Composition and Decomposition of Quality of Service Parameters in Distributed Component-Based Systems

Changlin Sun, Rajeev R. Raje,
Andrew M. Olson
Computer and Information Science
Indiana University Purdue University Indianapolis
723 W. Michigan Street, SL 280
Indianapolis, IN 46202, USA
{csun, rraje, aolson, csun}@cs.iupui.edu,
+1 317 274 5246/5174/9733

Mikhail Auguston
Department of Computer Science
New Mexico State University
Las Cruces, NM 88003, USA
mikau@cs.nmsu.edu
+1 505 646 5286

Barrett R. Bryant, Carol Burt
Computer and Information Sciences
The University of Alabama at Birmingham
1300 University Blvd.
Birmingham, Alabama 35294-1170, USA
{bryant, cburt}@cis.uab.edu, +1 205 934 2213

Zhisheng Huang
Computer and Information Science
Indiana University Purdue University Indianapolis
723 W. Michigan Street, SL 280
Indianapolis, IN 46202, USA
zhuang@cs.iupui.edu
+1 317 274 5246

Abstract

It is becoming increasingly acceptable that the component-based development is an effective, efficient and promising approach to develop distributed systems. With components as the building blcks, it is expected that the quality of the end system can be predicted based on the qualities of components in the system. UniFrame is one such framework that facilitates a seamless interoperation of heterogeneous distributed software components. As a part of UniFrame, a catalog of quality of service (QoS) parameters has been created to provide a standard method for quantifying the QoS of software components. In this paper, an approach for composition and decomposition of these QoS parameters is proposed. A case study from the financial domain is indicated to validate this model.

***Keywords**: Distributed systems, software components, quality of service (QoS), composition, decomposition*

1. Introduction

The development of distributed systems from reusable components is becoming increasing important because of its potential to reduce product development cost and time-to-market. Unfortunately, the current component-based approaches concentrate mainly on functional properties, and ignore the quality of service (non-functional) properties, which are crucial in many application domains. A few examples of quality of service parameters are: *dependability, reliability, availability, maintainability, adaptability, portability, evolvability, achievability, security, presentation, throughput, result, and turnaround time* [3]. In a component-based approach, it is relatively easy to glue components together to provide the desired system functionality, but it is difficult to guarantee the quality of service provided by a system made up of individual components. Hence, it is critical to determine the distribution of a system property into its component properties (decomposition) and how to reason the system property from the property of its individual components (composition). Currently, there is no common and accepted design standard that can facilitate such composition and decomposition.

In [1, 2] a framework, UniFrame, based on a unified meta-component model (UMM) and a unified approach (UA), is proposed for building distributed component-based systems. In UMM, components are autonomous entities that provide services and guarantee their quality. The creation of a distributed software system using UA has two levels: a) component level - developers create components, test and validate the appropriate QoS and deploy the components on the network, and b) system level – a collection of components, each with a specific functionality and QoS, enables a semi-automatic generation of a distributed software system. The focus of this paper is to study the mechanism of decomposition and composition of various QoS parameters so that the properties of the entire system can be inferred from the QoS properties of individual parameters and vice versa. The first step towards composition and decomposition is to identify and classify various QoS parameters.

2. Classification of QoS parameters

In [3], sixteen QoS parameters are identified and described in a catalog. The aim of this section is to study these parameters from the perspective of composition and decomposition, and classify them into different categories. Such a classification provides the developer of distributed systems the knowledge about how these QoS parameters should be treated during the creation of a software realization of a distributed system.

2.1 Static and dynamic QoS parameters

Static QoS parameters can be evaluated by examining the internal structure of a software component. These parameters are stable in different environments provided the internal structure of component is unchanged. The examples of static QoS parameters are *reliability, maintainability, portability, scalability, reusability, presentation, usability, security, priority, and parallelism constraints*. Dynamic QoS parameters, on the other hand, can be measured by observing the system behavior at run-time. These parameters are tightly associated with the deployment environment. Examples of dynamic parameters are *throughput, turnaround time, capacity, availability, and result.*

Static QoS parameters may compose well as they do not tend to change during system execution. However, the execution environment, which is not known in advance, influences dynamic QoS parameters and makes their composition a difficult task.

2.2 Application dependent and independent QoS parameters

Different application domains require the use of different QoS parameters. For example, in the E-commerce applications, availability, turnaround time, throughput and usability are important, while in the visualization applications, the frame rate is critcal. Some parameters are application dependent (e.g., throughput), while some others are application independent (e.g., reusability). Obviously, the application independent parameters are more convenient to deal with than the application dependent parameters, because the latter need application-specific information.

