

- [Blair97b] G.S. Blair, L. Blair, J.-B. Stefani, "A Specification Architecture for Multimedia Systems in Open Distributed Processing", *Computer Networks and ISDN Systems* 29, pp 473-500, Amsterdam: Elsevier Science, 1997.
- [Blair98a] G.S. Blair, L. Blair, H. Bowman, A. G. Chetwynd, "Formal Specification of Distributed Multimedia Systems", London: UCL Press, 1998.
- [Blair98b] L. Blair, G.S. Blair, A. Andersen, "Separating Functional Behaviour and Performance Constraints: Aspect-Oriented Specification", MPG-98-07, Submitted for publication, May 1998.
- [Blair98c] G.S. Blair, G. Coulson, M. Papatomas, P. Robin, "An Architecture for Next Generation Middleware", To appear in *Middleware'98*, The Lake District, U.K., September 1998.
- [Boiten96] E. Boiten, H. Bowman, J. Derrick, M. Steen, "Issues in Multiparadigm Viewpoint Specification", In *SIGSOFT'96 International Workshop on Multiple Perspectives in Software Development (Viewpoints '96)*, A. Finkelstein, G. Spanoudakis (eds), pp 162-166, ACM Press, October 1996.
- [Ho95] P.-H. Ho, H. Wong-Toi, "Automated Analysis of an Audio Control Protocol", *Proceedings of the 7<sup>th</sup> International Conference on Computer-Aided Verification (CAV'95)*, LNCS 939, pp 381-394, Berlin: Springer-Verlag, 1995.
- [Issarny96] V. Issarny, C. Bidan, "Aster: A Framework for Sound Customization of Distributed Runtime Systems", In *Proceedings of the 16<sup>th</sup> IEEE International Conference on Distributed Computing Systems*, pp 586-593, 1996. See also: <http://www.irisa.fr/solidor/work/aster.html>
- [Issarny98] V. Issarny, C. Bidan, T. Saridakis, "Non-functional Properties of Software Architectures", Internal report available from IRISA, Campus de Beaulieu, 35042 Rennes Cédex, France, 1998. See also: <http://www.irisa.fr/solidor/work/aster.html>
- [Kiczales97] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J.-M. Loingtier, J. Irwin, "Aspect-Oriented Programming", PARC Technical Report, SPL97-008 P9710042, February 1997. See also: <http://www.parc.xerox.com/spl/projects/aop/reports.html>
- [Lakas96a] A. Lakas, G. S. Blair, A. Chetwynd, "Specification and Verification of Real-Time Properties Using LOTOS and SCTL", *Proceedings of the 8th International Workshop on Software Specification and Design*, pp 75-84, Paderborn, Germany, March 1996.
- [Lakas96b] A. Lakas, G. S. Blair, A. Chetwynd, "A Formal Approach to the Design of QoS Parameters in Multimedia Systems", *Proceedings of the 4<sup>th</sup> International Workshop on Quality of Service*, Paris, France, March 1996.
- [Lakas96c] A. Lakas, G. S. Blair, A. Chetwynd, "Specification of Stochastic Properties in Real-Time Systems", *Proceedings of the 11<sup>th</sup> UK Performance Engineering Workshop for Computer and Telecommunications Systems (Liverpool, Sept. 1995)*, M. Merabti, M. Carew, F. Ball (eds), pp 202-216, Berlin: Springer-Verlag, 1996.
- [Matthijs97] F. Matthijs, W. Joosen, B. Vanhaute, B. Robben, P. Verbaeten, "Aspects Should Not Die", Presented at the Aspect-Oriented Programming Workshop (AOP'97), 1997. See <http://www.treese.cs.utwente.nl/aop-ecoop97/> <position papers>
- [Mens97] K. Mens, C. Lopes, B. Tekinerdogan, G. Kiczales, "Aspect-Oriented Programming Workshop Report", workshop held at ECOOP'97. See <http://www.treese.cs.utwente.nl/aop-ecoop97/> <workshop report>
- [Regan93] T. Regan, "Multimedia in Temporal LOTOS: a Lip-Synchronisation Algorithm", *Proceedings of the 13<sup>th</sup> International Symposium on Protocol Specification, Testing and Verification (PSTV XIII)*, Amsterdam: Elsevier Science, 1993.
- [Zave93] P. Zave, M.A. Jackson, "Conjunction as Composition", *ACM Transactions on Software Engineering and Methodology* II(4), pp 379-411, ACM Press, October 1993.
- [Zave96] P. Zave, M.A. Jackson, "Where Do Operations Come From? A Multiparadigm Specification Technique", *IEEE Transactions on Software Engineering*, XXII(7), pp 508-528, IEEE, July 1996.

