Some Results on Cross Viewpoint Consistency Checking

Howard Bowman, John Derrick and Maarten Steen

University of Kent at Canterbury, U.K. (hb5,jd1,mwas)@ukc.ac.uk.

The ODP multiple viewpoints model prompts the very challenging issue of cross viewpoint consistency. This paper considers definitions of consistency arising from the RM-ODP and relates these in a mathematical framework for consistency checking. We place existing FDTs, in particular LOTOS, into this framework. Then we consider the prospects for viewpoint translation. Our conclusions centre on the relationship between the different definitions of consistency and on the requirements for realistic consistency checking.

Keyword Codes: D.2.1, D.2.10
Keywords: Requirements/Specifications, Design

1. INTRODUCTION

Multiple viewpoints are a cornerstone of the Open Distributed Processing (ODP) model [12]; they enable a different perspective of a system to be presented to different observers. Each viewpoint is a partial view of the complete system specification. It is through this separation of concerns that the inherent complexity of a complete distributed system is decomposed. ODP supports five viewpoints: *enterprise*, *information*, *computational*, *engineering* and *technology*.

However, the subdivision of a system specification raises the issue of *consistency*. Descriptions of the same or related entities will appear in different viewpoints and it must be shown that the multiple specifications are not in conflict with one another. The development of tools and techniques to check the consistency of viewpoint specifications is of great importance, however, it is also extremely challenging. In particular, in its most general form, consistency checking requires specifications in different notations to be related. This is because it has been recognised that different notations are appropriate for different viewpoints. Relating model based specification notations, such as Z, to languages which explicitly model the 'temporal ordering' of abstract events, such as LOTOS or SDL, is particularly challenging.

This paper addresses the question: what is an appropriate definition for consistency? The RM-ODP is ambiguous in this respect. We will clarify the relationship between a number of possible consistency definitions and we will consider how different FDTs, in particular LOTOS, can be integrated into a consistency checking framework and then we will discuss the different options for translation. The results of the paper centre on the relative strengths of definitions and the information that needs to be made available in

[0]The work presented in this paper was partially funded by British Telecom Labs., U.K. and partially by the U.K. Engineering and Physical Sciences Research Council (grant number GR/K13035.)

order that an appropriate consistency check can be applied.

We consider consistency in very general terms. In particular, we do not consider specific instances of consistency, such as between the information and computational viewpoints. This reflects our adopted strategy, which is to clarify the general form of consistency as a relationship between arbitrary specifications before considering specific instances of consistency. This paper is reporting results of the initial, general, phase of our work.

The paper begins by exploring the extent of consistency relationships in ODP (in section 2). Section 3 discusses appropriate definitions of consistency arising from the RM-ODP and then section 4 relates these to a mathematical framework for consistency checking. Section 5 places existing FDTs into this framework. Then we outline a number of possible approaches to translation in section 6. Finally, we present concluding remarks in section 7.

2. THE EXTENT OF CROSS VIEWPOINT RELATIONSHIPS

Due to the central role viewpoints play, consistency relationships are extremely pervasive in ODP. Consistency arises in the following situations:-

Conformance Assessment. Conformance assessment for ODP is extremely broad. In particular, it encompasses both *conformance testing* (i.e. relating real implementations to specifications) and *specification checking* (i.e. specification to specification relationships), this distinction was particularly emphasised in PROST [7]. Verification of cross viewpoint consistency is an important example of specification checking.

System Development. The RM-ODP does not prescribe a particular system development methodology and a number of development methodologies could be envisaged. However, each viewpoint specification is, at least potentially, at the same level of abstraction; suggesting that viewpoints are related horizontally relative to a vertical system development. This is in contrast to classic waterfall development methodologies. PROST [7] has investigated such a, fully general, system development methodology for ODP. This is depicted in figure 1 and uses a number of specification to specification transformations, such as *translation*, *refinement* and *unification*, in order to generate a composite 'implementation' specification. Translation maps specifications into new languages, refinement has the usual meaning and unification is a transformation which enables specifications in the same language to be combined. Consistency is implicit in such a system development methodology. For example, two specifications would be viewed as inconsistent if a common unified specification did not exist. Thus, consistency arises during unification of specifications in models of ODP system development.

Architectural Semantics. The use of different FDTs in defining the ODP architectural semantics and the fact that the architectural semantics (when complete) will span a number of the viewpoint languages suggests consistency relationships will have relevance in this domain as well. Two forms of consistency relationship can arise. Firstly, there is a need to relate the architectural semantics of different viewpoints in order to determine that the FDT interpretations are consistent. Secondly, there is a need to demonstrate that descriptions in different FDTs of particular architectural semantics entities are consistent.

