Design Support for Distributed Systems: DSE4DS

D.H.Akehurst, B.Bordbar, J.Derrick, A.G.Waters

University of Kent at Canterbury
{ D.H.Akehurst, B.Bordbar, J.Derrick, A.G.Waters }@ukc.ac.uk

Distributed System design is a highly complicated and non-trivial task. The problem is characterized by the need to design multi-threaded, multi-processor, and multi-media systems. Design frameworks such as Open Distributed Processing (ODP), the ITU/ISO standard, provide significant, theoretical, guidance and aids towards the provision of successful distributed system designs. The purpose of the DSE4DS project is to develop these theoretical aids into practical 'design support' that will assist designers in the construction of distributed system specifications.

1 Introduction

The aim of the DSE4DS project is to extend facilities for the design of multimedia distributed systems, to ensure that they can effectively meet the needs of complex systems that will include the use of stream communication, multicasting and Quality of Service (QoS) constraints. To achieve this we augment designs with descriptions in sufficiently precise notations to enable assessments of designs to be made based on fitness for purpose, performance and functionality.

The pervasiveness of the Internet and the introduction of increasingly powerful network technologies including ATM have resulted not only in distributed systems becoming commonplace and critical to many walks of life, but also in the diversity of those systems. Expectations of new distributed services, such as multimedia conferencing and video distribution, are that they should match the quality of such facilities in broadcast media and home entertainment technologies.

However, users of distributed systems can easily be disappointed that this is not the case. Distributed systems are complex and rely heavily on suitable software design methodologies to make them work. In all modern systems, it is the software provision and maintenance that is the most expensive part of the system. In order to minimise costs, there is a need for software design tools that encourage rather than limit the design of effective distributed systems.

Traditional systems engineering methodologies are not focused on the development of distributed systems. However, because modern distributed systems are object-based there has been considerable interest in the use of the Unified Modelling Language (UML [1]) for the design of complex distributed systems.

Indeed UML has become a de facto standard for visualising, specifying, designing, and documenting object-oriented systems, and has emerged as the standard object-oriented analysis and design notation. Because it attempts to provide notation for most aspects of object-oriented design, users can select from a rich variety of design diagrams in UML and there are few guidelines on how precisely the design is defined. Therefore serious modelling and composition of components and frameworks in UML is not trivial. Industrial usage of UML in a distributed systems context has identified the need for a tailored design environment to be produced based upon UML.

The aims of the DSE4DS project respond to that need. In particular, we are building a set of modelling tools to provide such an environment, which focus on some of the key issues in the design of distributed multimedia systems. This involves the specialisation and extension of existing UML notations to produce frameworks that are both abstract and precise. Specifically, we are:

1. Augmenting UML to provide explicit support for distributed systems design, including provision for the use of stream communication, multicasting and QoS specification.
2. Defining techniques to check consistency of UML behavioural diagrams.
3. Enhancing existing techniques to produce performance validation and verification from designs.

This paper reviews some of this work. The structure is as follows: section 2 discusses existing work relevant to the project aims; section 3 describes the UML based language we propose for the structural specification of distributed systems and their Quality of Service; section 4 discusses the use of Statecharts as a language for behavioural specification within our framework.

Section 5 discusses the QoS analysis that is made possible by utilising our proposed method and languages for distributed systems design and section 6 concludes the paper.

2 Background

The design of distributed systems is non-trivial and the advent of multimedia systems has increased the complexity by requiring questions of performance to be central. Architectural frameworks, such as Open Distributed Processing (ODP [2]), have become important because they help standardised a number of architectural components such as streams and interfaces as well as providing approaches to QoS support and multimedia provision. The thinking behind ODP has also led to the implementation of CORBA and other middleware platforms that provide crucial implementation and application design support for computational concerns.

Architectural frameworks do not provide specific design notations, and there has been considerable interest in the use of UML as a design medium. However, the use of UML as it stands is not sufficient to address all the issues involved in the design of a distributed multimedia system [3] [4]. Problems arise due to the generality of UML, and therefore either precise meaning has been left open (to be tailored in specific application domains), or the support offered is weak in particular areas (e.g. its concurrency model).

