By Padmal Vitharana, Fatemah "Mariam" Zahedi, and Hemant Jain

Until recently, anecdotal evidence could only suggest CBSD superiority in requirements identification. Here is a set of testable hypotheses to help distinguish hype from fact.

DESIGN, RETRIEVAL, AND ASSEMBLY in Component-based Software Development

Component-based software development (CBSD) offers an effective approach to constructing software products. Grounded in the concept of component fabrication and assembly, CBSD can help the software industry realize quality and productivity gains similar to those achieved in the hardware and manufacturing industries [4, 10]. As an increasing number of software projects miss schedules, exceed budgets, and deliver defective products, industry experts have turned to component-based solutions to overcome the current software crisis [6]. The recent emergence of the Web services model for delivering component-based solutions over the Internet further underscores the importance of CBSD.

Although several initiatives have been directed at gaining insight into CBSD, relatively few formal methods and techniques for component design and application assembly exist. In recognition of this gap, we have undertaken a coherent research program to address key issues related to component design, retrieval, and assembly. We believe more formal methodologies are needed to make the CBSD paradigm into an effective development tool. Here, we present a summary of our research work.

The CBSD approach seeks to develop the components required to support various functions and processes for a particular domain or area. A component is a binary code that offers a set of related services through a published interface. The domain selected for componentizing can be as broad as manufacturing or as narrow as accounts receivable. Once domain components are designed and developed, various applications in that domain are constructed by first selecting and then assembling chosen components. Appropriate selection and sequencing of components provide the opportunity to customize the application. Because components in a domain might number in the hundreds or even thousands, systematic cataloging of components for subsequent search and retrieval is critical. Classification

and coding of components in a knowledge base such as a repository allows application assemblers to easily find the components that match user requirements and support the desired function or process of a particular domain application. Figure 1 summarizes our conceptualization of the CBSD life cycle. Here, we present our research efforts on second, third, and fourth phases of the CBSD life cycle.

A Model for Component Design

Although component technology has been around for quite some time, our understanding of how to design components is relatively weak. In fact, Szyperski [10] observes the "modeling of component-based systems is still a largely unresolved problem." Nonetheless, because of the close relationship between object and component paradigms, many industry watchers [4, 10] advocate adopting object-oriented (OO) fundamentals to develop components. Although a few methodologies such as Catalysis and Rational have emerged over the years, they often take a more technical approach to designing components.

Our aim was to develop an approach driven by the component developer's business strategy, as reflected in its managerial goals. These managerial goals are translated into technical features of components, a key aspect lacking in existing component design methodologies. We adopt an OO approach to domain modeling, in which the domain is analyzed and objects[1] representing the domain are identified [11]. These objects are then used as building blocks for constructing components. In designing components from the domain model, the challenge is to identify decision criteria for grouping objects into sets of components that support the business strategy of the component developer. Component developers need to address the interests of their external and internal customers, who purchase components and assemble them into applications.

Since component design is analogous to manufacturing and product design, we referred to the existing literature to identify cost effectiveness, ease of assembly, reusability, customizability, and maintainability as relevant managerial goals. The achievement of these managerial goals depends on various technical features or attributes that characterize component design. By reviewing applicable literature from OO and component design, we have identified the following technical features: coupling between components, cohesion of objects within a component, number of domain components, component size or granularity, and complexity. A summary of managerial goals and technical features is provided in Table 1.

Managerial Goals	Technical Features
Cost Effectiveness (cost effectiveness in creating components from objects)	***Intercomponent Coupling*** (extent to which the component is coupled with other components)
Ease of Assembly (extent of ease in assembling components into applications)	***Intracomponent Cohesion*** (strength of association between objects within a component)
Customization (capability of components to be used in developing customized applications)	***Number of Components*** (number of components representing the domain)
Reusability (component's potential for reuse in multiple applications)	***Component Size*** (number of objects in a component)
Maintainability (ease of maintaining or enhancing the component)	***Complexity*** (number of methods and complexity of their parameters in objects of a component)

Table 1. Summary of managerial goals and technical features.