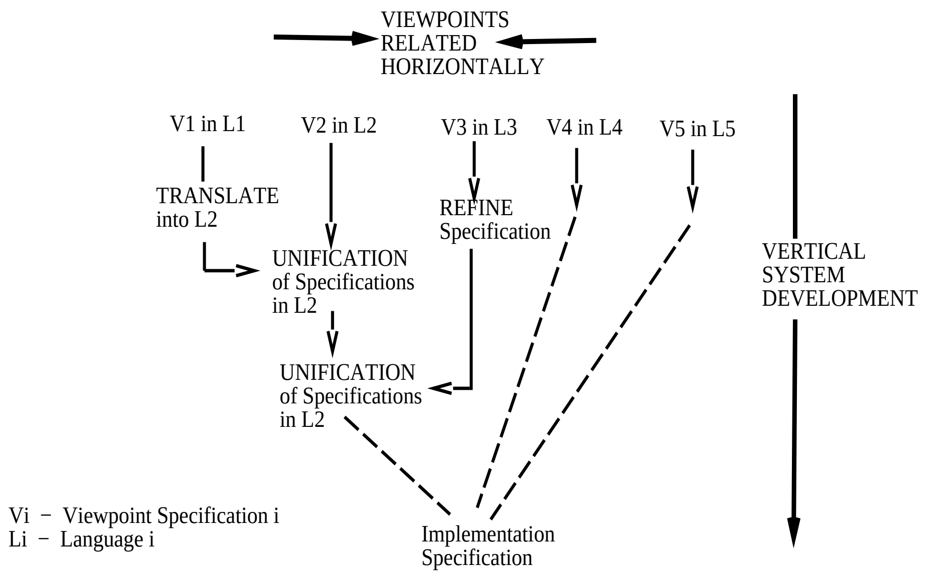

VIEWPOINTS
RELATED
HORIZONTALLY
V1 in L1
V2 in L2
V3 in L3
V4 in L4
V5 in L5
TRANSLATE into L2
REFINE Specification
UNIFICATION of Specifications in L2
UNIFICATION of Specifications in L2
VERTICAL SYSTEM DEVELOPMENT
Implementation Specification
Vi – Viewpoint Specification i
Li – Language i

Figure 1: PROST System Development Scenario

We strongly believe that a formal approach to consistency checking should be employed. In particular, the ability to reason rigorously about the specifications under consideration is of vital importance. We will assume the use of formal description techniques as viewpoint languages in the remainder of this paper.

3. CONSISTENCY DEFINITION

This section highlights three possible interpretations of consistency that appear in the RM-ODP, the first two appear in part 1 (clause 12.2) and the third apears in part 3 [12] (clause 10). Although, the first of these definitions is only alluded to; it is not formally proposed as a definition.

Definition 1
(1.1) Two specifications are consistent iff they do not impose contradictory requirements.
(1.2) Two specifications are consistent iff it is possible for at least one example of a product (or implementation) to exist that can conform to both of the specifications.
(1.3) Two specifications are consistent iff they are both behaviourally compatible with the other.

This last interpretation is a rewording of the RM-ODP definition. This is because the RM-ODP definition is expressed in terms of relating specific viewpoints. We are considering more generalised notions of consistency, thus, we have brought the definition into line with the other definitions in order to facilitate a direct comparison. In addition note, that all these definitions are symmetric, i.e. if a specification S is consistent with a specification R then R is consistent with S. This is a reasonable intuitive requirement for consistency.

Behavioural compatibility is defined as follows:

Definition 2 (Behavioural Compatibility) *An object is behaviourally compatible with a second object, with respect to a set of criteria, if the first object can replace the second* *object without the environment being able to notice the difference in the objects behaviour on the basis of the set of criteria.*

These three consistency interpretations blur over the fact that specifications may be in different FDTs and that it may not be possible to relate specifications directly without some element of translation. In fact, in the RM-ODP the third of these definitions includes a notion of translation which is described in terms of 'information preserving' transformations between languages. Translation will be discussed in section 6.

Each of these notions of consistency is intuitively reasonable. However, the question arises: what is the relationship between the interpretations and, in particular, are these definitions of consistency themselves consistent? In fact, the different interpretations are likely to be applicable in different settings. For example, definition 1 is relevant to consistency checking in a logical setting, e.g. in an FDT such as Z which is based on first order logic.

We seek to reconcile these interpretations through formalisation. We formalise the first notion of consistency as follows,

Definition 3 $S_1\ C_1\ S_2$ *iff* $\neg(\exists \psi\ s.t.\ S_1 \models \psi\ \wedge\ S_2 \models \neg\psi)$

where $\models$ is the satisfaction relation of the specification's logic. This definition states that two specifications are consistent if and only if there is no property that holds over one of the specifications and its negation holds over the other specification.

To interprete consistency 1.2 we need a formal interpretation of conformance. There is a difficulty here because conformance relates implementations to specifications and implementations are not amenable to formal interpretation. The classical approach to handling this difficulty is to only consider conformance up to a, so called, *implementation specification*. This is a specification that describes a real implementation in as much detail that a direct mapping from the implementation specification to the real implementation can be found. Thus, it is normal just to consider conformance relations between specifications, see [4] [5] [14] for typical approaches. However, implementation specifications relate to real implementations in different ways for different FDTs and, in particular, for some FDTs not all implementation specifications are implementable. For example, a Z specification that contains an operation $[n! : N | n! = 5 \wedge n! = 3]$ has no real implementation.

