Quality of Service (QoS) Standards for Model Driven Architecture[1]

Carol C. Burt[2], Barrett R. Bryant[2], Rajeev R. Raje[3], Andrew Olson[3], Mikhail Auguston[4]

Abstract

A number of middleware technologies have evolved over the last ten years to address specific business problems such as enabling process optimization via systems integration, rapid development of new applications, web enabling features for customers, and mechanization of supply chains. Software architects increasingly utilize models to represent different viewpoints of a business solution. Separation of concerns is a key characteristic of good software design. In an effort to facilitate the design of business systems in a platform independent matter, the Object Management Group (OMG) is currently progressing the Model Driven Architecture (MDA)[1]. Model Driven Architecture maintains a clean separation of Platform Independent Models (PIMs) that represent the domain from Platform Specific Models (PSMs) that expose details related to a middleware technology. In this way a single PIM can be mapped to multiple implementation technologies (such as OMG CORBA®, Sun J2EE, Microsoft COM+, and W3C Web Services). While all of the component-based technologies have established a concise means for describing functional contracts (the interface or services offered by a component in the architecture), none of these technologies have embraced architecture or a vocabulary for specifying non-functional Quality of Service (QoS) contracts. Non-functional contracts are necessary to analyze the ability of a service to meet QoS constraints when used in a composition. A component may, for example, guarantee a certain performance level (given a fixed set of constraints) or guarantee protection of features and/or information classified as a protected resource. The UniFrame [2] research (which includes identification and progression of requisite standards activities) envisions a plug and play component environment where QoS contracts are part of a component description and middleware bridges and quality of service instrumentation are generated by component integration toolkits. That is, business components that utilize diverse platform technologies may be easily integrated, and they will offer both functional and non-functional service contracts. A difficulty in progressing this work is the lack of a standard vocabulary for software component Quality of Service (QoS). Following this work, syntax for expressing QoS in component models and the mappings that form the transformations from diverse viewpoints of a model must also be explored and standardized. This research includes supporting and participating in the progression of such standards. This paper introduces a path for standards in the area of Quality of Service and discusses how this standardization would progress the goal of using commercial off the shelf (COTS) components in a heterogeneous system composition.

---

[2] Department of Computer and Information Sciences, The University of Alabama at Birmingham, Birmingham, AL 35294, USA, {cburt, bryant} @cis.uab.edu
[3] Department of Computer and Information Science, Indiana University Purdue University Indianapolis, Indianapolis, IN 46202, USA, {rraje, aolson} @cs.iupui.edu
[4] Computer Science Department, New Mexico State University, Las Cruces, NM 88003, USA, mikau@cs.nmsu.edu

Introduction

Enterprises are increasingly dependent upon multiple middleware technologies that enable new business paradigms by weaving together legacy systems with advanced technology. These technologies support core business functionality, enable distributed business systems, integrate business processes and allow companies to communicate with customers, suppliers, and business partners. While it is possible to construct such systems, it requires that the developer be aware of the nuances of the diverse middleware technologies. This problem must be resolved for the promise of software component technology (plug & play / off the shelf interoperability) can be fully realized. In addition, the increased complexity of this environment makes it impossible to predict the non-functional aspects of such a system until after it is constructed. That is, static QoS relies on the design expertise of software architects and engineers and metrics and test scenarios must be hand crafted on a case-by-case basis to determine if a composition is acceptable. As distributed systems become omni-present with many mission-critical, the notion of QoS-oriented software development will become essential. Such an approach is necessary to ensure the reliability and high confidence of distributed software systems.

The Unified Component Meta Model Framework (UniFrame) [2] research project is an attempt to unify the existing and emerging distributed component models under a common meta-model for the purpose of enabling the discovery, interoperability, and collaboration of components via generative software techniques. A great deal of work is underway in standards organizations such as OMG and World Wide Web Consortium (W3C) that provides the foundation for UniFrame. The UniFrame research builds upon work in standards organizations, targeting the dynamic assembly of distributed software systems from components built in different component models and the ability to express quality of service (QoS) requirements in such a way that generative design and implementations can utilize them. This is a necessity to enable timely (perhaps dynamic) assembly of e-business relationships (for example to select a replacement component when a primary component fails to deliver on QoS guarantees). It is also necessary in time and safety critical environments where failure to meet quality of service requirements results in significant system failures.

