as a case study.

© 2002 Elsevier Science B.V. All rights reserved.

Keywords: Business object; Business component; Web information system architecture; Distributed object; Industrial information systems

1. Introduction

Web-based industrial information systems are widely accepted and recognized as one of the most challenging for the development and implementation of modern industrial systems (e.g. [1,2]). The challenge stems from the fact that Internet, and more specifically, the Web, is a completely different computing environment compared to conventional computer-based environments.

Currently, there are basically two types of approaches to Web-based information system development or Web engineering [3]. The first is extended and/or modified Web site development as shown in [4]. The approach emphasizes Web site management, a document-centric development as opposed to an application platform. Since Web-based information systems are database-intensive systems, such approaches should focus not only on presentation, but also business logic, including databases as well.

The second type of approach has its origin in software development and is based on the premise that a Web-based information system is a product of software engineering (e.g. [5]). These approaches extend
or modify existing software engineering approaches for Web-based information systems [6]. Although, results from the analysis and design phases for both software engineering and Web engineering of a particular business system are the same, the software implementation model does not fit the Web implementation model [3]. Most software engineering approaches are based on the object-oriented paradigm and provide powerful mechanisms such as encapsulation, inheritance, and reusability [7,8]. The use of objects as building blocks, however, does not represent directly functional requirements and, therefore, such approaches need further enhancement in order to master dynamic systems.

Snoeck et al. [9] stated that a business process is divided into function-oriented (activities) and process-oriented parts (relationship between activities). The part, which is function-oriented, represents object behavior while the process-oriented part acts as a collaboration of these activities. Thus, emphasizing structural aspects is mainly a result of modeling individual object behavior but neglects identifying functional aspects of the system as a whole.

The most crucial issues in modern industrial information systems are flexibility, adaptability, and maintainability. To obtain these qualities, a component-based software development approach implementing business processes with software components is needed. Whenever a business process changes, the corresponding software component is also adapted appropriately. Furthermore, any existing software component can be reused/rewired for different enterprises to reduce development time.

This paper proposes the business process-based methodology (BPBM) for developing Web information systems constructed using components. This approach blends advantages of the structured [10] and object-oriented paradigms [7,8,11,12] for identifying and designing components called business components. The central idea of business component modeling is reusability of elementary units, which are business activities. An elementary unit that represents an atomic changeable business process can be implemented with a portable set of Web-based software components.

The BPBM should be able to answer at least the following two research questions. First, which methodologies are the most suitable for industrial information systems based on a Web environment? Second, how do these methodologies provide mechanisms in supporting data interchange? The requirements and infrastructure for information systems on the Web are reviewed in Section 2. The basic definitions are defined in Section 3. The proposed methodology for developing Web information systems is described in Section 4. A case study demonstrating the approach is shown in Section 5. The conclusions and some recommendations for future work are outlined in Section 6.

2. Information systems on the Web

2.1. Requirements

Traditionally, information systems on the Web that support Business-to-Customer (B2C) requirements rely on the typical three-tier Web-based information system architecture—client-tier (user interface), Web server-tier (business logic) and data source-tier based on Hypertext Transfer Protocol (HTTP) and HTML document. Another requirement for Web-based information systems is to support the Enterprise Application Integration (EAI) or Business-to-Business (B2B) relations. Common requirements of such systems are:

- **Heterogeneous data sources**: Almost all Web-based information systems are database-oriented systems implemented with various database sources. These databases are handled by their own proprietary vendors or by standard database engines such as ActiveX Data Objects (ADO), Open Database Connectivity (ODBC) or Java Database Connectivity (JDBC). To provide more independence between Web application server software and database accessing, such systems must have a mediator for handling the interface.

- **Universal clients accessibility**: Clients should not be required to have a high performance machine. At the same time, presentation techniques of clients should support a variety of output formats (e.g. HTML, PDF, etc.) and support various client devices (e.g. Personal Digital Assistant (PDA), wireless phone, etc.).

Using XML-based document [13] as middle-tier distributed object to interface the client-tier and the
data source-tier based on HTTP is a powerful solution to support the heterogeneous data sources accessing. XML documents combined with a variety of XSL stylesheets [14] can also support various client requirements.

2.2. Web-based information systems infrastructure

The foundation infrastructure of a Web-based information system, HTTP and HTML-based document, is enhanced with distributed object infrastructure such as OMG’s CORBA/IIOP, Microsoft’s DCOM and Java’s RMI shifting from traditional Web information systems to distributed computing information systems. However, this foundation not only causes shortcomings such as browser incompatibility and solid network—a permanent connection between client and server for each session [15], but also requires a client machine to be of high performance, which conflicts with the requirements for universal client accessibility.

XML, as a standard document representation, supports both kinds of requirements as mentioned above and provides Web-native programming development. Most commercial Web browsers (e.g. IE and Netscape) and Web servers (e.g. Apache’s Cocoon [16] provides XML-enabled Servlet) support XML-based Web information system development that offers features of electronic data interchange (EDI) [17]. XML documents submitted from servers are either rendered for displaying for client machines for the B2C or are sent to other information systems (clients or servers) for the B2B.

