

Design Support Environments for Distributed Systems (DSE4DS)

EPSRC project GR/M69500/01

Project Review

Gill Waters, Computing Laboratory, University of Kent at Canterbury, Canterbury, Kent CT2 7NF;

John Derrick¹, Behzad Bordbar², David Akehurst³

1 Background/Context

The aim of this project was to extend facilities for the design of multimedia distributed systems, to ensure that they can effectively meet the needs of complex systems that include the use of stream communication, multicasting and Quality of Service (QoS) constraints. To achieve this, we have augmented the design environment with descriptions in sufficiently precise notations to enable assessments of designs to be made based on fitness for purpose, performance and functionality.

Design methodologies for distributed systems have centred around frameworks such as the Open Distributed Processing (ODP) standard [1]. Significant features of ODP include object-based design, the use of transparencies to hide aspects of distribution and five design viewpoints. The framework is important because it has standardised a number of architectural components such as streams and interfaces as well as providing approaches to QoS support and multimedia provision. Middleware platforms such as CORBA aim to hide distribution issues from the application design and therefore support the instantiation of the computational viewpoint of ODP. ODP, and other relevant standards, provide frameworks for architectural issues without providing specific design notations. This project has addressed the need for design notations for multimedia distributed systems.

The Unified Modelling Language (UML) is widely used as a design medium for object-oriented systems and was an obvious candidate on which to base our work. UML is supported by a number of tools and its continuing development is addressed by the Object Management Group (OMG), but it was not sufficient to address all the issues involved in the design of a distributed system [2]. UML has very general applicability, and therefore either precise meaning was left open (to be tailored in specific application domains) and the support offered was weak in particular areas (e.g. its concurrency model). In addition, it was possible to specify behaviour within UML in a number of ways, leading to potential consistency problems. The emergence of the Object Constraint Language (OCL) has gone some way to help.

For these reasons we chose to augment UML, informed by the ODP architecture, to enable the required precise designs to be produced. We have provided explicit support for distributed systems design, including provision for the use of stream communication, multicasting and QoS specification, developed techniques to check consistency of UML behavioural diagrams and developed techniques to predict performance, and to offer validation and verification of designs.

The DSE4DS project has built on our experience of the generation of performance simulation models for distributed systems described in UML [3]; on experience of analytic approaches using formal methods (e.g. model checking probabilistic and real time automata), and the dual language approaches to QoS specification, validation and consistency e.g. [4] and on experience of metamodels and model transformations [a], [c] and [d]. An early overview of the project can be found in [H].

¹ now at Department of Computer Science, University of Sheffield

² now at School of Computer Science, University of Birmingham

³ now at Department of Electronics, University of Kent

2. Key Advances and Supporting Methodology

In this section, we describe the achievements of the project related to the original project objectives.

2.1 Distributed Systems Design using UML

Objective 1: Augment UML to provide explicit support for distributed systems design, including provision for the use of stream communication, multicasting and QoS specification.

a) Stream interfaces and communication

As discussed above, the design of distributed systems is a complex matter raising a number of demanding issues. In [A], we based distributed systems designs on RM-ODP concepts whilst using UML notations and techniques to support them. We have concentrated on the ODP computational viewpoint as this is at the heart of the software design view. ODP defines an abstract syntax but does not define a concrete syntax for the viewpoints. We have defined a bespoke Computational Viewpoint Language (CVL) [B] from which we can build and populate our designs of specific distributed systems. Our approach to this has been to create a metamodel that specifies the CVL language itself, its components and the relationship between them. The CVL uses stereotypes of the UML class to define ODP Computational Viewpoint Object Templates and Interface Signatures. It includes all the appropriate concepts required by the computational viewpoint definition to enable us to describe the design of specific systems using UML class and template diagrams and state-charts. We have defined UML icons associated with each stereotype label to allow either or both the icon or label to be used for each component in a UML class diagram describing the Computational Viewpoint of a specific system.

In addition to template specifications, it is often useful to illustrate particular configurations of objects and their interfaces. The UML term for such diagrams is a *Snapshot*. Our CVL allows ODP Snapshots to be illustrated using diagrams similar in concept to that of UML Object diagrams, although they use a notation more familiar to the ODP community.

Our language for designing a distributed system is thus familiar to UML users, is easy to use and is supported from many standard UML tools. Binding objects provide an abstraction of the mechanisms required to provide a communications path between object interfaces; they enable complex binding behaviours to be defined, and provide a focus for the specification of communication related QoS specifications, multiple interfaces and multiple instances. The designer of a specific system (e.g. a video on demand application) can use the CVL to assemble the appropriate computational components, such as objects, bindings and interfaces and can also create appropriate snapshots to illustrate representative configurations of the system.