This framework connecting managerial goals with quantified technical features is the basis of our component design model, which supports the component developer's business strategy. The business strategy then represents the trade-offs among conflicting management goals. For instance, if the company's business strategy is to provide low-cost and easy-to-assemble components, then higher weight is given to the cost effectiveness and ease of assembly goals. By adopting a strategy-based component development approach, managers can strike a balance among managerial goals to determine the optimal design that delivers the greatest benefit. Based on OO and component-based literature, we identified relationships between managerial goals and technical features,[2] and developed expressions for the quantification of technical features.[3] In this context, the task of determining the best design becomes an optimization problem, in which the chosen compo-

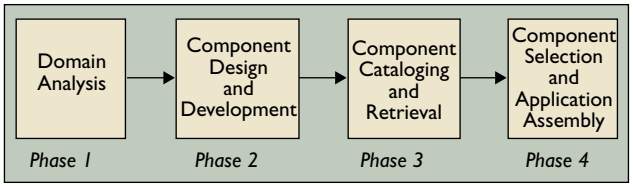

Figure 1. Component-based software development (CBSD) life cycle.

[1]A class of objects (interchangeably referred to as an object) represents an abstraction of similar elements.

[2]For example, research literature posits that the technical feature component size is positively related to the managerial goal ease of assembly, but negatively related to the managerial goal reusability.

[3]For example, the technical feature component size is measured in terms of the number of objects in a given component.

nent design strategy is optimized in terms of high-level managerial goals, measured in reference to their relationships with quantified technical features.

To determine the strength of the relationships between managerial goals and technical features, we conducted a survey of industry professionals. The survey was posted on selected online newsgroups frequented by OO and component industry experts. In this survey, respondents were requested to indicate the relative strength of the relationships between managerial goals and technical features on a scale of 1 to 10. Furthermore, to assess expert opinion on relative weights for the managerial goals, we also asked respondents to allocate 100 points among the five managerial goals. All of these weights could be subsequently modified according to the preferences of the component designer applying our component design model.

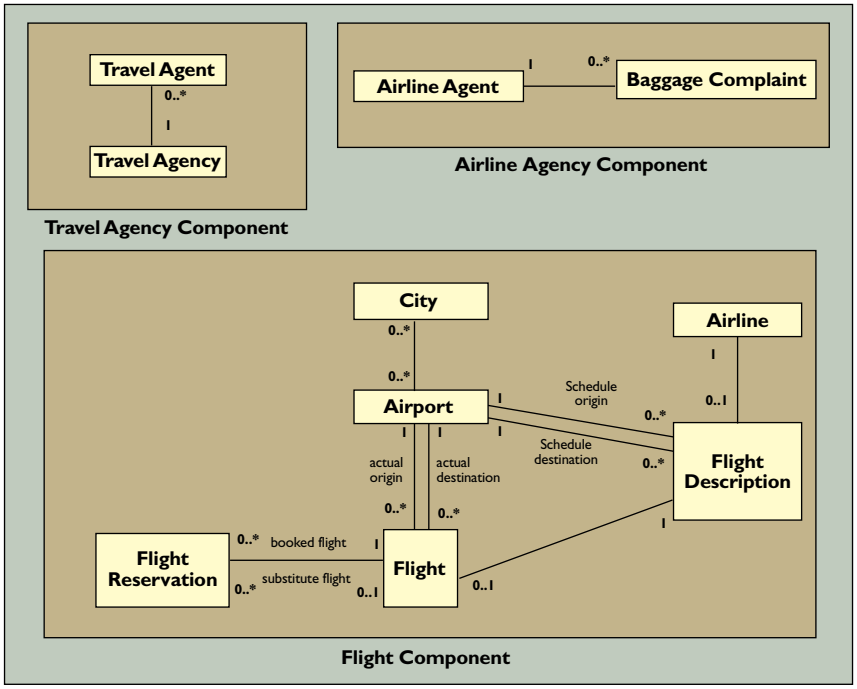

Figure 2. Sample components for the airline reservation case.

To validate the model, we conducted two case studies. One study utilized life insurance data from Tata Consulting Services (TCS), an international software consultancy firm. The other study utilized data from a published source regarding a well-known airline reservation scenario. The component design model was applied to the object models for these two cases using the GAMS optimization software package. In analyzing the results,[4] it was evident the component set for each of the two cases had a clear, logical structure. Figure 2 illustrates three of the resulting components and corresponding objects for the airline reservation case.

To further validate our model, we selected three component design strategies and applied the model to them. The component design strategies chosen were: maximum-maintainability strategy; maximum-reuse strategy; and balanced strategy, which involves applying equal weights to all managerial goals. The resulting component configurations were then presented to a group of IS professionals engaged in CBSD at TCS. When queried, these experts generally agreed that the derived designs are feasible and match the chosen component design strategy. This provided additional support for the validity of our component design model.