It is therefore necessary to tailor UML to enable better support to be provided for the key architectural concepts and components. There are a number of relevant approaches and we will draw upon these to provide support for the specific areas outlined above. Work on the relationship between ODP and UML includes that by OMG in their Object Oriented Analysis and Design and Meta-Object Facility proposals [5] [6]; and general applications of UML to distributed systems design includes [7] [4]. This work provides an excellent baseline (as does our own Permabase work [8]) but none provide support for the areas of multimedia design we address in this project. Relevant work on tailored solutions to UML includes the work by Objectime in association with Rational on use of UML for real-time systems [9], and that of Douglas [10] that concentrates on embedded systems.

The correct specification of behaviour is particularly important in a complex multimedia system, and within UML behaviour can be specified in a number of ways (e.g. using different forms of behavioural diagram). Support is therefore required in order to keep the UML diagrams consistent within a specification and throughout its evolution. Relevant work here includes that at UKC [11] [12] [13] [14] on the use of formal languages in individual ODP viewpoints, and on how these partial specifications can be checked for consistency. Also relevant is the work on defining formal UML semantics [15], which is necessary for UML designs to be interpreted consistently.

The emergence of OCL (Object Constraint Language [16]), which is heavily influenced by the formal methods Z and Object-Z, is also relevant as a means to support some of the behavioural issues such as QoS specification, which need a precise description language. However, OCL is still in the process of evolving, and the work of this project makes a timely contribution to its definition and use.

The resultant support for precise QoS description will enable performance aspects to be addressed. Work that is relevant to the performance issues includes analytic approaches using formal methods (e.g. model checking probabilistic and real time automata [17] [13]), and the dual language approaches to QoS specification and validation [18]. Model checking represents perhaps the most feasible approach to large-scale verification. Dual language approaches provide a means to separate out the specification of QoS requirements from the description of the underlying resource. The latter fits well with the simulation models which can be generated from UML capture of distributed systems and their execution environments [19]. This approach to simulation together with that of model checking UML designs is novel. Current approaches to verification of performance models derived from UML have centred around the integration and analysis of queuing models into UML [20]. This work, which is not targeted towards distributed systems support, is complimentary to our approach.

3 Design Languages

The ODP-RM defines five Viewpoints from which to design a system [2]. Each of these viewpoints forms a complete specification of the system, although each one addresses different aspects of the system design, abstracting away from parts of the design addressed in the other viewpoints. The five viewpoints defined are Enterprise, Information Computational, Engineering and Technology. The focus of the research in the DSE4DS project surrounds the aspects of design addressed by the Computational and Engineering viewpoints.

The ODP-RM standard does not prescribe any particular language or notation for use as a specification language of any of the viewpoints. However, to leave the choice of language up to an individual designer is not appropriate. Our decision is to use the UML as a specification language for both Computational and Engineering viewpoints.

Given the differences between UML objects and ODP objects we do not directly use the UML as a design language. Instead we use the UML (or MOF [6]) to describe a UML like language that is

appropriate as a Computational Viewpoint language. The primary difference between UML objects and ODP objects is the relationship between objects and interfaces. ODP objects support one or more interfaces each of which is defined by one or more interface signatures; Interfaces in UML are realised by classes and UML objects can be accessed as a particular interface type.

3.1 Streams, Interfaces and Communication

Within the context of the ODP-RM framework, objects are instantiated from an Object Template specification and interfaces are instantiated from Interface Signature specifications. We can use the UML notation (and notion) of a class and interface to specify these ODP concepts. To clarify that we are depicting an ODP specification, the UML class and interface concepts are given stereotypes to identify them as ODP concepts. UML interfaces are stereotyped as one of the three different types of Interface Signature (Stream, Operation, Signal) and UML classes are stereotyped as either Computational Object Templates or one of the three types of Binding Object Template (Stream, Operation, Signal).

Binding objects provide an abstraction over the mechanisms required to provide a communications path between object interfaces. Binding objects enable complex binding behaviours to be defined, and provide a focus for the specification of communication related QoS specifications.

