The Development of an Architectural Semantics for ODP

Richard Sinnott,
Department of Computing Science,
University of Stirling,
Stirling FK9 4LA,
Scotland
email: ros@cs.stir.ac.uk

Abstract

This paper provides an introduction to the role of Formal Description Techniques (FDTs) in the development of an architectural semantics for Open Distributed Processing (ODP). It gives a brief introduction to ODP in general and the Reference Model for ODP (RM-ODP). Following this, an outline of the reasons for the development of an architectural semantics, the problems associated with the development of an architectural semantics and issues surrounding the possible solutions to these problems are presented.

1 Introduction

The need for information in today's computerised world is vital. However, often this information may be located at a different place from where it is required. It may also be on a different type of computer or operating system using different languages and storage facilities.

ODP is an attempt at addressing the problems involved in making this distributed information available throughout a single distributed system in a consistent and reliable way.

In order to achieve this, a common understanding of the basic concepts and overall architectures of such systems is required. To this end, the RM-ODP is being [1] developed. This provides not only the common definitions and terms used when considering a distributed system, but also a general framework from which the coordinated production of standards for distributed systems may be achieved. This has the added advantage of enabling compatible standards to be produced concurrently.

This paper focuses on one particular part of the RM-ODP: Part 4, the Architectural Semantics. It identifies the necessity for the work, the problems associated with the work and the possible solutions to these problems.

The rest of the paper is organised as follows. Section 2 gives a brief introduction to FDTs, ODP and the RM-ODP. Section 3 identifies the need for the architectural semantics work. Section 4 identifies the approach taken in the architectural semantics work and the technical problems that exist in it. Section 5 deals with the limitations of the current work plan. Section 6 deals with problems in extending the scope of the architectural semantics work plan. Section 7 identifies future work necessary for the architectural semantics work to be successful. Finally, section 8 draws some conclusions on the architectural semantics work.

[1] At the time of writing the different parts of the RM-ODP are at different stages of development.

2 Background to Formal Descriptive Techniques

It is common knowledge that natural language is inadequate for giving precise specifications. Indeed this was the initial motivation behind the development of FDTs by ISO (International Organisation for Standardisation) and CCITT (International Consultative Committee for Telephony and Telegraphy).

FDTs allow for the unambiguous representation of requirements. The main advantages in their application with regard to standards development are in the improvement of quality and the expediency with which standards are produced.

FDTs arrived a little too late to have a major impact on the development of OSI (Open Systems Interconnection) standards. However, ODP represents a fresh start in which the benefits of the precision available in specifying systems through FDTs can be achieved.

2.1 Background to Open Distributed Processing

One definition of a distributed system is one in which the user of the system is unaware of the differences in a collection of computers and operating systems in which processes may run. The distinction between a distributed system and a networked system (*e.g.* as in OSI) is provided by the software rather than the hardware. That is, it is up to the software to provide the necessary layers of transparency that allow a system to be distributed.

ODP is already a major effort between ISO and CCITT [1, 2, 3, 4] which will lead to significant product development in the coming years. The ODP work identifies and attempts to provide a framework for distributed systems so that a variety of computers and networks provided by different vendors may be combined to form communicating systems. This has many far reaching consequences for information sharing and provision. It also requires many levels of transparency so that users of one system are unaware of the possible non-heterogeneity of the other system being used to satisfy requests, *e.g.* access, location and migration transparencies to name but three. These transparencies bring forward several other areas that need to be addressed by the ODP community, *e.g.* security.

To deal with the complexity of ODP systems, the ODP community has identified five different viewpoints from which an open distributed system should be considered. These are:

- *Enterprise Viewpoint*: this focuses on the expression of purpose, policy and boundary for an ODP system. That is, this viewpoint captures the requirements that justify and orient the design of the distributed processing system.

- *Information Viewpoint*: this focuses on the information and information processing functions in an ODP system. That is, the information structures and information flows are modelled and the constraints that apply to these are expressed, both from manipulation and management standpoints.

- *Computational Viewpoint*: this focuses on the expression of the functional decomposition of an ODP system, and of the interworking and portability of ODP functions. That is, this viewpoint deals with the operational and computational characteristics of the distributed processing system, and the processes which change the information.

- *Engineering Viewpoint*: this focuses on the expression of the infrastructure required to support distributed processing. From this viewpoint issues such as performance, dependability and distribution transparencies are dealt with.