Our approach then is to divide conformance testing into two parts. Firstly, we consider conformance up to implementation specifications, using a relation $conf \subseteq SPEC \times SPEC$, and then we consider conformance of implementation specifications to real implementations, using a relation $\mathbf{conf} \subseteq SPEC \times IMP$ [1]. Where $SPEC$ is the set of possible ODP specifications and IMP is the set of possible ODP implementations.

By way of clarification, $S_1 conf S_2$ expresses the property that specification S_2 conforms to specification S_1, i.e. according to tests derived from S_1, S_2 cannot be distinguished from S_1. It should be noted that we have not specified how and what form of tests are derived from S_1; there are many options for such derivation [4] [5]. In a similar way $S\mathbf{conf}I$ expresses the property that I conforms to S. Interpretation 1.2 is now formalized as:-

[1]The order of our relations is in accordance with Z conventions and is opposed to LOTOS conventions

Definition 4 $S_1 \; C_{2.1} \; S_2$ *iff* $\exists S \in SPEC, I \in IMP \; s.t. (S_1 conf S \; \wedge \; S_2 conf S) \; \wedge S$ **conf** I.

i.e., two specifications are consistent iff an implementation specification which conforms to both and a real implementation of the implementation specification can be found. This definition is correct, but is not very useful since it uses **conf**, which is not subject to formal interpretation. In order to resolve this difficulty we introduce the concept of *internal validity* which holds whenever a specification is implementable:-

Definition 5 *S is internally valid, denoted $\Psi(S)$, iff $\exists I \in IMP$ s.t. S***conf** *I*

Ψ acts as a receptacle for properties of particular FDTs that make specifications in that FDT unimplementable. For example, a Z specification which contains contradictions would not be internally valid. Now we can redefine C_2 in a more usable way:

Definition 6 $S_1 \; C_{2.2} \; S_2 \; iff \; \exists S \in SPEC \; s.t. \; (S_1 conf S \; \wedge S_2 \; conf S) \; \wedge \; \Psi(S)$.

The third and final consistency interpretation hinges on the notion of behavioural compatibility which is defined in terms of an environment and unspecified criteria. We will consider specific instantiations of behavioural compatibility when we look at specific FDTs; at this stage we formulate the interpretation completely generally, for bc a particular instantiation of behavioural compatibility.

Definition 7 $S_1 \; C_3 \; S_2$ *iff* $S_1 \; bc \; S_2 \wedge S_2 \; bc \; S_1$.

Since consistency checking will occur at the specification checking stage of conformance assessment we actually need a mechanism to assess consistency that uses only specification checking relationships, i.e. refinement, unification and equivalence. We will seek to define natural interpretations of refinement, unification and translation and then consider how the different definitions of consistency can be related to the above three consistency interpretations.

4. A SPECIFICATION CHECKING FRAMEWORK

Translation. It seems natural to require that translation enforces equivalence, i.e. a translation of a specification should be equivalent to the original specification. The actual notion of equivalence required will be FDT dependent. However, we would certainly want translation to preserve equivalence due to conformance, which we denote $\equiv_{cf}$:

Definition 8 $S_1 \equiv_{cf} S_2$ *iff* $\{S : S_1 conf S\} = \{S : S_2 conf S\}$

Intuitively, two specifications are equivalent iff they determine exactly the same set of valid implementation specifications through $conf$. It should be pointed out that $\equiv_{cf}$ does not imply standard semantic equivalence; the equivalences of FDTs (such as observational and testing equivalences of process algebra) are likely to be stronger than $\equiv_{cf}$.

Refinement. Following [14] we define that S_2 is a refinement of S_1 as:-

Definition 9 $S_1 \sqsubseteq S_2$ *iff* $\{S : S_2 conf S\} \subseteq \{S : S_1 conf S\}$

i.e. refinement restricts the set of conformant implementation specifications. But, importantly, the implementations of a refinement are also implementations of the original specification.

Unification. Unification takes two specifications in the same language and produces a unified version which is a combination of the two specifications. By *combination* of specifications, we mean that unification should satisfy the property of common refinement, i.e. that $T_1, T_2 \sqsubseteq U(T_1, T_2)$, since an implementation that conforms to $U(T_1, T_2)$ should also conform to the original specifications T_1, T_2. In fact, we characterize unification as the least refinement of two specifications, with the following construction: $U(T_1, T_2) \in \{T : T_1, T_2 \sqsubseteq T \text{ and if } T_1, T_2 \sqsubseteq S \text{ then } T \sqsubseteq S\}$, see [8] for a discussion.

Consistency. A natural specification checking definition of consistency is that two specifications are consistent if their unification can be implemented.

Definition 10 *Given S_1 in language L_1 and S_2 in language L_2. Then $S_1 C_4 S_2$ iff there exists a specification language L_3 such that $S_1 \equiv_{cf} T_1, S_2 \equiv_{cf} T_2$ and there exists a $U(T_1, T_2)$ in L_3 such that $\Psi(U(T_1, T_2))$ for some T_1, T_2 in L_3.*

Notice in particular that the internal validity condition guarantees that a conformant implementation of the unification exists. In addition, this is our first interpretation of consistency that embraces translation. Properties of refinement, equivalence, unification and consistency can be found in appendix (ii).

