IDEA GROUP PUBLISHING

701 E. Chocolate Avenue, Hershey PA 17033-1117, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

ITP4112

Business Objects and Components for Web-Based Information Systems Development

Dentcho N. Batanov and Somjit Arch-int
Computer Science and Information Management Program, School of Advanced Technologies, Asian Institute of Technology
P.O. Box 4 Klong Luang, Pathum Thani, 12120, Thailand, batanov@cs.ait.ac.th, somjit@kku.ac.th

ABSTRACT

Global competition among today's enterprises forces their business processes to evolve constantly, leading to changes in corresponding Web-based information systems. Most existing approaches that extend the traditional software engineering to develop Web-based information systems are based on object-oriented methods. Such methods emphasize on modeling individual object behaviors instead of system behavior. This paper proposes the Business Process-Based Methodology (BPBM) for developing such systems by using a business process as a unified conceptual framework for analyzing relationships between a business process and associated business objects, for identifying business activities and designing object-oriented components called business components. These business components can represent more clearly semantic system behaviors than linkages of individual object behaviors. A change made to one business process impacts some encapsulated atomic components within the respective business component without affecting other parts of the system. A business component is divided into parts suitable for implementation of multi-tier Web-based information systems.

INTRODUCTION

The increasing competition caused by world wide businesses forces the enterprise's strategies to evolve frequently. Whenever a strategy has been changed, the associated business processes must be also re-modeled which in turn requires that the corresponding Web-based information systems also be re-implemented and installed quickly.

Most existing approaches that extend the traditional software engineering to develop Web-based information systems are based on object-oriented methods. Although the object orientation provides powerful mechanisms such as encapsulation, inheritance, and reusability [4, 13], objects used as building blocks in early phase of the development result in individual object behaviors instead of system behavior. Further, since Web-based information systems emphasize both business logic and presentation, the traditional software implementation model does not fit to the Web implementation model [9]. Therefore, such existing approaches (e.g., [6, 10]) need further enhancement in order to master dynamic and sophisticated systems.

The *BPBM* blends advantages of the structured [1] and object-oriented paradigms [4, 13, 14, 15] for identifying and designing business components based on the notion that a business process consists of a *function-oriented* part (activities) representing object behavior, and a *process-oriented* part acting as a collaboration of these activities [18]. The benefits of the proposed business component model are not only taking advantages of the structural approach, which itself meets naturally related functional requirements, but the model also conforms to the powerful Model-View-Controller (MVC) architecture in its implementation in a Web-based environment.

The remainder of this paper is organized as follows. Information systems on the Web are previewed in Section 2. Basic definitions are described in Section 3. The proposed approach for business components and Web implementation modeling are described in Section 4 and Section 5, respectively. Conclusions are outlined in Section 6.

INFORMATION SYSTEMS ON THE WEB

Growth of Internet technology has allowed modern enterprise to move business systems onto the Internet [16]. That environment requires electronic data interchange (EDI) between enterprises to be a prime feature for providing a variety of Internet applications, e.g. E-commerce (B2B or B2C), including Web-based information systems. Making the move to the EDI Web-environment raises the need to address two additional requirements, *heterogeneous data sources* and *universal client accessibility*:

1. Heterogeneous data sources used within one or across several distinct application platforms prevent interoperability. To gain the ability to provide data exchanges between enterprises, including decoupling the accessing of data from different tiers, such systems should have a *mediator* as a handling interface.
2. Clients should not be required to have high performance machines. Moreover, a client's presentation method should support a variety of output formats (e.g., HTML, PDF, etc.) and support various client devices (e.g., Personal Digital Assistants (PDAs), wireless phones, etc.).

An appropriate resolution for these challenges is the use of Extensible Markup Language (XML) based technology [19]. An XML is a "metalanguage" that provides metadata for describing other data in a document. An XML document uses semantics tags that are validated with their XML Document Type Definition (XML DTD) to describe the document's content, which enables communication across platforms. Based on such semantic tags, XML allows developers design their own customized markup languages for limitless different types of documents.