## 5. Ongoing research

There is still much work to be done in this area. For example, our tool support needs to be extended to handle the composition of alternative languages (other than LOTOS and temporal logic), particularly for the automata described in [Blair98b]. One desirable feature of such a tool would be the ability to synthesise values (or bounds) for certain QoS constraints, as is possible in HYTECH [Ho95].

We also plan to consider further examples related to dynamic QoS management functions since it seems that it is these adaptive algorithms that are most challenging to the current state-of-the-art of formal specification. Similarly, further aspects, such as those relating to security and fault-tolerance, will also be considered.

On the practical side, our aim is for our monitors or controllers to have the ability to interact with an actual system (or prototype) thus permitting the run-time behaviour of the system to be directly monitored or controlled as required. To achieve this, we are currently addressing the mapping of such aspects to *components* in a reflective middleware platform [Blair98c].

## 6. Conclusions

Our work distinguishes between three aspects of a system's behaviour: functional behaviour, QoS constraints, and QoS management policies. As has been discussed above, there are many advantages of using an aspect-oriented approach for the specification of system behaviour. However, there are also drawbacks relating to the complexity of the validation procedure and the consistency across the different aspects of a specification.

In this paper, we have described our experiences with this approach, including an example based on QoS management. This example illustrates a powerful feature of our work. Whereas other work in formal specification has addressed the specification of QoS constraints, little work has been done on the specification of more complex issues associated with QoS management policies. We believe that the specification of such systems cannot be satisfactorily achieved without exploiting the diversity of different languages and adopting an aspect-oriented approach to specification.

## Acknowledgements

We would like to acknowledge the financial support of EPSRC (grant reference number GR/L28890). We would also like to thank Howard Bowman<sup>2</sup>, John Derrick and Jeremy Bryans (University of Kent at Canterbury) for their input during our collaborative V-QoS project meetings. Finally, thanks to Abderrahmane Lakas for his work on early algorithms and tool support.

## References

- [Ates96] A. F. Ates, M. Bilgic, S. Saito, B. Sarikaya, "Using Timed CSP for Specification, Verification and Simulation of Multimedia Synchronization", IEEE Journal on Selected Areas in Communications, 14(1), New York: IEEE, January 1996.
- [Blair95] L. Blair, G.S. Blair, H. Bowman, A.G. Chetwynd, "Formal Specification and Verification of Multimedia Systems in Open Distributed Processing", Computer Standards and Interfaces 17, pp 413-436, Amsterdam: Elsevier Science, 1995.
- [Blair97a] G.S. Blair, J.-B. Stefani, "Open Distributed Processing and Multimedia", Harlow: Addison-Wesley, 1997.

---

<sup>2</sup> Currently on leave at CNR-Instituto CNUCE, Via S. Maria 36, 56126 Pisa, Italy.

Temporal LOTOS specification, this policy is entwined together with the rest of the behaviour. It is not *explicit* and, as with the first modification, changes to it require a significant part of the specification to be rewritten. However, by considering the policy as a separate aspect, the affected behaviour can easily be *identified* and the required *changes* can be made. Note that, in this example, the functional aspect of the specification requires no changes.

### 3.4. Dynamic QoS management policies

More recently, we have started to look at the issue of dynamic (or *adaptive*) QoS management policies. This adds yet another dimension of complexity to systems and consequently imposes even more requirements on formal specification techniques. With dynamic policies, it is necessary to be able to *monitor* and *control* the behaviour of a system. [Blair98b] describes an extension to the stream example, where the buffer of the data sink is monitored and, depending on its status, the rate at which frames are sent across the stream can be controlled (increased or decreased as required). Whilst this specification continues to make use of LOTOS and temporal logic, we now augment this with automata representing the dynamic behaviour. This behaviour is distinct from both the functional behaviour and the specification of QoS constraints. Consequently, QoS management policies are considered as a separate aspect of our specification. As with the synchronisation policy described above, this means that QoS management policies (and their associated monitors and controllers) are *explicit* and are not merged with the rest of the behaviour.

The monitors and controllers of this example illustrate a further benefit of our approach. Automata seem a natural choice of language for this aspect of the specification. The fact that we are not tied to one language for the entire specification is a major advantage: we can use a different language as and when *appropriate*. It could be argued that we could write the entire specification using automata. However, by doing so we would sacrifice the power of techniques such as LOTOS, with respect to their handling of abstraction, structuring and decomposition, data handling, etc, all of which grow in importance as larger, more complex systems are considered. With an aspect-oriented approach, we can *exploit* the *diversity* of different languages.

Note that we will address issues related to the synthesis of “real-world” monitors and controllers from these automata in section 5.

