

Generative Composition of Distributed and Heterogeneous Components^{*}

Wei Zhao Barrett R. Bryant
Computer and Information Sciences
University of Alabama at Birmingham,
Birmingham, AL 35294-1170, U.S.A.
{zhaow,bryant}@cis.uab.edu

Mikhail Auguston
Computer Science
New Mexico State University
Las Cruces, NM 88003, U.S.A.
mikau@cis.nmsu.edu

Rajeev R. Rajee Andrew M. Olson
Computer and Information Science
Indiana University Purdue University Indianapolis
Indianapolis, IN 46202, U.S.A.
{rraje, aolson}@cs.iupui.edu

Carol C. Burt
Computer and Information Sciences
University of Alabama at Birmingham
Birmingham, AL 35294-1170, U.S.A.
cburt@cis.uab.edu

“A component is a unit of composition with contractually specified interfaces and explicit context dependencies only” [8]. Components by their nature can be deployed independently and are subject to composition by third parties. Large-scale, distributed systems have inherent complexity, high frequency of change, and differences among the component model, business model, level of requirements, location, programming language, operating system, etc. “Component systems adhere to the principle of divide and conquer for managing complexity of distributed computing” [3].

The first imperative objective of component-based systems is to realize “business-to-business electronic commerce” [3] by managing rapid changing of systems. For that reason, a well-defined architecture and a formal component specification are needed so that components can be easily replaced and re-assembled. In our research, UMM (Unified Meta-component Model) [7] is the desired architecture represented internally by TLG (Two Level Grammar), an executable formal specification language [2]. Based on this architecture, the interoperability and standard environment can be built among distributed heterogeneous components regardless of the aforementioned various differences in distributed component systems. For the component developer, the

UMM may look like a registration form or a graphic representation for this component; for the end system assembler, the UMM specification assures the components selected are of the right form for plugging in. The main parts of this specification include the component interface, QoS (Quality of Service) parameters [1], and the generation rules for the assembly and installation helper files.

In the crusade of pursuing the dream of reusability of Commercial Off-the-shelf (COTS) components to realize the large gain of productivity in the ultimate component market, we adopted the GDM (Generative Domain Model) [4] in our framework. By “domain,” we focus on the domains inside the infrastructure as opposed to the domains of business areas for our immediate goal is to build the interoperability among components adhering to different models as opposed to integrating the business components; the latter is higher level than the first one and is more likely the task of the business application system assembler. UMM has established a context to apply GDM, and TLG is used to specify GDM. Our approach can be outlined as below:

- Every component has a UMM associated with it [7]. Configuration knowledge, feature modeling of domain and generation rules for helper files needed in component assembly are embedded in UMM and can be extracted out to form the GDM (denoted in Figure 1 as the big cube in the middle of the figure) for the new system. In Figure 1 fragments of specifications of two components forming GDM are shown as jigsaw pieces, from which an instance of system configuration can be generated automatically. For example, a particular configuration for two

^{*} This research is supported by the U. S. Office of Naval Research under the award number N00014-01-1-0746.

components of client-server category could be: a proxy client for a server component and a server proxy for a client component (denoted in Figure 1 as two exploding stars), and a bridge driver (denoted as a well matched jigsaw puzzle plate). The two proxies are model specific access points and the bridge driver establishes the common context between two proxies and thus manages the session of the two components. Upon the success of the generation of configuration for the system, this new system consisting of component A and component B is assembled.

- The implementation of manufacturing the software members out of the GDM is powered by TLG for a TLG specification file can be turned into an implementation program by first converting it to VDM++ (an object-oriented extension of the Vienna Development Method [6]), and then it may be converted into an executable program written in Java or C++ by taking the advantage of VDM++ tool support [5].

References:

[1] Brahmamath, G. J., Raje, R. R., Olson, A. M., Auguston, M., Bryant, B. R., and Burt, C. C., "A Quality of Service Catalog for Software Components," To appear in *Proceedings of (SE)² 2002, the Southeastern Software Engineering Conference*, 2002.

[2] Bryant, B. R., Lee, B.-S., "Two-Level Grammar as an Object-Oriented Requirements Specification Language," *Proceedings of the 35th Hawaii International Conference on System Sciences*, 2002.

[3] Cheesman, J., Daniels, J. *UML Components: A Simple Process for Specifying Component-Based Software*. Addison-Wesley, 2000.

[4] Czarnecki, K., Eisenecker, U. W. *Generative Programming: Methods, Tools, and Applications*. Addison-Wesley, 2000.

[5] IFAD, The VDM++ Toolbox User Manual, <http://www.ifad.dk>, 2000.

[6] Larsen, P. G., et al., *Vienna Development Method - Specification Language - Part I: Base Language*, ISO/IEC 13817-1, December 1996.

[7] Raje, R. R., Auguston, M., Bryant, B. R., Olson, A. M., and Burt, C. C., "A Unified Approach for the Integration of Distributed Heterogeneous Software Components," *Proceedings of the 2001 Monterey Workshop on Engineering Automation for Software Intensive System Integration*, 2001, pp. 109-119.

[8] Workshop for Component-Oriented Programming. <http://www.research.microsoft.com/~cszypers/Events/WCOP2002/>

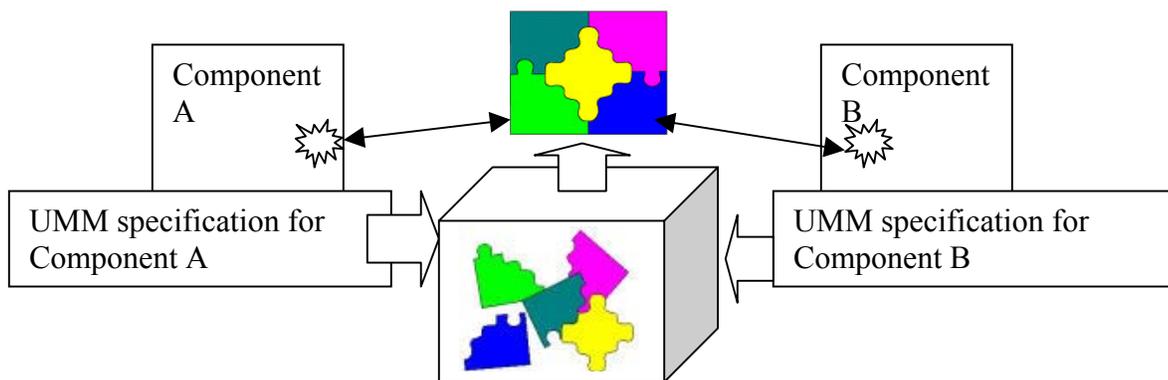


Figure 1. System Structure