Discussion. We now have four definitions of consistency C_1, $C_{2.2}$, C_3 and C_4. The first three of these arise from the ODP reference model and the third is a natural specification checking definition, which links notions of conformance to specification checking relationships such as refinement, unification and equivalence. We would clearly like to relate these definitions. However, a number of aspects of these definitions are FDT dependent. We will make the required FDT dependent comparison in the next two sections. We can, though, clarify our general approach, which is the following. Firstly, we view C_1 as a specialised form of consistency which is relevant to consistency checking in a logical setting and it will be captured by the internal validity property where it is relevant. The main focus of this paper, though, will be the relationship between $C_{2.2}$, C_3 and C_4 which are clearly in the same domain of reference.

The specification checking relationships of a particular FDT will not be equivalent to the corresponding definitions in our framework. However, our interpretation in this respect is that FDT relations that are stronger or equal to the framework definitions are appropriate, but relations that are either weaker or only partially intersect with the corresponding framework definition are not appropriate. Our intuition behind this interpretation is that consistency checking occurs during specification checking and that the specifier has knowledge about the nature of the specifications under consideration that is relevant to consistency, thus, at this stage of system development we can be more discriminating than is implicit in the framework. For example, the specifier may know that a specification is a functionality extension of another specification; that two specifications are strictly equivalent or that two specifications are related by reduction of non-determinism. This extra information should be used at the specification checking phase as long as it does not contradict the weaker conformance oriented definitions.

5. INSTANTIATING PARTICULAR FDTs

5.1 LOTOS Consistency Checking Relationships

Existing LOTOS relations can be instantiated into the consistency framework as follows:-

Conformance. A natural instantiation of our *conf* relation is the LOTOS conformance relation, which we denote $\underline{conf}$ (a definition of $\underline{conf}$ can be found in appendix i).

Internal Validity. The internal validity concept is targetted at FDTs such as Z where specifications can exist which do not have implementations. All LOTOS specifications can, at least 'theoretically', be implemented (and we apologize for the circularity here). Thus, we view all LOTOS specifications as internally valid.

$\mathbf{C_3}$. Consistency definition 1.3 is dependent upon the interpretation of behavioural compatibility, which in turn hinges on the interpretation of a specification's environment and the criteria imposed on that environment. The looseness of the definition of behavioural compatibility implies that one of a number of interpretations of C_3 could be made. It is our view that C_3 could be interpreted as any of the following:-

Definition 11
(i) $S_1 C_3^{\sim} S_2$ *iff* $S_1 \sim S_2$ - *Strong Bisimulation*
(ii) $S_1 C_3^{\approx} S_2$ *iff* $S_1 \approx S_2$ - *Weak Bisimulation*
(iii) $S_1 C_3^{te} S_2$ *iff* $S_1 te S_2$ - *Testing Equivalence*
(iv) $S_1 C_3^{cs} S_2$ *iff* $S_1 \underline{conf} S_2 \wedge S_2 \underline{conf} S_1$ - $\underline{conf}$ *symmetric*

Definitions (11.i) and (11.ii) view the environment as an unconstrained observer, in the sense of standard observational equivalences. In contrast, (11.iii) and (11.iv) view the environment as a tester for the specifications. The distinction between (11.iii) and (11.iv) is that (11.iii) implies robustness testing and (11.iv) implies restricted testing, see [4] [5] for a discussion of these alternatives. In the remainder of this paper we will concentrate on C_3^{cs}. Our reasons for this choice are two fold. Firstly, this interpretation agrees with the LOTOS definition of behavioural compatibility in Part IV of [12] and, secondly, we will show that, in comparison with $C_{2.2}$ and C_4, C_3^{cs} is a strong interpretation of consistency. Furthermore, C_3^{cs} is the weakest behavioural compatibility definition. Thus, since $C_3^{\sim} \Longrightarrow C_3^{\approx} \Longrightarrow C_3^{te} \Longrightarrow C_3^{cs}$, from process algebra theory, C_3^{cs} bounds the relationship between C_3 and the other consistency definitions.

Refinement. We will focus on two of the most important LOTOS refinement relations, *extension* (which we denote $\underline{ext}$) and *reduction* (which we denote $\underline{red}$), see appendix 1 for definitions. Intuitively, the former of these characterizes when a specification validly extends the behaviour of another specification and the latter relation characterizes refinement through reduction of non-determinism. In order to accept $\underline{ext}$ and $\underline{red}$ as suitable refinement relations we must show that both imply $\sqsubseteq$. Extrapolating from the results of [14] we get that $\underline{ext} \Rightarrow \sqsubseteq$, but $\underline{red} \not\Rightarrow \sqsubseteq$ and $\underline{red} \not\Leftarrow \sqsubseteq$. Thus, $\underline{ext}$ can be instantiated without any difficult, but $\underline{red}$ causes problems. We resolve this problem by considering a relation $\underline{red*}$ which we define as follows: $\underline{red*} = \underline{red} \cap \sqsubseteq$.

