IDEA GROUP PUBLISHING
701 E. Chocolate Avenue, Suite 200, Hershey PA 17033, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

ITP4842

Business Components Based Design for Enterprise Systems

Tarun K. Sen
Department of Accounting and Information Systems
Pamplin College of Business, Virginia Tech
7054 Haycock Road
Falls Church, Va 22043
Phone: 703 538 8413, Fax: 703 538 8415
Email: tksen@vt.edu

David Tegarden and Reza Barkhi
Department of Accounting and Information Systems
Pamplin College of Business, Virginia Tech
Blacksburg, Va 24061
Phone: 540 231 6099, Phone: 540 231 5869
Fax: 540 231 2511, fax: 540 231 251
Email: david.tegarden@vt.edu, Email: reza@vt.edu

ABSTRACT

Business components can be used to design systems that adapt to an organization's business processes. Components implement specific business functions that when connected appropriately produces adaptable business processes. Components are logical constructs that allow interoperability and reusability across businesses. We develop a method to define business processes using the component paradigm. The component-based design approach is described using an example university information system.

1. INTRODUCTION

The genesis of object-oriented design (OOD) can be traced to the need for defining components that can be reused with ease. OOD allows the encapsulation of data and related functions into atomic units (objects), which can be integrated in dynamic ways with the ultimate objectives of reducing design and development times and to produce sustainable applications. The expectation is that OO applications can adapt to evolving user needs in a shorter timeframe than traditional application design methodologies. Over the last decade, the object-oriented paradigm has been used extensively in commercial and scientific applications with enormous success. While suited for application design, OOD does not support the inclusion of business processes into the design phase. This often leads to a high-level of application design complexity and ultimately may not address business processes in a holistic manner. This drawback is often the reason that systems designers do not employ OO design techniques for traditional business applications.

Most enterprise systems are built using off-the-shelf packages, like Enterprise Resource Planning (ERP) systems and Customer Relationship Management (CRM) systems. Adopting these packages forces organizations to deviate from their traditional business processes often resulting in failed implementations (Bingi et al. 1999). In this paper, we present a business process-based design framework for component based systems based on the Object-Oriented (OO) paradigm. Our framework, called Component Based Design (CBD) synthesizes the desirable features of Object-Oriented Design (OOD) with business process components (Baster et al., 2001). Using CBD, an organization can design applications that are process-oriented and object-based (Allen 2002). Since CBD is an extension of OOD, it supports flexibility, reusability, and flexible adaptation to user needs while capturing business process workflows.

2. SUPPORTING A BUSINESS PARADIGM USING COMPONENTS

Component Based Design (CBD) is a business-process-based paradigm. Application design in CBD starts with process definitions and through several layers of conceptual and physical design leads to detailed application design. But in developing such a comprehensive framework, several challenges have to be mitigated. These challenges are focused on the fact that in order to create a robust system that supports business processes in a flexible manner, the designers must take an enterprise-view of the requirements and design. However, logical this is, the challenge of doing this in large, multi-tiered organizations is significant. Our framework promotes an incremental and top-down approach to tackling these challenges. CBD views system design as a synthesis of four concepts:

- Enterprise business processes
- Business process components
- Object design
- Architectural design

2.1 Enterprise Business Process Representation

One of the toughest challenges faced by an organization is the definition of the enterprise's business processes. It is not unusual to find several definitions of a simple process in a large organization. Getting stakeholders to agree on an enterprise-definition of a process requires executive commitment and investment. But even before stakeholders can begin to understand the current processes and to define new processes, a clear representation of desired business processes is necessary. The conceptual view of the business process can be represented in a variety of ways. CBD facilitates this by creating an enterprise business process blueprint (EBP blueprint). One of the challenges is to come up with a model that will allow users to create the enterprise business process blueprint using generally accepted business components.

2.2 Process Design

Once the enterprise's business processes have been represented using the CBD conceptual view, the next step is to define the components for each of these processes. A business component is *a clearly identifiable and replicable business function that can be expressed in a generic manner so that its functionality can be transportable from one business environment to another.* A grammar for business components and its interconnections needs to be defined for reusability and defining business processes.

