Viewpoint consistency in ODP, a general interpretation*

H. Bowman, E.A. Boiten, J. Derrick and M.W.A. Steen
University of Kent at Canterbury
Computing Laboratory, University of Kent, Canterbury, CT2 7NF, UK
(Phone: + 44 1227 827913, Fax: + 44 1227 762811,
Email: {H.Bowman,E.A.Boiten, J.Derrick,mwas}@ukc.ac.uk.)

Abstract

Multiple viewpoints are used in Open Distributed Processing (ODP) in order to decompose the complexity inherent in specifying distributed systems. Multiple viewpoints prompt the issue of consistency between viewpoints. The ODP reference model alludes to three different interpretations of consistency. This paper responds to this uncertainty by proposing a single all embracing interpretation of consistency. We show that our interpretation, firstly, satisfies all the basic requirements of a definition of consistency and, secondly, can be specialised to any of the three ODP reference model definitions. The generality of our definition will be illustrated through instantiation in the FDT LOTOS.

Keywords

Viewpoints, Consistency, ODP, Formal Description Techniques, LOTOS

1 INTRODUCTION

Reference Model for Open Distributed Processing (RM-ODP) [Lin95] provides an architectural framework for the construction of *open* distributed systems. The architecture is now mature, with the main components of the reference model (parts 2 and 3) having recently completed their progress to international standards. One of the central tenets of the architecture is the use of multiple viewpoints in order to decompose the description of systems. Five viewpoints are defined in the architecture *enterprise*, *information*, *computational*, *engineering* and *technology*; each of these viewpoints is applicable to a different viewer of the system. For example, the computational viewpoint is targetted at the applications programmer. Thus, viewpoints offer a fundamental *separation of concerns* for the specification of distributed systems.

Importantly though, the imposition of a multiple viewpoints model prompts the issue

*This work was partially funded by British Telecom Research Labs., Martlesham, Ipswich, U.K. and the Engineering and Physical Sciences Research Council under grant number GR/K13035.

of viewpoint consistency. Specifically, it is essential that multiple views of a system are shown not to conflict with one another, i.e. to be, in some sense, *consistent*. In order for such a relationship to be checked a formal approach should be adopted. This is because without recourse to formal semantics it is highly improbable that multiple specifications can be related in a uniform manner. Thus, consistency should be investigated in the context of the application of formal description techniques (FDTs) to ODP. In this paper we will illustrate our work on consistency using the FDT LOTOS, which is one of the most important FDTs being applied in ODP.

Unfortunately, there has been very little work on viewpoint consistency in ODP. Furthermore, from amongst the work that has been performed there is little agreement on the basic definition of consistency to use. As a reflection of this uncertainty, a committee draft of the reference model for ODP (RM-ODP) contained three different definitions. In its current form the reference model has backed away from prescribing particular interpretations because it was felt that none of the candidate definitions was a fully general interpretation. However, the three interpretations of the earlier committee draft are still alluded to as "possible interpretations". Here we will build upon our work in [BDS95], which provided a formal interpretation of the three definitions, by presenting a new definition of consistency, which we argue is, firstly, intuitively reasonable and, secondly, general, in the sense that it embraces the other main interpretations of consistency. Thus, this paper seeks to resolve the disagreement surrounding interpretations of consistency by proposing a single all embracing definition.

Structure of paper. Section 2 discusses the nature of consistency in ODP and formally interprets the three RM-ODP definitions (this section summarises the main results of [BDS95]). Section 3 gives an informal intuitive interpretation of ODP consistency and formalises this interpretation as our central definition of consistency. Section 4 highlights the generality of our interpretation by reconciling the RM-ODP definitions against our definition. Section 5 considers instantiations of the consistency definitions in LOTOS and section 6 presents some concluding remarks.

2 CONSISTENCY IN ODP

2.1 The nature of consistency in ODP

Figure 1 depicts the relationships that are involved in relating ODP viewpoints. *Development* yields a specification that defines the system being described more closely. Because all five viewpoint specifications will eventually be realized by one system, there must be a way to combine specifications from different viewpoints during development; this is known as *unification*. For specifications in different FDTs to be combined or unified, a *translation* mechanism is needed to transform a specification in one language to a specification in another language. *Consistency* is a relation between (pairs of) specifications.

In our work on consistency we distinguish between *intra* and *inter* language consistency checking. Intra language consistency considers how multiple specifications in the same language can be shown to be consistent, while inter language consistency considers relations between specifications in different FDTs. The latter issue is a significantly more demanding topic than the former.

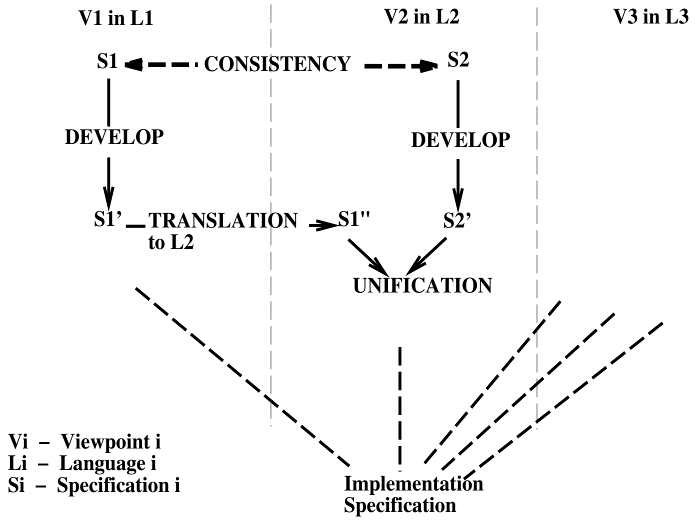

Figure 1 Relating viewpoints.