Key advance 1: Design Techniques and Languages for Specifying Multimedia and Group Based Applications

- Based on ODP concepts
- Use of existing notations: UML (Class/Template Diagrams, State-charts); CQML; OCL.
(See also below.)

Key advance 2: Meta-Models for the ODP Computational Viewpoints

b) QoS specification

The specification and implementation of QoS is increasingly important in distributed systems to address questions of performance, particularly for systems involving multimedia. Consequently, statements of QoS need to be introduced early in the design process. QoS describes both performance aspects and functional aspects of distributed systems. The quality of real-time multimedia systems is influenced particularly by delays, errors etc. We were therefore interested in the categories of

timeliness, for example end-to end delay (latency) and jitter (variability of delay) and of *volume*, e.g. the ability to deliver a video stream at the required rate.

In [C], we show that QoS specifications are needed at the interfaces of the stream bindings of our computational model and at the interfaces of objects using the stream interfaces. Our approach was to annotate the interfaces with textual QoS clauses, specifically to describe the QoS *requirements* of each input interface to a binding and the QoS *provision* offered at each output interface. The binding encompasses behaviour that ensures that, provided it receives the appropriate incoming QoS, it can present the appropriate outgoing QoS to its receivers; otherwise it does not meet its requirements. QoS invariants were defined to describe these characteristics. Our first approach to QoS annotations used QL [5], which is a first order real-time logic based on RTL (Real-Time Logic) and invariants were described using OCL. This technique gave us a great deal of insight into describing and evaluating QoS behaviour.

Further consideration [B] led us to use the Component Quality Modelling Language (CQML) [6] for our annotations instead of QL. CQML is associated with UML and OCL and is a feature-rich lexical language for QoS specification. We have found CQML to be expressive in describing QoS expectations and obligations; it is easy to use and to integrate with our other UML-based languages within the project – an essential characteristic for tool-building.

Key advance 3: Precise specification of QoS in the design of distributed systems

c) Multi-party facilities

Multi-party facilities are characterised by potentially large number of users. The ODP viewpoints are particularly useful here in separating out application functionality from the communication mechanism. In [D], we defined a concrete syntax and a mechanism for using UML notations to describe group based systems within both the computational and engineering viewpoints. In the computational viewpoint, we defined a mechanism to syntactically specify group objects (illustrated as a stack of member objects) within a snapshot diagram. This allows a particular object template to be bound (using an interface signature) to multiple other objects. For example, for a simple conferencing system, a group member would have an interface signatures for a conference and for a conference binding offering both transmission to and reception from the other group members. (Such bindings can also include QoS clauses.) The technique also allows snapshots to depict an initial snapshot of the system (null membership) and either a fixed or variable number of members.

The computational objects and their interactions can be mapped onto one of several alternative specifications in the engineering viewpoint, where nodes and both control and data channels can be defined. Different configuration of the end system objects can be modelled for specific communication paradigms. For example, we have shown different combinations of stubs, binders and protocol objects that specify native IP multicasting or application layer multicasting alternatives such as a complete user-to-user mesh, a central group server or hierarchies of group servers.

Key advance 4: Specification of multi-party facilities in both engineering and computational viewpoints; Metamodel for the Engineering viewpoint.

d) Future tool support

The project concentrated on producing mechanisms that are capable of being included in toolsets [B], [A] and [E]. These are based on both visual and textual notations and are drawn wherever possible from UML, with its existing range of support tools and related techniques. Some tool support has been produced by generation of a repository and browser using facilities of the Kent Modelling Framework (KMF) tool. The KMF project [a], [b] (currently supported jointly by DSE4DS and Reasoning with Diagrams (EPSRC grant GR/R63509/01)) is developing a set of tools to support model driven software

development. At the core of the KMF is a tool to generate modelling tools from the definition of modelling languages expressed as metamodels. KMF is supported by Java libraries that include capabilities for dynamic evaluation of OCL constraints, support for checking well-formedness of models and an implementation of the XML Metadata Interchange (XMI) standard.

Key advance 5: Demonstration of the potential capabilities of a design tool via development of the KMF tool for generating modelling tools.