We will denote the instantiation of $\underline{ext}$ as the refinement relation in C_4 as $C_4^{\underline{ext}}$ and, similarly, the instantiation of $\underline{red*}$ in C_4 as $C_4^{\underline{red*}}$

Results. The following results arise from applying LOTOS relations to consistency:-

Proposition 1 *For $\underline{conf}$ all pairs of LOTOS processes are consistent by $C_{2.2}$*

Proof This follows from [13] which provides an algorithm that determines a common extension (i.e. $\underline{ext}$) for any pair of LOTOS processes and since $\underline{ext} \Longrightarrow \underline{conf}$.

Proposition 2 *For $\underline{conf}$, $C_3^{cs} \subset C_{2.2}$*

Proof All we have to do is to demonstrate a pair of processes that are not related by $\underline{conf}$. This is straightforward. For example, for the processes, $S_1 := b; stop[]i; a; stop$ and $S_2 := b; c; stop[]i; a; stop$, $\neg(S_2 \underline{conf} S_1)$. This is because $Ref(S_1, b) \not\subseteq Ref(S_2, b)$, e.g. $c \in Ref(S_1, b)$ but $c \notin Ref(S_2, b)$.

Proposition 3 $C_4^{\underline{ext}} = C_{2.2}$

Proof This follows from the results of [13].

Proposition 4 *(i) $C_3^{cs} \cap C_4^{\underline{red*}} \neq \emptyset$, (ii) $C_3^{cs} \not\subseteq C_4^{\underline{red*}}$ and (iii) $C_4^{\underline{red*}} \not\subseteq C_3^{cs}$.*

Proof We provide example LOTOS processes to demonstrate each of the properties.
(i) Consider the following trivial example. Take $S_1 = S_2 :=$ a; b; stop. Clearly, $S_1\ C_3^{cs}\ S_2$. In order to show that also $S_1\ C_4^{\underline{red*}}\ S_2$, we choose their common refinement to be $S = S_1 = S_2 :=$ a; b; stop. Obviously, $S_1 \underline{red*} S$ and $S_2 \underline{red*} S$.
(ii) Take $S_1 :=$ a; stop and $S_2 :=$ i; a; stop [] b; c; stop. Now $S_1 C_3^{cs} S_2$, but we will show that $\neg(S_1 C_4^{\underline{red*}} S_2)$. Firstly, the only possible reduction of both S_1 and S_2 is the process $S :=$ a; stop. Now, take the implementation $T :=$ a; stop [] b; stop. This is a valid implementation with respect to S, i.e. $S \underline{conf} T$. However, we can see that $\neg(S_2 \underline{conf} S)$, because S refuses action c after the trace b. Therefore, $S_2 \underline{red*} S$ does not hold.
(iii) Take $S_1 :=$ a; (b; stop [] i; stop) and $S_2 :=$ a; b; stop [] i; stop. We can easily check that $\neg(S_2 \underline{conf} S_1)$ and $\neg(S_1 \underline{conf} S_2)$. Therefore, we have $\neg(S_1\ C_3^{cs}\ S_2)$. However, $S_1\ C_4^{\underline{red*}}\ S_2$, which can be shown by taking $S :=$ a; b; stop as the common refinement of S_1 and S_2. This is because $S_1 \underline{red} S$ and $S_2 \underline{red} S$, since all non-determinism in S_1 and S_2 has been resolved in S. In addition, as $Tr(S) = Tr(S_1) = Tr(S_2)$ we know that $S_1 \underline{ext} S$ and $S_2 \underline{ext} S$. Moreover, since $\underline{ext} \Rightarrow \sqsubseteq$, from [14], we know that $S_1 \sqsubseteq S$ and $S_2 \sqsubseteq S$.

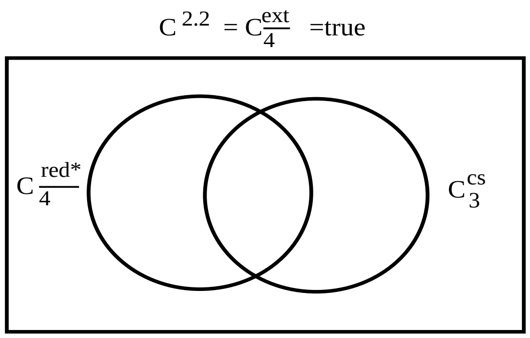
These results are depicted in figure 2. Interestingly, though unification construction algorithms can be given which demonstrate that $C_3 \subset C_4^{\underline{red}}$ and $C_3 \subseteq C_4$, these algorithms will not always yield the same unification, thus $C_4^{\underline{red}} \cap C_4 \neq C_4^{\underline{red*}}$. For further discussion of these relations see [16]. The following implications can be drawn from these results.

1. For LOTOS $C_{2.2}$ is very weak. In fact, it does not distinguish any processes.

2. In contrast, C_3 is a strong relation for LOTOS. In particular, none of the specification checking consistency relationships, i.e. $C_4^{\underline{red*}}$, $C_4^{\underline{red}}$, $C_4^{\underline{ext}}$, imply C_3.

3. The relationship between $C_4^{\underline{red*}}$ and C_3^{cs} is not very satisfactory and contrasts with the more natural relationship of $C_4^{\underline{red}}$ and C_4 with C_3^{cs}.