In order to inform our investigation of possible definitions of consistency it is worth considering what we require of such a definition. We offer the following list as an initial set of requirements. The consistency definition we seek must,

- be applicable *intra language* for many different FDTs, e.g. must make sense between two Z specifications and also between two LOTOS specifications;
- be applicable *inter language* between different FDTs, e.g. relate a Z specification to a LOTOS specification.
- support different *classes of consistency check*. There are many different forms of consistency and the appropriate check to apply depends on the viewpoint specifications being considered and the relationship between these viewpoints [BDS95]. For example, it would be inappropriate to check two specifications which express exactly corresponding functionality with the same notion of consistency that is applicable to checking consistency between specifications which extend each other's functionality.
- support global consistency. To date research on consistency has generally only considered the n=2 case (what we will call *binary consistency*); for full generality we need any arbitrary n>0.
- allow viewpoints to relate to the target system in different ways. Thus, not only are there different forms of consistency check, but within a consistency check, specifications are related in different ways. For example, the enterprise specification is likely to express global requirements, while the computational specification defines an interaction model. Thus, the relationship between the system being developed and the enterprise specification is very different from the relationship of the system to the computational specification.

This final point prompts our work on, so called, *unbalanced consistency* in which each viewpoint is potentially related to the system under development by a different development relation. For example, the enterprise viewpoint may be related by a logical satisfaction relation while the computational viewpoint may be related by a behavioural conformance relation. Note also that unbalanced consistency is needed to support inter language consistency. This aspect of our work represents a significant departure from existing theoretical work on relating partial specifications, e.g. [ACGW94], which has universally looked at, what we call, *balanced consistency.*

2.2 ODP definitions

This section highlights the three interpretations of consistency that currently appear in the RM-ODP, the first two appear in part 1 (clause 12.2) and the third appears in part 3 (clause 10). Although, the first of these definitions is only alluded to.

Definition 1
(1.1) 2 specifications are consistent iff they do not impose contradictory requirements.
(1.2) 2 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) 2 specifications are consistent iff they are both behaviourally compatible with the other.

Behavioural compatibility is defined as follows:

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

The RM-ODP definition of this concept is expressed in terms of objects, however, in order to be more general than this we have presented the concept in terms of specifications.

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

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

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.

Consistency 1.2 is interpreted as,

Definition 4 $S_1\ C_2\ S_2\ \mathit{iff}\ \exists S\ s.t.\ S\ \mathit{conf}\ S_1\ \wedge S\ \mathit{conf}\ S_2\ \wedge\ \Psi(S).$

The definition uses a conformance relation, *conf*, which relates specifications that conform under some class of testing. It also uses internal validity, denoted Ψ, which is a check to determine that the conformant specification is implementable. We will discuss internal validity in some depth in section 3. The definition states that two specifications are consistent if and only if a third specification can be found which conforms to both original specifications and the third specification can be realised in an implementation.

Consistency interpretation 1.3 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 a specific FDT; at this stage we formulate the interpretation completely generally, for bc a particular instantiation of behavioural compatibility.

Definition 5 *S_1 C_3 S_2 iff S_1 bc $S_2 \wedge S_2$ bc S_1.*

We will often make the parameterisation here explicit and denote the interpretation as C_3^{bc}. These definitions are limited in a number of ways.

- Each definition is a specialized notion of consistency that is applicable in a certain setting, e.g. C_1 to consistency in Z, but none of the definitions gives the "big picture" and is general enough to be instantiated reasonably for many FDTs and many notions of consistency.
- The definitions blur over the fact that specifications may be in different FDTs.
- The definitions are restricted to binary consistency checking.
- Unbalanced consistency is not supported.

3 A GENERAL DEFINITION OF CONSISTENCY

This section responds to the deficiences just highlighted. We will give general definitions of the consistency checking relationships: *consistency*, both *intra* and *inter* language, and *unification*. First though we will present the notation that we will work with. Importantly, this notation reflects the search for a general interpretation of consistency by defining very general notational conventions. These conventions will be specialized for particular FDTs and particular forms of consistency.

Notation. We begin by assuming a set DES of formal descriptions, which contains both formal specifications in languages such as LOTOS and Z and semantic descriptions in notations such as labelled transition systems and ZF set theory.

We assume a set $DEV \subseteq \mathcal{P}(DES \times DES)$ of *development relations*. These are written dv and if X dv X' then, in some sense, X is a valid development of X'. Our concept of a development relation generalises all notions of evolving a formal description towards an implementation and thus embraces the many such notions that have been proposed. In particular, DEV contains *refinement* relations, *equivalences* and relations which can broadly be classed as *implementation* relations such as the LOTOS conformance relation **conf**. These different classes of development are best distinguished by their basic properties. Refinement is typically reflexive and transitive (i.e. a preorder); equivalences are reflexive, symmetric and transitive; and implementation relations are only reflexive.

In general though we do not require that development relations support any specific properties. In fact, we cannot even assume reflexivity in the general case. This is because, in order to support inter language consistency checking, we allow development to relate descriptions in different notations. In these circumstances reflexivity is not a sensible concept.

Descriptions are written in formal techniques. A formal technique is characterised by the set of possible descriptions in the notation and a set of associated development relations. For a particular formal technique ft we denote the set of all descriptions in ft as DES_{ft} and the set of all development relations as DEV_{ft}.

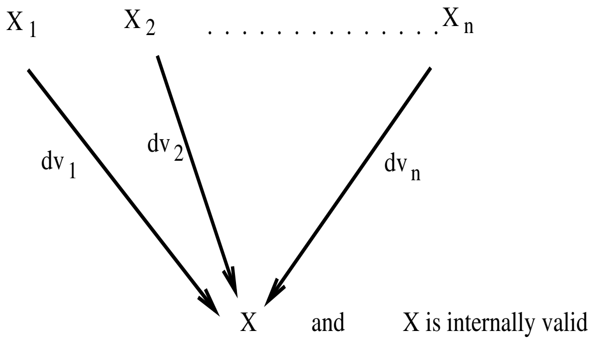

Figure 2 A consistency check.

