December 1999

Automating Reuse for Systems Design

Tae-Dong Han
Georgia State University

Automating Reuse for Systems Design

Tae-Dong Han, Georgia State University, than@gsu.edu

Abstract

Reuse is as an important approach to conceptual object-oriented design. A number of reusable artifacts and methodologies to use these artifacts have been developed that require the designer to select to a certain level of granularity and a certain paradigm. This makes retrieval and application of these artifacts difficult and prevents the simultaneous reuse of artifacts at different levels of granularity. The purpose of this research, therefore, is to develop an actionable approach to lowering barriers to reuse. The approach is materialized in automating the conceptual design stage of the systems development process by reusing a new kind of design artifacts, which we call *design fragments*, which are synthesized with analysis patterns. The goal of the study includes the development of machine learning algorithms generating reusable design fragments and effectively storing/retrieving them.

Introduction

As the frequency of product and service innovations has increased dramatically, the corresponding need for new software systems has grown. The software design process is under considerable pressure to support these increased demands. Although software productivity has steadily increased over the past 30 years, the gains have not been sufficient to close the gap between demands placed on the software industry and what the state of practice can deliver (Gibbs 1994). Several decades of research have confirmed that reuse is an important approach to bridging this gap (Krueger 1992; Mili et al. 1995). Reuse involves the design of new systems from prefabricated reusable artifacts or higher level specifications (Setliff 1993).

Many research and industry efforts have focused on creating reusable artifacts such as class libraries (Sirkin et al. 1993), components (Szyperski 1998), and framework (Fayad and Schmidt 1997). However, realizing the benefits of reuse requires more than the development of reusable artifacts. It needs the creation of tools, techniques and approaches to facilitate design and construction of new systems with reuse (Mili et al. 1995). The latter is important because every instance of reuse has an overhead cost consisting of time and effort spent in: (a) searching for appropriate reusable artifacts; (b) applying them to the problem at hand; and (c) integrating the reused artifact with other parts of the solution. Unless these barriers are lowered, realizing the benefits of reuse will remain as an illusive goal (Barnes and Bollinger 1991).

The objective of this research, therefore, is to develop an actionable approach to lowering barriers to reuse. The approach is materialized in automating the conceptual design stage of the systems development process by reusing a new kind of design artifacts, which we call *design fragments*. Design fragments are synthesized with analysis patterns in the study. It is proposed that automation of reuse process using design fragments can reduce designers' time and effort in searching for appropriate reusable artifacts, in modifying them to fit to the problem at hand, and in generating the integrated solution. The goal of the study includes the development of machine learning algorithms generating reusable design fragments and effectively storing/retrieving them.

Use of Analysis Pattern

In general, there are two approaches to reuse: component-based and model-based. The component-based approach involves assembly of existing components to create new applications. The model-based approach involves adapting standard, generic domain models to create new applications. Both approaches have advantages and disadvantages. Components are difficult to understand and search for, but do not require a great deal of modification. Models are easier to understand, but require adaptation to and instantiation in a specific environment. The model-based approach reuses a model itself at a high level of abstraction, not at the individual object level. The component-based approach has been

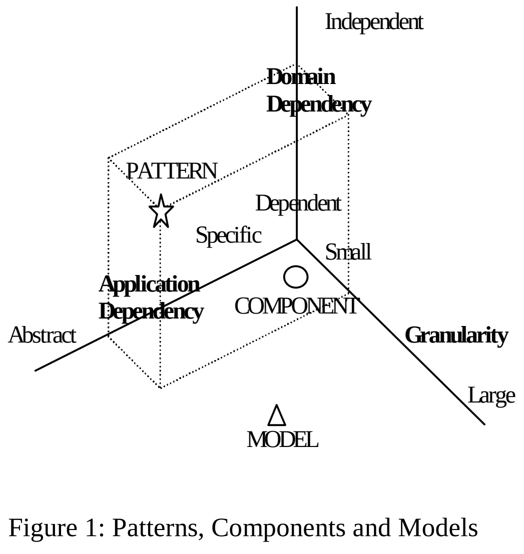

Figure 1: Patterns, Components and Models

adopted already to the actual system development at the implementation phase.

