Towards Specifying QoS-Enabling Software Architectures

Valérie Issarny, Christophe Bidan, Frédéric Leleu and Titos Saridakis
IRISA / INRIA
Campus de Beaulieu, 35042 Rennes Cédex, FRANCE
{issarny, bidan, leleu, saridaki}@irisa.fr

Extended Abstract

1 INTRODUCTION

The term QoS has been originally used in the context of network communication resources and then in the context of system resources needed for multimedia applications. It now stands for specific resource management as required by applications and relating to non-functional execution properties implemented by the processing environment (e.g. availability, security, responsiveness). In that framework, there exists a wide spectrum of application domains having QoS requirements (e.g. multimedia, military, parallel, telecommunication applications). In the same way, there is a growing diversity of available software platforms implementing system support for guaranteeing a certain QoS (e.g. multimedia architectures (Nahrstedt & Smith 1996), standard distributed processing environments (TINA-C 1995)). The goal of our research is to provide a development environment that eases the construction of QoS-demanding applications through the re-use of existing QoS-related techniques. Our solution relies on the description of the application's *software architecture*, which includes the specification of the execution properties that must be provided by the underlying Distributed Processing Environment (DPE). Execution properties subdivide into two categories: (i) *interaction properties* that capture the communication patterns among the application's modules and that allow to reason about the adequacy of a given communication platform with the communication protocol required by the running modules, and (ii) *QoS properties* that capture the resource management policies implemented transparently by the underlying DPE and that enable to reason about the adequacy of the DPE with respect to the application QoS requirements. The next section outlines description of software architectures together with the specification of associated execution properties. Conclusions then address related work and our current and future work.

2 DESCRIBING QOS-ENABLING SOFTWARE ARCHITECTURES

Our proposal builds on results produced in the software architecture research field of software engineering (Perry & Wolf 1992). Ongoing research work aims at providing a sound basis for the specification of various styles of software architectures. An architectural style identifies the set of patterns that should be followed by the system organization, that is, the kinds of components to be used and the way they interact.

2.1 Describing Software Architectures.

Although the software architecture field is continuously evolving, it is now accepted that the description of an architecture decomposes into at least three abstractions (e.g. (Shaw, DeLine, Klein, Ross, Young & Zelesnik 1995)):

1. *components* that abstractly define computational units written in any programming language;
2. *connectors* that abstractly define types of communications (e.g. pipe, RPC) between components;
3. *configuration* that defines a system structure (i.e. a software architecture) in terms of the interconnection of components through connectors.

Development environments that are based on the software architecture paradigm integrate an Architectural Description Language (ADL) that allows system specification in terms of the three above abstractions, and runtime libraries that implement base system services (including primitive connectors). Such an architectural description fosters software reuse, evolution, analysis and management. However, in order to be able to reason about the correctness of a software architecture from the standpoint of resource management implemented by the underlying DPE, the description of connectors should be enriched with the specification of associated execution properties that subdivide into interaction and QoS properties.

2.2 Specifying interaction properties.

Interaction properties relate to the communication patterns that are implemented by the DPE. Thus, the specification of execution properties allows to verify that the use of communication protocols by the application modules is correct with respect to the protocols that are implemented by the DPE. Formal specification of interaction properties is addressed by the Wright ADL introduced in (Allen & Garlan 1994). Using Wright, a component is described as a set of *port* processes that are the component interaction points, and the *coordination* process defining coordination among ports. The behavior of a component is then the parallel composition of port and coordination processes. Similarly, a connector is described as a set of *role* processes that implement communications among components, and the coordination process that defines the coordination among roles; and the behavior of a connector is defined as the parallel composition of role and coordination processes. Finally, a configuration is described as a set of components and connectors, which are interconnected through appropriate bindings between ports and roles. Given the use of CSP-like notations, the correctness of a configuration with respect to the communication protocols that are used can be checked through the *refinement* concept. Basically, it must be verified that port processes *refine* the role processes with which they are bound.

2.3 Specifying QoS properties.

QoS properties describe abstractly the resource management policies that are transparently implemented by the DPE. Hence, the specification of such properties allows to verify that the DPE guarantees the application constraints relating to various non-functional properties (e.g., availability, responsiveness, security, timeliness). The QoS-oriented description of distributed software architectures constitutes the basis of the ASTER configuration-based programming environment (Bidan, Issarny, Saridakis & Zarras 1997). Using the ASTER environment, it can be checked automatically whether a given distributed platform (characterized using connectors) meets the QoS requirements of communicating components with respect to the specified QoS. Such a verification is achieved using a logical tool that implements specification matching.

ASTER further supports customization of a platform from the standpoint of QoS by selecting within the software repository, the complementary software components that are needed to support a given QoS. As mentioned in the introduction, there is a wide diversity of execution properties relating to QoS. Up to now, we have been mainly concentrating on security and availability issues. From the standpoint of responsiveness and timeliness, we are looking for solutions based on existing work in this area (e.g., (Demeure, Farhat-Gissler & Gasperoni 1996, Lakas, Blair & Chetwynd 1996)).

3 CONCLUSION

The specification of QoS-enabling software architectures is not a new concern. In particular, let us mention the RM-ODP standardization effort (ISO/IEC 1994) and the Telecommunication-oriented TINA distributed software architecture that is being specified (TINA-C 1995). These proposals prescribe notations for specifying the QoS demanded by applications. However, QoS specification is addressed with respect to a specific distributed software architecture. On the other hand, architectural descriptions that have been addressed in Section 2 enable to describe different types of QoS-enabling software architectures. In addition, let us mention that the ASTER environment allowing QoS-oriented description of software architectures can be considered as a programming environment implementing the TINA's computing architecture.

We have been experimenting our approach to deal with QoS properties relating to general purpose distributed applications and to telecommunication applications (Bidan et al. 1997). We are now working on the complete specification of timeliness constraints so as to address specification of multimedia-dedicated software architectures. In addition, we are working on the specification of interaction properties based on the proposal of (Allen & Garlan 1994) so as to reason about the adequacy of a DPE for a given application with respect to the complete set of execution properties required by the application.

REFERENCES

Allen, R. & Garlan, D. (1994), Formalizing architectural connection, *in* 'Proceedings of the Sixteenth International Conference on Software Engineering'.

Bidan, C., Issarny, V., Saridakis, T. & Zarras, A. (1997), ASTER Project Published papers, URL: `http://www.irisa.fr/solidor/aster`, IRISA-INRIA, Rennes, France.

Demeure, I., Farhat-Gissler, J. & Gasperoni, F. (1996), A scheduling framework for the automatic support of temporal QoS constraints, *in* 'Proceedings of the Fourth International IFIP Workshop on Quality of Service'.

Lakas, A., Blair, G. S. & Chetwynd, A. (1996), A formal approach to the design of QoS parameters in multimedia systems, *in* 'Proceedings of the Fourth International IFIP Workshop on Quality of Service'.

Nahrstedt, K. & Smith, J. M. (1996), 'Design, implementation and experiences with the OMEGA end-point architecture', *IEEE JSAC*.

Perry, D. E. & Wolf, A. L. (1992), 'Foundations for the study of software architecture', *ACM SIGSOFT Software Engineering Notes* **17**(4).

Shaw, M., DeLine, R., Klein, D., Ross, T., Young, D. & Zelesnik, G. (1995), 'Abstractions for software architecture and tools to support them', *IEEE Transactions on Software Engineering* **21**(4).

TINA-C (1995), Overall Concepts and Principle of TINA – Version 1.0, Technical Report TB_MDC.018_1.0_94, TINA-C Document.