- *Technology Viewpoint*: this focuses on the expression of suitable technologies to support distributed processing, including the hardware and software that comprise the local operating systems, input and output devices, etc.

Each of these viewpoints represents a different abstraction of the same original system. Using a viewpoint results in a consideration of the system focusing on a particular concern. Different viewpoints address different concerns, however there is likely to be some common ground between them. Each of these viewpoints has its own associated language through which the concepts and entities identified from the viewpoint may be dealt with.

2.2 Background to the Reference Model of ODP

Due to the rapid growth of distributed processing, a requirement for a co-ordinated framework for standardisation has been identified. This has been set out in the Reference Model of ODP (RM-ODP). It contains an architecture through which distribution, interworking and portability can be achieved, where portability may be regarded as the ease with which a given system may adapt to evolution.

The RM-ODP is based upon precise concepts derived from current distributed processing developments and, as far as possible, on the use of FDTs to specify the architecture. The RM-ODP uses an approach based upon the notion of objects, where an object may be regarded as an identifiable encapsulated aspect of some real world entity. This allows for the easier development of systems where inessential details of objects may be ignored. Some of the advantages of this technique in system development are given in [14] and [15].

The RM-ODP consists of four main parts. These are:

- *Overview and Guide to Use* (**Part 1**) [1]: contains an overview of ODP giving scope, justification, explanation of key concepts, an outline of the ODP architecture and explanatory material on how the RM-ODP is to be understood and applied by its users.

- *Descriptive Model* (**Part 2**) [2]: contains the definition of the concepts and analytical framework and notation for the description of distributed processing systems.

- *Prescriptive Model* (**Part 3**) [3]: contains the specification of the required characteristics that qualify distributed systems as open, that is, the constraints to which ODP standards must conform.

- *Architectural Semantics* (**Part 4**) [4]: "currently" contains a formalisation of the ODP modelling concepts defined in the Descriptive Model. This formalisation is achieved through interpreting each modelling concept in terms of different FDTs.

The aims of this paper are to address the current problems in the architectural semantics work (Part 4), specifically through consideration of the development of an architectural semantics for ODP using the FDTs LOTOS [6] and Z [7]. It should be pointed out however, that there is no real restriction to LOTOS and Z as to the choice of FDT used to develop the architectural semantics. Other possibilities exist which have an equally valid claim to be suited to the development of an architectural semantics for ODP, *e.g.* ESTELLE [8], SDL [9], RAISE [10]. The only limitation placed upon the choice of FDT is that it be either standardised or very widely regarded. However, for the purpose of this paper, LOTOS and Z will be studied as the main FDTs of international relevance to distributed systems, one reason for this being that they represent disparate FDTs, with LOTOS being constructive and algebraic whilst Z is more abstract and state oriented.

3 Necessity for Architectural Semantics

As stated previously, the use of natural language to define an architectural semantics is flawed in that the architecture is often ambiguous or open to interpretation. Thus FDTs have been chosen to generate an architectural semantics as they allow for the precise, mathematical definition of concepts. Apart from the direct advantage of providing precision that would not otherwise be available, using FDTs to give an architectural semantics allows for a comparison to be made of different FDTs when used to provide a formal description of the same standard. An architectural semantics also enables the semantic models of the FDTs to be compared to the standard modelling concepts.

It is often the case that writing specifications proves to be difficult due to poor specification structures to begin with. Thus having a correct architecture upon which specifications can be based removes many of the difficulties involved in the actual writing of specifications. By a similar argument, specifications written without being based on a well structured architecture tend, not only to be difficult to write, but hard to understand and difficult to modify and extend.

Having a good specification architecture is also very useful for problems that are not well defined by requiring detailed consideration of the informal problem statements. Thus attempting to formalise "messy" problems directly leads to "messy" specifications.

An architectural semantics may be regarded as a set of precisely defined architectural concepts. Examples of architectural concepts include notions such as action, object, interaction, interaction point, type and subtype to name but a few. These concepts may be termed the "building bricks" for more complex designs such as services and protocols. It should also be noted that the building bricks developed in an architectural semantics may themselves be built from other building bricks, thus the architectural semantics is hierarchical. The most basic building bricks of all are concepts such as action and interaction point.

It should be noted here that the objectives of an architectural semantics are not to redefine the concepts they are modelling, but to provide clarity so that formal interpretations of the same concept are the same, or at least offer a level of compatibility. The architectural semantics work should ensure a uniformity of style in writing formal descriptions of standards.