## 4. Further Issues

As can be seen from the above discussion, there are many advantages to an aspect-oriented approach to specification. However, there are also drawbacks that must be considered carefully. The most significant drawback is the added *complexity* introduced when validation is undertaken. As described in section 2 above, we have developed validation techniques over the composition of two languages (LOTOS and a temporal logic). However, this work must be extended to handle the inclusion and composition of further languages such as automata. A further drawback is that, as more languages are used (and more aspects are considered), the issue of *consistency* across the specification becomes crucial [Boiten96]. It becomes harder for the specifier (or specifiers) to keep track of the inter-relationship between different aspects. Indeed, this is one issue that we are addressing by separating them out in the first place. However, it is quite possible that two aspects result in some inconsistency or contradiction. For example, suppose we consider a further aspect, such as fault-tolerance, which requires some replication of data. This may contradict other requirements for real-time performance (or efficiency). Mechanisms must thus be included in the validation procedure to flag such inconsistencies.

quantitative behaviour. With such systems, any non-functional (quantitative) behaviour is often referred to as a *quality of service* property or constraint [Blair97a][Issarny98]. An important issue related to these properties is *quality of service management*, that is ensuring that the desired quality of service constraints are attained and, in the case of continuous media, sustained. The specifications introduced in sections 3.2. and 3.3. address QoS properties whilst our more recent work considers the additional impact of QoS management (section 3.4.).

### 3.2. A media stream

A stream consists of a flow of data of a single continuous media type (such as audio or video) from a source object to a sink object. The stream may encapsulate compression and decompression objects, and/or encryption and decryption objects. Specifications of this can be found in [Blair95] and [Blair98a]. These specifications use an approach based on a *separation of concerns* between the functional (qualitative) and the quantitative aspects of the system behaviour. In our approach, the process algebra, LOTOS, is used to specify the functional behaviour whilst a real-time temporal logic is used to specify the (quantitative) quality of service constraints. Probabilistic and stochastic behaviour relating to a stream are handled by using a probabilistic temporal logic as described in [Lakas96a] and [Lakas96b].

This example illustrates some important benefits to our approach. Firstly, by separating out the quantitative (real-time, probabilistic and stochastic) aspects, the remaining specification of behaviour contains no performance or implementation considerations and, in this sense, is totally *abstract*. Secondly, the QoS constraints are immediately *identifiable* and can easily be *changed* if necessary. The example in section 3.3. will illustrate this point further. An additional benefit is that the approach allows us to exploit *existing* languages (and tools) without having to develop extensions or search for one language to cover all behaviour.

### 3.3. Lip-synchronisation

The lip-synchronisation problem requires the synchronisation, at the presentation device, of an audio stream and a video stream. This is an interesting problem since it not only requires temporal constraints to be achieved and maintained along each individual stream, but also *across* streams. Various formal descriptions of this problem have been published which use a single specification language, for example Temporal LOTOS [Regan93], Timed CSP [Ates96] and Esterel [Blair97a]. These specifications all make the assumption that, although jitter is allowed on the video presentation, no jitter is allowed on the sound presentation. An aspect-oriented specification of this lip-synchronisation algorithm can be found in [Blair97b] and [Blair98a]. Importantly, the latter compares our specification with one using a more traditional (single language) approach, namely that of Temporal LOTOS [Regan93]. The comparison highlights the concise and comprehensive nature of the quantitative behaviour when compared to the same information in Temporal LOTOS. More importantly, the work also considers the impact on each specification of two possible modifications to the lip-synchronisation algorithm:

- firstly, relaxing the presentation of sound to allow a small measure of jitter, and
- secondly, adopting a more complex lip-synchronisation strategy (removing some non-determinism relating to the presentation time to achieve the best possible lip-synchronisation).

The first of these modifications illustrates the ease with which *changes* in real-time behaviour can be accommodated in our approach. In contrast, significant changes and restructuring are required to the Temporal LOTOS specification since fragments of real-time information are scattered throughout the specification. The second modification addressed a change in lip-synchronisation strategy. In the

functional behaviour as one aspect of our specification. However, as will be seen in section 3.4., the quantitative behaviour incorporates both QoS constraints and QoS management policies. We view these as two further (distinct) aspects of a specification.

## 2.2. Choice of languages

As mentioned in the introduction, there are many different specification languages to choose from. Here is not the place to discuss the merits of each individual approach, nor is the choice crucial to the principle of aspect-oriented specification. What is important is that we do not require the same language to be used for different aspects. Our work has focused on a process algebra (LOTOS) for an abstract specification of functional behaviour and has evaluated temporal logic and automata (and real-time and probabilistic extensions to each) for the specification of quantitative aspects (see, for example, [Lakas96a], [Blair98a] and [Blair98b]). Other work elsewhere (e.g. [Zave93], [Zave96]) has considered the use of Z along with automata, Petri nets and formal grammars.