Basic Definition. In its general form consistency is a check which takes any number of descriptions, $X_1, X_2, ..., X_n$, and returns true if all the descriptions are consistent and false otherwise. This check will be performed according to a list of development relations, $dv_1, dv_2, ...dv_n$, one per description, and is denoted, $C[dv_1, dv_2, ...dv_n](X_1, X_2, ..., X_n)$. The validity of the check has two elements: *type correctness* and *consistency.*

Type correctness ensures that the consistency check being attempted is sensible. For example, it would prevent a development relation being applied to a specification written in a different language to that which the development relation is defined over. Type correctness becomes an issue when determining an appropriate inter language consistency check to apply. For simplicity, in this paper all consistency checks will be assumed to be type correct.

Intuitively we view n specifications $X_1, X_2, ..., X_n$ as consistent if and only if there exists a physical implementation which is a realization of all the specifications, i.e. X_1, X_2 through to X_n can be implemented in a single system. However, we can only work in the formal setting, so we express consistency in terms of a common (formal) description, X, and a list of development relations, $dv_1, dv_2, ..., dv_n$. The definition states that n descriptions are consistent if and only if a description can be found which is a development of X_1 according to dv_1, X_2 according to dv_2, through to X_n according to dv_n, and the description is internally valid, written $\Psi(X)$. The structure of the consistency check is depicted in figure 2 and is formalized in definition 6. We denote this interpretation of consistency as C.

Definition 6 (Consistency)
$C[dv_1, ...dv_n](X_1, ..., X_n)$ *holds, iff* $\exists X \in DES$ *s.t.* $(X\ \ dv_1\ X_1 \wedge ... \wedge X\ dv_n\ X_n) \wedge \Psi(X)$.

The internal validity check in the above definition formalizes the notion of implementability. It is required because descriptions relate to physical implementations in different ways for different languages and, in particular, for some FDTs not all specifications are implementable. For example, a Z specification that contains an operation $[n! : \mathbb{N} | n! = 5 \wedge n! = 3]$ has no real implementation. Thus, for some FDTs it is possible to find a description which is a common development of a pair of specifications, but is not itself implementable. The property $\Psi(X)$ is true if and only if the description X has a real implementation. Thus, Ψ acts as a receptacle for properties of particular languages that make descriptions in that language unimplementable. For example, a Z specification which contains contradictions would not be internally valid. This ensures that definition 6 in the case that n=1 coincides with what is commonly called "consistency" of a single specification.

Unification is the mechanism by which descriptions are composed in such a way that the composition is a development of all the descriptions.

Definition 7 (Unification Set)
$\mathcal{U}[dv_1, dv_2, .., dv_n](X_1, X_2, ..., X_n) = \{X \in DES \ : \ X \ dv_1 \ X_1 \ \wedge ... \wedge \ X \ dv_n \ X_n\}.$

The unification set is the set of all common developments of a list of descriptions, i.e. the set of all unifications. Clearly, $C[dv_1, dv_2, .., dv_n](X_1, X_2, ..., X_n)$ holds if and only if $\exists X \in \mathcal{U}$ such that $\Psi(X)$. In fact, one approach to consistency checking is to perform a unification and then to show that this unification is internally valid.

Our interpretation of consistency, C, meets the requirements for a definition of consistency that we highlighted earlier, in the following ways:

- Different development relations can be instantiated which are appropriate both to different FDTs and to assessing different forms of consistency.
- Both intra and inter language consistency are incorporated. In particular note that in most cases $X_1, X_2, ..., X_n$ in the above definition will all be specifications, however, X will commonly be a semantic representation. In particular, if some of $X_1, X_2, ..., X_n$ are in different languages then X is likely to be in a common semantic notation.
- Consistency checking between an arbitrary number of descriptions can be supported and checked according to a list of development relations. Binary consistency is just a special case of this global consistency, e.g. $C[dv_1, dv_2](X_1, X_2)$. Binary consistency is a binary relation and is often written, $X_1 \ C_{dv_1,dv_2} \ X_2$.
- Both balanced and unbalanced consistency are incorporated. Unbalanced consistency arises if $dv_i \neq dv_j$ for $i \neq j$.

It is beyond the scope of this paper to fully document the properties of our interpretation of consistency, the interested reader is referred to [BBDS95], however, a number of classes of consistency will be used later in this paper and are, thus, reviewed in the following subsections.

Implementation Complete. There are a number of languages in which all specifications are internally valid. Thus, we introduce the following notation:-

Notation 1 (Implementation Complete)
A formal technique ft is called implementation complete iff $\forall X \in DES_{ft}, \ \Psi(X)$.

Balanced Consistency. Balanced consistency reflects the situation in which the specifications being checked for consistency are at the same level of abstraction; balanced consistency is written: $C[dv](X_1, X_2, ..., X_n)$. It should be noted that some of our previous papers have only considered balanced consistency, e.g. [BDS95] and presented this as consistency in its entirety. This paper presents a generalization of that work.

Definition 8 (Balanced Consistency)
$C[dv_1, dv_2, .., dv_n](X_1, X_2, ..., X_n)$, *is balanced iff* $dv_i = dv_j, \ \forall dv_i, dv_j$ *s.t.* $1 \leq i, j \leq n$.

Once again we can consider the special case of binary balanced consistency, $C[dv](X_1, X_2)$, which is often written as C_{dv}.