2.3 OO Design

The enterprise business process blueprint and the process blueprints together form the foundation for the object design. In this phase, traditional OOD techniques are used to define reusable components *across* business processes and ultimately across the enterprise. CBD's strength lies in the fact that object design is based on clearly specified enterprise-wide process components defined in the earlier steps even though the implementation of the objects can be done over time and subsets of processes.

2.4 Architecture Design

Once the objects have been defined at the conceptual level that would describe the business processes, software developers need to define the architecture of the application. This definition includes hardware and software specification and blueprints. All implementation efforts will be based on these architectural blueprints.

3. THE SEMANTIC DEFINITION OF BUSINESS COMPONENTS

CBD's top down approach has clear benefits based on a breakdown of complex enterprise-wide design into individual business process based components (Fellner and Turowski 2000). CBD not only provides a guideline for how a process-based design approach should be done, it also includes a semantic framework for defining the four concepts presented in the previous section. Using the semantic framework, organizations can define enterprise business processes and individual business processes with ease and flexibility (Hall 1999).

Each enterprise process consists of a series of interacting components. The semantics of these components and their interactions are encapsulated in a process component. Each process component has the following attributes:

- Data
- Functions
- Security Constraints

Process data includes the information that is "flowing" through the component. The data item may be modified by the component or other processes may be triggered based on the value of the data item. Functions are tuples of the form <condition, action>, where condition indicates the existence of specific instance of the data item and action indicates the function to be performed when the condition is true. Every component has a set of security constraints to which it adheres. These constraints are necessary because different organizations require different levels of security for the same component. Even within the same organization the security level can vary from one set of users to another.

3.1 Component Definition

We view the business application as a set of interconnected components K_i where K_i is a function of the triplet (D, M, S). D is set of data, M is the set of public methods, and S is the set of security constraints on the components. $D = f(d_1, d_2, d_3, \dots, d_n)$ where d_i can be atomic or complex sets of data that are used in specific applications. For example, in a student registration component, atomic data could be the student information such as name and social security while an example of a complex data set could be the list of courses the student is registered for and the set of grades received for those courses.

The methods (M) are (c, a) tuples where c is the trigger condition that activates the action "a". The condition "c" is a function of a set of data d_i and operators o_j. For example, in the student registration component, the condition may be that a student cannot sign up for a junior level class when he or she is a freshman. The data is the course that he or she cannot take and the operator is the assignment of a student to that course. The action "a" is also a set of data and operators. So, for example, the operator should be able to assign a student to a course if registration conditions are not violated.

The security constraints (S) is a set of tuples (u, m) were u µ U and m µ M where U is the domain of users and M is the domain of methods in a component. The security constraints can be best viewed as a table with a column of methods for a component and a list of users (groups and individuals) that have access to the methods.

We view the design of business applications as an ensemble of interacting components that satisfies the workflow and the processes in that workflow. An ensemble is a set of interactions among components that cooperate through these interactions to provide some useful and predicted aggregate behavior (Wallnau, Hissam, and Seacord, 2002). Components can communicate by sending messages to other components. This will require each component to know the functionality of other components to request a favor from that component. Alternatively, each component that has a request can post the request on a common communication component, i.e., a blackboard (Nii, 1989). The blackboard can be scanned by components that may be interested in placing bids on specific requests.

3.2 Connecting Components Using Contracts

Business processes are created using components that can be connected using contracts (Cheesman and Daniels 2001). A contract T is a triplet of the following form: (K_i, T_{ij}, K_j) where T_{ij} is a set of possible contracts between the server component K_i and the client component K_j. T_{ij} is basically a condition-action tuple (c, a). For example, an application component can be connected to an admission component in the University information system by using the following condition-action tuple: if a student meets acceptance criteria, then make the admission component available. The server component initiates the connection, the client component accepts the responsibilities assigned by the server component. The client component may also deliver certain "contracted" information back to the server. It should be noted that a pair of components can switch roles of server and client by using different contracts. Also, a contract may involve more than two components.