Related Work

Although QoS parameters and associated metrics have been widely used in networking, there is no standard vocabulary for discussing QoS as it relates to distributed computing and component based solutions. For example, the CORBA® Components Specification only uses the term "quality of service" with regard to events and whether or not they are transactional in nature [3]. The Java2 Enterprise Edition (J2EE) specification [4] states, "We expect J2EE products to vary widely and compete vigorously on various aspects of quality of service. Products will provide different levels of performance, scalability, robustness, availability, and security. In some cases this specification requires minimal levels of service. Future versions of this specification may allow applications to describe their requirements in these areas." Today, there is no standard vocabulary that we can utilize to define QoS requirements for any component technology.

In the fall of 2000, the OMG Analysis and Design Task Force considered a draft Request for Proposal (RFP) for a UML profile for Modeling Quality of Service as it relates to real-time systems [5]. This RFP called for a framework and categorization of QoS characteristics. In January 2002, the OMG Analysis and Design task force issued an RFP for a "UML™ Profile for

Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms" [6].   This RFP solicits proposals for a UML profile or Meta Object Facility (MOF) meta-model that defines standard paradigms of use in modeling quality of service and fault-tolerance aspects of systems. This is the first of a series of RFPs that have the goal of significant benefits to the UML user community engaged in high-quality robust system development.   The key mandatory requirements of this RFP are listed in Figure 1.

A General Quality of Service Framework

To ensure consistency in modeling various qualities of service, submissions shall define a standard framework or, reference model, for QoS modeling in the context of the UML. This shall include:

- A general categorization of different kinds of QoS; including QoS that are fixed at design time as well as ones that are managed dynamically

- Integration of different categories of QoS for the purpose of QoS modeling of system aspects.

- Identification of the basic conceptual elements involved in QoS and their mutual relationships. This shall include the ability to associate QoS characteristics to model elements (specification), a generic model of the system aspects involved in QoS-associated collaboration and their functional interactions and use cases (usage model), and a generic model of how QoS allocation and decomposition is managed.

- A coherent set of stereotypes, tagged values, and constraints as necessary to represent the identified QoS properties constructing a UML Profile.

A Definition of Individual QoS Characteristics

Submissions shall define QoS characteristics, particularly those important to real-time and high confidence systems, which describe the fundamental aspects of the various specific kinds of QoS based on the QoS categorization identified in the framework. These shall include but are not limited to the following:

- time-related characteristics (delays, freshness)

- importance-related characteristics (priority, precedence)

- capacity-related characteristics (throughput, capacity)

- integrity related characteristics (accuracy)

- fault tolerance characteristics (mean-time between failures, mean-time to repair, number of replicas)

A coherent set of stereotypes, tagged values, and constraints as necessary to represent the identified QoS properties constructing a UML Profile.

Figure 1: OMG RFP – UML Profile for QoS - Mandatory Requirements

As part of the UniFrame research, we have outlined our approach to a QoS-based framework for creating distributed heterogeneous software components [7]. The QoS-based method in UniFrame is made up of three steps:

1. The creation of a catalog for QoS parameters (and/or metrics),
2. A formal specification of these parameters, and
3. A mechanism for ensuring these parameters, both at each individual component level and at the entire system level

UniFrame leverages work by Zinky, Bakken & Schantz [8] with the goal of providing a catalog of QoS parameters, indicating how parameters might be described. There are many possible QoS parameters that a component (and its developer) can use to indicate the associated service. Some of these parameters may be general in nature, while others may be pertain to a specific domain. The goal of creating the QoS catalog is two fold: a) it assists the component developer (or the system integrator) in selecting the necessary QoS parameters for the component (or system) under construction, and b) it enables the developer (or integrator) to ensure the necessary QoS guarantees by integrating the selected QoS parameters into the assurance process.

We have recently published the initial version of our QoS catalog [9]. At present the following QoS parameters have been selected for inclusion in the catalog.