A B2B environment focuses on interchanging data among enterprises instead of communications between distributed objects. These enterprises may implement their systems with various kinds of applications, databases, and platforms. An interaction between two distinguished systems based on HTTP can be accomplished using XML documents as distributed objects for interfacing such systems. As the rendering and displaying of an XML document for client machines request no additional software package, universal client devices can access XML-based Web information systems. Simple Object Access Protocol [18] is an example of the protocol based on XML-enabled Web servers that supports EDI using only HTTP and XML technology.

This paper provides guidelines for implementing the business component model following the three-tier Web-based information system architecture based on HTTP and XML infrastructure.

3. Basic definitions

3.1. Business objects

A business object (BO) is an object with well-defined boundaries and an identity that encapsulates a business state and behavior. A business state is a structural property represented by attributes or instance variables while a behavior is a behavioral property represented by methods that operate on the attributes. BOS represent business resources in a business domain such as employee, product, payment, information, process, etc. More details on the formal definition of this notion by the Object Management Group can be seen in [19].

Let BO be a business object composed of \( m \) attributes, \( A = \{a_1, a_2, \ldots, a_m\} \), and \( n \) methods, \( M = \{M_1, M_2, \ldots, M_n\} \). In terms of overall behaviors, a BO can be defined as:

\[
BO \equiv \{M_1(A_1), M_2(A_2), \ldots, M_n(A_n)\}
\]

where \( A_i \subseteq A \), for all \( i = 1, 2, \ldots, n \). \( M_i \) operates on a set of attributes, \( A_i \). Note that \( A_i \) and \( A_j \) may contain common elements but are processed by different methods, \( M_i \) and \( M_j \) respectively.

A number of business objects participate cooperatively in a particular business process. This cooperation can be represented by sharing interactions between the associated business objects [20]. More specifically, each business object of a particular business object pair of an interaction must provide a public method for interfacing. Such interactions determine business object dependencies.

3.2. Business components

Traditionally, a software component is defined as “a self-contained piece of software with well-defined interface or set of interfaces [21]”.

A larger-grained component called a business component (BC), as defined by Herzum and Sims [22], focuses on a business concept as “the software
implementation of an autonomous business concept or business process. It consists of all software artifacts necessary to represent, implement, and deploy a given business concept as autonomous, reusable element of a larger distributed information system”.

No matter how a component is named (e.g. component, distributed component, or business component), it has the following basic characteristics [21]:

- A component is a self-contained software construct: a self-contained piece of software unit is one that can be independently (autonomously) delivered, installed, and deployed.
- A component socket (plug-out for other components to plug-in) provides a well-defined and well-known run-time interface. This means that it can be easily combined with other components to provide useful functionality.
- A component is built for composition and collaboration with other components.

Component-based development approaches such as CBSD [22] and Business Object Component Architecture (BOCA) [23] provide mechanisms based on the above properties and support component composition at multiple layers of abstractions.

Our approach focuses on identifying business activities instead of business objects. The associated atomic components with respective activities are then composed to build a larger component, called business component. Thus, the approach supports the reusability and component composition in several levels of granularity, from fine-grained components (atomic-level component) that represent business activity through large-grained components (business components) that represent business processes to coarse-grained components that represent domain subsystems.

4. Business process-based methodology for Web-based information system development

The two main tasks of the methodology are business component modeling (Section 4.1) and Web implementation modeling (Section 4.2). The first task describes methods and techniques in identifying and generating business components. The result of this task is a business component model, which is used to represent business components.

4.1. Business component modeling

The key of the BPBM approach is the use of a business process as a unified conceptual framework for analyzing relationships between a business process and related BOs, for identifying business activities and designing business components.

4.1.1. Analysis process

A business system can be decomposed into a set of business processes and each business process can be also decomposed into business activities resulting in a two-level hierarchy of the business processes model. A business process is performed by participation/ cooperation of a number of business resources called business objects. These business objects can be either activator or business state. Activators represent actors who initiate a business process when a business process event occurs. Business states (business data or data objects) are created and/or used by a business process while performing the process. These business objects are modeled as conceptual business objects [24], using models such as the Entity–Relationship (E–R) model or the Object Definition Language [25].

There are several components of an enterprise model: a business goal, business resources (business objects), business rules, and business processes. Among these components, business processes are the active part of the enterprise [26]. Business objects are input objects transformed or consumed as part of the business process governed by business rules to accomplish the business process goal in the form of output objects (also business objects). Business objects operated by the business process are actually processed by business activities of that business process. Each business activity requires business objects as input data and produces business objects as outputs and, therefore, it can be represented as a set of interactions between business objects.

Let us represent such interactions using the E–R model specifying graphical notations: a rectangle symbol represents a BO, an ellipse symbol represents a business process (or a business activity), and a directed arrow labeled with a kind of operations represents an operation that the business object operates on its business state. For example, the Request Order business process shown below requires two business objects, Branch and Material, to operate
on their business states with \( R \) (read or retrieve) operations called \( R-R \) interaction.

At the low-level of abstraction, a business process is decomposed into business activities and all interactions between BOs for each business activity are identified and labeled with explicit kinds of operations. For example, a business process, \( BP_1 \), is decomposed into four business activities \( BA_i (i = 1, 2, \ldots, 4) \), involving five \( BO_i (i = 1, 2, \ldots, 5) \), in different ways (Fig. 1 with omitted operation labels).