The following simple results relate the characteristics of the development relation used to the induced balanced consistency. They will be valuable when we seek to relate behavioural compatibility to our interpretation of consistency.

Proposition 1
(i) If dv is reflexive and $\Psi(X_1)$, then $X_1\ dv\ X_2 \implies X_1\ C_{dv}\ X_2$.
(ii) If dv is symmetric and transitive then $X_1\ C_{dv}\ X_2 \Longrightarrow X_1\ dv\ X_2$.

Proof
(i) Assume $X_1\ dv\ X_2$ and $\Psi(X_1)$; from reflexivity of dv we get X_1 is the required common development.
(ii) Assume $\exists X\ s.t.\ X\ dv\ X_1\ \wedge\ X\ dv\ X_2\ \wedge\ \Psi(X)$; then from symmetry $X_1\ dv\ X$ and from transitivity $X_1\ dv\ X_2$ as required. $\square$

Corollary 1
For ft an implementation complete formal technique and dv an equivalence $dv = C_{dv}$ for all descriptions in ft.

4 GENERALITY OF THE DEFINITION

Our definition embraces one of the RM-ODP definitions directly and the other two through imposition of constraints on the development relation used. We consider these results here.

Reconciling C_2. The following proposition makes the required instantiation.

Proposition 2
If dv' is instantiated as conformance then $C_2 = C_{dv'}$.

Reconciling C_1. Our approach is to define a development relation with the required characteristics and instantiate this into C. We define dv as:

(dv) $X_1\ dv\ X_2 \iff (X_1 \models \gamma \impliedby X_2 \models \gamma)$

This constraint is not unreasonable, e.g. it could be defined for Z. Also, a consequence of (dv) is that dv is reflexive. In addition, we give internal validity a natural interpretation as,

$\Psi(X) \Longleftrightarrow \neg(\exists\gamma\ s.t.\ X \models \gamma, \neg\gamma)$

We need a simple lemma.

Lemma 1
$X_1\ C_1\ X_2 \implies (\neg(\exists\gamma\ s.t.\ X_1 \models \gamma, \neg\gamma)\ \wedge\ \neg(\exists\gamma'\ s.t.\ X_2 \models \gamma', \neg\gamma'))$.

Proof
We will show that $X_1\ C_1\ X_2 \implies \neg(\exists\gamma\ s.t.\ X_1 \models \gamma, \neg\gamma)$. We will use contradiction, so assume $\exists\gamma\ s.t.\ X_1 \models \gamma, \neg\gamma$. Now if we consider X_2 it is clear that either $X_2 \models \gamma\ \vee\ X_2 \models$

$\neg\gamma$. However, if either of these hold then C_1, i.e. $\neg(\exists\gamma''\ s.t.\ X_1 \models \neg\gamma'' \ \wedge\ X_2 \models \gamma'')$, is contradicted. We can make a similar argument to show that $\neg(\exists\gamma'\ s.t.\ X_1 \models \gamma', \neg\gamma')$ follows from C_1. $\square$

Now we can prove the equality that we want.

Proposition 3
$C_1 = C_{dv}$.

Proof
($C_1 \subseteq C_{dv}$) Assume C_1, i.e. $\neg(\exists\gamma\ s.t.\ X_1 \models \neg\gamma \ \wedge\ X_2 \models \gamma)$. We can draw the following implications:-

$$\neg(\exists\gamma\ s.t.\ X_1 \models \neg\gamma \ \wedge\ X_2 \models \gamma) \implies \forall\gamma.\neg(X_1 \models \neg\gamma \ \wedge\ X_2 \models \gamma) \implies$$
$$\forall\gamma.\neg(\neg(X_1 \models \gamma) \ \wedge\ X_2 \models \gamma) \implies \forall\gamma.\neg(\neg(X_1 \models \gamma) \ \wedge\ \neg(\neg(X_2 \models \gamma))) \implies$$
$$\forall\gamma.(X_1 \models \gamma \ \vee\ \neg(X_2 \models \gamma)) \implies \forall\gamma.(X_1 \models \gamma \ \Longleftarrow\ X_2 \models \gamma)$$

thus $X_1\ dv\ X_2$, by (dv). By reflexivity of dv we also have that $X_1\ dv\ X_1$. So, X_1 is a common development and from lemma 1 we have that $\Psi(X_1)$. Thus, $X_1\ C_{dv}\ X_2$ as required.
($C_1 \supseteq C_{dv}$) We will use contradiction. Thus, assume C_{dv} and the negation of C_1:
$\exists X\ s.t.\ X\ dv\ X_1 \ \wedge\ X\ dv\ X_2 \ \wedge\ \neg(\exists\gamma\ s.t.\ X \models \gamma, \neg\gamma)$ and $\exists\gamma'\ s.t.\ X_1 \models \gamma' \ \wedge\ X_2 \models \neg\gamma'$, but (dv) and these assumptions give $X \models \gamma', \neg\gamma'$ which is a contradiction. $\square$

Reconciling C_3. As a concept, behavioural compatibility is extremely general; the notion is, firstly, FDT dependent and, secondly, can be interpreted a number of ways for each FDT, thus, a direct relating of C_3 and C is not possible. However, we can give strong evidence that C_3 can be fully embraced. In particular, the following result gives a general relationship for implementation complete formal techniques, it follows immediately from corollary 1.

Proposition 4
For an implementation complete language and ω an equivalence $C_3^\omega = C_\omega$.

Thus, if ft is implementation complete and behavioural compatibility induces an equivalence on C_3 we can make a straightforward instantiation of behavioural compatibility in the development relation and obtain an equivalent definition. Furthermore, the restriction to implementation complete formal techniques is not overly restrictive, since the target of C_3 is the behavioural portion of notations such as, LOTOS, Estelle and SDL, which can be viewed to be inherently implementation complete [†].