1. *Dependability:* a measure of confidence that the component is free from errors.
2. *Security:* a measure of the ability of the component to resist an intrusion.
3. *Adaptability:* a measure of the ability of the component to tolerate changes in resources and user requirements.
4. *Maintainability:* a measure of the ease with which a software system can be maintained.
5. *Portability:* a measure of the ease with which a component can be migrated to a new environment.
6. *Throughput:* indicates the efficiency or speed of a component.
7. *Capacity:* indicates the maximum number of concurrent requests a component can serve.
8. *Turn-around Time:* a measure of the time taken by the component to return the result.
9. *Parallelism Constraints:* indicates whether a component can support synchronous or asynchronous invocations.
10. *Availability:* indicates the duration when a component is available to offer a particular service.
11. *Ordering Constraints:* indicates the order of returned results and its significance.
12. *Evolvability:* indicates how easily a component can evolve over a span of time.
13. *Result:* indicates the quality of the results returned.
14. *Achievability:* indicates whether the component can provide a higher degree of service than promised.
15. *Priority:* indicates if a component is capable of providing prioritized service.
16. *Presentation:* indicates the quality of presentation of the results returned by the component.

Uniframe also leverages work on Quality Objects and adaptive middleware. Quality Objects [10] is a framework for providing QoS to software applications composed of objects distributed over wide area networks. QuO bridges the gap between socket-level QoS and distributed object level QoS, emphasizing specification, measuring, controlling, and adapting to changes in QoS. RAPIDware [11] is an approach to component-based development of adaptable and dependable

middleware. It uses rigorous software development methods to support interactive applications executed across heterogeneous networked environments.

Frolund & Koistinen [12] point out that deciding which quality of service properties should be provided by individual components is an important part of the design process. They define a Quality-of-Service specification language (QML) and they show how the Unified Modeling Language (UML) can be extended to support the concepts of QML. In addition, they suggest a technique for representation of QML constructs in terms of ISO IDL [12] [13]. Frolund and Koistinen were working with a platform specific model (CORBA). The OMG currently is progressing an RFP for a UML profile for Quality of Service that will provide the meta-model necessary to use UML to model QoS in a platform independent manner.
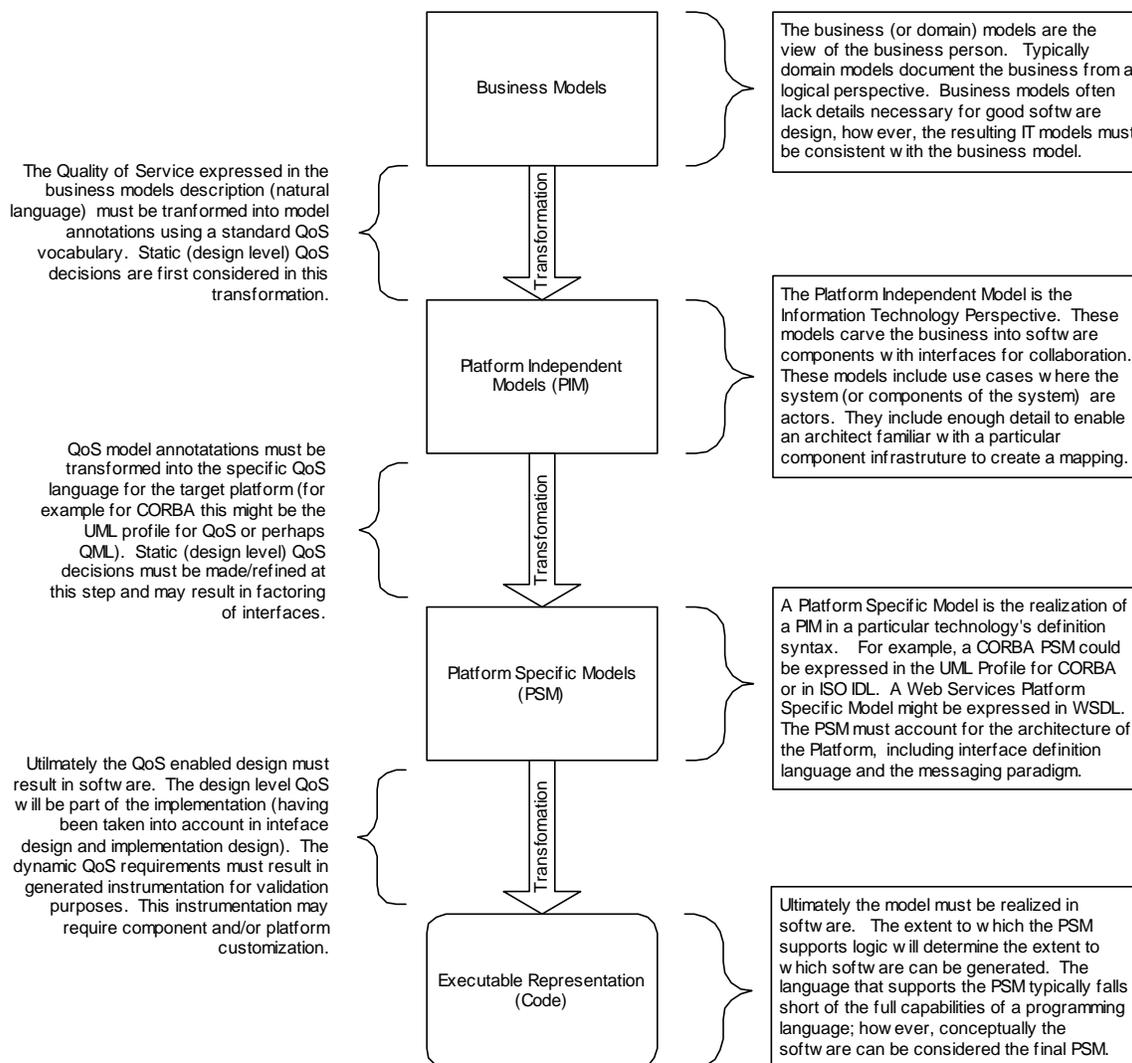
The OMG work will standardize how static (design related) and dynamic (environmentally influenced) QoS characteristics are expressed in UML models. We are working within the OMG community to introduce our efforts, contribute to the analysis of the submissions, and ensure that our research is aligned with industry standard vocabulary as we progress techniques that enable QoS-aware systems to utilize generative software tools. We will also be experimenting with alternative syntax for representation of QoS characteristics such as event grammar [14].