4. Under C_4^{ext} all pairs of LOTOS specifications are consistent. This may seem a surprising result at first, but it reflects the fact that extension of functionality across pairs of specifications can always be reconciled.



Firgure 2: LOTOS Consistency Relations

Probably the most important implication of these results is that consistency checking must be performed selectively. In particular, it is inappropriate to view consistency checking as a single mechanism which can be applied to any pair of specifications. For example, it would be inappropriate to check two specifications which express exactly corresponding functionality with C_4^{ext}. Thus, in order to apply suitable consistency checks the relationship of the specifications being checked must be made available. The RM-ODP has no provision for the communication of such information. The correspondence rule concept is used in the reference model as a means to locate portions of viewpoint specifications that should be compared. However, there is no means to define how these portions of specifications should be related.

5.2 Z Consistency Checking Relationships

A conformance relation for Z does not exist, but refinement has been extensively investigated. Thus, our work on consistency checking in Z has focussed on instantiating the C_4 definition of consistency. As indicated earlier internal validity is a central issue with Z, specifically, we define:-

Definition 12 *For S, a Z specification, $\Psi(S)$ iff $\neg\exists\psi$ s.t. $S \models \psi, \neg\psi$.*

An algorithm can be given which will unify two Z specifications [8]. This algorithm is divided into three stages: normalization, common refinement (which we usually term unification itself), and re-structuring. Normalization identifies commonality between two specifications, and re-writes the specifications into normal forms suitable for unification. Unification itself takes two normal forms and produces the least refinement of both. Re-structuring is performed to re-introduce the specification structure that is lost during normalization.

The major issue with Z consistency checking is not demonstrating that a unification exists, rather it is showing that the unification is internally valid. This is in obvious contrast to LOTOS where finding a unification with respect to a refinement relation is the central task. Demonstrating internal validity of Z specifications using theorem proving tools is a central area of our current research. A companion paper [8] contains a full discussion of consistency checking for Z.

6. TRANSLATION - THE OPTIONS

There has been some success in relating FDTs that have similar underlying semantics, e.g. [15] [2], although, it should be pointed out that the common semantic form underlying these approaches is typically very ugly and significant research is required before usable translations can be generated. ODP consistency checking though, requires translation across FDT families. There are very few positive results on this topic, although a number of approaches could be considered, the following are the most likely:-

Syntactic Translation. Translation based upon a direct relating of syntactic terms in one FDT to terms in another FDT is a possible approach. However, it is difficult to envisage that such an approach could offer a general solution. In particular, a lot of semantic meaning will certainly be lost in such a crude relating of FDTs. Partial syntactic translations may though be feasible.

Common Semantic Model. Translation into a common semantic model is a more realistic approach. Such translation could either use the semantics of one of the FDTs as the intermediate semantics or use a third semantics. The former of these is not fully general, for example, Z and LOTOS are so fundamentally different that relating one to the others semantic model is very difficult to envisage. Relating FDTs using a third intermediate form is a more likely approach.

- There is a link between model based action systems (and thereby Z) and CSP made by showing that refinements (forwards and backwards simulation) in an action system are sound and jointly complete with respect to the notion of refinement in CSP [18].

- The requirement for highly expressive intermediate semantics suggests that logical notations may be appropriate. [10] and [3] consider logical characterisations of LOTOS in temporal logic. However, relating temporal logic to the Z first order logic remains an open issue. Categorical approaches and the theory of institutions offer a possible solution [3].

- An alternative logical approach is that by [19]. This work uses first order logic to express relationships between states and events. Thus, they offer a single notational link between model based specification and formal descriptions based on transition systems. The approach uses logical conjunction as composition and sketches how consistency checking can be performed in this framework. The pragmatic nature of this work reflects the compromises that will have to be made when performing translation in the ODP setting. Specifically, [19] acknowledge that their approach does not preserve the semantic equivalences of particular FDTs.

- A final alternative which has the benefit of being ODP specific is suggested by the work of [6]. This work offers a denotational semantics for the computational viewpoint language. These semantics could, theoretically, be used to relate different FDT interpretations of the computational viewpoint language. Clearly, this work does not give a complete solution to consistency as the semantics are restricted to a single viewpoint. However, it may be possible to extrapolate this approach to a general solution.

A further issue affecting translation is the role of the ODP architectural semantics. Specifically, Part 4 should provide a basis for relating FDTs. ODP concepts, in particular viewpoint languages, are defined in different FDTs in the architectural semantics. Thus, when relating complete viewpoint specifications in different FDTs these definitions can be used as components of a consistency check. However, it is important to note that the architectural semantics will only provide a framework for consistency checking. Actual viewpoint language specifications will extend the ODP architectural semantics, which are non prescriptive by nature, with FDT specific behaviour. There is then a need to combine the framework provided by the architectural semantics with actual consistency checking relationships arising from FDTs.

It is clear though that a usable translation mechanism is likely to represent a pragmatic, compromise solution. In particular, complete preservation of semantic meaning during translation will not be possible.