We will further justify that C_3 can be embraced by C by showing, in section 5.2.2, that all the obvious LOTOS instantiations of behavioural compatibility can be given an equivalent C interpretation. This is strong evidence since LOTOS is a main target for the behavioural compatibility concept. We will summarise these results here.

Firstly, using proposition 4 we can reconcile any LOTOS instantiation that interprets

[†] Note that consideration of the data languages associated with these techniques may invalidate implementation completeness. For example, contradictory equations can certainly be specified in ACT-ONE

C_3 as an equivalence, e.g. testing equivalence or weak or strong bisimulation. In addition, we will show that the single remaining interpretation can also be embraced. Under this interpretation behavioural compatibility is viewed as the LOTOS **conf** relation. Using a **conf** based related, denoted **xcs**, we can get the required relationship between C_3 and C.

Proposition 5 *For LOTOS specifications,* $C_3^{\mathbf{conf}} = C_{\mathbf{xcs}}$.

We will explain the relation **xcs** and prove this result in section 5.2.2.

5 CONSISTENCY IN LOTOS

Introducing LOTOS is beyond the scope of this paper, thus, this section will assume familiarity with the language. The objective of this section is to illustrate the generality of our definition by showing that LOTOS instantiations of the RM-ODP definitions can be embraced by our definition. We particularly focus on C_3, as behavioural compatibility is FDT dependent. The next section reiterates the standard definitions of the LOTOS development relations that we use in section 5.2 to instantiate the RM-ODP definitions. Section 5.2 contains the main theoretical results of this paper, which is the relating of $C_3^{\mathbf{conf}}$ to our definition.

5.1 Development relations

First we introduce some notation.

Notation. In the following P, P', Q, Q' stand for processes. $\mathcal{L}$ is the alphabet of observable actions associated with a certain process, while i is the invisible or internal action. We use the variable α to range over $\mathcal{L}$. Furthermore, $\mathcal{L}^*$ denotes strings (or traces) over $\mathcal{L}$. The constant $\epsilon \in \mathcal{L}^*$ denotes the empty string, and the variable σ ranges over $\mathcal{L}^*$. We assume the following definitions:

$P \xrightarrow{\mu} P'$ denotes a transition, i.e. P can do μ and evolve to P';
$\stackrel{\epsilon}{\Longrightarrow}$ denotes the reflexive and transitive closure of $\xrightarrow{i}$;
$P \stackrel{\alpha\sigma}{\Longrightarrow} P'$ iff $\exists Q, Q' \cdot P \stackrel{\epsilon}{\Longrightarrow} Q \xrightarrow{\alpha} Q' \stackrel{\sigma}{\Longrightarrow} P'$;
$P \stackrel{\sigma}{\Longrightarrow}$ iff $\exists P' \cdot P \stackrel{\sigma}{\Longrightarrow} P'$;
$P \stackrel{\sigma}{\not\Longrightarrow}$ iff $\nexists P' \cdot P \stackrel{\sigma}{\Longrightarrow} P'$;
$Tr(P) = \{\sigma \in \mathcal{L}^* \mid \mathcal{P} \stackrel{\sigma}{\Longrightarrow}\}$, denotes the set of traces of a process P;
P *after* $\sigma = \{P' \mid P \stackrel{\sigma}{\Longrightarrow} P'\}$, denotes the set of all states reachable from P by the trace σ;
$Ref(P, \sigma) = \{X \mid \exists P' \in (P \textit{ after } \sigma), \text{ s.t. } \forall \alpha \in X : P' \stackrel{\alpha}{\not\Longrightarrow}\}$, denotes the refusals of P *after* σ.

Conformance. The **conf** relation [BSS86], has been adopted as the primary interpretation of conformance in LOTOS, it is defined as follows:

Definition 9 (conformance)
P **conf** Q, *iff* $\forall \sigma \in Tr(Q),\ Ref(P, \sigma) \subseteq Ref(Q, \sigma)$

We will also use a development relation which is a symmetric subset of **conf**. This relation is called **conf** symmetric and is denoted **cs**; it will play a central role in instantiations of C_3. In particular, since the ODP architectural semantics adopt **conf** as their interpretation of behavioural compatibility **cs** is an obvious interpretation of C_3.

Definition 10 *(***conf symmetric***)*
*P **cs** Q iff P **conf** $Q \ \wedge \ Q$ **conf** P.*

Equivalences. We also assume the standard notions of equivalence: *testing equivalence* [BSS86], denoted **te**, *weak bisimulation equivalence* [Mil89], denoted $\approx$, and *strong bisimulation equivalence* [Mil89], denoted $\sim$.

Properties of the Development Relations. Apart from **cs** the properties of the development relations presented above have been well documented in the literature. We will review some of these properties here.

Proposition 6
(i) **te***, $\sim$, and $\approx$ are equivalences.*
(ii) **conf** *is reflexive, but neither symmetric or transitive.*
(iii) **cs** *is (a) reflexive and (b) symmetric, but (c) not transitive.*

Proof
(i) and (ii) are all standard results from the theory of LOTOS and process algebra in general, see for instance [Led91] and [Mil89]. However, (iii) needs some justification:-

(iii.a) This is a consequence of **conf** being reflexive.

(iii.b) This is immediate from the definition of **cs**.

(iii.c) The following counterexample justifies this. Let $P := b; stop[]i; a; stop$; $Q := i; a; stop$ and $R := b; c; stop[]i; a; stop$; then P **cs** Q, Q **cs** R, but $\neg(P$ **cs** $R)$. This is because $\neg(P$ **conf** $R)$ as P refuses c after the trace b, but R cannot refuse c after the trace. $\Box$

te, $\sim$, and $\approx$ can be classed together as *equivalences*; while **conf** and **cs** are weaker *implementation* relations. Notice we have not defined any of the standard LOTOS preorders trace preorder, reduction and extension; this is because they do not play a role in the next section. Instantiations of these preorders into our definition of consistency are extensively investigated in [SBD95].