Model Driven Architecture with QoS parameters

In addition to the work that OMG has done with distributed computing interoperability (CORBA®/IIOP), the OMG has also progressed standards in the domain of modeling and meta-modeling: the Unified Modeling Language (UML™) and Meta-Object Facility (MOF™). Some of the analysis and design standards include the precise mappings that define the transformation of model information into interface definitions in ISO Interface Definition Language (IDL).

The latest initiative - Model Driven Architecture (MDA™) - is the way that the OMG will standardize Platform Independent Models (PIMs) that can be mapped to multiple Platform Specific Models such as CORBA®, Java2 Enterprise Edition (J2EE), and Web Services for implementation. This approach holds promise for the standardization of components that could be used in collaborative environments as a result of a common semantic model. To fully realize the potential of this approach, Quality of Service (QoS) catalogs, formal parameterization of Platform Specific Models, and ultimately instrumentation mappings must also be standardized within the Model Driven Architecture roadmap.

Figure 2 outlines the type of models that are common in a MDA approach. Quality of Service parameters must be introduced into each model and the transformations (or mappings) that occur as models are refined must be standardized. The current RFP is merely the beginning – providing a vocabulary and syntax for expressing QoS in UML. As we move beyond the QoS catalog, our research will focus on the constraints that are placed on transformations as a result of the quality requirements and the generative techniques for ensuring that metrics can be gathered.

The business (or domain) models are the view of the business person. Typically domain models document the business from a logical perspective. Business models often lack details necessary for good softw are design, how ever, the resulting IT models must be consistent w ith the business model.

Business Models

The Quality of Service expressed in the business models description (natural language) must be tranformed into model annotations using a standard QoS vocabulary. Static (design level) QoS decisions are first considered in this transformation.