A business activity that involves BOs operating on a business state with the manipulation operation is the main business activity of a particular business process. These BOs, therefore, are represented as the core BOs of the business process. Using the example in Fig. 1, \( BA_1 \) and \( BA_2 \) can be defined as follows:

\[
BA_1 \equiv \{ I(BO_1, BO_4) \} \quad \text{and} \quad BA_2 \equiv \{ I(BO_4, BO_5) \}
\]

where \( I(BO_i, BO_j) \) is an interaction between \( BO_i \) and \( BO_j \) and each BO provides a respective public method for interacting. All computations on business states of each BO are performed by its own local methods.

In the example, \( BA_1 \) and \( BA_2 \) interact by involving a shared common \( BO_4 \). Note, however, that \( BA_1 \) and \( BA_2 \) may involve the \( BO_4 \) with different methods, \( M_{BO_4}^+ \) and \( M_{BO_4}^\sim \), respectively. Thus, \( BA_1 \equiv \{ I(M_{BO_1}, M_{BO_4}) \} \) and \( BA_2 \equiv \{ I(M_{BO_4}, M_{BO_5}) \} \). In fact, a business process may require several interactions and there may be several business processes involving a common BO. This BO is one way to represent interactions between business processes, including business activities.

### 4.1.2. Design processes

The design process has the task to identify and construct business components and their atomic components (business activities) as independent reusable and composable software units. More specifically, BOs belonging to each atomic component, implementing respective business activity should be identified and represented as working elements.

To be considered as a component, the functionality of the respective business activity should be in correspondence with requirements for high cohesion within the component and low coupling with the rest of the system.
components of the system [27]. The dependencies between BOs for a particular business activity are the key to identification of business components.

4.1.2.1. Business activity identification. At high-level design, conceptual dependencies between BOs due to manipulation operations can be determined using static analysis. We introduce the concept of a degree of an operation, which can be classified as high-level (for Creation (C) and Deletion (D) operations), and low-level (for Update (U) and Retrieve (R) operations). C and D are operations at instance level while U is an operation at attribute level and R is a read-only operation. Given three kinds of operations, there are six possible interactions patterns: C–C, C–U or U–C, C–R or R–C, U–U, U–R or R–U and R–R. These six patterns of interactions can be used for analysis of the degree of coupling or degree of interactions between BOs. We use high coupling (due to C and/or D operations) and low coupling (due to U and/or R operations) degree of interactions, respectively.

Let a business activity, BA, involve \( n \) BOs \{BO\(_1\), BO\(_2\), \ldots, BO\(_n\)\}. For any interaction of a business objects pair (BO\(_i\), BO\(_j\)), if one or both BOs operate on their respective business states with a high-level operation, they should be selected to belong to the BA (see Table 1).

In fact, one BO can interact with several other BOs with various degrees of interaction. More specifically, the distinct interaction patterns of the same set of associated BOs may cause different business activities. Thus, a BO can belong to several business activities. Moreover, every BO must have at least one C operation for manipulating its own business state and also the operation may cause at least one interaction. For example, in Case 4, the both BOs, BO\(_1\) and BO\(_2\), should belong to other business activities which require these BOs to operate on their business states with C operations.

The other criterion for identifying business activities is the degree of cohesion measured in terms of a proportion between degrees of internal interactions and external interactions. An internal interaction is an interaction between two BOs in the same business activity. An external interaction is an interaction where a BO in one business activity interacts with a BO of another business activity. The degree of each interaction is evaluated using the respective degree of operations, which operate on their business states.

As mentioned previously, a business activity should provide high cohesion and low coupling in order to be considered as an acceptable component.

4.1.2.2. Business component identification. All identified business activities, determined by which BOs belong to them, are composed into business components using coupling and cohesion measurements. The technique is to combine all business activities with high coupling of interactions into a single business component with the objective to reduce coupling between identified business components.

An interaction between business activities is actually represented with an interaction between BOs, which belong to different business activities. As for business activity identification, we classify degree of interactions between business activities into high-level (for C and D operations) and low-level (for U and R operations). Suppose that two business activities BA\(_1\) and BA\(_2\) interact through BO(s) with various kinds of operations. The two business activities should be accepted to be members of a considered business

<table>
<thead>
<tr>
<th>Coupling between BO(_1) and BO(_2) of BA</th>
<th>BO(_1) C/D</th>
<th>U/R</th>
<th>BO(_2) C/D</th>
<th>U/R</th>
<th>Set of BOs belonging to BA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 1</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td>[BO(_1), BO(_2)]</td>
</tr>
<tr>
<td>Case 2</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>[BO(_1)]</td>
</tr>
<tr>
<td>Case 3</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td>[BO(_2)]</td>
</tr>
<tr>
<td>Case 4</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td>{}</td>
</tr>
</tbody>
</table>
component, if they satisfy the coupling criterion as shown in Table 2.

The classification of degrees of coupling of interactions between business activities is described as follows:

- $\text{C} \rightarrow \text{C}$ interaction: The interaction between two business activities forces the BO of each business activity to operate on their own BO with $C$ operations. Since these two business activities have high degrees of interaction, they should be selected to be members of the same BC (Case 1).