5.2 Relating the RM-ODP definitions

We begin by giving LOTOS instantiations of relevant definitions and, in particular, the RM-ODP definitions; these instantiations are related to C in the following subsection.

5.2.1 RM-ODP instantiations

Proposition 7
$\forall P \in DES_{LOTOS}$, $\Psi(P)$, i.e. LOTOS is implementation complete.

This follows intuitively from considering the nature of LOTOS specifications. At least it follows if we ignore the ACT-ONE data language. Thus, here we are really considering just basic LOTOS. In particular, at least theoretically, we can view all basic LOTOS specifications as implementable. Even degenerate specifications, such as those containing deadlocks, for example, have a physical implementation equivalent. This is a fundamental characteristic of behavioural languages that distinguishes them from logically based specification notations. This result is important as it considerably simplifies the class of consistency that must be considered for LOTOS. Furthermore, we assume that all consistency checks are type correct. This is reasonable since we are only considering consistency intra the LOTOS language.

Of the specific RM-ODP definitions, we could relate C_1 via an interpretation of LOTOS in logic, this is a complex interpretation with a number of subtle issues. Thus, we will view this as beyond the immediate scope of our work and we will not consider C_1 further in the context of LOTOS. In contrast, C_2 and C_3 are immediately appropriate to LOTOS. We will consider these in turn.

Instantiation of C_2. This is very straightforward, we give the following definition:-

Definition 11 *For $P_1, P_2, P \in DES_{LOTOS}$ P_1 C_2 P_2 iff $\exists P$ s.t. P $\mathbf{conf}$ P_1 $\wedge P$ $\mathbf{conf}$ P_2.*

It should be noted that this instantiation is dependent on the interpretation of conformance adopted. **conf** is a weak interpretation, in particular, it does not enforce the preservation of safety properties (although, liveness properties are preserved). However, **conf** is a realistic reflection of the capabilities of conformance testing and is the basis of work on test case generation for LOTOS [BSS86].

Instantiation of C_3. Consistency definition C_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 12
(i) P_1 $C_3^{\sim}$ P_2 iff $P_1 \sim P_2$ - Strong Bisimulation
(ii) P_1 $C_3^{\approx}$ P_2 iff $P_1 \approx P_2$ - Weak Bisimulation
(iii) P_1 $C_3^{\mathbf{te}}$ P_2 iff P_1 $\mathbf{te}$ P_2 - Testing Equivalence
(iv) P_1 $C_3^{\mathbf{cs}}$ P_2 iff P_1 $\mathbf{cs}$ P_2 - Conf symmetric

Definitions 12(i) and 12(ii) view the environment as an unconstrained observer, in the sense of bisimulation equivalences. In contrast, 12(iii) and 12(iv) view the environment as a tester for the specifications. The distinction between 12(iii) and 12(iv) is that 12(iii) implies robustness testing and 12(iv) implies restricted testing, see [BSS86] for a discussion of these alternatives. Amongst these definitions $C_3^{\mathbf{cs}}$ is particularly important for a number of reasons. Firstly, this interpretation agrees with the LOTOS definition of behavioural compatibility in the RM-ODP architectural semantics. In addition, as indicated in the following proposition, $C_3^{\mathbf{cs}}$ is the weakest of the LOTOS interpretations of C_3.

Proposition 8
$C_3^{\sim} \subset C_3^{\approx} \subset C_3^{\mathbf{te}} \subset C_3^{\mathbf{cs}}$.

Proof
$C_3^{\sim} \subset C_3^{\approx} \subset C_3^{\mathbf{te}}$ are standard process algebra results. $C_3^{\mathbf{te}} \subset C_3^{\mathbf{cs}}$ requires some justification. Firstly, it is straightforward to see that $\mathbf{te} \subseteq \mathbf{cs}$. In addition, we can provide the two processes $P := a; stop[]i; b; stop$ and $Q := i; b; stop$ as counterexamples to justify that $\mathbf{cs} \not\subseteq \mathbf{te}$, since P $\mathbf{cs}$ Q, but $\neg(P \ \mathbf{te}\ Q)$ as the trace sets of the two processes are not equal. $\Box$

Furthermore, [BDS95] has shown that C_3 is the strongest of the RM-ODP interpretations of consistency, thus, $C_3^{\mathbf{cs}}$ bounds the relationship between C_3 and the other RM-ODP consistency definitions and warrants particular attention.

5.2.2 Relating definitions

This subsection specializes the results of section 4 to LOTOS.

Reconciling C_2**.** The following result is immediate from a comparison of instantiations.

Proposition 9
For LOTOS $C_2 = C_{\mathbf{conf}}$.

Reconciling C_3**.** Three of the interpretations made in section 5.2.1 can be related using corollary 1 to our general definition easily.

Proposition 10
(i) $C_3^{\sim} = C_{\sim}$, *(ii)* $C_3^{\approx} = C_{\approx}$ *and* *(iii)* $C_3^{\mathbf{te}} = C_{\mathbf{te}}$

Thus, interpretations of behavioural compatibility in LOTOS which are based on one of the language's equivalences are easily reflected in our general definition of consistency. But, $C_3^{\mathbf{cs}}$ is not transitive, c.f. proposition 6, so corollary 1 does not get us a relationship between $C_3^{\mathbf{cs}}$ and $C_{\mathbf{cs}}$. In fact, we have the following result.