Transformation

Platform Independent Models (PIM)

The Platform Independent Model is the Information Technology Perspective. These models carve the business into softw are components w ith interfaces for collaboration. These models include use cases w here the system (or components of the system) are actors. They include enough detail to enable an architect familiar w ith a particular component infrastruture to create a mapping.

QoS model annotatations must be transformed into the specific QoS language for the target platform (for example for CORBA this might be the UML profile for QoS or perhaps QML). Static (design level) QoS decisions must be made/refined at this step and may result in factoring of interfaces.

Transformation

Platform Specific Models (PSM)

A Platform Specific Model is the realization of a PIM in a particular technology's definition syntax. For example, a CORBA PSM could be expressed in the UML Profile for CORBA or in ISO IDL. A Web Services Platform Specific Model might be expressed in WSDL. The PSM must account for the architecture of the Platform, including interface definition language and the messaging paradigm.

Utilmately the QoS enabled design must result in softw are. The design level QoS w ill be part of the implementation (having been taken into account in inteface design and implementation design). The dynamic QoS requirements must result in generated instrumentation for validation purposes. This instrumentation may require component and/or platform customization.

Transformation

Executable Representation (Code)

Ultimately the model must be realized in softw are. The extent to w hich the PSM supports logic w ill determine the extent to w hich softw are can be generated. The language that supports the PSM typically falls short of the full capabilities of a programming language; how ever, conceptually the softw are can be considered the final PSM.

Figure 2 – QoS considerations during model transformations

Quality of Service Issues related to Interface Generation

It should be noted that there are existing standards for using generative techniques to create interfaces from UML models. For example, the OMG Meta Object Facility (MOF) [15] allows the generation of interfaces from Unified Modeling Language UML models. A careful analysis of the resulting interface specifications makes it clear, however, that distribution is not a key factor in the algorithms used. This has a direct impact on the ability to meet quality of service requirements in a distributed solution. For example, in a distributed system, quality of service requirements for performance, scalability and/or security would dictate the use of iterators, the factoring of interfaces into separate query and administrative operations and the use of structure and/or objects passed by value. The current standards in this area tend to focus on data access with accessors and mutators and relationship traversal. This is acceptable (perhaps even desirable) in a single machine environment, but unacceptable for highly distributed

communications and collaborations [1]. It has long been accepted that systems with distributed components require specialized design to ensure performance and ease of security administration. That is, there is a need to take QoS parameters and use-cases into account when designing component interfaces. For a model driven generative technique to be successful, models must be parameterized with QoS standard parameters as defined in a catalog. In addition, use case scenarios must also be formally expressed so that they can be used as input to an interface generator. That is, given a parameterized domain model, semantically equivalent interfaces (and the bridges between them) must be generated.

The OMG Architecture Board produced a paper that describes the technical details of the Model Driven Architecture (MDA) [1]. This document outlines several areas where significant research is required before the MDA vision can be fully realized. One of the most important areas is directly related to the UniFrame research. It states: "It is generally agreed that the MOF-IDL mapping is in need of upgrading. [Footnote: The problem is that the generated interfaces are not efficient in distributed systems. Firstly, the mapping predates CORBA valuetypes and thus does not make use of them. Secondly, a class with N attributes is always mapped to a CORBA interface with N separate getter/setter operations. In a distributed system one would want to group attributes based upon use cases, cache attribute values, or implement other optimizations to reduce the number of distributed calls]. Realistically we will probably have to accept the fact that for the foreseeable future, the automatically generated transformation from PIM to PSM will have to be enhanced by humans. As we gain more experience we will be able to define various patterns and allow them to be selected in some way."

This is recognition that a generated interface must be optimized using quality of service and usage scenarios requires research in techniques for integrating QoS into a generative programming model. It also recognizes that we do not currently have a way to express the quality of service requirements in such a way that generative techniques can be trusted during the design process.