An XML document acts as a *middle-tier distributed object* to interface multi-tier Web information systems components through the HTTP protocol. An XML document produced by a variety of propriety database vendors can be distributed either to other distinct application platforms for EDI-enabled applications or to a requesting client. In addition to the second challenge, XML documents combined with a variety of XSL stylesheets [20] can also support various client requirements.

BASIC DEFINITIONS

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. More details on the formal definition of this notion by the Object Management Group can be found in [17].

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, *BO* can be defined as

$$BO\ ^{\circ}\{M_1(A_1), M_2(A_2), \ldots, M_n(A_n)\}$$

where $A_i \subseteq A$, for all $i=1, 2, \ldots, n$. M_i operates on a set of attributes A_i.

Business Components

A business component (BC) defined by Hersum et al. [11] is a software unit that implements a business concept. Such a business component is large-grained, which means 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. A BC is a kind of components, which has some basic characteristics e.g., self-contained software construct and well-defined and well-known run-time interface [5].

Our approach in developing business components focuses on identifying business activities instead on business objects as in, for example, *component-based software development (CBSD)* [11] and *Business Object Component Architecture (BOCA)* [8]. The associated atomic components with respective activities are then composed to build a larger component—business component.

BUSINESS COMPONENT MODELING

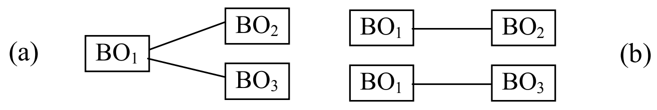
The key of the *BPBM* approach in modeling business components is the use of business process as a unified conceptual framework for analyzing relationships between a business process and associated BOs, for identifying business activities and designing business components. A business system is decomposed into a set of business processes and each business process is then decomposed into business activities resulting in *a two-leveled hierarchy of the business processes model*.

A business process is performed by participation/cooperation of a number of business resources called *business objects*, which 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. Each business activity of a particular business process requires business objects as generations of input and output data and, therefore, it can be represented as a set of interactions between business objects.

In high-level design, conceptual dependencies between BOs due to accessing operations can be determined using static analysis. We classify the degree of operations into three levels, *Creation(C) and Deletion(D)*—instance level, *Update(U)*—attribute level, and *Retrieve(R)*—read-only operation. Given three kinds of operations, there are six possible interaction patterns between business objects: *C-C*^{1}, *C-U* or *U-C*, *C-R* or *R-C*, *U-U*, *U-R* or *R-U* and *R-R*.

A business activity is composed of interactions among related business objects. More specifically, a business activity may require a business object to interact with one or more business objects. In Fig.4.1 for example, the BO_1 interacts with BO_2 and BO_3 in a particular sequence. The BO_1 may interact with both business objects by using a respective method. It is more convenient to separate such interactions into two different pairs of interactions, $I_1(BO_1, BO_2)$ and $I_2(BO_1, BO_3)$, as shown in Figure 1 (b).

Figure 1: Details of interactions of a business activity



For example, the *R-C* interaction pattern requires one business object to retrieve business state and another business object to create a new instance. In addition, such an interaction includes the case that a business object operates itself without interacting with others. This kind of interactions is occurred due to the business activity, which needs to manipulate business state of individual business objects

For example, let a business process, *BP*, is decomposed into four business activities BA_i (i=1,2,…,4), involving five BO_i (i=1,2,…,5), in different ways (Figure 2 with omitted explicit operation labels).

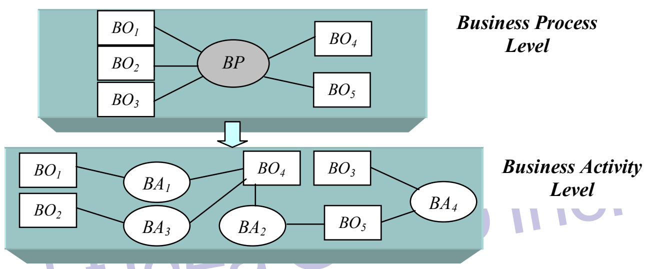
Using the example in Figure 2, BA_1 and BA_2 can be defined as follows:

$BA_1 \equiv \{ I(BO_1, BO_4) \}$ and $BA_2 \equiv \{ I(BO_4, BO_5) \}$

In general, a business activity consists of a set of interactions between business objects as formulated below:

$BA \equiv \{I_1(BO_1, BO_2), I_2(BO_3, BO_4), \ldots, I_n(BO_p, BO_q) \}$

Figure 2: Two-level interaction model of a business process



where $I_k(BO_i, BO_j)$ is a given interaction representing a public method invocation of business objects BO_i and BO_j.