7. CONCLUDING REMARKS

We have described how consistency arises in ODP. We have formalized a number of possible definitions of consistency, three of which are presented in the RM-ODP. We have considered instantiations of these consistency definitions with particular FDTs, viz, LOTOS and Z and finally we have discussed the thorny issue of translation between FDTs.

We believe that consideration of consistency is timely, not just from an ODP perspective. In particular, a number of recent software engineering methodologies consider relating multiple specifications of a single system, e.g. [19] [1]. The interest in such approaches reflects a general move away from classical single threaded waterfall system development scenarios. Furthermore, OO methodologies, require specifications to be related horizontally. Related issues can be found in OSI [9].

There are very few published results on consistency checking for Open Distributed Processing, [17] and [11] are exceptions to this. Both of these consider strong notions of consistency based on process algebra equivalences and in this sense take a quite different approach to us. The work presented in this paper suggests the following concrete results:-

1. The consistency interpretations arising in the RM-ODP have very different meanings. In particular, for LOTOS, all pairs of specifications are consistent by $C_{2.2}$, while C_3 is significantly stronger. In addition, by defining suitable conditions on the relationship between $conf$ and $\models$ we can use C_1 'consistently' with our conformance definitions. We can guarantee that $C_4 \Rightarrow C_{2.2}$ and $C_4 \Rightarrow C_1$, thus, C_4 provides an important link between logical notions of consistency and conformance notions.

2. It is appropriate to determine consistency using stronger relationships than the basic conformance definitions, since the extra knowledge available during specification checking enables system developers to apply consistency more discriminatingly.

3. With LOTOS all instantiations of C_4 with LOTOS refinement relations (trivially) imply $C_{2.2}$, while none of the instantiations imply C_3.

4. Consistency checking in Z and in LOTOS have a very different character. With LOTOS the central issue is finding a unification, while with Z the central issue is demonstrating that a unification does not contain any contradictions and can thus be implemented.

5. Pragmatic approaches to translation, in which some semantic information is lost, will have to be accepted.

We make the following recommendations; these are all required if realistic cross viewpoint consistency checking is to be undertaken:-

1. More specification to specification information must be made available to the consistency checking process. The nature of the consistency relationship to be checked must be made known. In addition, knowledge of the specification style used will be of value in performing consistency checking. It may even be necessary for specifiers to highlight particular cross viewpoint assertions that need to be tested.

2. Work on Part 4 of the RM-ODP must be undertaken as a priority. The architectural semantics provide an essential basis for consistency checking. In addition, the architectural semantics must themselves be shown to be 'consistent'. i.e. different FDT interpretations must not conflict.

3. Examples of multiple viewpoint specifications must be undertaken and be made available to the ODP community. Without realistic examples, consistency checking research will be poorly focussed.

In conclusion then, our inital results suggest that reasonable intra language consistency relationships can be found, however, inter language consistency checking remains a very challenging proposition. It is likely that this will only be possible with considerable prescriptive help from viewpoint language specifiers and in a pragmatic manner. However, this challenge must be met since without a realistic approach to maintaining the consistency of specifications across multiple viewpoints the potential of the existing and ongoing work on the ODP model cannot be fully realised.

References

[1] M Ainsworth, AH Cruickshank, LJ Groves, and PJL Wallis. Viewpoint specification and Z. *Information and Software Technology*, 36(1):43–51, February 1994.

[2] D. Bert, M. Bidoit, C. Choppy, R. Echahed, J.-M. Hufflen, J.-P. Jacquot, M. Lemoine, N. Lévy, J.-C. Reynaud, C. Roques, F. Voisin, J.-P. Finance, and M.-C. Gaudel. Opération SALSA: Structure d'AccueiL pour Spécifications Algébriques. Rapport final, PRC Programmation et Outils pour l'intelligence Artificielle, 1993.

[3] H. Bowman and J. Derrick. Towards a formal model of consistency in ODP. Technical Report 3-94, Computing Laboratory, University of Kent at Canterbury, 1994.

[4] E. Brinksma. A theory for the derivation of tests. In S. Aggarwal and K. Sabnani, editors, *Protocol Specification, Testing and Verification, VIII*, pages 63–74, Atlantic City, USA, June 1988. North-Holland.

[5] E. Brinksma, G. Scollo, and C. Steenbergen. Process specification, their implementation and their tests. In B. Sarikaya and G. V. Bochmann, editors, *Protocol Specification, Testing and Verification, VI*, pages 349–360, Montreal, Canada, June 1986. North-Holland.

[6] AFNOR cont. *A direct computational language semantics for Part 4 of the RM-ODP*. ISO/IEC JTC1/SC21/WG7 approved AFNOR contribution, July 1994.

[7] G. Cowen, J. Derrick, M. Gill, G. Girling (editor), A. Herbert, P. F. Linington, D. Rayner, F. Schulz, and R. Soley. *Prost Report of the Study on Testing for Open Distributed Processing*. APM Ltd, 1993.

[8] J. Derrick, H. Bowman, and M. Steen. Maintaining cross viewpoint consistency using Z. In *ICODP'95*, Brisbane, Australia, February 1995.

[9] A. Fantechi, S. Gnesi, and C. Laneve. Two standards means problems : A case study on formal protocol descriptions. *Computer Standards and Interfaces*, 9:11–19, 1989.