2.2 Consistency of UML Behavioural Diagram

Objective 2: Develop techniques to check consistency of UML behavioural diagrams

In undertaking this work item we began by reviewing the state of other work regarding the behavioural aspects of UML. This assessment showed that a significant body of work known as the UML Action Semantics has arisen as a result of the OMG's Request for Proposals on this issue. The original aims of this work package involved plans for some similar work to that achieved within the Action Semantics community and, in order not to duplicate this work, we have instead focused on investigating the application of a more general consistency framework within the context of UML.

Previous work at Kent [7] has developed techniques for checking consistency between multiple viewpoint specifications, in the context of the RM-ODP, each of which could be written in a different language. In [F], we have investigated the use of these techniques in the context of treating the multiple diagrams and diagram types of a UML specification as different viewpoints. Further work is planned in this area, possibly in conjunction with others working on similar problems.

No single diagram in UML defines the complete behaviour of complex systems; this is only given by their collective constraints. Indeed, the philosophy of UML is that complex systems are best specified via a number of different views of a model. Thus each diagram represents a partial description of the complete system. Such a partial description is usually called a partial specification or a viewpoint. It is of course essential that the total specification formed from the collection of partial specifications is consistent, otherwise an implementation of the system could not be built.

In order to be able to check the consistency of multiple viewpoint specifications, we first define what is meant by consistency. Many different notions of consistency exist; however, the definition in [7] and [F] is based on the notions of refinement relations and correspondences. Correspondences are the definition of linkage between overlapping viewpoints and refinement is the usual notion of development. Consistency is thus defined as follows:

A set of viewpoint specifications are consistent if there exists a specification that is a refinement of each of the viewpoint specifications with respect to the identified refinement relations and the correspondences between viewpoints. This common development is called a unification.

The focus of such a perspective on consistency is on behavioural consistency. In the context of UML there are diagram types (viewpoints) for the specification of both structure and behaviour. Our definition of consistency means that we take a view that all that matters is the behavioural specifications. Thus two or more different UML diagrams can be considered consistent if a third (not necessarily UML) specification can be written that is a common refinement of the behaviour defined in each.

For this approach to truly solve the consistency issues of UML it would be necessary for a common semantic model to be adopted across the multiple diagram types, and this feeds into, and depends upon, other ongoing work in the UML community.

Key advance 6: Use of Refinement Techniques for addressing Consistency between UML Diagrams

2.3 Performance, validation and verification

Objective 3: Enhance existing techniques to predict performance, and offer validation and verification of designs

The stated objective of this work package was to investigate ways to prove aspects about the usability of a system that has been designed using products of the other work packages. In particular the original aim was to take a function design augmented with non-functional Quality of Service (QoS) specifications and map this to a formal language, such as the language of Timed Automata. The formal specification was then to be used as a means to verify the performance of the functional specification against the required QoS.

We have developed techniques for transforming QoS specifications into state based, Timed Automata (TA,) specifications to enable model checking of functional behaviour against the QoS-based temporal constraints [E], [G]. This is achieved by the definition of particular patterns of TA that can be used to represent particular QoS characteristics. State-based behaviour specifications are also mapped to TA specifications. The combination of these two sets of TA is used as input into the Uppaal model checker. Error states in the TA are used to indicate whether or not a QoS constraint has been invalidated by the functional behaviour.

Drawing on previous UKC work [8], patterns of TA have been developed that model the QoS characteristics for latency, anchored jitter and throughput; variations in those patterns have also been developed for use in the context of an expected or obliged QoS specification. To complete the mapping we have also developed a framework of TA specifications onto which we can map the specification of a configuration of distributed objects. The functional behaviour of these objects should be specified using the UML state diagram (or related Statechart) notation and, drawing also on the work of [9], we map these hierarchical state descriptions to the flat model required by Uppaal. We use the Uppaal model-checker tool as a means to perform the functional vs non-functional performance verification.

The TA patterns for QoS characteristics are design with error states that cause a deadlock if there is a QoS violation; thus the resulting set of TA can be model-checked against a test for no deadlocks. If no deadlocks are found in the system it can be asserted that the functional specification of the system performs in accordance with the non-functional specification. Further context and input to this component of the work has been provided by that conducted under the EPSRC funded project VQoS: *A Specification Architecture for the Validation of Real-time and Stochastic Quality of Service* [10]. Further work on verification of the timeliness QoS properties in multimedia systems can be found in [I].

Key advance 7: Techniques for mapping Designs to Formal Specifications (Timed Automata) in-order to verify QoS against functional behaviour using a model checker.

