Issues in multiparadigm viewpoint specification

Eerke Boiten, Howard Bowman, John Derrick and Maarten Steen
Computing Laboratory, University of Kent, Canterbury, CT2 7NF, UK.
(Email: J.Derrick@ukc.ac.uk.)

Abstract

This paper discusses the issues of specification style and refinement that arise in connection with viewpoint modelling. In particular, we consider the support needed in order to deal with viewpoints written at different levels of abstraction. The motivation for this work arises from the use of viewpoints in distributed systems design, in particular in the Open Distributed Processing standard.

Key words: Consistency; representation; Open Distributed Processing; Z; LOTOS.

1 Introduction

In this paper we discuss some issues in multiparadigm viewpoint specification. These issues have arisen from our work on the use of viewpoints in distributed system design, and in particular the need for consistency checking of viewpoints.

Open Distributed Processing (ODP) [11] is a joint standardisation activity of the ISO and ITU. A reference model has been defined which describes an architecture for building open distributed systems. Central to this architecture is a *viewpoints* model. This enables distributed systems to be described from a number of different perspectives. There are five viewpoints: *enterprise*, *information*, *computational*, *engineering* and *technology*. Requirements and specifications of an ODP system can be made from any of these viewpoints.

The reference model is thus a very general framework that aims to cover all aspects of distributed systems, but the viewpoints have a fixed and pre-determined role, covering a number of different aspects of design. In this respect it differs from some other viewpoint approaches, for example [10], which focus on one particular phase of the software engineering life-cycle, e.g. requirements specification.

Inherent in any viewpoint approach is the need to check or manage the *consistency* of viewpoints and to show that the different specifications do not impose contradictory requirements. The mechanisms needed to do this depend on the viewpoint languages used, and we have a particular interest in the use of formal techniques because the ODP reference model places an emphasis on the use of formalism. The reference model includes an architectural semantics which describes the application of formal description techniques (FDTs) to the specification of ODP systems. Of the available FDTs, Z is likely to be used for at least the information, and possibly other, viewpoints (the information viewpoint of the ODP Trader specification is being written using Z), whilst LOTOS is a strong candidate for use in the computational and engineering viewpoints.

Our concerns thus focus on how to relate viewpoints which have been specified in different formal languages at different levels of abstraction. Because we are dealing with different languages and levels of abstraction, our work relies heavily on the use of refinement. We define a collection of viewpoints to be consistent if and only if a common refinement can be found, i.e. a specification that refines all the original viewpoints (each with respect to a particular refinement relation). Of course the choice of refinement relation to apply to each of the different viewpoints is critical.

The strategy we envisage to check the consistency of one ODP viewpoint written in Z with another written in LOTOS is as follows. First translate the LOTOS specification to an observationally equivalent one in Z using results from [9], then use the mechanisms defined in [2, 4] to check the consistency of the two viewpoints now both expressed in Z. These mechanisms attempt to find a common Z refinement of the two viewpoints - if one exists the viewpoints are consistent.

However, these mechanisms deal with viewpoints writ-

To appear in the Viewpoints in Software Development Workshop at ACM SIGSOFT '96, Fourth Symposium on the Foundations of Software Engineering (FSE4), San Francisco, October 1996.

ten at the same level of abstraction, and they need to be extended to deal with differing levels of abstraction by using appropriate methods of refinement. Equally importantly, we need to develop specification styles that naturally allow a separation of concerns compatible with viewpoint modelling and consistency checking. We discuss these issues in section 3, whereas section 2 describes our current work in this area.

2 Current work

There have been four main thrusts to our work on ODP viewpoint consistency: a general framework for defining and interpreting notions of consistency, techniques for LOTOS, techniques for Z, and techniques for relating LOTOS and Z.

A general framework

At one time, the ODP reference model alluded to three different definitions of consistency. This is clearly an undesirable situation, which can be resolved by adapting a formal framework like we have described in [5]. Our definition of consistency is general enough that it encompasses all three ODP definitions of consistency between specifications, and even the usual notion of consistency within a specification. A crucial role in this framework is played by (what we call) *development relations* which formally relate specifications during the development process (e.g., conformance relations, refinement translations, or semantic mappings). Because the viewpoints have such different roles, for every viewpoint potentially a different development relation applies (for example, it might be conformance for the engineering viewpoint and refinement for the information viewpoint), even if the viewpoints are specified in the same language. The issue of consistency only comes up because viewpoints may overlap in the parts of the envisaged system that they describe. For example, the enterprise viewpoint may prescribe a security policy which will have to be implemented in the engineering viewpoint. In simple examples, these parts will be linked implicitly by having the same name and type in both viewpoints – in general however, we may need more complicated descriptions for relating common aspects of the viewpoints. Such descriptions are called *correspondences* in ODP. We define a collection of viewpoints to be consistent if a common development of all of them via the respective development relations exists which respects the correspondences between the viewpoints.