Both, however, are fairly domain dependent. In order to leverage the advantages of both, and mitigate the disadvantages, reuse can be attempted at a level that (a) is higher in granularity than individual components, (b) does not require adoption of the complete domain model (application independent), and (c) allows reuse across domains (domain independent) as seen in Figure 1. At this level, the reusable artifact can be identified as patterns (Coad 1995; Gamma et al. 1995; Buschmann et al. 1996). Patterns are a group of generic objects with stereotypical responsibilities and interactions. They provide a possible path to implementation - i.e., mapping to components. They can also be assembled to generate domain models.

However, the current approach of using patterns does not free designers from understanding patterns, selecting one or more of them, and modifying them for the proposed system. Although a pattern determines the basic structure of the solution to a particular design problem, it does not specify a fully detailed solution. A pattern provides a scheme for a generic solution to a family of problems, rather than a prefabricated module that can be used ‘as is’. Designer must implement this scheme according to the specific needs of the design problem at hand. A pattern helps with the creation of similar units. These units can be alike in their broad structure, but are frequently quite different in their detailed appearance.

In this study, we use analysis patterns (Coad 1995; Fowler 1997) to bridge model-based and component-based reuse. This study constructs a design base in which each design is generated by synthesizing analysis patterns with the help of automated reasoning and learning heuristics. During this process, *design fragments* are identified. Design fragments are the common sets of analysis patterns shared among different sets of designs, and can be served as starting points for the development of the future designs. Since design fragments are collections of analysis patterns, they provide higher granularity than analysis patterns, which means easier reuse. Designs in the design base are clustered/indexed based on design fragments. Since these indexed designs can be directly suggested to the designer, our approach can reduce the designer’s burden of finding an appropriate design(s) and provide immediate solution for the new system without much modification once an appropriate design is found.

Research Design and Methodology

This study will develop a prototype system that addresses the objective of the research. Architecture of the proposed system is shown in Figure 2. The research design is composed of three stages. The first stage is to

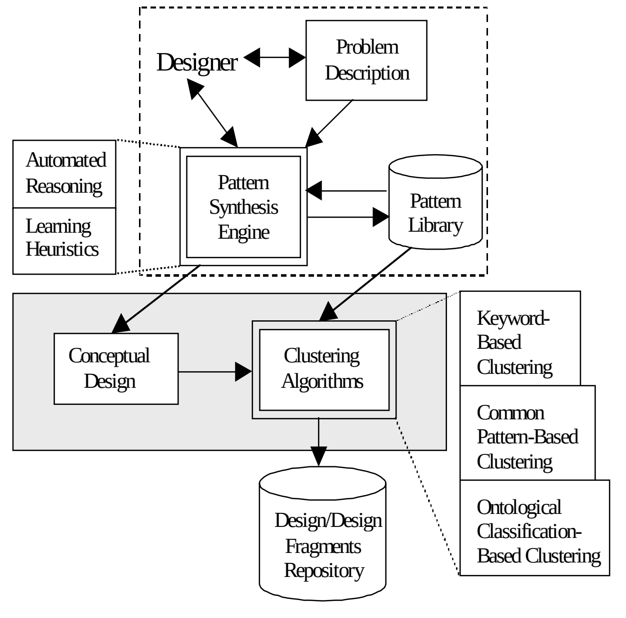

Figure 2: Proposed Architecture

generate reusable designs from the requirements statements (area enclosed in the dotted line). The approach automates this process as retrieval, instantiation and synthesis of analysis patterns from pattern libraries. The result is a synthesized preliminary design. Analysis patterns are identified from the requirement statements through simple natural language process. The pattern synthesis process is aided by heuristic machine learning algorithms. Large part of this stage has been already done through several studies (Purao, Storey, and Han 1998; Purao and Storey 1997a, 1997b). Requirement statements are collected from two sources: students enrolled in the graduate program of Information Systems at Georgia State University, and the designers via the Internet in the future. Students and system designers on the Internet are instructed to create a short statement of requirements for a selected domain. The amount of student data is about 160 statements from about 7 systems development related classes.

The second stage is to identify design fragments and to build a repository that stores them and the conceptual designs generated at the first stage. The designs will be indexed and clustered using three clustering algorithms that will be developed in the study: keyword-based clustering, common pattern-based clustering, and ontological classification-based clustering. Document classification using word frequency and weighting has been studied in Chen et al. (1994) and Salton (1989). The designs are clustered by keyword frequency and weighting in the requirements statements. For this process, this study also uses mechanisms analogous to conceptual clustering (Michalski and Stepp 1983). The designs are also clustered based on commonalties in patterns used among the designs, which are, in turn, *design fragments*. Finally, we adopt the ontology of Storey et al. (1997, 1998) to understand the semantics of an object in a pattern. Designs are clustered based on ontological classifications of instantiations of objects.