2.4 Case Studies

Objective 4: Perform several case studies

During the development of our techniques, we have used a number of case studies. Typically, simple case studies were used for preliminary explorations and these were expanded to cover further capabilities or exceptions and to cross-check that the techniques were applicable across a range of distributed systems. The case studies covered have included an interactive multimedia kiosk [A], a near video-on demand system [B], lip synchronisation between audio and video streams [8], [G] and a multi-party message conferencing system [D].

3. Summary and Further Research

As can be seen from the above description, we have achieved many key advances in the project's objectives. The project benefitted from the variety of expertise of the research team, including QoS and multicasting in distributed systems and networks, verification, validation and refinement, timed automata and modelling and metamodelling. In common with many other Computer Science projects, the progress of related research in the community has affected some of our initial objectives. For example, the UML consistency work has been superseded by Action Semantics, reducing the scope of the initial consistency requirements, hence work in this area was reduced in favour of other areas.

The members of the research team have now dispersed and their current research interests are outlined below. Parallels with the work in the DSE4DS project are clear.

Dr. Gill Waters (principal investigator), Computing Laboratory, University of Kent. *Interests: Support for applications involving groups and multicasting, especially optimisation for multicast routing and Application Layer Multicasting.* Current work concentrates on the Osmunda project (Optimisation in Support of MULTicast NetworkED Applications), which examines application layer structures for distributed systems, looking particularly, but not exclusively, at multicasting. The novelty of this work is that we are looking at different techniques for organising these structures so that they can be built to required optimisation constraints. In addition to graph theory, we consider clustering algorithms and computational geometry.

<http://www.cs.kent.ac.uk/people/staff/agw/>

Prof. John Derrick (co-investigator), now at the Department of Computer Science, University of Sheffield. *Research interests include: refinement in state-based systems; integrated formal methods; viewpoint specification; verification and testing Erlang code; policy specification; frameworks for distributed systems design.*

<http://www.dcs.shef.ac.uk/~jd/>

Dr. Behzad Bordbar (project RA), now at School of Computer Science, University of Birmingham. *Current research interests include: QoS in Wireless systems, Model Transformation in Web Services, Verification of QoS in multimedia.*

<http://www.cs.bham.ac.uk/~bxb/>

Dr. David Akehurst (project RA), now in Department of Electronics, University of Kent. *Interests: Model Transformation / Translation, visual Languages, model Checking, performance Modelling, UML and OCL.* Current work is in applying the OMG's Model Driven Architecture (MDA) to Distributed Systems and Embedded Systems. The MDA initiative proposes the idea of Platform Independent Models (PIMs) and Platform Specific Models (PSMs), with transformations between them. A PSM for a distributed system is composed of a number of different models; i.e. the model of the code, the model of the configuration, model of security policies for a platform, etc. In addition, frameworks such as the RM-ODP suggest the use of multiple viewpoints or PIMs. Current work applies these modeling techniques to embedded systems.

<http://www.cs.kent.ac.uk/archive/people/staff/dha/> or

http://www.ee.kent.ac.uk/department/staff_detail.aspx?id=191

4. Publications and References (Project Web site: <http://www.cs.kent.ac.uk/projects/dse4ds/>)

Project Publications (techniques)

- [A] B. Bordbar, J. Derrick, and A. G. Waters, "A UML Approach to the Design of Open Distributed Systems," in H. Miao (eds) proceedings Formal Methods and Software Engineering, 4th International Conference on Formal Engineering Methods, ICFEM 2002, Springer, LNCS, 2495, Shanghai, China, pp. 561-572, October 2002.
- [B] D. H. Akehurst, J. Derrick, and A. G. Waters, "Addressing Computational Viewpoint Design," in proceedings Enterprise Distributed Object Computing Conference, EDOC 2003, Brisbane, Australia, pp. 147-158, September 2003.
- [C] B. Bordbar, J. Derrick, and A. G. Waters, "Using UML to specify QoS constraints in ODP," *Computer Networks*, vol. 40, pp. 279-304, 2002.
- [D] D. H. Akehurst, A. G. Waters, and J. Derrick, "A Viewpoints Approach to Designing Group Based Applications," in proceedings Design, Analysis and Simulation of Distributed Systems 2004, ed. Herwig Unger, Advanced Simulation Technologies Conference, Arlington, Virginia, pp. 83-93, April 2004,
- [E] D. H. Akehurst, B. Bordbar, J. Derrick, and A. G. Waters, "Design and Verification of Distributed Multi-media Systems," University of Kent Computing Laboratory Technical Report 1-03, January 2003.
- [F] J. Derrick, D. H. Akehurst, and E. Boiten, "A framework for UML consistency," in Z. Huzar (eds) proceedings <<UML>> 2002 Workshop on Consistency Problems in UML-based Software Development, pp. 30-45, October 2002.
- [G] D. H. Akehurst, J. Derrick, and A. G. Waters, "Design and Verification of Distributed Multi-media Systems," in P. Stevens (eds) proceedings Formal Methods for Open Object-Based Distributed Systems, FMOODS 2003, Springer, Lecture Notes in Computer Science 2884, Paris, pp. 276-292, November 2003.
- [H] D. H. Akehurst, B. Bordbar, J. Derrick, and A. G. Waters, "Design Support for Distributed Systems: DSE4DS," in A. Montessoro (eds) proceedings 7th Cabernet Radicals Workshop, Bologna, Italy, October 2002.
- [I] B. Bordbar and K. Okano, "Verification of Timeliness QoS Properties in Multimedia Systems," in proceedings International Conference on Formal Engineering Methods (ICFEM), 2004.