2.3 Parameters with different ranges of decomposition

The dependence of a system level property on the component level property leads to several special decompositions of QoS parameters: *universal, subset, existential* and *component-specific* decomposition. For universal decomposition of QoS parameters, the system level property decomposes into all of the components in the system. Most of the QoS parameters have universal decomposition, such as, availability, reliability, security, etc. For subset decomposition of QoS parameters, the system level property decomposes into a subset of components in the system. For existential decomposition of QoS parameters, the system level property decomposes into any component in the system. Mobility is an example of QoS parameters with existential decomposition. For component-specific decomposition of QoS parameters, the system level property decomposes into a particular component. For example, presentation of a system is decomposed into the presentation provided by the user-interface component of the system.

2.4 Parameters with different aggregation rules

In the physical world, some properties show different aggregation rules. For example, the mass or the energy of a system is the sum of the mass or the energy of subsystems. The density or the temperature of a system is the average of the density or the temperature of subsystems. The strength of a system is limited by the strength of the subsystem with the minimum value of strength. Similarly, for systems built from software components, different QoS parameters may abide by different composition rules. For example, the turn-around time of a system is the sum of the turn-around time of each component in the system. The maintainability of a system is an average of the maintainability of each component in the system. The security of a system is limited by the component with the minimum value of security.

3. System decomposition and composition models

In this section, a decomposition and composition model of QoS parameters is proposed. The model includes the decomposition process, the composition process, and the corresponding rules.

3.1 Decomposition rules

The decomposition process factorizes the system QoS parameters to QoS parameters of components and provides a rough estimate of the values for the QoS parameters of individual components. To decompose the QoS parameters of systems, we identify following properties*: at-least-one property*, *universal property*, *subset property*, *at most one property*, *at-least-one-X property*, *universal-X property*, *subset-X property* and *at-most-one-X property*.

A property X is an at-least-one property if, when any system has property X, at least one component of that system has property X. A property X is a universal property if, when any system has property X, all components of that system have property X. A property X is a subset property if, when any system has property X, a subset of components of that system have property X. A property X is an at-most-one property if, when any system has property X, at most one component of that system has property X. A property Y is an at-least-one-X property if, when any system has property Y, at least one component of that system has a certain property called X. A property Y is a universal-X property if, when any system has property Y, all components of that system have a certain property called X. A property Y is a subset-X property if, when any system has property Y, a subset of components of that system have a certain property called X. A property Y is an at-most-one-X property if, when any system has property Y, at most one component of that system has a certain property called X.

Each QoS parameter needs to be classified using one of these properties. For example, from the decomposition point of view, mobility is an at-least-one property, security is a universal property, and frame rate is a universal-X property, where X is throughput. Based on the definitions of these properties, the QoS parameters for individual components can be classified into one of these properties. Given the value of system-level QoS parameters, an upper or lower bound of the value of the QoS parameters of an individual component can be estimated. For example, for turn-around time, the component turn-around time TAT_i (i=1, 2, …, n) need to satisfy $0 < TAT_i < TAT$, where TAT is the system turn-around time, while for security, the component security S_i (i=1, 2, …, n) need to satisfy $S_i > S$, where S is the system security requirement.

3.2 Composition rules

During the composition process, the system QoS parameter is reasoned from the QoS parameters of components. Due to the causal link between the property of the system and the properties of components in the system, we assume the property of composed system depends on the properties of components in the system. The proposed equation for composition can be written as:

$$P = f(p_1, p_2, \ldots, p_n) \qquad (1)$$

where P is the property of composed system, and p_i (i=1,2, …,n) is the property of component i in the system.

The equation (1) can be approximated as a weighted sum, as indicated below:

$$P = w_1 * p_1 + w_2 * p_2 + \ldots, + w_n * p_n \qquad (2)$$

Where w_i (i=1, 2, …, n) is a constant coefficient (weight), for the component I, within the range [0, 1]. The determination of w_i is based both on the analysis and experimentation.

For each QoS parameter, the equation (2) can be simplified. For example, in the case of maintainability, equation (2) becomes:

$$P = w_1 * p_1 + w_2 * p_2 + \ldots, + w_n * p_n \qquad (3)$$

where $w_i = \frac{LOC_i}{\sum_{j=1}^{n} LOC_j}$, LOC=Lines of Code.