Contracts can be used to make the connection between components that forms a business process. Contracts are based on the legal notion that both components have obligations to fulfill. In a sense it is a set of constraints and guarantees. These constraints can be written in a natural language like English or can be formalized using UML's object constraint language (Warmer and Kleppe, 1999). The notion of contracts comes from the object-oriented design paradigm (Wirfs-Brock, Wilkerson, Weiner, 1990, Dennis et al. 2002). A contract must specify the conditions that need to be fulfilled before an action can be taken. In a contract, typically there is a server component and a client component. When the server component meets a set of conditions then the client component is sent a message upon which it acts. The client then fulfills its obligations, by sending an appropriate message back to the server. In a given scenario, a component can be a client or a server. However, the same component that is a client in a business process can be a server in another part of the business process. Also, multiple contracts are possible between a pair of components. When linking two components not all contracts have to be made. Contracts can link more than two components in a given process. Contracts allow components to be linked to form different business processes.

In the following section we use an example from a university information system to illustrate the principles underlying component based design.

4. THE UNIVERSITY INFORMATION SYSTEM EXAMPLE

We first develop an enterprise business process representation of the University environment using components. Figure 1 shows a high

Figure 1. Business Process Diagram Using ComponentsFigure 3: Example Contract

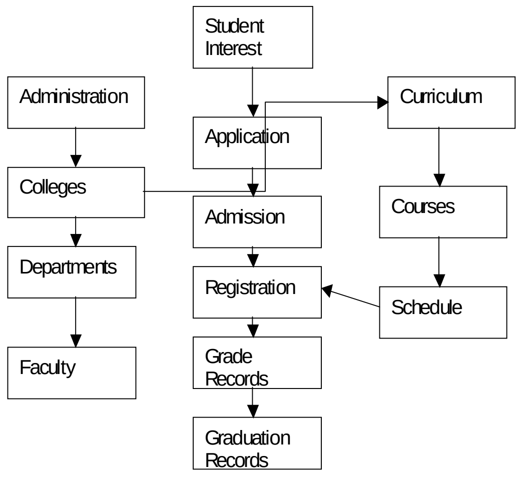
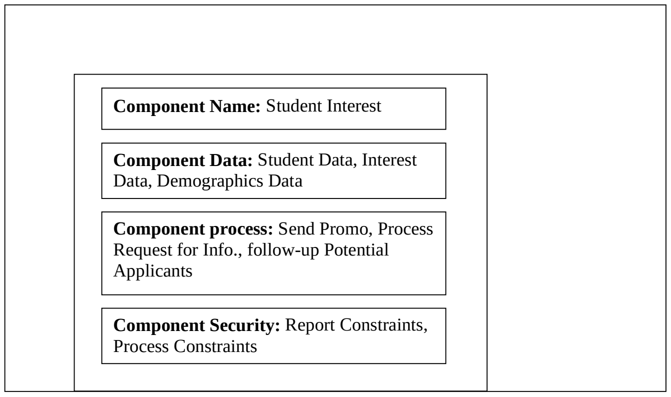


Figure 2: Component Specification for Student Interest


Component Name: Student Interest
Component Data: Student Data, Interest Data, Demographics Data
Component process: Send Promo, Process Request for Info., follow-up Potential Applicants
Component Security: Report Constraints, Process Constraints

level business component diagram which could be viewed as the business process that the university uses to manage its constituents. The process is initiated when a student requests information from the University. The process ends when a student graduates and becomes an alumnus. Each box in the diagram is a component. These components are used as examples. They may not represent the best selection of components for this environment. The ideal set of components for a university environment has to be determined by functional experts in this area.

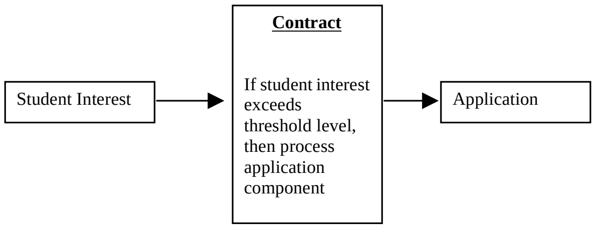
The connections between the components are established by using contracts. This, in effect, results in establishing a business process for the University. Note that the process shown here is one of many possible business process configurations that could have been obtained by using the same set of components. This illustrates the advantage of using business components design where flexibility can be availed without compromising reusability significantly.