- $\text{C} \rightarrow \text{U}$ and $\text{C} \rightarrow \text{R}$ interactions: The business activity that requires its BO to operate on its business state with $C$ operation is selected to be a member of the BC (Cases 2 and 3).

- Case 4 shows that $\text{BA}_1$ and $\text{BA}_2$ have no common BO. Both business activities show no interaction. Thus, they may belong to other different business components.

In case a particular business activity does not have any BO belonging to it, such business activity is called an independent business activity, and can be a member of any business component. However, every business activity must belong to one and only one business component due to the effectiveness of deployment. The coupling criterion for such business activity cannot be applied for identifying which business component an independent business activity should belong to.

A solution is to determine the core BO among BOs involved by an independent business activity using cohesion measurement. For example, the core BO of a particular business activity, including an independent business activity, is identified using the maximum proportion of operated attributes comparing all attributes of all BOs involving in the business activity. The proportion is measured in terms of attributes operated by business object methods, which have no interaction with other BOs. Since every BO should belong to at least one business activity including the main business activity, the independent business activity should be selected as a member of the same business component as the main business activity and has responsibility to provide services for other business components.

Since business components are components that can be composed into a larger-grained component, they should also be reusable and composable. Therefore, the properties of an ideal business component are low coupling and high cohesion.

### 4.1.3. Business component model

The primary elements of a business component (BC) are business activities that specify required BOs and structural linkages between them. These business activities should be processed correctly in a well-defined sequence. Thus, every business component should have one controller called business process controller (BPC) to control and manage the collaboration of its business activities.

Moreover, a business component may provide services for other components so that there are additional component elements as formulated below:

$$BC = \{\{\text{BO}_1, \text{BO}_2, \ldots, \text{BO}_i\}, \{\text{BA}_1, \text{BA}_2, \ldots, \text{BA}_j\}, \text{BPC}, \{\text{PI}_1, \text{PI}_2, \ldots, \text{PI}_k\}\}$$

where

- $\{\text{BO}_1, \text{BO}_2, \ldots, \text{BO}_i\}$ is the set of business objects.

As mentioned in Section 3.1, a BO can be involved in several business activities and $\{\text{BO}_1, \text{BO}_2, \ldots, \text{BO}_i\}$ is represented separately from $\{\text{BA}_1, \text{BA}_2, \ldots, \text{BA}_j\}$.

<table>
<thead>
<tr>
<th>Interaction of $\text{BA}_1$ and $\text{BA}_2$ through BO</th>
<th>$\text{BA}_1$</th>
<th>$\text{BA}_2$</th>
<th>Set of business activities belonging to BC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\text{C/D}$</td>
<td>$\text{U/R}$</td>
<td>$\text{C/D}$</td>
</tr>
<tr>
<td>Case 1</td>
<td>$\text{x}$</td>
<td>$\text{x}$</td>
<td>${\text{BA}_1, \text{BA}_2}$</td>
</tr>
<tr>
<td>Case 2</td>
<td>$\text{x}$</td>
<td></td>
<td>${\text{BA}_1}$</td>
</tr>
<tr>
<td>Case 3</td>
<td></td>
<td>$\text{x}$</td>
<td>${\text{BA}_2}$</td>
</tr>
<tr>
<td>Case 4</td>
<td>$\text{x}$</td>
<td></td>
<td>${\text{}}$</td>
</tr>
</tbody>
</table>
• \( \{ \text{BA}_1, \text{BA}_2, \ldots, \text{BA}_j \} \) is the set of business activities that is represented in terms of interactions between business objects as:
\[
\text{BA} \equiv \{ I_1(\text{BO}_1, \text{BO}_2), I_2(\text{BO}_3, \text{BO}_4), \ldots, I_n(\text{BO}_p, \text{BO}_q) \}
\]
where \( I_k(\text{BO}_i, \text{BO}_j) \) is a given interaction that will be implemented as an invocation between public methods of the \( \text{BO}_i \) and \( \text{BO}_j \). In addition, there are several local methods of each business object involved in this interaction, but they are encapsulated within the respective business object.

There are two ways to initiate an interaction: by a business process event, or by a business object event. A business process event is stimulated usually by the end user through the user interface while a business object event is normally activated automatically by function call of a BO and is always invisible for the end user.

• BPC is a business process controller for collaborating business activities. A particular business component consists of a set of main business activities and a set of independent business activities. These main business activities have to be controlled by the BPC for collaborating in the required sequences. The sequences can be initialized in a business activity activated with a business process event and the business activity activates other business activities to be executed until the business process is completed/terminated.

• \( \{\text{PI}_1, \text{PI}_2, \ldots, \text{PI}_k\} \) is the set of public interfaces that might contain interfaces of two kinds of services, business activities and selected public business object methods represented as:
\[
\text{PI} \equiv \{[\text{BA}_1, \text{BA}_2, \ldots, \text{BA}_j], [\text{M}_1, \text{M}_2, \ldots, \text{M}_k]\}
\]

Another consideration of modeling is the architecture. Layering the architecture is a design technique used to classify business component functions. We classify our business components, using service-based measurement [28], into two layers: the common business component (CBC), which provides general purposes services and the application business component (ABC) that performs more specific tasks.