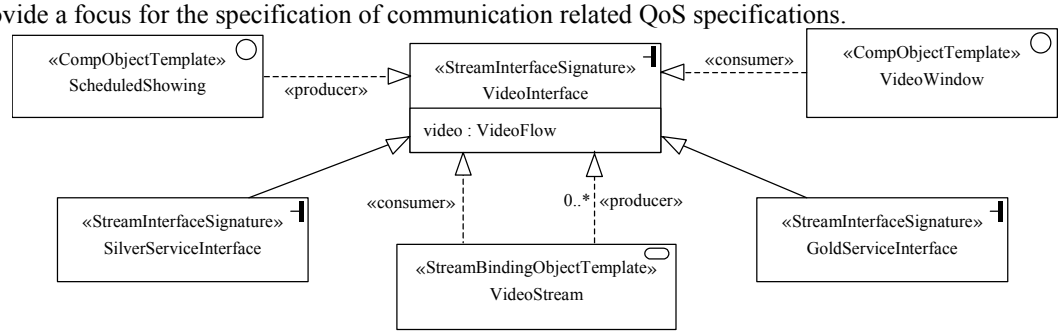

Figure 1 A Computational Template Diagram

The Templates Diagram shown in Figure 1 forms part of a Computational Viewpoint specification of a Video on Demand system. The figure illustrates the use of our UML based notation to define the computational object templates, interfaces signatures and the relationships between them. In addition to the stereotypes, icons are used to aid the pictorial distinction between the different concepts. This particular diagram shows the definition of a computational object template called *ScheduledShowing* that supports the stream interface *VideoInterface* in the role of a producer (i.e. it is the source of a video stream flow). The object template *VideoWindow* is shown as supporting a consumer interface of the *VideoInterface* signature. In addition a Stream Binding Object Template is specified, which supports a single *VideoInteface* interface as a consumer, and which supports multiple *VideoInterface* interfaces in the role of producer. This defines the *VideoStream* binding object to be a communications path that supports 1-to-many multi-cast communication. The *GoldServiceInterface* and *SilverServiceInterface* signatures are specified as sub-types of the *VideoInterface* signature; thus instances of these interfaces may be substituted wherever a *VideoInterface* interface is expected. These alternative interfaces are provided in order to support the two levels of QoS that we define in the following section.

In addition to template specifications, it is often useful to be able to illustrate particular configurations of objects and their interfaces. The UML term often used for such diagrams is a Snapshot. Our Computational Viewpoint language 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. An example snapshot is shown in Figure 2.

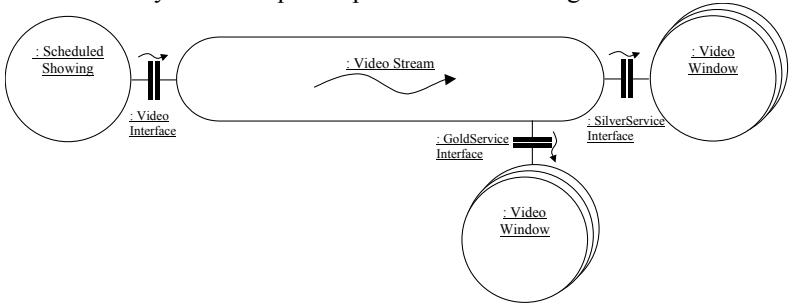

Figure 2 A Computational Snapshot Diagram

These ODP snapshots illustrate instances of object templates as circles, instances of binding object templates as elongated circles and instances of interface signatures as 'T' shapes. Arrows are placed near to interfaces to indicate the direction of causality, to indicate that two interfaces are bound they are either placed next to each other (as shown in the figure) or connected via a dashed line. Multiple objects can be illustrated using stacked circles. The name and type of each instance can be shown on the objects using the UML notation of an underlined 'name : Type'; where both the name and type labels are optional.

Figure 2 shows a single ScheduledShowing object bound via a VideoStream binding object to several VideoWindow objects. Some of the VideoWindow objects are bound via instances of the SilverServiceInterface signature and some via the GoldServiceInterface signature.

3.2 Quality of Service

Quality of Service (QoS) is a general term that refers to the non-functional aspects of a system. In general there are requirements on a system indicating the functions that are to be performed and the QoS with which they should be performed.