A business activity that requires BOs to operate on business states with the *C* operation is the *main* business activity of a particular business process. These BOs, therefore, are represented as the *core* BOs of the business process.

Analysis Model Representation

To make the analysis model simple and precise in identifying business activities and business components using coupling and cohesion measures, the interactions between business objects are represented using an undirected graph [12, 21] called a *business object interaction graph* (BOIG) which is similar to the *input/output dependence graph* [2].

The six interaction patterns are classified into three groups depending on the direction of interaction:

1. *Two directions:* This group consists of *C-C*, *C-U*, and *U-U* interaction patterns and requires both of associated business objects to create instances or to update business state. Business objects involving in such interactions are masters.
2. *One direction:* This group consists of *R-C* or *R-U* interaction pattern. Such an interaction requires only one business object, which is the master while the other is the slave. The master business object, in this case, creates instances or updates business state.
3. *No direction:* This group consists of *R-R* interaction pattern. Such an interaction requires both of associated business objects to be slaves.

A business activity consists of interactions among associated business objects. The result of one interaction can be an input for other interactions. Such a result can be either a new instance or updated business state of business objects due to *C* or *U* operation respectively, or singly data due to *R* operation. Hence, an interaction between two business objects is represented as bi-directional regardless of its belonging to any direction group of interactions.

Definition 1: *BOIG*

Let *OP*={*C, U, R*} be the set of operations and *I*={*C-C, C-U, C-R, U-U, U-R, R-R*} the set of interaction patterns between a particular two business objects. BOIG of a business activity *BA* is an undirected graph $G_{BA}(V, E)$ where V is the set of the associated business objects, and E is the set of edges connected (interacted) between vertices (business objects) such that $E = \{<x, y> \in V x V \mid \exists i \in I$: (x and y interaction through *i* operation)$\}$.

The *loop* graph [21] is used to represent interactions that involve *only one* business object such that $E = \{<x> \in V \mid \$op \in OP$: (x operates on its business state with *op* operation)$\}$.

In Figure 2, for example, there are four business activities each one of which can be represented using BOIGs, for example, BA_1 and BA_2 as shown below:

- $G_{BA1}(V, E)$ represents the business activity BA_1 where $V = \{BO_1, BO_4\}$ and $E = \{<BO_1, BO_4>\}$.
- $G_{BA2}(V, E)$ represents the business activity BA_2 where $V = \{BO_4, BO_5\}$ and $E = \{<BO_4, BO_5>\}$.

Every business activity consists of at least one interaction. Further, a business activity should not be isolated so that it has at least one interaction with the outside world. Hence, |E| is greater than zero.

Business Activity Identification and Measurement

An associated business object must be identified explicitly by the business activities to which it should belong. An identified business activity is viewed as a modular unit with respect to an atomic software component. To be considered as a component, the functionality of a respective business activity should be in correspondence with requirements for *high cohesion* within the component and *low coupling* with the rest of components of the system [7].

Definition 2: *Coupling between Business Objects—Business Object Interaction (CBO-BOI) is the number of interactions between two BOs, which operate on their business state with various kinds of operations.*

Let a conceptual *BA* involves *n* business objects, $BO_{BA}=\{BO_1, BO_2, \ldots, BO_n\}$. For $BO_i \in BO_{BA}$, it is strongly recommended to *belong* to the *BA* if CBO-BOI(*BA*) = $|E_i|+|E_j| > 0$ such that

(1) $E_i = \{<BO_i, BO_j> \in VxV | (\exists op \in OP)$: ($BO_i$ operates on its own business state with *C* operation) ^ (BO_j operates on its own business state with *op* operation)} and

(2) $E_j = \{<BO_i> \in V$: (BO_i operates on its own business state with *C* operation)}.