Proposition 11
$C_3^{\mathbf{cs}} \subset C_{\mathbf{cs}}$.

Proof
Firstly, $P_1\ C_3^{\mathbf{cs}}\ P_2 \implies P_1\ C_{\mathbf{cs}}\ P_2$, follows immediately from the reflexivity of **cs**, i.e. either of P_1 or P_2 could act as the required common **cs**-development.

In addition, we can provide a counterexample to show that, $C_{\mathbf{cs}} \not\subseteq C_3^{\mathbf{cs}}$. Consider, $P_1 := i; a; stop[]b; c; stop$, $P_2 := i; a; stop[]b; stop$ and $P := i; a; stop$. Now, P $\mathbf{cs}$ P_1 and P $\mathbf{cs}$ P_2, but $\neg(P_1\ \mathbf{cs}\ P_2)$. This is because $\neg(P_2\ \mathbf{conf}\ P_1)$ as P_2 refuses c after the trace b, but P_1 cannot refuse c after the same trace. $\Box$

This result is disappointing, but interesting. The counterexample provided is one of the few situations in which the unification has a smaller trace set than both the original specifications and furthermore a unification with a larger trace set does not seem to exist for this example. This observation motivates the following, which considers a development relation in which the trace set increases. Thus, we define extended **conf** symmetric, denoted **xcs** as:-

Definition 13 P_1 **xcs** P_2 *iff* P_1 **cs** P_2 $\wedge$ $Tr(P_1) \supseteq Tr(P_2)$.

An alternative derivation of **xcs** is: P_1 **xcs** P_2 iff P_1 **ext** P_2 $\wedge$ P_2 **conf** P_1 (a definition of **ext** can be found in [SBD95]). So, we have added a trace extension constraint on the development. Note in particular that using **xcs** as development relation in C will rule out the counterexample used in the previous proposition. So let us try to relate $C_{\mathbf{xcs}}$ and $C_3^{\mathbf{cs}}$.

Proposition 12
$C_{\mathbf{xcs}} \subseteq C_3^{\mathbf{cs}}$

Proof
Assume $P_1\ C_{\mathbf{xcs}}\ P_2$, i.e., $\exists P \cdot P\ \mathbf{conf}\ P_2 \wedge P_2\ \mathbf{conf}\ P \wedge Tr(P) \supseteq Tr(P_2) \wedge P\ \mathbf{conf}\ P_1 \wedge P_1\ \mathbf{conf}\ P \ \wedge\ Tr(P) \supseteq Tr(P_1)$
which expands to:-

$(i)\ \forall \sigma\ \in Tr(P_2),\ \ Ref(P,\sigma) \subseteq Ref(P_2,\sigma) \wedge$
$(ii)\ \forall \sigma'\ \in Tr(P),\ \ Ref(P_2,\sigma') \subseteq Ref(P,\sigma') \wedge$
$(iii)\ \forall \sigma''\ \in Tr(P_1),\ \ Ref(P,\sigma'') \subseteq Ref(P_1,\sigma'') \wedge$
$(iv)\ \forall \sigma'\ \in Tr(P),\ \ Ref(P_1,\sigma') \subseteq Ref(P,\sigma') \wedge$
$(v)\ Tr(P) \supseteq Tr(P_1), Tr(P_2)$

From properties (i), (iv) and (v) we get:-
$\forall \sigma_1 \in Tr(P_2),\ \ Ref(P_1,\sigma_1) \subseteq Ref(P,\sigma_1) \subseteq Ref(P_2,\sigma_1)$
i.e. P_1 **conf** P_2. Similarly, P_2 **conf** P_1 since properties (iii), (ii) and (v) give us:
$\forall \sigma_2 \in Tr(P_1),\ \ Ref(P_2,\sigma_2) \subseteq Ref(P,\sigma_2) \subseteq Ref(P_1,\sigma_2)$
Notice these relationships can only be derived because $Tr(P) \supseteq Tr(P_1), Tr(P_2)$. $\square$

So, we have the direction of implication that we could not get with $C_{\mathbf{cs}}$, but now the other implication direction is more difficult as we need to show a unification with trace extension exists. Before we consider this we need a simple result, which is a consequence of the definition of **cs**.

Proposition 13
$P_1\ \mathbf{cs}\ P_2 \implies \forall \sigma \in Tr(P_1) \cap Tr(P_2),\ \ Ref(P_1,\sigma) = Ref(P_2,\sigma).$

We will use the following unification construction:-

Denote $\mathcal{U}_x(P_1,P_2)$ as the set of all LOTOS specifications characterised by the following constraints:-

$$Tr(\mathcal{U}_x(P_1,P_2)) = Tr(P_1) \cup Tr(P_2) \quad \wedge \qquad -(a)$$
$$\forall \sigma \in Tr(\mathcal{U}_x(P_1,P_2)),$$
$$\sigma \in Tr(P_1) \cap Tr(P_2) \implies Ref(\mathcal{U}_x(P_1,P_2),\sigma) = Ref(P_1,\sigma) = Ref(P_2,\sigma) \quad \wedge \qquad -(b)$$
$$\sigma \in Tr(P_1) - Tr(P_2) \implies Ref(\mathcal{U}_x(P_1,P_2),\sigma) = Ref(P_1,\sigma) \quad \wedge \qquad -(c)$$
$$\sigma \in Tr(P_2) - Tr(P_1) \implies Ref(\mathcal{U}_x(P_1,P_2),\sigma) = Ref(P_2,\sigma) \qquad -(d)$$

Notice that (b) is only possible because of proposition 13. It should also be noted that this construction is well founded and will always yield a LOTOS specification. One justification for this is that Leduc [Led91] performs the same construction with his rooted failure tree model (definition 6.4.1 on page 153) and shows that the resulting tree is well-formed, i.e. can be mapped to a labelled transition system.