The service-based measurement is measured in terms of export services that a business component provides for other business components. Export services are represented using external interactions with ordered pair interactions, \( R-C \), \( R-U \), \( R-R \), and \( U-U \). For example, \( R-C \) interaction provides a service that forces a BO within the business component to operate on a business state with \( R \) operation while another BO of the request service operates on its business state with \( C \) operation.

The business component model can be represented using an XML-based description in order to be applied efficiently to Web computing environments. The respective XML document type definition (XML DTD) is shown in Fig. 2.

4.2.Web implementation modeling using the business component model

Web implementation modeling is guidance for implementing the business component model on a specific B2B Web-based information system infrastructure.

The XML-based business component model representation is a meta data model offering information about software component implementation. The implementation of primary component elements of the model on the Web follows the model described below.

• Each business component has only one main Web page (\( M-WP \)), containing a set of sub-Web pages (\( sub-WPs \)) which represent the main business activities of the business component. In addition to the CBC, the independent Web pages (\( I-WPs \)), which implement the independent business activities can be constructed and deployed independently of the \( M-WP \) and are to be linked to other Web pages (see Fig. 3).

• Recall that a business activity may be implemented as a sub-WP of the \( M-WP \) or an \( I-WP \). However, every business activity can be represented with a set of interactions involving one or more BOs. Such kinds of interactions are implemented with different component elements of Web information systems. Suppose that,
\[
\text{BA}_1 \equiv \{I(\text{BO}_1, \text{BO}_2)\}
\]
\[
\equiv \{I(\text{M}_{\text{BO}_1}(\text{BDU}_i), \text{M}_{\text{BO}_2}(\text{BDU}_j))\}
\]
and
\[
\text{BA}_2 \equiv \{I(\text{BO}_3)\} \equiv \{I(\text{M}_{\text{BO}_3}(\text{BDU}_k))\}
\]
These two business activities can be implemented by Web information system components that support the different tasks shown in Fig. 4.

In the example, BA1 requires the interaction, which involves BO1 and BO2. Since BO1 and BO2 operating with their respective BDUs enable the creation of their own Web pages, the interaction between BO1 and BO2 within the BA1 represents an interaction between business activities. However, an interaction between two BOs may have no user interfaces (Web pages).

In the following, we explain the main tags:

- The `<BusinessObject>` tag represents BO and is used to describe the structure (using the `<Structure>` tag) and behavior (using the `<Behavior>` tag) of BO. The structural description is eventually implemented as persistent object and stored on data source-tier while the behavioral description (business logic) is implemented as business object methods and resided on middle-tier (Web application server-tier).
From a user’s perspective, a business object should be visible to an end user at client machines through BO’s views for both business state presentation and manipulation. A method and a set of attributes operated by the method of a particular BO allow straightforward creation of the view. A complex view may be composed of several sub-views, which are constructed by different methods.

- The `<BusinessDataUnit>` tag (BDU) describes a virtual data unit that is accessed by BO methods. Each BDU is a primary information source in designing view contents (Web page contents). A BDU can be accessed by different methods for different objectives, so that it enables creation of several different Web pages. Such Web pages can be implemented using a single XML schema but may differ in XSL stylesheets and respective business object methods.
- The `<BusinessActivity>` tag represents a business activity in terms of internal and external interactions between BOs using `<Interaction>` tag. An internal interaction represents an interaction between BOs,

\[
BA_1 = \{ I(M_{BO_1}(BDU_1)), M_{BO_2}(BDU_2) \}
\]

\[
BA_2 = \{ I(M_{BO_3}(BDU_3)) \}
\]

Fig. 3. Implementations of application business component (ABC) and common business component (CBC).

Fig. 4. Implementations of two kinds of interactions within a business activity.
which belong to the same business activity through method attributes of Method1 and/or Method2 elements. Method1 and/or Method2 of external interaction are methods from two BOs that belong to different business activities. An <Interaction> can be viewed as Web page navigation and communication between business objects.

A method that operates on one BDU enables creation of one Web page (or a part of a Web page). An interaction between different methods forces implementation of a link from one Web page (due to the method defined in <Method1> tag) to another (due to the method defined in <Method2> tag) as a simple scenario following. Whenever a method is invoked by a business process event from one Web page, the dynamic content is generated at the Web server and downloaded for display at a client machine as a new Web page. Moreover, if the new Web page has hyperlinks to other Web pages due to the BO interaction, the contents of those new Web pages are constructed and presented. More generally speaking, a BDU defined using <BusinessDataUnit> tag specified by Method2 can be implemented as an XML document. This XML document acts as a distributed object for communication between related BOs or as a middle-tier for interface between client-tier and data source-tier.

With respect to support EDI, the communication between tiers of BOs can be implemented using HTTP and XML-based infrastructure as follows. A client sends a request to a Web server through HTTP in form of, for example, HTML/XML format and then Web server interprets the request to perform an appropriate function (e.g. calling a class method), which is executed at the server-side. In database-intensive Web information systems, those server-side class methods always access a database for retrieving data. The result of this execution is XML data combined with an XSL stylesheet or HTML formatted document. In turn, a client can send XML-based data to the Web server for updating the BO in the data source.