More specifically, the high degree of interactions between BOs and their respective functionality should be encapsulated in a module to reduce an inter-business object coupling. If CBO-BOI(*BA*) = 0, there is no business object belonging to the *BA* and such a *BA* is called an *independent* business activity.

Definition 3: *Coupling between Business Activity—Interaction-based (CBA-I) of an identified BA is the proportion of the number of internal interactions (II) and the number of external interactions (EI).*

Let an identified *BA* consists of *n* member business objects, $BO_{BA}=\{BO_1, BO_2, \ldots, BO_n\}$. Given a $BO_i \in BO_{BA}$,

(1) BO_i interacts with other business objects belonging to the same *BA* (II) if there exist edges E_{II} ($|E_{II}|> 0$) such that
$E_{II} = \{<BO_i, BO_j> \in VxV | (\exists i \in I)$: ($BO_i$ and BO_j interact through an *i* interaction)}. The more $|E_{II}|$ implies the higher quality of the internal relationships in the *BA*.

(2) BO_i interacts with other business objects, which belong to different business activities if there exist edges E_{EI} ($|E_{EI}|> 0$) such that
$E_{EI} = \{<BO_i, BO_j> \in VxV | (\exists BO_j \notin BO_{BA})(\exists i \in I)$: ($BO_i$ and BO_j interact through an *i* interaction)}. The more $½E_{EI}½$implies the higher coupling of the *BA* with other business activities.

The CBA-I(*BA*) value is calculated in interval [0, 1]:

$$\text{CBA-I}(BA) = \frac{|E_{EI}|}{|E_{EI}| + |E_{II}|}$$

The higher CBA-I(*BA*) value, the higher coupling of the *BA*. We specify a critical point to be 0.5 for comparing the external interactions to the total interactions of a business activity.

- If CBA-I(*BA*) > 0.5, there exist more external than internal interactions. This implies that the *BA* should be split into two or more business activities. A technique for breaking up is to split the functionality and their responsive BOs.
- In contrast, if $|E_{EI}|<|E_{II}|$or the CBA-I(*BA*) is close to *0.0*, there is a large number of internal interactions, which need not to be split up.

Definition 4: *Lack of Cohesion of Business Activity(LCBA).*

In theory, for any identified *BA* which has been constructed as an independent component, the number of internal-method invocations should represent the degree of cohesion. In practice, however, such interactions may enable business object *"clusters"*, operating on disjoint sets of business objects.

The LCBA measure is a refinement of the notion of the Lack of Cohesion in Methods (LCOM) [7] to determine the number of business object clusters. Let *BA* denotes an identified business activity and BO_{BA} the set of member business objects. Let $G_{BA}(V, E)$ be undirected graph with V = BO_{BA}. To obtain an inverse measure of cohesion, LCBA(*BA*) = $|E_c|$ is defined as the number of cohesion clusters of BOs within the same *BA* such that

$E_c = \{<x, y> \in VxV \mid z \in BO_{BA} : \neg((\exists <x, z> \in VxV) \vee (\exists <y, z> \in VxV))\}$.

If LCBA(*BA*)>*n*, this implies that *BA* should be split into *n* sub business activities, each containing a cluster of BOs and their responsive functionality. On the other hand, if LCBA(*BA*)=*0*, *BA* needs not to be split.

In summary, an identified *BA* should satisfy both criteria such that CBA-I(*BA*)<0.5 and LCBA(*BA*)=*0*.

Business Component Identification and Measurement

A business component is composed of identified business activities which satisfy both coupling and cohesion measurements. The first step of the technique is to combine a set of identified business activities with *high coupling* into a single business component with the objective of reducing coupling between identified business components. Then, the identified business component is evaluated for possible split into more business components based on the cohesion measurement.

Definition 5: *Coupling between Business Activity—Business Object Interaction (CBA-BOI) is the number of interactions between BOs of the two business activities.*

Let BO_{BA1} and BO_{BA2} be the set of business objects belonging to two identified business activities BA_1 and BA_2, respectively, and $BO_i \in BO_{BA1}$ and $BO_j \in BO_{BA2}$. There are two kinds of interactions between two business activities, *direct* and *indirect* interactions.