For QoS parameters: security, adaptability, capacity, equation (2) becomes:

$$P = Min(p_1, p_2, \ldots, p_n) \qquad (4)$$

For the QoS parameter turn-around time, equation (2) becomes:

$$P = p_1 + p_2 +, \ldots, + p_n \qquad (5)$$

Theoretically, for each QoS parameter, a corresponding composition rule can be derived from equation (2) based on analysis and experimentation.

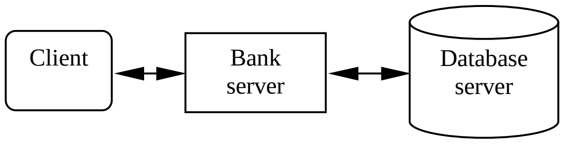

Figure 1. Bank account system architecture

4. A case study

To illustrate the composition and decomposition of QoS parameters in developing distributed component-based system, a simple bank account system (financial system) is discussed below. As shown in Figure 1, the system consists of three components: client, bank server, and database server. For this bank account system, the following QoS parameters, based on the functionality, are identified: availability, turnaround time, security, throughput, reliability, and usability. As turn-around time is an important dynamic QoS parameter for most applications, its composition rule is validated through experiments as indicated below.

The experimental system, as shown in Figure 2, consists of three Ultra-250, SPARC Sun workstations. The workstations Phoenix and Magellan are connected using a 100Mbit Ethernet and the workstation Raleigh is connected via a 10Mbit Ethernet. Phoenix is a file server for the local area network.

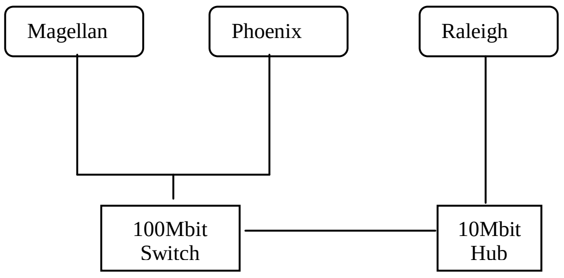

Figure 2. Experimental setup

The three components in the bank account system are implemented using Java RMI. For the purpose of the validation, each component has an instrumented code to measure the dynamic QoS parameters during the execution time.

Initially, the turnaround time for each component is measured by running them (in isolation) on each of the three workstations multiple times. These experiments yielded the following average turnaround times for the three components under consideration: 34 ms (client component), 119 ms (bank server component) and 126 ms (database server component). Based on the composition rule for turn-around time, the predicted turn-around time for the entire system is the summation of the individual turnaround times, i.e., the turnaround time for the system is predicted to be 278 ms. To experimentally validate this predicated value, the three components were deployed using two distributed configurations: a) the client on Raleigh, bank server on Phoenix and database server ob Magellan, and b) the client on Raleigh and both the servers on Magellan. In both the cases, the system level turn-around time was measured. The error between the predicted turnaround time and the actual turnaround time, for both the configurations, was found to be of 3.3% and 3.1% respectively. Hence, it can be concluded from these simple experiments that the model presented here allows the prediction of values for the turnaround time with a good accuracy. Similar empirical studies for validating the composition rules for other parameters are being carried out. However, for the sake of brevity, and to adhere to the space constraints, these are not reported in this paper.

5. Conclusions

The UniFrame approach provides a framework for the development of distributed software systems based on components by highlighting not only the functional but also the QoS requirements. The QoS feature of a system can be predicted by applying the composition and decomposition rules to QoS attributes of the individual components. These rules are based on the classification of different QoS parameters. A simple case study presented here empirically validates the composition/decomposition rules described in this paper.

6. Acknowledgement

This material is based upon work supported by the US Office of Naval Research under award number N00014-01-1-0746.

References

[1] R. R. Raje, M. Auguston, B. R. Bryant, A. M. Olson and C. Burt. A Unified Approach for the Integration of Distributed Heterogeneous Software Components. Proceedings of the 2001 Monterey Workshop, Monterey, California, June 2001, pp. 109-119.

[2] R. Raje, M. Auguston, B. Bryant, A. Olson, C. Burt. A Quality of Service-based Framework for Creating Distributed Heterogeneous Software Components. Technical Report, Department of Computer and Information Science, Indiana University Purdue University Indianapolis, 2001.

[3] G. Brahnmath, R. Raje, A. Olson, M. Auguston, B.Bryant, C. Burt. A Quality of Service Catalog for Software Components. Proceedings of the Southeastern Software Engineering Conference, Huntsville, Alabama, April 2002.