- Since the main Web page is composed of the implementation of the main business activities, its contents contain attribute values of the BDUs operated by methods defined in <Interaction> tags of these business activities. Other elements of the main Web page are hyperlinks to sub-Web pages depending on the number of interactions of respective business activities. The main Web page can be implemented, for example, as Microsoft’s ActiveServer Page or Sun’s JavaServer Page, with smaller components such as JavaBeans.
- <Interface> tag represents the public interfaces. An <Interface> description can be implemented as a software declaration unit containing business activities and/or methods descriptions. These descriptions provide an interface for a client object to communicate with server objects based on distributed object infrastructure.

5. A case study: inventory subsystem of an industrial information system

5.1. Description

Let us assume that an industrial company has several company branches located at different areas controlled by the central office. Requested raw materials for production of each product must not exceed pre-evaluated quantities. Whenever raw materials are requested with a specified usage date, each branch submits an electronic request order to the central office for approval. The request order is used to create purchase orders, which are sent, respectively, to appropriate suppliers. Requested raw materials are delivered directly by suppliers to branches in the specified time. The (supplier) invoice is finally passed through the purchase department in the central office for registration and confirmation.

For the Inventory subsystem, the request order, the purchase order, and the invoice management (supplier invoice) processes work together as a stock-in process. The stock-out process occurs when a particular branch submits a usage order for these raw materials.

Let us assume that the system to be developed is an online computer-based system. Once a branch submits a request order, it can be seen immediately by the purchase department. In turn, the branch can monitor the progress in order to follow up the request.

There are a number of information subsystems in a particular industrial system, for example, Human Resources, Inventory, Finance, Account subsystems, etc. The case study uses only the Inventory subsystem developed for illustration.
5.2. Business process and business activity analysis

The Inventory subsystem is divided into the following business processes: Request Order, Purchase Order, Invoice Management, and Usage Order. Each business process is then decomposed into business activities. In the meantime, all BOs involved in each BP, including business activity are determined as RequestOrder, RequestOrderLine, and NumberGeneration. These BOs act as input and/or output of business activities in terms of interactions between BOs illustrated as follows:

- **Select Branch**: This business activity requires the Branch to retrieve its instance and forces the RequestOrder to update the object identifier (OID) of the Branch instance as a reference.

- **Request Material Entry**: This business activity requires several interactions among BOs involved in sub-activities to accomplish its task. Create sub-activity requires RequestOrder to create an instance and then forces the RequestOrderLine to create new

![Diagram](image)

Table 3

<table>
<thead>
<tr>
<th>Business activity</th>
<th>Request order</th>
<th>Purchase order</th>
<th>Usage order</th>
<th>Invoice</th>
<th>Warehouse</th>
<th>Number generation</th>
<th>Supplier</th>
<th>Branch</th>
<th>Material</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request order business process</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Select branch</td>
<td>Ref</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>Request material entry</td>
<td>C/U/D/Ref</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>Generate request number</td>
<td>R</td>
<td></td>
<td></td>
<td>U</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Print request order</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Purchase order business process</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Select supplier</td>
<td>Ref</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>Purchase material entry</td>
<td>U</td>
<td>C/U/D/Ref</td>
<td></td>
<td></td>
<td>C/D</td>
<td>U</td>
<td></td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>Generate purchase number</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td>U</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>print purchase order</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>Invoice management business process</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Select supplier</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>C/U/D/Ref</td>
<td>U</td>
</tr>
<tr>
<td>Invoice entry</td>
<td>R</td>
<td>Ref</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>U</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>Generate invoice number</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td>U</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Print invoice</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>Usage order business process</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Select branch</td>
<td>Ref</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>Usage material entry</td>
<td>C/U/D/Ref</td>
<td></td>
<td></td>
<td></td>
<td>U</td>
<td></td>
<td></td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>Generate usage number</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td>U</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Print usage order</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>R</td>
<td></td>
</tr>
</tbody>
</table>

R: Read/Retrieve; Ref: Reference; C: Create; U: Update; D: Delete.

[u] The conceptual single BO actually contains two BOs, e.g. RequestOrder: {RequestOrder, RequestOrderLine}.
instance(s). Whenever the RequestOrderLine creates/deletes an instance or update attributes, the Update sub-activity requires the RequestOrder update the related instance automatically. Another sub-activity is Get Material. Since each instance of RequestOrderLine refers to an instance of Material, Get Material provides an OID of its instance as an input and update into RequestOrderLine as a reference.

The business object event from RequestOrderLine enables the Update subactivity to update RequestOrder automatically.

- **Generate Request Order Number**: The business activity requests a service for generating a Request Order Number from the NumberGeneration. This number is updated within the RequestOrder as a reference (Ref) and then the NumberGeneration increases the running number and updates (U) itself.

- Since the Supplier, Branch, NumberGeneration, and Material have the main responsibility for manipulating their own business state (omitted in the table) and provide some services (R-operation) for other business activities, each of these BOs can be identified independently to be a common business component.