Besides a definition of consistency in this framework, we have also investigated methods for constructively establishing consistency [3]. An important notion in that context is that of a *unification*: a (least developed) common development of two specifications according to their respective development relations. Using unifications as the intermediate results, global consistency of a set of viewpoints can be established by a series of binary consistency checks, assuming a few reasonable restrictions on the development relations involved.

Consistency in LOTOS

What makes viewpoint consistency checking in LOTOS a particularly challenging task is the existence of a large collection of development relations for LOTOS. These characterise different ways in which LOTOS specifications can be viewed as partial specifications, with development being e.g. conformance, functionality extension, or reduction of non-determinism. We have characterised and compared many of the consistency relations induced by various combinations of development relations [5]. Also, we have obtained syntactic definitions of various kinds of unifications, some of which are guaranteed to be least common developments [13].

Consistency in Z

For Z as a viewpoint specification language we have so far assumed the established states-with-operations specification style, with an eye towards encapsulation of these as in object-oriented variants of Z [8]. In this style, finding a unification, i.e. a least common refinement, is an almost syntactical operation. For any two viewpoint specifications, we can construct a candidate unification, which *is* the unification if one exists [2]. Two relatively simple conditions characterise whether it is indeed a refinement. Other work in this area includes [1].

A vital ingredient in Z unification is the correspondence between the viewpoints. Our work has shown that we cannot rely on naming alone to determine which parts of which viewpoints refer to the same object – in particular, if one variable has different types between the viewpoints, we need to relate these types. For this, we require an explicit correspondence relation, and unification and consistency are relative to this correspondence relation.

Relating LOTOS and Z

Using viewpoints written in LOTOS and Z requires that we bridge a gap between completely different specification styles. Both languages can be viewed as dealing with states and behaviour – in Z the states are fully detailed, and operations (transitions) are only there to provide us with the next state; in LOTOS, the behaviour is what gets specified, and states form a useful concept for capturing information about what transitions might happen next. Our solution for consistency checking between these two languages so far has adapted a more behavioural interpretation of Z. A common semantics for LOTOS and a subset of Z in extended transition systems is used to validate a translation from LOTOS into Z [9]. Then the unification techniques for Z can be applied to determine consistency. However, knowing that both viewpoints are consistent (after translation) with respect to Z refinement may not always be enough. The LOTOS viewpoint had an associated development relation, which does not necessarily correspond to Z refinement under translation. Thus, we have begun to investigate how the development relations in Z and LOTOS relate, with interesting and promising results [6].

3 Issues

As has been suggested already the heart of our consistency checking strategy is to identify common refinements for the multiple specifications. Such refinements can also be viewed as common *models* for the multiple specifications. Such a common model will typically be expressed in terms of some set of primitive entities, examples of typical entities are:

- actions or operations, e.g. *acceptMessage* and *deliverMessage* in a communication protocol or *pickfork* and *putfork* in a dining philosophers specification.
- data variables, e.g. the value of a variable *seqno* which models the sequence numbers in a communication protocol.

The common semantic notation used in [9] incorporates both these kinds of primitive entities into a single *Extended Transition System* notation.

However, the approach of seeking a common model expressed in terms of a set of primitive actions is problematic. Firstly, if the multiple specifications are developed completely independently of one another it is almost certain that the primitive entities used in the two specifications will be quite different. This then poses the problem of what primitive entities should the common model be expressed in terms of. The notion of correspondence helps, as it enables corresponding entities in the source specifications to be related and then suitable renaming can take place to locate a primitive set of entities which is common to all the source specifications. However, this is certainly not a complete solution. Different specifications will be expressed at different levels of abstraction (this is especially true of ODP viewpoints), thus, identifying one-to-one correspondences is almost certain to be impossible. In fact, these correspondences can be extremely complex with what are primitive entities in one viewpoint being related to whole portions of behaviour in another viewpoint. For example, the execution of a remote procedure call operation in the computational viewpoint would actually correspond to a body of primitive interactions in the engineering viewpoint, e.g. interactions between stub objects, binding objects and protocol objects in order to invoke an RPC transport protocol.

Such changes of abstraction level are extremely hard to handle in viewpoints modelling and consistency checking, since the models of the two viewpoints are expressed in terms of different (but non-independent) primitives, thus, hindering the search for a common model. There are two different approaches that we envisage for resolving this problem:

- Action Refinement; and
- Promotion.

We will consider these in turn.