[4]That is, the number of components and the assignment of domain objects to those components.

Component Cataloging and Retrieval

In CBSD, one of the greatest challenges facing the component assembler is the discovery of suitable components for constructing customer applications. Because the need for an application is driven by a set of user requirements, finding the components that match those requirements is critical to delivering the desired product. Hence, classification and coding (C&C) of components for subsequent search and assembly play a vital role in CBSD success [4]. Many scholars have identified the need for a knowledge base, or repository, approach to cataloging components [2].

Although a survey of current research does not reveal a comprehensive C&C scheme that facilitates storage and retrieval of business components, lessons learned in other disciplines, such as manufacturing and software engineering, might help us derive a C&C scheme for components. In manufacturing, part classification in general and group technology in particular address C&C issues. For example, part characteristics such as geometry (as in shape), function (as in fastener), or material (as in plastic) are often used to classify manufactured parts [3]. In software engineering, several approaches to classifying traditional reusable artifacts have emerged. These include attribute-value, keyword, hypertext, and faceted classification. Of these, faceted classification for archiving reusable artifacts has recently gained some attention. In faceted classification, a set of facets, or categories, and corresponding descriptions are identified for a particular subject area. Reusable artifacts are then classified according to some facet-description pairs.

Faceted classification, successfully applied to catalog traditional reusable software artifacts, offers the following advantages over other methods [9]:

- Fosters the development of a standard vocabulary for items of interest
- Accommodates a continually expanding collection of items
- Facilitates finding reusable items that are similar, not just exact matches
- Provides high descriptive power for items to be cataloged

By adopting key principles from manufacturing and software engineering, we developed a C&C scheme for components, and have used it as the basis for the design of a component knowledge base [12]. In deriving the C&C scheme, we examined the inherent structure of the component, services offered by various components, and how components differ from each other. Note that each component adheres to a particular structure representing various levels of detail. A component consists of a set of interfaces and each interface consists of attributes, methods, and exceptions. This inherent structure of the component facilitates its cataloging for subsequent search and retrieval in two ways. First, at each level of detail, classifiers can be identified to distinguish components from each other. Second, such a structure enables the assembler to start with a much broader search and subsequently, through repeated iterations, to narrow the search to a handful of components for closer scrutiny.

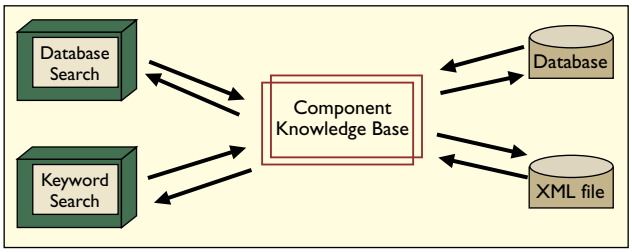

Figure 3. Component knowledge base.

Because of the clear advantage of faceted classification, we adopted it in our C&C scheme. At each level of the component structure, we identified a set of facets for describing the appropriate level of detail about the component. The facet selection process was based on a review of literature and an examination of component characteristics. Our intention was to identify facets useful in distinguishing components from each other in order to facilitate the subsequent search and retrieval, but not necessarily to derive an exhaustive facet list. For instance, at the component level (the highest level in the component structure), the role facet that describes a component's role in potential applications is used to classify that component. As an example, the travel agency component shown in Figure 2 might be used in a Web-based airline reservation system. Because each component typically adheres to certain rules, we also used the rule facet to characterize a component. The remaining facets we selected include functions of the component, such as travel agency management; elements associated with a component, such as travel agency; events associated with a component, such as flight reservation; and users of the component, such as travel agents. Table 2 illustrates sample facets for the travel agency component. Similarly, appropriate facets are identified for subsequent levels of the component structure, including interfaces, attributes, methods, and exceptions.

Although facets are predefined and structured, their descriptions are mostly semistructured or unstructured. For example, the role facet examples in Table 2 represent semistructured textual narratives of the travel agency component's role in a potential application.

Components may also be cataloged based on a set of well-defined classifiers. For example, based on type of use, we classify each component as one of system, such as an operating system; algorithm, such as statistical analysis; or application, such as travel agency. Furthermore, we place each component into standard industry categories such as airlines, banking, and so on (such as those based on SIC codes). Classifying each component according to its intrinsic structured and semistructured characteristics facilitates the subsequent search and retrieval of components during application assembly.