- **Print Request Order**: The core BO is the RequestOrder. The business activity consists of sub-activities. Getting an instance of RequestOrder (R) requires retrieving other BOs, retrieving (R) the instance of the reference BO—Branch and a set of related instances of RequestOrderLine. Retrieving an instance of RequestOrderLine, forces to retrieve (R) automatically the instance of the reference BO—Material.

- The independent business activity should be a member of the business component as the main business activity. For example, Print Request Order should be a member of the same business component as the Request Material Entry.

The business component model and layered architecture for this particular case are illustrated in Fig. 5.

The other business processes are analyzed in the same manner. All interactions are shown in Table 3.

5.3. Business component design

Using Table 3, business activities, including business components of the Inventory subsystem can be identified and modeled as described and illustrated below:
5.4. Assembling components for Web-based environment

The Branch_BC model represented in Appendix A is composed of two business activities. Only business activity ManipulateBranch_BA is implemented as a main Web page (Fig. 6(a)). The contents of the main Web page are attribute’s values of the Branch_BO and hyperlinks to other sub-Web pages (Fig. 6(c)) according to interactions defined using <Interaction> tag. The link to the Web page for adding (activated by pressing “Add” button by end user) and modifying (activated by pressing “Modify” button) illustrated in Fig. 6(c) is from the main Web page.
Another business activity is the SelectBranch_BA. Since the business activity is for servicing other business components, it is implemented as an independent Web page (Fig. 6(b)) without linking with the main Web page, but other Web pages of other business components can be linked to it.

Processing of the SelectBranch_BA requires the GetBranch_M to retrieve SelectBranch_BDU (two attributes) from Branch_BO at data source-tier and to be transformed into an XML document at server-side, which acts as a distributed object between tiers. This XML document will be sent back to the requested client either to be processed in the EDI manner or to be rendered and presented for end user at client side.

Appendix B show the business component Request_BC which is composed of four business activities and identified with <BusinessActivity> tags has responsibility to do a specific task without providing any service for other business components. Thus, all business activities are implemented as and controlled by the main Web page (RequestOrder Page), which implements the BPC of the business component. The primary responsibility of the main Web page is to control the correct sequences of the processing of business activities by enable or disable buttons according to various conditions.

The main Web page may link to a number of sub-Web pages depending on a number of both internal and external interactions of each business activity. For example, the RequestEntry_BA consists of six internal interactions, CreateRequestOrder_I, UpdateRequestOrder_I, DeleteRequestOrder_I, CreateRequestLine_I, UpdateRequestLine_I, and one external interaction, GetMaterialOfOrderLine_I. Some interactions are implemented and included within the main Web page but some are constructed independently and linked by the main Web page. Fig. 7 shows an external interaction between RequestOrder_BO and Branch_BO, which enables navigation from the RequestOrder Page to the SelectBranch Page in view of end users.

However, some interactions are implemented as invisible Web pages for end users, for example, the external interaction between two methods, GetRequestNumber_M of the Request_BO and GenRequestNumber_M of the NumberGeneration_BO.

6. Conclusion

This paper proposes a methodology for identifying and use of business components in software development, based on process-oriented and object-oriented
A suitable business component model is introduced to support the software development process, including measuring the degree of cohesion and coupling between the components. XML with its metadata description capabilities is used for representation of and working with the model. Such a representation allows convenient and functionally flexible adaptation to Web-based computing environment for developing distributed applications.

The basic idea of the methodology is that the components relate primarily to business processes and activities instead to objects as constituent parts. This is more natural approach to analysis of complex business domains and systems. It also allows for building information systems, which are more flexible and easily adaptable to frequently changing functional requirements.

Applicability of the methodology is illustrated on a real example of developing a subsystem of an industrial information system on the Web.

Work to improve the methods and techniques, regarding, for example, software quality and complexity measurement within the proposed methodology continues. One direction for additional research is to seek for more formal and accurate techniques for measuring the degree of coupling and cohesion, which is the basis for identifying business components. This would substantially help the automatic identification and generation of components. In accordance with the enormous efforts toward transition to a new generation Web, we believe that the proposed XML-based model description is a good basis for extension to not only structural but also semantic representation and processing of business components.
Appendix A