In Figure 2, we take one of the components (student interest) and show how this component can be described using data, methods, and security constraints. The description shown is at a very high level and low level specifications can be provided using UML or other implementation languages.

In Figure 3, we show how components can be connected using contracts. For example, we connect a student interest component to an application component using one of the contracts available for this component set. Different business processes can be obtained by using different contracts. Of course, the flexibility of obtaining different business processes is limited to the extent of the different contracts that are available for a given pair of components

One of the biggest challenges in designing business processes using components is to formalize the connection mechanism between components. Although, having several open ended connections between components allow for greater flexibility in defining different business processes, this creates interoperability challenges. One must seek a requisite balance between flexibility and interoperability. Thus the number of connections between a pair of components should be limited and well defined. The other challenge defining connection mechanisms between components is to come up with a formal language to represent it. The contract-based specification provides a potential framework for representing connection mechanisms. The language has to be formalized for business component based systems to provide greater reusability and interoperability.

Figure 3: Example Contract


Student Interest
Contract
If student interest exceeds threshold level, then process application component
Application

5. CONCLUSION

In this paper we have tried to show the principles underlying component-based design for business applications. We have focused primarily on the highest level of design for this approach. This level is called the enterprise business process. The next level that is called the business process level uses tools like UML to expand upon the enterprise business model using components. Finally, we can use object design approaches and architectural blueprints to specify the system more completely.

There are several approaches to building components. For example, in a very general sense, there seems to be two general types of components: fine-grained and course-grained. IBM's San Francisco patterns focuses on fine-grained components. In this paper we have focused on course-grained components. In a nutshell a course-grained component seems to be equivalent to the idea of a sub-system. In the OO world, these are known as subjects, subsystems, collaborations, clusters, and packages. Furthermore, a business definition of a course-grained component could be a set of collaborating objects that make sense to package together into a component for sale. If it does not make sense to sell it, it probably does not make sense to create it as a component. In this paper, we have attempted to define these course-grained components. After defining them, we show how they can be connected to create business processes. We are currently working on the translation of these business components into architectures that can be implemented using object-oriented technology.

REFERENCES

1. Allen, P. Realizing e-Business with Components, Addison Wesley, London, 2001.

2. Baster, G., Konana, P., and Scott, J. , " Business Components: A Case Study of Bankers Trust Australia Limited," *Communications of the ACM*, Vol. 44, No. 5, 2001, pp. 92-98.

3. Bingi, P., Sharma, M.K., and Godla, J., "Critical Issues Facing an ERP Implementation," *Information Systems Management*, vol. 16, no. 3, 1999 pp. 7-14.

4. Cheesman, J., and Daniels, J. UML Components: A Simple Process for Specifying Component-Based Software, Addison Wesley, 2001, Upper Saddle River, NJ.

5. Dennis, A., Wixom, B.H., and Tegarden, D. Systems Analysis and Design: An Object-Oriented Approach with UML, John Wiley and Sons, 2002.

6. Fellner, K., and Turowski, K., "Classification Framework for Business Components," *Proceedings of the 33rd. Hawaii International Conference on System Sciences*, 2000.

7. Hall, P.A.V. "Architecture Driven Component Reuse", *Information and Software Technology*, vol. 41, no. 14, November 1999, pp. 963-968.

8. Nii, H.P. Chapter XVI: Blackboard Systems in The Handbook of Artificial Intelligence, Vol IV, ed. By A. Barr, P.R. Cohen, and E.A. Feigenbaum, Addison-Wesley, 1989.

9. Wallnau, K.C., Hissam, S.A., and Seacord, R.C. Building Systems from Commercial Components, Addison-Wesley, 2002.

10. Warmer, J., and Kleppe, A. The Object Constraint Language: Precise Modeling with UML, Addison-Wesley, 1999.

11. Wirfs-Brock, R., Wilkerson, B., and Wiener, L. Designing Object-Oriented Software, Prentice-Hall, 1990.