Thus when completed, the architectural semantics work should allow standards developers, in this case the standards that are to be generated from the RM-ODP framework, to have a clear unambiguous interpretation of the concepts contained within the RM-ODP. It should also provide specifiers with a library of definitions which they may use to specify ODP standards with.

Before continuing with the approach taken by [4], it is necessary to consider examples where an architectural semantics was not provided and the subsequent problems this caused.

Problems occurred in the Reference Model for Open System Interconnection in several of the basic concept definitions. For instance that of service primitive, service access point, service data unit, entity and endpoint to name but a few of the problems. For example service data units were not defined precisely enough which caused problems when specifications were written that contained these concepts. The main problem with this concept was that its atomicity was not defined. This had repercussions on whether protocol entities could forward parts of a service data unit before it arrived fully. Similarly with service primitives which were defined in [5], these could be interpreted in different ways. For example, imprecision existed in that it was not stated whether service primitives were atomic or not, or whether they could be engaged in by service users and service providers at the same time, or even whether invalid sequences of service primitives could exist. Similar problems of imprecision existed for the other concepts mentioned above.

Thus the generation of an architectural semantics for ODP is vital in uncovering these problems at as early a stage as is possible.

4 Approach and Current Scope of Architectural Semantics Work

The approach taken in the Part 2 of the RM-ODP [2], and hence in the architectural semantics work is an object-oriented one. One definition of an object-oriented approach [15] is an approach that has "classes and objects, inheritance and communication with messages". This is particularly useful for the ODP approach in that it enables inessential details of objects to be ignored, thereby making the problems involved in obtaining the necessary transparencies for an ODP system more manageable.

This object-oriented approach causes problems however, when it comes to the architectural semantics work. These problems primarily stem from the fact that the FDTs presently used are not object-oriented, that is they were developed before notions of object-orientation were widely used.

4.1 Limitations of FDTs

As the architectural semantics work has to deal with object-oriented concepts, two main possibilities arise. Either, an object-oriented style of specification with the present FDTs may be adopted or the use of an object-oriented FDT may be made. Each of these solutions causes its own problems however.

4.1.1 The Role of Specification Style

LOTOS and Z both represent two disparate FDTs, neither of which however, provides an object-oriented approach directly. They both allow for specifications styles giving varying degrees of "object-orientedness". More information on the role of specification styles for ODP may be found in [20], however, this does not consider the object-oriented style.

Specification Style in Z

The FDT Z does not handle many of the object-oriented concepts very well because, generally speaking, a Z specification may only be used to describe a single object. Hence notions such as communication and interaction between objects are not easily dealt with. Similarly notions such as type, deletion of an object, introduction of an object are problematic too.

Despite these problems, Z is still a very useful FDT with regard to the architectural semantics work, as it allows the formal description of a single object to be done clearly and succinctly.

There also exist approaches in which Z can be specified in an approximate object-based way [12]. A possible definition of an object-based approach [17] is one which allows for objects and classes, but does not allow for inheritance or communication with messages. Here object identifiers are used to obtain a system with more than one object (so called); the effect of an operation on an object is promoted (with a framing schema) to an effect on the system. This approach, however, does not handle classes of objects or allow for inheritance or communication with messages. It also does not allow for operations on more than one object. This style goes as far as possible without extending Z. It does however, allow for a sound theoretical base. It should

be noted, however, that with this approach the "objects" are not a language feature or regarded as first class values, so this approach is not really object-based using Wegner's definition [17, 18].

The conclusions to this with regard the architectural semantics work are either to accept the limitations of Z with regard to object-oriented concepts and use it to describe a single object, for which it is still a very useful FDT, or to agree upon a stable object-oriented version of Z, *e.g.* [11], and possibly use it instead of Z (or alongside it) in [4]. This however, requires a solid semantics being established for the object-oriented version and widespread agreement on its usage.

Specification Style in LOTOS

The FDT LOTOS allows for a formalisation (interpretation) of many of the object-oriented concepts contained in [2]. Often this requires that a given style of LOTOS is used. This brings with it new problems, however, as far as the architectural semantics work is concerned, namely prescriptivity and scalability. Other work with regard to the object-oriented style of specification in LOTOS may be found in [19] and [21].