(1) There is a *direct* interaction between BA_1 and BA_2 if there exists E_D such that
$E_D = \{<BO_i, BO_j> \in VxV\ \exists k \in I$: ($BO_i$ and BO_j interact through a *k* interaction)}

(2) There is an *indirect* interaction between BA_1 and BA_2 if there exists E_{ID} such that
$E_{ID} = \{<BO_i, BO_j> \in VxV\ |(\exists BO_p \in BO_{BAX})$ such that $|<BO_i, BO_p>|>0$ and $|<BO_p, BO_j>|>0$ }. That is, there is a business object BO_p, which enables two interactions BO_i and BO_p, and BO_p and BO_j.

The CBA-BOI measure is used to identify business components as follows. Let I_{CC}={C-C} be the highest coupling degree of interaction patterns. Two identified business activities BA_1 and BA_2 should be encapsulated as an identified business component if CBA-BOI(BA_1, BA_2) > 0 such that

$$\text{CBA-BOI}(BA_1, BA_2) = |E_D| + |E_{ID}|$$

$$\text{where } |E_D| > 0, \text{ if } \exists k \in I_{CC} \text{ and } |E_{ID}| > 0, \text{ if } \exists l, m \in I_{CC}$$

From the underlying object-oriented perspective, we can define coupling and cohesion measurements of business components as they are defined for business activities.

Definition 6: *Coupling between Business Component—Interaction-based (CBC-I) of an identified business component (BC) is the proportion of the number of internal interactions (II) and the number of external interactions (EI).*

By using the same criterion as CBA-I(*BA*) we can define CBC-I(*BC*) to measure the coupling between business components for creating independent business components. In addition to the LCBC(*BC*), it can be also defined using the LCBA(*BA*) for measuring lack of cohesion in business components.

Since business components can be assembled into a larger-grained component, the properties of an ideal business component are *low coupling and high cohesion* using the previous defined measurements. More specifically, a required business component should satisfy both criteria such that CBC-I(*BC*)<0.5 and LCBC(*BC*)=*0*.

Business Component Model

Since XML is a standard language, XML-based representation is helpful and meaningful for exchanging design information with other developers varying on different platforms and with development tools. Moreover, an XML-based model provides information about a business component that is easy to bundle and deploy in a particular commercial business components environment such as Enterprise JavaBeans [22].

The primary elements of a business component are business activities that specify required BOs. Moreover, a business component may provide services for other components so that there is an optional component element, *interface*. The business component model is represented using an XML-based description in order to be applied efficiently to Web computing environments. The respective XML DTD is shown in Figure 3.

In addition to requiring that business activities be processed correctly in a well-defined sequence, every business component should have one controller called a *business process controller (BPC)* to control and manage the collaboration of its business activities. More specifically, a set of *main* business activities has to be controlled by the BPC for collaborating in the required sequences.

Further, we classify our business components, using the *service-based measurement* [3], into two layers: *common business component* (CBC), which provides general purpose services and *application business component* (ABC) that does specific tasks. The *service-based measurement* is measured in terms of *export* services that a business component provides for other business components.