Using the C&C scheme, we designed a knowledge base for archiving, searching, and retrieving components. Because databases generally facilitate the storage, search, and retrieval of structured information, the component's structured information is stored in a relational database. In contrast, the semistructured part of the C&C scheme requires a more flexible vehicle. Since markup languages are appropriate for cataloging semistructured text-based information, we coded the facet-based portion of the classification scheme in Extensible Markup Language (XML). Thus, to find desired components in the knowledge base, one could use a structured search that queries the database, a semistructured keyword search that queries the text-based facet description stored in the XML format, or a combination of the two. Figure 3 illustrates the design of the proposed knowledge base.

A prototype of the knowledge base was developed

and used in an empirical study to gauge the effectiveness of the proposed C&C scheme. The components from the airline reservation case were cataloged and placed in the prototype knowledge base. The experimental task was to search and retrieve components satisfying three separate requirements with different levels of complexity. Subjects were instructed to first read each of the three tasks, or requirements specifications, in succession, and then search for component(s) matching the corresponding requirement. The effectiveness of the treatment (C&C scheme in the knowledge base) and control (no C&C scheme) groups was analyzed using retrieval metrics of recall, precision, search effort, satisfaction, and ease of use. According to the results, except for search time, all other measures for retrieval effectiveness indicated the treatment group outperformed the control group. This provided evidence for the validity of our C&C scheme for cataloging components, and its usefulness in the proposed knowledge base for the storage, search, and retrieval of components. Subsequently, the C&C scheme was successfully applied to a life insurance case provided by TCS.

Facet	Description	Example
1. Synonym	Other possible names for the component	Travel bureau
2. Role	Roles the component could play in potential end-applications	Travel agency in a Web-based airline reservation system
3. Rule	Important high-level rules applicable to the component	Travel agency must employ at least one travel agent
4. Function/task	What the component does	Manage functions of a travel agency
5. Element/part	Elements or parts associated with the component	Travel agency
6. Action/event	Actions and events related to the component	Flight reservation
7. User	Person/thing using the component	Travel agent

Table 2. Component-level facets for the travel agency component.

Requirements Analysis and Application Assembly

CBSD clearly signals a paradigm shift in software construction. Instead of building applications from scratch, the CBSD paradigm facilitates software development through component fabrication and assembly. CBSD not only discourages construction of new software, but also promotes assembly of prebuilt components in developing software applications. Accordingly, we expect fewer firms will specialize in the component fabrication business, with most application assembly conducted in-house or by third-party vendors.

When compared to the traditional paradigm, CBSD fosters closer alliance between users and analysts. CBSD allows users and analysts to work jointly to identify components satisfying a given set of requirements and assemble them in an application. Consequently, the requirements identification process in CBSD may be conducted in a markedly different manner than that practiced in the traditional paradigm. Components can advertise the services they offer through the component knowledge base. Armed with a vague notion of "needs and wants," users and analysts will be able uncover relevant application requirements as they traverse the component knowledge base. Querying the knowledge base to find the needed reusable artifacts increases users' knowledge in the problem domain [7]. Moreover, because the knowledge base holds the domain knowledge, the knowledge base should eventually contain the best practices of that domain or industry. CBSD empowers users to identify software components that satisfy their requirements. Supported by the knowledge base and analysts, users can actively participate in identifying requirements.

We are currently pursuing a research study to assess the relative effectiveness of the requirements identification process in CBSD and traditional systems development environments. Requirements identification is an exercise in problem solving that involves processing a tremendous amount of information [1, 5]. Herbert Simon and his colleagues have developed the information processing theory (IPT) to study human problem solving [8]. Accordingly, we developed an IPT-based assessment model to determine the effectiveness of requirement identification in the two paradigms.

Based on this IPT framework, we derived a set of testable hypotheses for assessing the efficiency and effectiveness of the CBSD approach to identifying requirements. Previously, we had only anecdotal evidence suggesting CBSD superiority in requirements identification. Because requirements identification is a precursor to component selection and application assembly, a scientific assessment such as the one proposed here is crucial for providing the support CBSD proponents need to distinguish hype from fact.