To specify QoS we use the language CQML [21]. This language has at its core the OCL and hence it integrates well with UML like languages. The CQML specification below describes the QoS of the binding specified in Figure 1.

```
QoSprofile videoQuality for VideoStream {
  uses fastAndSteady(consumerVideoInterface.video.SignalsRecieved);
  provides
       producerVideoInterface->select(
           oclIsKindOf(SilverServiceInterface))->forAll( vi |
                    silverService(consumerVideoInterface.video, vi.video) )
       and
       producerVideoInterface->select(
           oclIsKindOf(GoldServiceInterface))->forAll( vi |
                    goldService(consumerVideoInterface.video, vi.video) );
}
```

This first segment of CQML defines the QoS expectations (uses) and obligations (provides) of VideoStream bindings. The 'uses' clause states that VideoStream binding objects expect a fast and steady input of video on the input 'consumer' interface; the details of 'fast and steady' are defined below. The 'provides' clause is a conjunction of two parts, which specify the obligations to output interfaces of either Gold or Silver service quality. There are many output (or producer) interfaces of the binding as it is a 1-to-many multicast binding. The type of the interface indicates the QoS that the binding object is obliged to provide at that interface.

```
quality fastAndSteady(seq : Sequence(Event)) {
  statFrameRate(seq).mean >= 25 and
  jitter(seq) <= 20
}
```

The 'fastAndSteady' quality is defined as a mean frame rate that is 25 frames per second or more, with an inter frame jitter of no more than 20 milliseconds. The silver and gold service qualities are defined in CQML as follows:

```
quality silverService(src : VideoFlow, sink : VideoFlow) {
  statLatency(src, sink).mean < 100 and
  statLatency(src,sink).variance < 10 and
  statFramerate(sink).mean >= 20
}

quality goldService(src : VideoFlow, sink : VideoFlow) {
  statLatency(src, sink).mean < 50 and
  statLatency(src,sink).variance < 5 and
  statFramerate(sink).mean >= 25
}
```

The silver service quality is defined to have a mean latency of less than 100 milliseconds, with a variation in latency of less than 10 milliseconds and a frame rate of no less than 20 frames per second. The gold service quality is defined as a mean latency of less than 50 milliseconds, with a variation less than 5 milliseconds and a frame rate of no less than 25 frames per second.

3.3 Tool Support

The notations have been designed such that they are compatible enough with standard UML so that any typical UML tool can be used to create the specifications. Even the ODP snapshot diagrams can be illustrated using UML Object diagram notation, by representing interfaces as UML objects linked to the ODP object (also represented as a UML object).

4 Behaviour Specification

To specify the functional behaviour of computational objects we use the Statechart language. This language, already part of UML, gives a precise and well-integrated behavioural specification technique. In addition, Sequence and Collaboration Diagrams can be used to specify particular traces of behaviour, which can subsequently be validated against the Statecharts.

Primitive operations, such as object and binding creation, or system calls such as getting the current time, are supported by the ODP functions. These functions form an 'Engineering Virtual Machine' [22] upon which the computational behaviour executes.

5 QoS Analysis

There are two aspects to QoS analysis with respect to computational viewpoint specifications:

1. Inter-Object QoS issues: If we bind the output interface of one object to the input interface of another object will the obligations of the first meet the expectations of the second?
2. Intra-Object QoS issues: Given a particular functional behaviour specification, can the specified obligations be met with respect to the defined expectations?

We address the first of these issues by enabling comparisons to be made between specified QoS obligations and QoS expectations. These comparisons provide a true or false value indicating whether or not the obligations meet the expectations.

The second issue is more complex, requiring analysis by some form of model checking. Currently we have addressed this by mapping QoS statements and Statechart specifications onto the Stochastic Automata language used by the UPPALL tool [23]. Conveniently, this tool takes XML as input; hence we can automatically generate Automata from our specifications and use the UPPALL tool to check for deadlock in the generated automata. If deadlock occurs, then we deduce that the functional specification will not support the specified QoS.

6 Conclusion

In this paper we have given an overview of the research currently underway within the DSE4DS project at the University of Kent at Canterbury. We have given a strong motivation for why additional design support for distributed systems is appropriate, specifically within the context of multimedia systems.

Subsequently, we have shown that the use of UML and OCL based languages (such as CQML) are appropriate for the specification of structural, behavioural and QoS aspects of distributed multimedia systems. Further more, we have indicated that use of such languages enables additional design support to be provided by way of verification of the specifications through the use of model-checking techniques.