Finally, the third stage is to validate the effectiveness of the learning/clustering algorithms developed at the previous stages and of the proposed approach of this study. This process will be performed through the hypotheses testing for each stage. Hypotheses are as follows:

Hypothesis 1a: The natural language process is as effective as human analysts in identifying analysis patterns.

Hypothesis 1b: The design automation by synthesizing analysis patterns is as effective as human designer.

Hypothesis 2a: Designers take less time to find an appropriate design for the new system from the design / design fragments repository than designing from scratch or other types of reusable artifacts.

Hypothesis 2b: Designers take less time to integrate the design / design fragment found from the repository into the new system than modifying other types of reusable artifacts.

References

1. Barnes, B. and Bollinger, T. "Making Reuse Cost-Effective," *IEEE Software*, January 1991, pp. 13-24.
2. Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., and Stal, M. *A System of Patterns: Pattern-Oriented Software Architecture*, Wiley, 1996.
3. Chen, H., Hsu, P., Orwig, R., Hoopes, I., and Nunamaker, J. "Automatic concept classification of text from electronic meetings," *Communications of the ACM*, (37:10), October 1994, pp. 56-73.
4. Coad, P., *Object Models: Strategies, Patterns, and Applications*, Prentice Hall, 1995.
5. Fayad, M. and Schmidt, D. "Object-Oriented Application Frameworks," *Communications of the ACM* (40:10), October 1997, pp. 32-38.
6. Fowler, M., *Analysis Patterns: Reusable Object Models*, Addison-Wesley, 1997.
7. Gamma, E., Helm, R., Johnson, R., and Vlissides, J. *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1995.
8. Gibbs, C. W. "Software's Chronic Crisis," *Scientific American*, September 1994, pp. 86-95.
9. Krueger, C., "Software Reuse," *ACM Computing Surveys* (24:2), June 1992, pp. 131-184.
10. Michalski, R. and Stepp, R. "Learning from observation: Conceptual clustering," In *Machine Learning, An Artificial Intelligence Approach*, Michalski, R., Carbonell, J., and Mitchell, T. (Ed), Morgan Kaufmann, 1983, pp. 331-364.
11. Mili, H. et al., "Reusing Software: Issues and Research Directions," *IEEE Transactions on Software Engineering*, June 1995, pp. 528-562.
12. Purao, S. and V. Storey, "Intelligent Support for Retrieval and Synthesis of Patterns for Object-Oriented Design," *Proceedings of 16th International Conference on Conceptual Modeling-ER'97*, Los Angeles, CA., November 2-7, 1997a.
13. Purao, S. and V. Storey, "APSARA: A Web-based Tool to Automate System Design via Intelligent Pattern Retrieval and Synthesis," *Proceedings of the 7th Workshop on Information Technologies & Systems (WITS 97)*, Atlanta, GA, December 1997b, pp. 180-189.
14. Purao, S., Storey, V., and Han, T. "Improving Reuse-based System Design with Learning," *Working Paper*, Georgia State University, 1998.
15. Salton, G., *Automatic text processing*, Addison-Wesley, Mass., 1989.
16. Setliff, D. et al. "Practical Software Synthesis," *IEEE Software*, May 1993, pp. 6-10.
17. Sirkin, M., Batory, D., and Singhal, V. "Software Components in a Data Structure Precompiler," *Proceedings of the 15th International Conference on Software Engineering*, Baltimore, MD, May 17-21, 1993, pp. 437-446.
18. Szyperski, C., *Component Software: Beyond Object-Oriented Programming*, Addison-Wesley, 1998.
19. Storey, V., Dey, D., Ullrich, H., and Sundaresan, S., "An Ontology-Based Expert System for Database Design," *Data and Knowledge Engineering*, 1998.
20. Storey, V., Ullrich, H., and Sundaresan, S., "An Ontology to Support Automated Database Design," *Proceedings of the 16th International Conference on Conceptual Modeling (ER'97)*, Los Angeles, 3-6, November 1997, pp.2-16.