## 2.3. Aspect-weaving

In aspect-oriented *programming*, it is the aspect-weaver's job to combine the behaviour of the component program (equivalent to our functional specification) and any aspect programs. The result is a program consisting of "tangled" code (in C or Java)<sup>1</sup>. It is clear that we will need an equivalent process for aspect-oriented *specification*.

In the world of specification, the crucial feature which underpins everything else is the *semantics*: they must be clear, concise and, above all, unambiguous. For our aspect-oriented specification technique, we must additionally deal with the semantics of the *composition* of two or more aspect specifications. In our approach, the result of this composition is an automata-style *global model*. Because of the nature of the languages we have chosen, our global model can be generated from the composition of a labelled transition system (derived from LOTOS), *event schedulers* derived from temporal logic formulae, and from automata directly. A description of event schedulers and an algorithm for their composition with a labelled transition system is presented in [Lakas96a].

## 2.4. Validation

Given that we are in the specification world, the issue of validation is also of vital importance. Having generated our global model, there are various validation techniques that can be applied. One way would be to consider user requirements (perhaps written in a logic), and to carry out a formal proof that these requirements satisfy the overall specification, for example based on model-checking techniques (two such techniques are presented in [Blair98a] and [Lakas96a]). A second possibility would be to do some performance analysis on our global model (algorithms underpinning a non-exhaustive simulation tool that we have developed are presented in [Lakas96b] and [Lakas96c]). A third possibility, based on the synthesis of monitors and controllers, is the subject of ongoing work (see also section 5.).

# 3. Experiences with Aspect-Oriented Specification

## 3.1. Overview

The following sections summarise our experiences of applying an aspect-oriented approach to the specification of *distributed multimedia systems*, focusing on the distinction between functional and

---

<sup>1</sup> Note that at least one presentation in [Mens97] proposed that certain aspects should not die, but should survive separately (e.g. to permit run-time adaptation of an aspect) [Matthijs97].

# **The Impact of Aspect-Oriented Programming on Formal Methods (Position Paper)**

*Lynne Blair, Gordon S. Blair*

*Computing Department, Lancaster University, Bailrigg, Lancaster, LA1 4YR*

*Internal Report No: MPG-98-08, May 1998*

*e-mail: lb,gordon@comp.lancs.ac.uk*

## **Abstract**

This position paper considers how the principles of aspect-oriented programming can be applied to formal methods. By specifying different aspects of a system in different formal languages, the diversity and power of existing specification languages can be exploited. We describe our experiences of applying such an approach in the field of distributed multimedia systems. The significance of an aspect-oriented approach to the specification of adaptive QoS management functions is also considered.

## **1. Introduction**

There are many different formal specification techniques already in existence. Importantly, these languages exhibit great diversity both with respect to the style of language (syntax and semantics) and the domain of applicability. This broad spectrum of techniques can be classified into various categories such as specification logics, process algebraic techniques, automata-based techniques, Petri nets, synchronous languages, and more software engineering oriented languages (such as Z). A survey covering such categories can be found in [Blair98a].

In general, extensions to these languages are regularly being developed. Typically, such extensions address a shortcoming of many specification techniques: whilst they are very good at capturing the functional (qualitative) behaviour of a system, few have the ability to specify the more quantitative aspects of systems. For example, many extensions address the specification of systems exhibiting real-time behaviour, hybrid behaviour and probabilistic/ stochastic behaviour.

However, all such extensions, by their very nature, have one thing in common: the functional (qualitative) behaviour of the system and the quantitative behaviour become entwined, or tangled together, in the specification. For fairly simple specifications, this may cause no problem. However, as larger and more complex specifications are tackled, the entwining of different aspects makes it difficult to develop, understand and make modifications to a specification. This is precisely the same problem addressed by aspect-oriented programming. In this paper we present, from a specification perspective, some of our experiences regarding the separation of different aspects of a system.

## **2. Aspect-oriented specification**

### **2.1. Choice of aspects**

Following the principles of aspect-oriented programming [Kiczales97], we propose an aspect-oriented approach to specification. An important question is, however, what aspects should we consider? As in the programming world, there are different answers to this question for different applications. For example, the Aster project at IRISA distinguishes between functional, interaction and non-functional properties (examples of the latter include timeliness, fault-tolerance, concurrency control, etc. [Issarny96][Issarny98]). Our work has, to date, focused on the distinction between the functional (qualitative) behaviour and the quantitative behaviour of a specification. We view the