A Problem of Prescriptivity

This problem is best seen through an example. The older version [2] of [4] interpreted the definition of an object from [2] as "a template instantiation — i.e. an instance of a LOTOS process", where a template was defined as "a process definition with formal parameters (though the value parameter list may be omitted)". This at first sight appeared a valid interpretation of an object in LOTOS. However upon further consideration it became apparent that it stated nothing about objects having unique identification.

In order for this to be correctly handled it puts restrictions on the specification style of LOTOS that may be used. That is, each instantiation of a process definition must have associated with it a unique identifier which will represent the object identifier.

In LOTOS there are different ways that a process definition may be assigned this unique identifier. For instance, when the process definition is instantiated or by the process definition having some initial behaviour which generates a unique identifier. Thus it is possible to have choices even within a given specification style. The architectural semantics work should be general enough to allow for these choices, but precise enough to ensure that when specified, the concept should satisfy one possible choice given in the architectural semantics work.

The question might be asked as to whether it is valid to restrict the users of the architectural semantics work to a certain style of specification. This will have repercussions in that it may not be easy to take any specification and identify the architectural semantic concepts contained within it. This can be countered however, in that it is up to the specifiers of ODP systems to use the architectural semantics work. Hence the restriction to a style of specification may be seen as a valid restriction.

It should be noted here, however, that the issue of prescriptivity may also have repercussions on the compatibility of the architectural semantics work for the given FDTs. That is, an operation in a LOTOS specification should have the same effect on a given system as an operation in a Z specification when used to describe the same system (standard). *i.e.* the specifications of the same system should describe the same behaviours. Prescriptivity may limit this. Ideally, there should be some form of compatibility between LOTOS and Z specifications of the same system. This will affect the re-usability of the standards generated from the RM-ODP. That

[2] The version before the last international ODP meeting in Torino, November 1993.

is, it is not just isolated standards that are striven for, but standards that can be combined and used by other standards. This search for commonality is not easy, *e.g.* LOTOS is more constructive whilst Z is more abstract. The architectural semantics work should identify and provide discussion on the problems here. The work contained in [22] addresses some of these problems in its attempt to combine Z and CSP [23], upon which the process part of LOTOS is partly based.

A Problem of Scalability

The problem of being too prescriptive introduces the problem of scalability. For instance it is possible to specify a form of inheritance in LOTOS provided that a very restrictive style of LOTOS is used, requiring that the inherited process have **exit** functionality. This means that any process having **noexit** functionality may never be inherited from. This is a severe restriction upon the specifier and one which is clearly not scalable.

The architectural semantics work should provide guidance on how specifiers may specify concepts such as inheritance. It should also however, provide warnings of the problems that the specification of these concepts may induce.

4.2 Introduction of Object-Oriented FDTs

Solutions to many of the problems that occur in the architectural semantics work would follow if an object-oriented FDT were used [3], some examples of which include POOL [16] and Object-Z [11]. However, an FDT's use in [4] requires it to be either standardised or stable. As no object-oriented FDT is either standardised or at the moment has the necessary stability to be included in [4], this approach has not been taken. It is likely that unless considerable work and international agreement takes place fairly soon, then it may be too late to include such object-oriented FDTs into the architectural semantics work of ODP.

4.3 Technical Difficulties with Architectural Semantics

Besides the more indirect problems facing the architectural semantics work like prescriptivity and scalability, several more immediately apparent technical problems exist. The notion of type and hence notions such as class, subtype and subclass are particularly difficult to interpret in the FDTs LOTOS and Z. Similarly, the notion of a location in space and time as identified in [2] proves to be problematic.

4.3.1 The Problem of Type

A type as defined in [2] is simply a predicate. An $< \mathbf{X} >$ is of the type, or satisfies the type, if the predicate holds for that $< \mathbf{X} >$. Here $< \mathbf{X} >$ might be an object, an interface or an action. The notion of type is implicitly associated with the notion of class, where a class is given by the set of all $< \mathbf{X} >$'s satisfying some type. As may be seen, this is a very general notion of type and one which proves to be problematic when interpreted in the FDTs LOTOS and Z. It should be noted however, that it is generally difficult to give a formal definition to the notion of type using an FDT, and not only the definition contained in [2] which proves to be problematic.