Currently the work for the project has developed UML based design languages, and some algorithms for automated analysis. Future work is to provide appropriate *bespoke* tool support and to extend the techniques to encompass Engineering Viewpoint specifcations.

7 References

[1] OMG, "The Unified Modeling Language Version 1.4," Object Management Group formal/01-09-67, 2001.

[2] ITU-T Recommendation X.901 (1995) | ISO/IEC 10746-1:1998, Information Technology - Open Distributed Processing - Reference Model: Overview

[3] J. Ø. Aagedal and A. Berre, "ODP-Based QoS-Support in UML," in proceedings First International Enterprise Distributed Object Computing Workshop (EDOC'97), 1997.

[4] J.-M. Jezequell, A. L. Guennec, and F. Pennaneac'h, "Validating Distributed Software Modelled with UML," in proceedings <<UML>>'98 Beyond the Notation, Mullhouse, France, June 1998.

[5] Rational et al., "OA&D CORBAfacility Interface Definition," OMG.

[6] BEA Systems Inc et al., "Meta Object Facility (MOF) Specification," OMG.

[7] M. M. Kande, S. Mazaher, O. Prnjat, L. Sacks, and M. Wittig, "Applying UML to Design an Inter-Domain Service Management Application - A Case Study Based on the ACTS Project TRUMPET," in proceedings <<UML>>'98 Beyond the Notation, Mulhouse, France, June 1998.

[8] 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.

[9] A. Lyons, "UML for Real-Time Overview," ObjectTime Limited.

[10] B. P. Douglas, *Real-time UML: developing efficient objects for embedded systems*: Addison Wesley, 1998.

[11] H. Bowman, J. Derrick, P. F. Linington, and M. Steen, "Cross viewpoint consistency in Open Distributed Processing," *IEE Software Engineering Journal*, vol. 11, pp. 44-57, January 1996.
[12] E. Boiten, H. Bowman, J. Derrick, and M. Steen, "Viewpoint consistency in Z and LOTOS: A case study," in proceedings FME'97: Industrial Applications and Strengthened Foundations of Formal Methods, Springer-Verlag, 1313, September 1997.
[13] H. Bowman, E. Boiten, J. Derrick, and M. Steen, "Strategies for consistency checking based on unification," *Science of Computer Programming*, 1998.
[14] E. Boiten, J. Derrick, H. Bowman, and M. Steen, "Constructive consistency checking for partial specification in Z," *Science of Computer Programming*, December 1999.
[15] S. Kent, A. Hamie, J. Howse, F. Civello, and R. Mitchell, "Semantics Through Pictures: towards a diagrammatic semantics for object-oriented modelling notations," in proceedings ECOOP'97 workshop on Precise Semantics for Object-Oriented Modelling Techniques, Technical Report TUM-I9725, 1997.
[16] A. Kleppe and J. Warmer, "The Object Constraint Language and its application in the UML metamodel," in proceedings <<UML>>'98 Beyond the Notation, Mullhouse, France, June 1998.
[17] R. Gawlick, R. Segala, J. Sogaard-Anderson, and N. Lynch, "Liveness in Timed and Untimed Systems," in proceedings 21st ICALP, 1994.
[18] G. Blair, L. Blair, H. Bowman, and A. Chetwynd, *Formal Specification of Distributed Multimedia Systems*: University College London Press, 1997.
[19] A. G. Waters, P. F. Linington, D. H. Akehurst, and S. A, "Communications software performance prediction," *13th UK Workshop on Performance Engineering of Computer and Telecommunication Systems*, pp. 38/1-38/9, July 1997.
[20] R. Pooley and P. King, "The Unified Modelling Language and Performance Engineering," *IEE Proceedings - Software*, vol. 146, pp. 2-10, February 1999.
[21] J. Ø. Aagedal, "Quality of Service Support in Development of Distributed Systems," PhD thesis, Department of Informatics, Faculty of Mathematics and Natural Sciences, The University of Oslo, 2001
[22] G. Blair and J.-B. Stefani, *Open Distributed Processing and Multimedia*: Addison Wesley, ISBN 0-201-17794-3, 1997.
[23] K. G. Larsen, P. Pettersson, and W. Yi, "UPPAAL in a Nutshell," *Springer International Journal of Software Tools for Technology Transfer*, vol. 1, October 1997.