[10] A. Fantechi, S. Gnesi, and G. Ristori. Compositional logic semantics and LOTOS. In L. Logrippo, R.L. Probert, and H. Ural, editors, *Protocol Specification, Testing and Verification, X*, Ottawa, Canada, June 1990. North-Holland.

[11] K. Farooqui and L. Logrippo. Viewpoint transformations. In J. de Meer, B. Mahr, and O. Spaniol, editors, *2nd ICODP*, pages 352–362, Berlin, September 1993.

[12] ISO/IEC JTC1/SC21/WG7. *Basic reference model of Open Distributed Processing - Parts 1-4*, July 1993.

[13] F. Khendek and G. v. Bochmann. Merging behaviour specifications. Technical Report 856, University of Montreal, Department of Computing, 1993.

[14] G. Leduc. A framework based on implementation relations for implementing LOTOS specifications. *Computer Networks and ISDN Systems*, 25:23–41, 1992.

[15] R. Reed, W. Bouma, J.D. Evans, M. Dauphin, and M. Michel. *Specification and Programming Environment for Communication Software*. North-Holland, 1993. ISBN 0 444 89923 5.

[16] M. Steen, H. Bowman, and J. Derrick. Consistency in LOTOS. Technical Report in preparation, Computing Laboratory, University of Kent at Canterbury, 1995.

[17] A. Vogel. *Entwurf, Realisierung und Test von ODP-Systemen auf der Grundlage formaler Beschreibungstechniken.* PhD thesis, Humboldt-Universität zu Berlin, 1993. submitted.

[18] J.C.P. Woodcock and C.C. Morgan. Refinement of state-based concurent systems. In D. Bjorner, C.A.R. Hoare, and H. Langmaack, editors, *VDM '90 VDM and Z - Formal Methods in Software Development*, LNCS 428, pages 340–351, Kiel, FRG, April 1990. Springer-Verlag.

[19] P. Zave and M. Jackson. Conjunction as composition. *ACM Trans. on Soft. Eng. and Method.*, 2:379–411, 1993.

APPENDIX (i): LOTOS Relations. P, P_1 and P_2 are processes; $\mathcal{L}$ is the alphabet of observable actions; $\mathcal{L}^*$ denotes strings over $\mathcal{L}$; $Tr(P)$ denotes the set of traces of P and $Ref(P,\sigma)$ denotes the refusal set of P after the trace σ.

Definition 13
(i) P_2 $\underline{conf}$ P_1 *iff* $\forall \sigma \in Tr(P_2) : Ref(P_1,\sigma) \subseteq Ref(P_2,\sigma)$.
(ii) P_2 $\underline{red}$ P_1 *iff* $Tr(P_1) \subseteq Tr(P_2) \wedge P_1 \underline{conf} P_2$.
(iii) P_2 $\underline{ext}$ P_1 *iff* $Tr(P_1) \supseteq Tr(P_2) \wedge P_1 \underline{conf} P_2$.
(iv) P_1 $\underline{te}$ P_2 *iff:* $Tr(P_1) = Tr(P_2) \wedge \forall \sigma \in \mathcal{L}^* : Ref(P_1,\sigma) = Ref(P_2,\sigma)$.

APPENDIX (ii): Further Results. Proofs of these results can be found in [3].

Proposition 5 *Properties of* $\sqsubseteq$
(i) $\sqsubseteq$ *is a pre-order (i.e. reflexive and transitive)*
(ii) $S_1 \equiv_{cf} S_2$ *iff* $S_1 \sqsubseteq S_2$ *and* $S_2 \sqsubseteq S_1$ *(i.e.,* $\sqsubseteq$ *is a partial order with respect to equivalence)*
(iii) $(\sqsubseteq \circ\, conf) = conf$
(iv) For all R*, we have* $R \subseteq \sqsubseteq$ *iff* $(R \circ\ conf) \subseteq\ conf$
(v) For all R*, we have* $Id \subseteq R$ *implies that* $(R \subseteq \sqsubseteq)$ *iff* $(R \circ\ conf = conf)$
(vi) $\sqsubseteq$ *is the least relation* R *such that* $R \circ\ conf = conf$.

Proposition 6 *Unification satisfies the following properties:*
(i) $U(T_1,T_2) = U(T_2,T_1)$ *- commutativity*
(ii) $U(T_1,U(T_2,T_3)) = U(U(T_1,T_2),T_3)$ *- associativity*
(iii) $T_1,T_2 \sqsubseteq U(T_1,T_2)$ *- common refinement*
(iv) If $T_1 \sqsubseteq T_2$ *then* $U(T_1,T_2) = T_2$

Proposition 7 *Properties of consistency:*
(1) Consistency is a symmetric relation, but it is neither reflexive nor transitive.
(ii) $S_1 C_4 U(S_2,S_3)$ *iff* $S_2 C_4 U(S_1,S_3)$ *iff* $S_3 C_4 U(S_1,S_2)$.
(iii) Global consistency of three or more specifications implies pairwise consistency.
(iv) Pairwise consistency does not imply global consistency.