Action Refinement. This approach applies to the problem of relating actions at different levels of abstraction in multiple specifications. It fits most naturally into a process algebra setting where actions serve as the primitive unit of computation. We will thus, discuss it with reference to LOTOS. The basic approach is to incorporate into refinement a change of action granularity. For example, if we consider a specification of the behaviour of an end-to-end communication as follows:

$$acceptMessage;\ commMessage;\ deliverMessage;\ stop$$

The only LOTOS syntax we have used is action names, *acceptMessage*, *commMessage* and *deliverMessage*; the deadlock behaviour *stop*, which does nothing; and action prefix - ; - which states that an action must precede a behaviour. The specification states that a message is accepted (at a sender side) some communication action is performed and then the message is delivered (at the receiver side). This behaviour could be action refined to the following:

$$acceptMessage;\ conSetup;\ transmitData;\\ conDisconnect;\ deliverMessage;\ stop$$

where the action *commMessage* has been action refined into the "partial behaviour" *conSetup*; *transmitData*; *conDisconnect*. The first behaviour could be viewed as more "abstract" in its modelling of the transmission process; the actual mechanism for communication is abstracted away from and represented by a single action.

This is exactly the kind of relationship between primitives that we would like to employ. It would enable us to relate specifications at different levels of abstraction to the same unification. For example, a first viewpoint specification, expressed in terms of coarse grain primitives, could be action refined to a model that is expressed in terms of the finer grained primitives of a second viewpoint specification. This would fit nicely into the consistency checking framework that we have already identified.

Such action refinement has been quite extensively investigated within the process algebra field. Although, it should be pointed out that little work has to date been performed in the context of LOTOS. Action refinement has proved a hard problem to resolve. In particular, it has been realised that it is difficult to handle in the context of an interleaving semantics (which is the standard approach). This is because central to the interleaved interpretation of independent parallelism is the assumption that actions are atomic. For example, consider the behaviour:

$$a;\ stop\ |||\ b;\ stop$$

where ||| denotes independent parallelism and states that two behaviours will evolve concurrently without any communication. This specification would be modelled equivalently as:

$$a;\ b;\ stop\ []\ b;\ a;\ stop$$

where [] is the choice operator, which states that either the action a will happen before b or b will happen before a. This is only a reasonable interpretation of concurrency if the occurrences of a and b cannot overlap in time. Clearly, if actions can be refined into arbitrarily complex behaviours, the assumption of atomic actions is lost.

Current research has suggested that true concurrency models are more well behaved in the presence of action refinement [14]. True concurrency models do not rely on the assumption of atomic actions. All our work to date has been performed in an interleaving setting. We are currently investigating the feasibility of moving to a true concurrency setting in order to offer a resolution of this problem.

Promotion

Promotion is a technique often used in Z specifications for combining specifications at different levels of abstraction. This technique can be profitably used for specifying *viewpoints* at different levels of abstraction as well, as shown in [7] with viewpoints defining the dining philosophers problem, and in the specification of a telephone system in [12].

Promotion works when a global operation on a number of components is defined in terms of a local operation on a single component. Lapsing into Z, the global operation could be defined by (where *Local* is the local state, and *Promote* is a special promotion schema)

$$\exists \Delta Local \bullet Promote \wedge LocalOperation$$

Global operations could even be defined in terms of multiple (possibly different) local operations on different instances of the local state, i.e. they may change the state of several local components at once. An example of this is an operation which represents one telephone user ringing another: the state of one telephone is changed from *dialling* to *ringingtone*, and the other state from *free* to *ringing*, where both of these state changes would be represented by local operations.

This promotion technique can be used in a very powerful way across viewpoints, giving possibilities for top-down decomposition of operations and modularisation — which are the main consequences of having viewpoints at different levels of abstraction. The way to do this is as follows: one viewpoint defines the global operations, but also this viewpoint includes the local state and its operations – but only their signatures (in Z terms, by including them as empty schemas). The global viewpoint thus does not make any assumptions about the local state and operations apart from their existence. Another viewpoint will then actually *define* the local state and its operations. This models the situation where one viewpoint provides the implementations of standard components to be used in another one. This is exactly the relationship that arises between a number of the ODP viewpoints. For example, the engineering viewpoint provides standard communication components that are assumed when describing a computational viewpoint specification. A welcome advantage of this specification style is that unification techniques as we have defined them for Z will deliver the correct combination of viewpoints when such viewpoints are specified in this promotion style – namely, the syntactic inclusion of the local viewpoint.

Using this technique allows us to not only have Z viewpoints at an equal level of abstraction, but also to model the situation where one viewpoint provides an implementation module for another. This partially resolves the problem mentioned earlier, that correspondence relations could be extremely complicated. With this viewpoint specification style, most of the complexity gets moved into the global viewpoint.