A common business component—Branch_BC representation

```xml
<BusinessComponent name="Branch_BC">
  <BusinessObject id="Branch_BO">
    <Structure>
      <Attribute id="Identifier" type="String" />
      <Attribute id="Name" type="String" />
      <Attribute id="Address" type="String" />
    </Structure>
    <BusinessDataUnit id="SelectBranch_BDU" BO_id="Branch_BO">
      <Attribute id="Branch_BO.Identifier" />
      <Attribute id="Branch_BO.Name" />
    </BusinessDataUnit>
    <BusinessDataUnit id="Branch_BDU" BO_id="RequestOrder_BO">
      <Attribute id="ALL" />
    </BusinessDataUnit>
    <!--include all attributes of the Branch_BO -->
    <Behavior>
      <Method name="GetBranch_M" input="SelectBranch_BDU" output="Identifier" />
      <Method name="CreateNewBranch_M" input="Branch_BDU" output="Branch_BDU" />
      <Method name="UpdateBranch_M" input="Branch_BDU" output="Branch_BDU" />
      <Method name="DeleteBranch_M" input="Branch_BDU" output="Branch_BDU" />
    </Behavior>
  </BusinessObject>
  <BusinessObject id="SelectBranch_BA">
    <BusinessObjectList>
      <BusinessObjectRef BC_name="Branch_BC" BO_id="Branch_BO" />
    </BusinessObjectList>
    <Behavior>
      <Method id="GetBranch_F" name="Branch_BO.GetBranch_M" />
    </Behavior>
  </BusinessActivity id="ManipulateBranch_BA">
    <BusinessObjectList>
      <BusinessObjectRef BC_name="Branch_BC" BO_id="Branch_BO" />
    </BusinessObjectList>
    <Behavior>
      <Method id="ConstructInstance_F" name="Branch_BO.CreateNewBranch_M" />
      <Method id="ModifyInstance_F" name="Branch_BO.UpdateBranch_M" />
      <Method id="DestructInstance_F" name="Branch_BO.DeleteBranch_M" />
    </Behavior>
  </BusinessActivity id="SelectBranch_BA">
  <Interface>
    <BusinessActivityList>
      <BusinessActivityId name="SelectBranch_BA" />
    </BusinessActivityList>
    <PublicMethod>
      <MethodName name="GetBranch_M" />
    </PublicMethod>
  </Interface>
</BusinessComponent>
```
Appendix B

Application business component—parts of Request_BC representation

```xml
<BusinessComponent name="Request_BC">
  ...
  <BusinessActivity id="BranchReference_BA">
    <BusinessObjectList>
      <BusinessObject BC_name="Branch_BC" BO_id="Branch_BO" />
      <BusinessObject BC_name="RequestOrder_BC" BO_id="RequestOrder_BO" />
    </BusinessObjectList>
    <Interaction name="BranchReferenceEntry_1">
      <Method1 name="RequestOrder_BO.GetBranch_M" />
      <Method2 name="Branch_BO.GetBranch_M" />
    </Interaction>
  </BusinessActivity>
  <BusinessActivity id="RequestNumber_BA">
    <BusinessObjectList>
      <BusinessObject BC_name="NumberGeneration_BC" BO_id="NumberGeneration_BO" />
      <BusinessObject BC_name="RequestOrder_BC" BO_id="RequestOrder_BO" />
    </BusinessObjectList>
  </BusinessActivity>
  <BusinessActivity id="RequestEntry_BA">
    <BusinessObjectList>
      <BusinessObject BC_name="Material_BC" BO_id="Material_BO" />
      <BusinessObject BC_name="RequestOrder_BC" BO_id="RequestOrder_BO" />
      <BusinessObject BC_name="RequestOrderLine_BC" BO_id="RequestOrderLine_BO" />
    </BusinessObjectList>
    <Interaction name="CreateRequestOrder_1">
      <Method1 name="RequestOrder_BO.CreateRequestOrder_M" />
    </Interaction>
    <Interaction name="UpdateRequestOrder_1">
      <Method1 name="RequestOrder_BO.UpdateRequestOrder_M" />
    </Interaction>
    <Interaction name="DeleteRequestOrder_1">
      <Method1 name="RequestOrder_BO.DeleteRequestOrder_M" />
    </Interaction>
    <Interaction name="CreateRequestLine_1">
      <Method1 name="RequestOrder_BO.CreateRequestEntry_M" />
      <Method2 name="RequestOrderLine_BO.CreateRequestLine_M" />
    </Interaction>
    <Interaction name="UpdateRequestLine_1">
      <Method1 name="RequestOrderLine_BO.UpdateRequestLine_M" />
    </Interaction>
    <Interaction name="DeleteRequestLine_1">
      <Method1 name="RequestOrderLine_BO.DeleteRequestLine_M" />
    </Interaction>
    <Interaction name="GetMaterialOfOrderLine_1">
      <Method1 name="RequestOrderLine_BO.CreateRequestLine_M" />
      <Method2 name="Material_BO.GetMaterial_M" />
    </Interaction>
  </BusinessActivity>
  <BusinessActivity id="PrintRequestOrder_BA">
    ...
  </BusinessActivity>
</BusinessComponent>
```
References


Somjit Arch-int received the BSc degree in Statistics from Khon Kaen University and the MSc computer science from the National Institute of Development Administration, Thailand in 1987 and 1990, respectively. He is currently a doctoral student in computer science at the Asian Institute of Technology. His previous experiences include several industry systems development and consulting activities. His research interests are business component-based software development, object-oriented metrics, ontology-based e-business modeling, knowledge-based representation, and semantic Web. He is a member of IEEE Computer Society.

Dentcho N. Batanov received the MSc and PhD degrees from the Technical University, Sofia, Bulgaria, in 1970 and 1975, respectively. He is currently an associate professor in the Program of Computer Science and Information Management at the Asian Institute of Technology, Thailand, where he was a coordinator of the program from 1996 to 1999. His research interests are application of information technology systems development and consulting activities.
technologies in education, management, industrial engineering and manufacturing, knowledge-based and expert systems, object-oriented software engineering, distributed systems and technologies, component and framework-based software development, and e-business. He has published more than 100 papers in journals and conference proceedings, and numerous textbooks and manuals. He is currently a member of international board of several international journals and conferences. He is a member of the IEEE, and the ACM.