Both of the FDTs LOTOS and Z allow type predicates to be written. However, they do not allow for a general "all-purpose" notion of a type predicate to be written that can be applied to

[3] It should be noted that SDL'92 has claims to be object-oriented, however, it does not yet have a fully stable semantics.

all $< \mathbf{X} >$'s. The best that can be achieved is explicit examples of types given. Even through this approach establishing the type of an $< \mathbf{X} >$ may be a non-trivial task. [24] addresses many of the problems involved in establishing types and relationships between types.

4.3.2 The Problem of Location

[2] defines location as an interval of arbitrary size in time and space at which an action can occur. This causes problems in the FDTs LOTOS and Z in that notions of space and time are abstracted away from. To talk about explicit, absolute locations in metric space and time is outside the scope of the FDTs LOTOS and Z. The best that can be achieved is that an abstract interpretation of location based upon the structure of the given FDT is done.

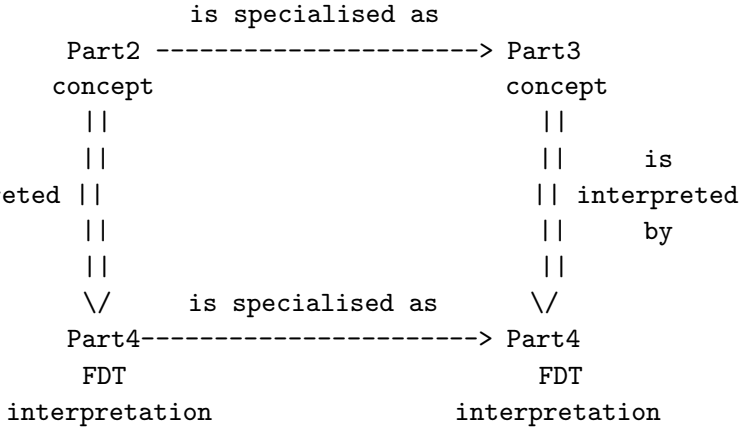
5 Limitations of Scope of Architectural Semantics Work

As stated above the current architectural semantics work considers the FDT interpretation of the basic modelling and specification concepts as found in [2]. This is a valid starting position in the development of an architectural semantics for ODP, however it is limited in its scope. If the architectural semantics work is to be regarded as a formalisation (or an interpretation) of the ODP concepts, then without addressing the specification of the required characteristics that qualify distributed systems as open, that is the concepts contained in [3], then the architectural semantics work will be severely limited in its use.

Ideally, the architectural semantics work should consist of a formal conceptualisation of the whole ODP-RM. Whilst this is a laudable goal it appears too difficult to achieve. Hence, the current architectural semantics work has been reduced to an FDT interpretation of basic modelling and specification concepts.

One could consider concepts contained in [3] to be a specialisation of the concepts contained in [2]. For example, a stream interface as found in [3] might be considered a specialisation of an interface as found in [2]. The problem with only considering an architectural semantics for the basic modelling and specification concepts as found in [2] is that there may be no interpretation of the specialisation of this concept. Consider the above example of an architectural semantics for the basic modelling concept of an interface and its specialisation into a stream interface in LOTOS. The basic modelling concept of an interface may be interpreted in LOTOS. However, its specialisation to a stream interface may not be directly interpreted in LOTOS, although an abstract representation of this concept may be achieved. The notion of flows of information and the event structure of LOTOS do not really go hand-in-hand.

Thus the relationship between the [2], [3], and their interpretation into [4] may be shown as



The top line of this rectangle is represented by the [2] to [3] relationship of the RM-ODP. The left and right hand sides of this rectangle represent the architectural semantics work.

Ideally the architectural semantics user should be able to specify systems from the different viewpoints (languages). Thus the bottom right corner is the ideal. However, the current architectural semantics work only addresses basic modelling and specification concepts, that is, the left hand side of this rectangle. As shown in the above example of interfaces in LOTOS, the bottom line may not exist directly. Where this is the case it should be noted and conclusions drawn. These conclusions need not necessarily make up part of the [4] document but could be included in a Technical Report.

The next question that arises from this is which of the viewpoint languages is to be dealt with. Ideally all of them, but realistically only the computational, information and engineering languages will be addressed in the time allowed [4]. Thus there should be a rectangle as above for these viewpoint languages and for each of the FDTs. It is expected that the FDTs LOTOS, Z and SDL[5] will be used to interpret the viewpoint languages. Other FDTs (ESTELLE, RAISE, etc) may also be used if member bodies are prepared to make contributions (and the FDT is very well known or standardised). If no contributions are received then it is likely that other FDTs will be rather left behind in the ODP work of Part 4.