Conclusion

The ability to provide QoS parameterization of models is recognized in the Object Management Group community and standards in this area will lead to the ability to generate Platform Specific Models that take quality of service characteristics into account. Since there has been very little work on progressing Quality of Service specifications for component based architectures, this work has the potential to impact how the Object Management Group (OMG) defines QoS parameterization for Model Driven Architecture and the ability to more clearly specify and measure component feasibility for a particular task. This standardization of QoS catalogs and parameters is a pre-requisite to benchmarking and service validation instrumentation. In addition, the Java Community Process (JCP) has a history of working with OMG to progress consistent standards. The expectation is that any Quality of Service parameters would be applicable for CORBA®, J2EE™, and Web Services component architectures. In addition, this standardization provides a foundation for future standards. Quality of Service characteristics must have syntax for expression in every artifact of the analysis, design and development process. The design patterns must be documented and exploited in such a way that generative techniques can be applied. In addition, formal specifications will allow instrumentation necessary for measuring quality of service to be come an integral part of middleware and component implementation frameworks.

The UniFrame research project is investigating techniques and patterns used when static QoS is a consideration during refinement of models and software design, and is utilizing emerging technology in generative programming for QoS instrumentation with a goal of progressing standards for QoS instrumentation when the technology matures. By ensuring that software components can be tested against standard Quality of Service feature sets we progress the goal of using more commercial off the shelf (COTS) components in heterogeneous system compositions.

References

[1] Object Management Group. 2001. *Model Driven Architecture: A Technical Perspective*. Technical Report. Document # ormsc/2001-07-01. Framingham, MA: Object Management Group. July 2001.

[2] Rajeev R. Raje, Barrett Bryant, Mikhail Auguston, Andrew Olson, Carol Burt. "*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, pp: 109-119, Monterey, California, 2001.

[3] Object Management Group. 2001. *CORBA 3.0 CORBA Component Model Chapters*. Document # ptc/2001-11-03. Framingham, MA: Object Management Group.

[4] Sun Microsystems. 2001. *Java 2 Platform Enterprise Edition Specification v1.3*, Available via ftp from www.java.sun.com. Sun Microsystems.

[5] Object Management Group. 2000. *UML Profile for Modeling Quality of Service as it relates to real-time systems*. Draft Request for Proposal. OMG document ad/00-12-07. Framington, MA. Note: This RFP was never issued.

[6] Object Management Group. 2002. *UML™ Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms*. Request for Proposal. OMG document ad/02-01-07. Framington, MA. Note: This RFP issued January 2002 with submissions due June 24, 2002.

[7] Rajeev R. Raje, Mikhail Auguston, Barrett Bryant, Andrew Olson, Carol Burt. 2001. *A Quality of Service-based Framework for Creating Distributed Heterogeneous Software Components*. Technical Report. Indiana University Purdue University Indianapolis.

[8] Zinky, J.A.,Bakken, D.E., and Schantz, R., 1995. *Overview of Quality of Service for Distributed Objects*, In Proceedings of the Fifth IEEE Dual Use Conference.

[9] G. Brahnmath, R. Raje, A. Olson, M. Auguston, B. Bryant and C. Burt. 2002. *Quality of Service Catalog for Software Components*. Technical Report #TR-CIS-0219-01. Indiana University Purdue University Indianapolis.

[10] BBN Corporation, 2001. *Quality Objects (QuO) Project*, URL: http://www.dist-systems.bbn.com/tech/QuO.

[11] Michigan State University, 2001. *RAPIDware: Component-Based Development of Adaptable and Dependable Middleware*, URL: http://www.cse.msu.edu/rapidware/.

[12] S. Frolund, J. Koistinen. 1998. *Quality of Service specification in Distributed Object Systems*, Proceedings of the 4th USENIX Conference on Object-Oriented Technologies and Systems (COOTS '98), Santa Fe, New Mexico, April 1998.

[13] S. Frolund, J. Koistinen. 1999. *Quality of Service Aware Distributed Object Systems*. 5th USENIX Conference on Object-Oriented Technologies and Systems (COOTS '99). May 1999.

[14] M.Auguston, 1998. *Building Program Behavior Models*, Proceedings of the European Conference on Artificial Intelligence ECAI-98, Workshop on Spatial and Temporal Reasoning, Brighton, England, August 23-28, 1998, pp.19-26.

[15] Object Management Group. 2000. *Meta Object Facility*. Document formal/2001-11-02. Framingham, MA, Object Management Group.