

A Unified Approach to Component Assembly Based on Generative Programming*

Wei Zhao¹ Barrett R. Bryant¹ Rajeev R. Raje² Mikhail Auguston³ Andrew M. Olson² Carol C. Burt¹

The UniFrame project consists of a unified meta-component model (UMM) for distributed component-based systems (DCS), and a Unified Approach (UA) for integrating components [7]. The core parts of the UMM are: *components*, *service and service guarantees*, and *infrastructure*. A creation of a software solution for a DCS using UA comprises of two levels: a) the component level -- developers create components, test and validate the appropriate functional and non-functional (Quality of Service -- QoS) features and deploy the components on the network, and b) the system level -- a collection of components, each with a specific functionality and QoS, are obtained from the network, and a semi-automatic generation of a software solution for a particular DCS is achieved.

It is assumed that different developers will provide on a network a variety of possibly heterogeneous components for specific problem domains. For a specific problem, a search process will identify relevant components from those available on the network. Once these are identified, the task is to integrate these disparate components in a specific solution for the DCS under construction. The UA assumes that the generation environment is built around a generative domain-specific model (GDM) [4, 5] supporting component-based system assembly. The distinctive features of UA are:

- The developer of a distributed application presents to the UA-based system a query, describing the required characteristics, in a structured form of a natural language. The query is processed using the domain knowledge (such as key concepts from a domain) and a knowledge-base containing the UMM descriptions of the components for that domain. The domain knowledge and the knowledge-base are parts of the GDM. From this query a set of search parameters is generated which guides “headhunters” to perform a component search of the networked environment.. Headhunters are special components responsible for locating components deployed by the developers for the specific domain under consideration [8].
- The headhunters discover a set of applicable components that satisfy the functional and QoS requirements as indicated by the developer of the distributed system. The developer expresses the QoS requirements by selecting an appropriate set of parameters from a catalog of parameters [2]. After the components are fetched, the distributed application is assembled according to the generation rules embedded in the GDM. This assembly requires the creation of glue/wrapper interface between various components. Two-Level Grammar (TLG) [3] is used to specify the generative rules and provides the formal framework for the generation of the glue/wrapper code. This is implemented according to the process of translating TLG specifications into executable code as described in [6].
- QoS parameters are divided into two categories: a) static and b) dynamic. Static QoS parameters (e.g. dependability) are processed during the generation. Dynamic QoS parameters (e.g., response time) result in the instrumentation of generated target code based on event grammars [1], which at run time produce the corresponding QoS dynamic metrics, to be measured and validated.

* This material is based upon work supported by, or in part by, the U. S. Army Research Laboratory and the U. S. Army Research Office under contract/grant numbers DAAD19-00-1-0350 and 40473-MA, and by the U. S. Office of Naval Research under award number N00014-01-1-0746.

¹ Department of Computer and Information Sciences, University of Alabama at Birmingham, Birmingham, AL 35294-1170, U. S. A., {zhaow, bryant, cburt}@cis.uab.edu.

² Department of Computer and Information Science, Indiana University Purdue University Indianapolis, Indianapolis, IN 46202, U. S. A., {rraje, aolson}@cs.iupui.edu.

³ Computer Science Department, New Mexico State University, Las Cruces, NM 88003, U. S. A., mikau@cs.nmsu.edu.

QoS parameters given in the query provide a special dimension to the generated code - the instrumentation necessary for the run-time QoS metrics evaluation. Based on the query, the user has to come up with a representative set of test cases. Next the implementation is tested using the set of test cases to verify that it meets the desired QoS criteria. If it does not, it is discarded. After that, another implementation is chosen from the component collection. This process is repeated until an optimal (with respect to the QoS) implementation is found, or until the collection is exhausted. In the latter case, the process may request additional components or it may attempt to refine the query by adding more information about the desired solution from the problem domain. If a satisfactory implementation is found, it is ready for deployment.

The case study for the generative approach is a bank account management system. It is assumed that different client and server components for a bank domain are available on the network. These components (belonging to a category, i.e., server/client) do offer the same functionality but different QoS features. After requesting the construction of this system, assume that the headhunters found the following components: **JavaAccountClient**, **JavaAccountServer**, and **CorbaAccountServer**. The first two adhere to the Java-RMI model and the third one is developed with CORBA technology. The UMM specifications associated with the components indicate that the two server components have the same functionality but **CorbaAccountServer** has better service guarantees and meets the QoS specified in the system query, thus the final system should be assembled from the Java client and the CORBA server. Based on the generation rules embedded in GDM, a proxy server for the Java client component, a proxy client for the CORBA server component, a bridge driver between the two proxies and some other installation helper files can be generated to form an integrated account system. The assembled system will be deployed if it meets the desired QoS criteria. If the system succeeds then a new set of UMM specifications will be generated for the integrated system to insure that it is available for the discovery by other head-hunters, i.e., to act as a component of other possible application systems.

References:

- [1] M. Auguston, A. Gates, M. Lujan, *Defining a Program Behavior Model for Dynamic Analyzers*, Proc. SEKE '97, 9th Int. Conf. Software Eng. Knowledge Eng., 1997, pp. 257-262.
- [2] G. J. Brahmamath, R. R. Raje, A. M. Olson, M. Auguston, B. R. Bryant, C. C. Burt, *A Quality of Service Catalog for Software Components*. Proc. Southeastern Software Eng. Conf. (to appear), 2002.
- [3] B. R. Bryant, B.-S. Lee, *Two-Level Grammar as an Object-Oriented Requirements Specification Language*. Proc. 35th Hawaii Int. Conf. System Sciences, 2002.
- [4] J. C. Cleaveland, *Program Generators with XML and Java*, Prentice-Hall, 2001.
- [5] K. Czarnecki, U. W. Eisenecker, *Generative Programming: Methods, Tools, and Applications*. Addison-Wesley, 2000.
- [6] B.-S. Lee, B. R. Bryant, *Automated Conversion from Requirements Documentation to an Object-Oriented Formal Specification Language*. Proc. ACM Symp. Applied Computing (to appear), 2002.
- [7] R. R. Raje, M. Auguston, B. R. Bryant, A. M. Olson, C. C. Burt, *A Unified Approach for the Integration of Distributed Heterogeneous Software Components*. Proc. Monterey Workshop Engineering Automation for Software Intensive System Integration, 2001, pp.109-119.
- [8] N. N. Siram, R. R. Raje, B. R. Bryant, A. M. Olson, M. Auguston, C. C. Burt, *An Architecture for the UniFrame Resource Discovery Service*. Submitted for publication, 2002.