Project Publications (tools)

- [a] D. H. Akehurst, S. Kent, and O. Patrascoiu, "A relational approach to defining and implementing transformations between metamodels," *Journal on Software and Systems Modeling*, vol. 2, pp. 215-239, November 2003.
- [b] D. H. Akehurst and S. Kent, "A Relational Approach to Defining Transformations in a Metamodel," in S. Cook (eds) proceedings The Unified Modeling Language 5th International Conference, LNCS, Springer, 2460, Dresden, Germany, pp. 305-320, 2002.
- [c] D. H. Akehurst and O. Patrascoiu, "Tooling Metamodels with Patterns and OCL," in J. S. Willans (eds) proceedings Metamodelling for MDA: First International Workshop, York, UK, pp. 203-215, November 2003.
- [d] D. H. Akehurst and O. Patrascoiu, "OCL 2.0 – Implementing the Standard for Multiple Metamodels," in proceedings UML 2003 Workshop, OCL 2.0 - Industry standard or scientific playground?, San Francisco, USA, October 2003.

References

- [1] ITU-T Recommendation X.901-5 10746-2 to 5:1996-99, Information Technology - Open Distributed Processing - Reference Model: All Parts
- [2] J. O. Aagedal and A. Berre, "ODP-Based QoS-Support in UML," in proceedings First International Enterprise Distributed Object Computing Workshop (EDOC'97), IEEE, Gold Coast, Australia, October 1997.
- [3] A. G. Waters, P. F. Linington, D. H. Akehurst, P. Utton, and G. Martin, "Permabase: Predicting the performance of distributed systems at the design stage," *IEE Proceedings - Software*, vol. 148, pp. 113-121, August 2001.
- [4] H. Bowman, J. Derrick, P. Linington, and M. Steen, "Cross Viewpoint Consistency in Open Distributed Processing," *IEE Software Engineering Journal*, vol. 111, pp. 44-57, January, 1996.
- [5] G. Blair and J.-B. Stefani, *Open Distributed Processing and Multimedia*: Addison-Wesley, 1997.
- [6] J. Ø. Aagedal and E. F. Ecklund, "Modelling QoS: Towards a UML Profile," in S. Cook (eds) proceedings <<UML>> 2002 The Unified Modeling Language: Model Engineering, Concepts, and Tools, Springer, LNCS 2460, Dresden, Germany, pp. 275-289, October 2002.
- [7] E. Boiten, H. Bowman, J. Derrick, P. F. Linington, and M. Steen, "Viewpoint consistency in ODP," *Computer Networks*, vol. 34, pp. 503-537, August 2000.
- [8] H. Bowman, G. Faconti, J.-P. Katoen, D. Latella, and M. Massink, "Automatic verification of a lip-synchronisation algorithm using uppaal - extended version," in J. V. Wamel (eds) proceedings FMICS'98 Third International Workshop on Formal Methods for Industrial Critical Systems, CWI, Amsterdam, The Netherlands, pp. 97-124, May 1998.
- [9] A. David and M. O. Moller, "From HUPpaal to Uppaal: A translation from hierarchical timed automata to flat timed automata," BRICS, Department of Computer Science, University of Aarhus, Research Series RS-01-11, March 2001.
- [10] V-QoS project, <http://www.cs.kent.ac.uk/research/groups/tcs/vqos/vqos.html>.