6 Possible Problems of Including Prescriptive Concepts in the Architectural Semantics Work

The above arguments identify the advantages to be gained from inclusion of the more prescriptive concepts contained in [3] into the architectural semantics work contained in [4]. Care should be taken before this is directly approached however. Problems may crop up which could cause confusion as to the role of the architectural semantics work. One example in LOTOS of this might be that an object (an instantiated process definition with some means of unique identification) may not always be used in the different viewpoints. That is, can this method of generating an object in LOTOS be used for all of the ODP viewpoints? Is it possible or desirable to model enterprise, information, computational, engineering and technology objects from this method of generating an object in LOTOS? For example, if an information object were to be modelled in LOTOS then it might be better to model this through ACT ONE. This would allow computational objects to deal with information objects directly in LOTOS.

The problem of the relationships [6] between the viewpoints also becomes much more difficult to model and interpret when different FDTs are used. That is, it is difficult enough trying to establish the relationship between, say, an information object in LOTOS and a computational object in LOTOS. This problem becomes much more acute when the relationship between an information object in Z and a computational object in LOTOS is attempted.

Hence the straightforward approach of simply adding the concepts contained in [3] into the architectural semantics work needs to be approached carefully and methodically. To account for this, [25] provides an initial feasibility study on the development of an architectural semantics for

[4] Much of the international community expects the architectural semantics work to go to Committee Draft status in June 1994.

[5] [26] contains an architectural semantics for the computational viewpoint language of [3] in SDL'92.

[6] It should be noted that the viewpoints may have no relationship. However, it is likely that there will generally be some relationship between them. An example of this is the relation between the information and computational viewpoints. If the information viewpoint is regarded as modelling the information of the ODP system and the computational viewpoint as the functions which deal with this information, then some relation between the viewpoints must exist.

the computational language of [3] in LOTOS, the results of which, in the opinion of the author, reflect favourably in the use of LOTOS for the development of an architectural semantics for the computational language of [3].

7 Future Work

The current version of the architectural semantics work, as stated, deals only with the basic modelling and specification concepts contained in [2]. In order to make the architectural semantics work particularly useful and approachable, new areas need to be addressed.

The inclusion of the more prescriptive concepts contained in [3] needs to be addressed. As identified above, however, caution is needed here so as not to confuse the whole of the architectural semantics work.

It is likely that due to time limitations and political pressures, the inclusion of the more prescriptive concepts of [3] may exist as a separate part of [4]. That is, to base the current document wholly around a new work area would cause problems. Hence it is likely that the newly introduced concepts would exist as a freestanding section of the architectural semantics document. Thus if the work proved to be too problematic, then the whole of the document would not necessarily have to be scrapped [7].

Also identified for future work has been the need for a large and complete example of the formal specification of an ODP system, showing how the different viewpoints are related to one another (if at all). It is likely that this large example may be attempted only when considerable work has been done on the development of an architectural semantics for the different viewpoint languages. This example may then be used as a guideline for would-be users of the architectural semantics work.

Work also needs to be done on determining which of the FDTs is most suitable for the different viewpoints. Suggestions may then be made to the users of the architectural semantics work as to which FDT is most applicable to their needs.

Finally, the success or failure of the architectural semantics work depends to a great extent on the tutorial nature of its content. The use of a large example would aid in this by giving a "how it can be done" approach. However, many smaller examples showing the relationship of the ODP concept to its representation in the respective FDTs would also aid greatly in the success of the architectural semantics work. Hence a tutorial guide for the architectural semantics work is necessary.

8 Conclusions

This paper has provided an insight into the architectural semantics work of ODP. It has identified the need for the work and has covered issues surrounding the current problems with the work and the difficulties in finding solutions to these problems, not all of which are of a technical nature. Time and politics may play a large role in the success or failure of this application of FDTs. If this is not going to be a case of FDTs yet again not being fully exploited, then collaborative international work is required immediately to provide a sound and apposite architectural semantics for ODP. Both the basic modelling and specification concepts, and the specification of the required characteristics that qualify distributed systems as open must be completed.

[7] To speed up the process of attempting to develop an architectural semantics for the more prescriptive concepts contained in [3], an international email discussion group has been established. For more information regarding this and access to the discussion group, the reader may in the first instance contact the author at the email address given.