Proposition 14
$C_3^{\mathbf{cs}} \subseteq C_{\mathbf{xcs}}$.

Proof
Assume $P_1\ C_3^{\mathbf{cs}}\ P_2$, i.e. P_1 **cs** P_2, then take $X \in \mathcal{U}_x(P_1, P_2)$, we suggest that X is a common **xcs** development of P_1 and P_2, as required by $C_{\mathbf{xcs}}$. Let us show that X **xcs** P_1. We will show first that X **conf** P_1, then that P_1 **conf** X and then that $Tr(X) \supseteq Tr(P_1)$.

(i) (X **conf** P_1). Take $\sigma \in Tr(P_1)$. Now, if σ is also a trace of P_2, by (b), $Ref(X, \sigma) = Ref(P_1, \sigma)$, however, if $\sigma \notin Tr(P_2)$, by (c), $Ref(X, \sigma) = Ref(P_1, \sigma)$.

(ii) (P_1 **conf** X). Take $\sigma \in Tr(X)$. We have the following cases:-
(a) $\sigma \in Tr(P_1) \cap Tr(P_2) \implies Ref(P_1, \sigma) = Ref(X, \sigma)$, by (b).
(b) $\sigma \in Tr(P_1) - Tr(P_2) \implies Ref(P_1, \sigma) = Ref(X, \sigma)$, by (c).
(c) $\sigma \in Tr(P_2) - Tr(P_1) \implies Ref(P_1, \sigma) = \emptyset$ this is because $\sigma \notin Tr(P_1)$, which implies that $Ref(P_1, \sigma) \subseteq Ref(X, \sigma)$

(iii) ($Tr(X) \supseteq Tr(P_1)$). This is immediate from (a).

Thus, X **xcs** P_1 and it can be similarly verified that X **xcs** P_2. □

Corollary 2
$C_3^{\mathbf{cs}} = C_{\mathbf{xcs}}$.

This result completes our relating of C_3 to C and shows that all obvious LOTOS instantiations of behavioural compatibility in C_3 can be given an equivalent formulation in C and justifies proposition 5.

Finally, it is worth pointing out that the purpose of defining the relation **xcs** is to embrace $C_3^{\mathbf{cs}}$ within our framework and not to develop a new practical conformance relation.

6 CONCLUDING REMARKS

We have proposed a general definition of consistency and shown that it embraces the three existing RM-ODP definitions. In addition, our interpretation fulfils the main requirements for a definition of consistency: it is applicable intra and inter language, supports different classes of consistency checking, supports global consistency and unbalanced as well as balanced consistency.

Viewpoint consistency is a very large and demanding research area. Here we have only been able to consider one aspect of the issue, however, we refer the interested reader to the following further work on the topic: a complete framework for consistency, a presentation of the properties of our definition of consistency and an investigation of consistency in LOTOS and in Z (including unification algorithms) can be found in [BBDS95] and an investigation of translation between FDTs can be found in a companion paper to this paper [DEBS96].

REFERENCES

[ACGW94] 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.

[BBDS95] E. Boiten, H. Bowman, J. Derrick, and M. Steen. Cross viewpoint consistency in Open Distributed Processing (intra language consistency). Technical Report 8-95, Computing Laboratory, University of Kent at Canterbury, 1995.

[BDS95] H. Bowman, J. Derrick, and M.W.A. Steen. Some results on cross viewpoint consistency checking. In K. Raymond and L. Armstrong, editors, *IFIP TC6 International Conference on Open Distributed Processing*, pages 399–412, Brisbane, Australia, February 1995. Chapman and Hall.

[BSS86] 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.

[DEBS96] 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. To appear.

[Led91] G. Leduc. *On the Role of Implementation Relations in the Design of Distributed Systems using LOTOS*. PhD thesis, University of Liège, Liège, Belgium, June 1991.

[Lin95] P. F. Linington. RM-ODP: The Architecture. In K. Raymond and L. Armstrong, editors, *IFIP TC6 International Conference on Open Distributed Processing*, pages 15–33, Brisbane, Australia, February 1995. Chapman and Hall.

[Mil89] R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.

[SBD95] M. Steen, H. Bowman, and J. Derrick. Composition of LOTOS specifications. In P. Dembinski and M. Sredniawa, editors, *Protocol Specification, Testing and Verification*, Warsaw, Poland, 1995. Chapman & Hall.

Dr. H. Bowman received his Ph.D. from and was a postdoctoral research associate at the University of Lancaster. He is currently a lecturer at the University of Kent where he is working on the application of formal techniques to the construction of distributed systems. His current research interests include the application of FDTs in ODP and the formal specification and validation of multimedia systems.

Dr. E.A. Boiten received his Ph.D. in 1992 from the University of Nijmegen, for a thesis on formal program development (algebraic specification, functional programming, transformations). He has since worked on various aspects of formal methods in post-doctoral research positions in Nijmegen and Eindhoven. Since 1995 he has been a research associate on the project 'Cross Viewpoint Consistency in ODP' at the University of Kent.

Dr. J. Derrick gained a D.Phil in 1987 from Oxford, he then worked on the RAISE project at STC Technology. Since 1990 he has been a Lecturer in Computer Science at the University of Kent. He has worked on the theoretical foundations of formal methods and their application to ODP and distributed computing. His current interests include developing techniques for the use of FDTs within ODP and formal definitions of consistency and conformance.

Ir. M.W.A. Steen obtained an MSc(Eng) in Computer Science from the University of Twente in 1993. He is currently supported by an E.B. Spratt bursary from the University of Kent to do a Ph.D. in Computer Science. His research interests include *refinement* and *partial specification* in process algebras and the application of formal methods to ODP.