Figure 3: XML DTD of business components

```
<?xml  version="1.0"   standalone="yes"?>
<!DOCTYPE          BusinessComponent[
<!ELEMENT          BusinessComponent     (BusinessObject+,
                                         BusinessActivity+, Interface*)>
<!ATTLIST          BusinessComponent     name    ID     #REQUIRED>
<!ELEMENT          BusinessObject        (Structure, Behavior)+>
<!ATTLIST          BusinessObject        id       ID      #REQUIRED>
<!ELEMENT          Structure             (Attribute)+>
<!ELEMENT          Behavior              (Method)+>
<!ELEMENT          Attribute             #PCDATA>
<!ATTLIST          Attribute             id        ID       #REQUIRED
                                         type  CDATA   #REQUIRED>
<!ELEMENT          Method                #PCDATA>
<!ATTLIST          Method                name  CDATA  #REQUIRED
                                         input ((BusinessDataUnit*,
                                           Attribute*)*)
                                         output  ((BusinessDataUnit*,
                                           Attribute*)*)>
<!ELEMENT          BusinessActivity      (BusinessObjectList,
                                         BusinessDataUnit+, Interaction+)
<!ATTLIST          BusinessActivity      id       ID       #REQUIRED>
<!ELEMENT          BusinessObjectList    (BusinessObjectRef)+>
<!ATTLIST          BusinessObjectRef     BC_name IDREF
                                           #REQUIRED
                                         BO_id IDREF #REQUIRED >
<!ELEMENT          BusinessObjectRef     #PCDATA>
<!ELEMENT          BusinessDataUnit      (AttributeId)+>
<!ATTLIST          BusinessDataUnit      id       ID       #REQUIRED
                                         BO_id IDREF #REQUIRED >
<!ELEMENT          AttributeId           #PCDATA>
<!ATTLIST          AttributeId           id    IDREF    #REQUIRED
<!ELEMENT          Interaction           (Method1, Method2*)+>
<!ATTLIST          Interaction           name      ID    #REQUIRED>
<!ELEMENT          Method1               #PCDATA>
<!ATTLIST          Method1               name  IDREF  #REQUIRED>
<!ELEMENT          Method2               #PCDATA>
<!ATTLIST          Method2               name IDREF #REQUIRED>
<!ELEMENT          Interface             (BusinessActivityList*,
                                           PublicMethodList*)*>
<!ELEMENT          BusinessActivityList(BusinessActivityId)*>
<!ELEMENT          PublicMethodList      (MethodName)*>
<!ELEMENT          BusinessActivityId    #PCDATA>
<!ATTLIST          BusinessActivityId    name IDREF   #REQUIRED
<!ELEMENT          MethodName            #PCDATA>
<!ATTLIST          MethodName            name   IDREF   #REQUIRED
]>
```

WEB IMPLEMENTATION MODELING

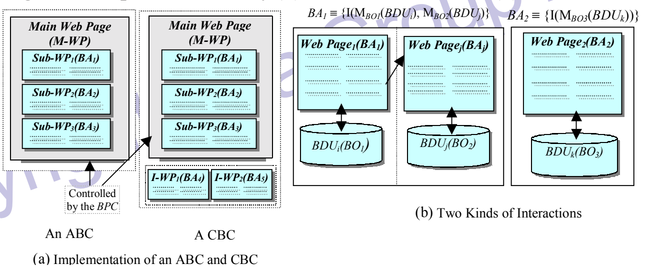
The XML-based business component model representation is a meta data model offering information about software component implementation. The implementation of primary component elements of the model on the Web follows the model described below.

- 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 will be linked to other Web pages (see Figure 4(a)).
- 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 applications. Suppose that,

  $BA_1 \equiv \{I(BO_1, BO_2)\} \equiv \{I(M_{BO1}(BDU_i), M_{BO2}(BDU_j)\}$ and
  $BA_2 \equiv \{I(BO_3)\} \equiv \{I(M_{BO3}(BDU_k)\}$

These two business activities can be implemented by Web application components that support different tasks (see Figure 4(b)). BA_1, for example, requires the interaction, which involves BO_1 and BO_2. Since BO_1 and BO_2 operating with their respective BDUs enable creation of their own Web pages, the interaction between BO_1 and BO_2 within the BA_1 represents an interaction between business activities. However, an interaction between two BOs may have no user interfaces (Web pages).

Figure 4: Implementation of (a) ABC and CBC and (b) Interactions



CONCLUSION

This paper proposes a methodology for identifying a business component suite based on process-oriented and object-oriented paradigms using coupling and cohesion measurements. The analysis model is represented using graph theory, which is a standard and formal representation, allowing development tools to be able to measure the coupling and cohesion automatically. 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 a more natural approach to analysis of complex business domains and systems.

An XML with its meta data description capabilities is used for representation of and working with the suitable business component model to support the software development processes. Such a representation allows convenient and functionally flexible adaptation to a Web-based computing environment for developing multi-tier distributed applications.

In accordance with enormous efforts toward transition to a new generation Web, we accept the proposed XML-based model description as a good basis for extension to not only structural but semantic representation and processing of business components.