References

[1] Basic Reference Model of ODP — Part 1: [Recommendation X.901 - ISO 10746-1] Overview and Guide to Use of the Reference Model.

[2] Basic Reference Model of ODP — Part 2: [Recommendation X.902 - ISO 10746-2] Descriptive Model.

[3] Basic Reference Model of ODP — Part 3: [Recommendation X.903 - ISO 10746-3] Prescriptive Model.

[4] Basic Reference Model of ODP — Part 4: [Recommendation X.904 - ISO 10746-4] Architectural Semantics.

[5] Information Processing Systems — Data Communications — Network Service Definition. ISO-IEC, April 1987, IS8348.

[6] Information Processing Systems — Open Systems Interconnection — LOTOS — A Formal Description Technique based on the Temporal Ordering of Observational Behaviour; ISO/IEC 8807, International Organisation for Standardisation, Geneva.

[7] J.M. Spivey. Understanding Z : A Specification Language and its Formal Semantics. Cambridge University Press, 1988.

[8] ISO (1989e) Information Processing Systems - Open System Interconnection - ESTELLE - A Formal Description Technique Based on an Extended State Transition Model, ISO/IEC 9074, International Organisation for Standardisation, Geneva.

[9] Specification and Description Language, CCITT Z.100, International Consultative Committee on Telegraphy and Telephony, Geneva.

[10] Tutorial Notes - FORTE'93, Sixth International Conference on Formal Descriptive Techniques, Boston, Massachusetts, USA, October 26-29, 1993. Edited by Richard L. Tenney.

[11] Object-Z: an Object-Oriented Extension to Z, D.A. Carrington, D. Duke, R. Duke, P. King, G.A. Rose, G. Smith in Formal Descriptive Techniques FORTE'89 (North Holland, 1987) pp 313-341, ed S. Vuong.

[12] Using Z as a Specification Calculus for Object-Oriented Systems, J.A. Hall in VDM'90: VDM and Z - Formal Methods in Software Development, Lecture Notes in Computing Science 1990, 428, pp 290-318.

[13] Object-Oriented Process Specification, S.A. Schuman, D.H. Pitt, P.J. Byers in Specification and Verification of Concurrent Systems, Workshops in Computing, Stirling (Springer-Verlag 1990), pp 21-70, ed C. Rattray.

[14] Object-Oriented Design with Applications, Grady Booch. (Benjamin-Cummins, 1991).

[15] Object-Oriented Design, Second Edition, P. Coad, E. Yourdon, Yourdon Press, Prentice Hall Building, Englewood Cliffs, NJ 07632.

[16] Issues in the Design of a Parallel Object-Oriented Language, P.H.M. America, Formal Aspects of Computing 1 (1989), pp 366-411.

[17] The Object-Oriented Classification Paradigm, P. Wegner, Research Directions in Object-Oriented Programming (MIT Press 1987), eds P. Wegner, B. Shriver.

[18] Dimensions of Object-Based Language Design, P. Wegner, OOPSLA'87 Proceedings, ACM SIGPLAN Not, 1987, 22, (12), pp 168-182.

[19] LOTOS Design-Oriented Specifications in the Object-Based Style, Technical Report 84, Department of Computing Science and Mathematics, April 1992, University of Stirling.

[20] Use of Formal Specification Techniques for ODP, ISO/IEC JTC1/SC21/WG7 N753.

[21] Object-Oriented Specification Style in LOTOS, W.H.P. van Hulzen, LOTOSPHERE Work Package One, T1.1/RNL/N002 July 1989.

[22] A Message Passing System. An Example of Combining CSP and Z., M. Benjamin, Z User Workshop, Workshops in Computing, ed J.E. Nicholls, Oxford 1989, Springer-Verlag.

[23] Communicating Sequential Processes, C.A.R. Hoare, Prentice-Hall International, Englewood Cliffs, New Jersey, 1985.

[24] The Formal Specification in LOTOS of a Basic Type Manager, R.O. Sinnott, University of Stirling (*in preparation*).

[25] An Initial Interpretation in the FDT LOTOS of the Computational Viewpoint Language of Part 3 of the Basic Reference Model of Open Distributed Processing, R.O. Sinnott, University of Stirling.

[26] Formalisation of the Computational Description Language, ISO/IEC JTC1/SC21/WG7 N773, November 1993.