4 Conclusions

Action refinement and promotion offer different solutions to representation and consistency in viewpoint modelling. Viewpoint modelling is dependent on specification styles adopted in the individual viewpoints and the development (or refinement) relations that are used on the viewpoints. Inappropriate styles or relations hinder the specification and development of viewpoints.

Promotion seeks to define a particular style of viewpoints and their relationship to each other, with the aim of providing natural separation of concerns between the viewpoints and ease of later combination and consistency checking. Action refinement seeks to provide a suitable development relation that can be used between viewpoints of different levels of abstraction.

A full account of viewpoint modelling in ODP would provide a viewpoint architectural semantics which defines the relationship between the viewpoints, both in terms of prescriptive templates (e.g. specification styles) and development relations between the viewpoints. Central to such an architectural semantics would be the correspondence rules between the viewpoints.

We are currently performing a medium-sized case study in ODP specification and consistency checking in which different abstraction levels and issues of state-based versus behaviour-oriented specification play a significant role. More information about our work (which is partially funded by British Telecom Research Labs. and the Engineering and Physical Sciences Research Council under grant number GR/K13035.) can be found at: `http://alethea.ukc.ac.uk/Dept/Computing/Research/NDS/consistency`

References

[1] M. Ainsworth, A. H. Cruickshank, L. J. Groves, and P. J. L. Wallis. Viewpoint specification and Z. *Information and Software Technology*, 36(1):43–51, February 1994.

[2] E. Boiten, J. Derrick, H. Bowman, and M.Steen. Consistency and refinement for partial specification in Z. In M.-C. Gaudel and J. Woodcock, editors, *FME'96: Industrial Benefit of Formal Methods, Third International Symposium of Formal Methods Europe*, volume 1051 of *Lecture Notes in Computer Science*, pages 287–306. Springer-Verlag, March 1996.

[3] H. Bowman, E. Boiten, J. Derrick, and M. Steen. Strategies for consistency checking, the choice of unification. Technical Report 5-96, Computing Laboratory, University of Kent at Canterbury, 1996.

[4] H. Bowman, J. Derrick, P. Linington, and M. Steen. FDTs for ODP. *Computer Standards and Interfaces*, 17:457–479, September 1995.

[5] H. Bowman, E.A.Boiten, J. Derrick, and M. Steen. Viewpoint consistency in ODP, a general interpretation. In *First IFIP International workshop on Formal Methods for Open Object-based Distributed Systems*, Paris, March 1996. Chapman & Hall.

[6] J. Derrick, H. Bowman, E. Boiten, and M. Steen. Comparing LOTOS and Z refinement relations. In *FORTE/PSTV'96*, Kaiserslautern, Germany, October 1996. Chapman & Hall. To appear.

[7] J. Derrick, H. Bowman, and M. Steen. Maintaining cross viewpoint consistency using Z. In K. Raymond and L. Armstrong, editors, *IFIP TC6 International Conference on Open Distributed Processing*, pages 413–424, Brisbane, Australia, February 1995. Chapman and Hall.

[8] J. Derrick, H. Bowman, and M. Steen. Viewpoints and Objects. In J. P. Bowen and M. G. Hinchey, editors, *Ninth Annual Z User Workshop*, LNCS 967, pages 449–468, Limerick, September 1995. Springer-Verlag.

[9] J. Derrick, E.A.Boiten, H. Bowman, and M. Steen. Supporting ODP - translating LOTOS to Z. In *First IFIP International workshop on Formal Methods for Open Object-based Distributed Systems*, Paris, March 1996. Chapman & Hall.

[10] A. Finkelstein, J. Kramer, B. Nuseibeh, L. Finkelstein, and M. Goedicke. Viewpoints: a framework for integrating multiple perspectives in system development. *International Journal on Software Engineering and Knowledge Engineering, Special issue on Trends and Research Directions in Software Engineering Environments*, 2(1):31–58, March 1992.

[11] ITU Recommendation X.901-904 — ISO/IEC 10746 1-4. *Open Distributed Processing - Reference Model - Parts 1-4*, July 1995.

[12] D. Jackson. Structuring Z specifications with views. Technical Report CMU-CS-94-126, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, 1994.

[13] M. W. A. Steen, H. Bowman, and J. Derrick. Composition of LOTOS specifications. In P. Dembinski and M. Sredniawa, editors, *Protocol Specification, Testing and Verification, XV*, pages 73–88, Warsaw, Poland, 1995. Chapman & Hall.

[14] R.J. van Glabbeek. The refinement theorem for ST-bisimulation semantics. In *Programming Concepts and Methods*. Elsevier Science Publishers, 1990.