ENDNOTE

1 This included D operation

REFERENCES

[1] Ritu Agarwal, Prabuddha De, and Atish P. Sinha, Comprehending Object and Process Models: An Empirical Study, IEEE Trans. Software Eng., vol. 25, no. 4, pp.541-555, July/August 1999.
[2] James M. Bieman and Byung-Kyoo Kang, Measuring Design-Level Cohesion, IEEE Trans. Software Eng., vol. 24, no. 2, pp.111-124, February 1998.
[3] Lionel C. Briand and and Sandro Morasca, Defining and Validating Measures for Object-Based High-Level Design, IEEE Trans. Software Eng., vol. 25, no. 5, pp.722-743, September/October 1999.
[4] G. Booch, Object-oriented analysis and design with Applications, second ed., Redwood City, Calif.: Benjamin/Cummings, 1994.
[5] Alan W. Brown and Kurt C. Wallnau, The Current State of CBSE, IEEE Software, September/October 1998.
[6] Jim Q. Chen and Richard D. Heath, Building Web Applications; Challenges, Architectures, and Methods, Information Systems Management, Winter 2001, pp.68-79.
[7] Shyam R. Chidamber and Chris F. Kemerer, A Metric Suite for Object Oriented Design, IEEE Trans. Software Eng., vol. 20, no. 6, pp. 476-493, June 1994.
[8] Tom Digre, Business Object Component Architecture, IEEE Trans. Software, September/October 1998.
[9] Martin Gaedke, Daniel Schwabe, Gustavo Rossi, and Hans-W. Gellersen, Web Engineering: Introduction to Minitrack, Proceedings of the 33rd Hawaii International Conference on System Science, 2000.
[10] G.Q. Huang and K.L. Mak, Issues in the development and implementation of Web applications for product design and manufacture, International Journal Computer Integrated Manufacturing, 2001, Vol.14, No.1, pp.125-135.
[11] Peter Herzum and Oliver Sims, Business Component Factory: A Comprehensive Overview of Component-Based Development for the Enterprise, John Wiley & Sons, Inc., 2000.
[12] Martin Hitz and Behzad Montazeri, Correspondence of Chidamber and Kemerer's Metric Suite: A Measurement Theory Perspective, IEEE Trans. Software Eng., vol. 22, no. 4, pp.267-271,April 1996.
[13] Ivar Jacobson and et al., Object-Oriented Software Engineering(OOSE): A Use Case Driven Approach, revised printing, Addison-Wesley Publishing Company, 1995.
[14] Ivar Jacobson, Maria Ericsson, and Agneta Jacobson, THE OBJECT ADVANTAVE: business process reengineering with object technology, Addison-Wesley Publishing Company, 1994.
[15] Ivar Jacobson, Martin Griss, and Patrik Jonsson, Software Reuse: Architecture, Process and Organization for Business Success, Addison-Wesley Publishing Company, 1997.
[16] R. Kocharekar, K-Commerce: Knowledge-Based Commerce Architecture with Convergence of E-Commerce and Knowledge Management, Information Systems Management, Spring 2001.
[17] OMG-Business Object Domain Task Force, Business Object Concepts, White paper, January 1999 OMG document: bom/99-01-01.
[18] Monique Snoeck-S and Guido Dedene, An Architecture for bridging OO and Business Process Modelling, IEEE Int'l Conference on Technology of Object-Oriented Language and System (TOOLS'00), 2000.
[19] World Wide Web Consortium, Extensible Markup Language (XML) 1.0 (Second Edition), W3C Recommendation 6 October 2000, http://www.w3.org/TR/2000/REC-xml-20001006.
[20] World Wide Web Consortium, Extensible Stylesheet Language (XSL) Version 1.0, W3C Candidate Recommendation 21 November 2000, http://www.w3.org/TR/xsl/.
[21] Kenneth H. Rosen, Discrete Mathematics and Its Applications, McGraw-Hill International editions, Fourth Editions, 1999.
[22] Sun Microsystems, Enterprise JavaBeans Specification Version 1.1, May 31, 2000.