Implications for Practice and Research

Our research program has important implications

for project managers and developers contemplating the adoption of a component-based approach to developing application systems. First, our component design model presents a formal strategy-driven methodology to help managers systematically construct business components based on managerial goals and preferences. Instead of building components on an as-needed basis, construction of components encompassing an entire domain facilitates the subsequent assembly of a wide range of domain applications. Second, the proposed C&C scheme supports orderly cataloging and subsequent retrieval of desired components in building customer applications. Because our C&C scheme incorporates both structured and semistructured characteristics inherent in components, as the number of components increases it will enable assemblers to match user requirements with prefabricated components described in the knowledge base. Finally, by comparing the relative effectiveness of the requirements identification process in component-based and traditional software development environments, we can assess the quality of the user requirements generated from these two paradigms. If CBSD proves to generate higher-quality requirements, managers have an added incentive to pursue component-based solutions to their enterprise systems.

Our research can be extended in a number of directions. Some may study how useful designed components are in developing various types of application software. Others might explore research that directly models components without the object intermediaries. Additionally, one could employ industry experts to test the C&C scheme for cataloging components in the field. We also need additional research with case-based, qualitative, and longitudinal studies that assess the robustness of our component design model and C&C scheme.

Furthermore, research streams applicable to traditional software development can be extended to CBSD. For example, the discipline of software metrics entails identifying various attributes that need to be measured, and determining how to measure them. Certainly, we need new measures for assessing traits peculiar to CBSD. For instance, managers need metrics for assessing a component's reliability and its degree of customizability in building applications in order to make informed decisions about component-based solutions. Specifically, we believe that CBSD presents new research opportunities in the following areas:

- Measurement of analysts' and developers' productivity in developing component-based applications
- Configuration management, as in versioning
- Cost estimation of CBSD projects
- Pricing of components and component-based applications
- Promotion, sale, and distribution of components (including certification and licensing)
- Changes to user, analyst, developer, and project manager roles and relationships
- Quality, reliability, and maintenance of component-based solutions
- Management and organizational issues in component fabrication and application assembly

As a relatively recent phenomenon, CBSD offers a gold mine of opportunities for researchers. Our component design model, the C&C scheme, and the corresponding knowledge base offer an early attempt in formalizing some of the critical aspects of creating and using components. C

References

1. Agarwal, R., Sinha, A.P., and Tanniru, M. Cognitive fit in requirements modeling: A study of object and process methodologies. *Journal of MIS 13*, 2 (Fall 1996), 137–162.
2. Allen, P. and Frost, S. *Component-Based Development for Enterprise Systems: Applying the SELECT Perspective*. Cambridge University Press, Cambridge, U.K., 1998.
3. Billo, R.E. and Bidanda, B. Representing group technology classification and coding techniques with object-oriented modeling principles. *IIE Transactions 27*, 4 (1995), 542–554.
4. Brown, A.W. *Large-Scale Component-Based Development*. Prentice Hall, Upper Saddle River, N.J., 2000.
5. Davis, A.M. *Software Requirements: Analysis and Specification*. Prentice Hall, Englewood Cliffs, N.J., 1990.
6. Fayad, M.E and Schmidt, D.C. Object-oriented application frameworks. *Commun. ACM 40*, 10 (Oct. 1997), 32–38.
7. Mili, R., Mili, A., and Mittermeir, R.T. Storing and retrieving components: a refinement based system. *IEEE Transactions on Software Engineering 23*, 7 (July 1997), 445–460.
8. Newell, A. and Simon, H. *Human Problem Solving*. Prentice-Hall, Englewood Cliffs, N.J., 1972.
9. Prieto-Diaz, R. Implementing faceted classification for software reuse. *Commun. ACM 34*, 5 (May 1991), 89–97.
10. Szyperski, C. *Component Software: Beyond Object-Oriented Programming*. ACM Press, New York, 1998.
11. Vitharana, P., Jain, H., and Zahedi, F.M. Strategy-based design of reusable business components. University of Wisconsin-Milwaukee, School of Business Administration Working Paper, Fall 2000.
12. Vitharana, P., Zahedi, F.M., and Jain, H. Classification and retrieval of reusable business components. University of Wisconsin-Milwaukee, School of Business Administration Working Paper, Fall 2000.

Tata Consulting Services (TCS), Nariman Point, Mumbia, India supported this case study and helped in data collection on alternative component designs.

Padmal Vitharana (padmal@syr.edu) is an assistant professor of information systems in the School of Management at Syracuse University.

Fatemah "Mariam" Zahedi (zahedi@uwm.edu) is a professor of information systems in the School of Business Administration, University Wisconsin-Milwaukee.

Hemant Jain (jain@uwm.edu) is a professor in the School of Business Administration, University Wisconsin-Milwaukee.