

On Interactions in the RM-ODP

Guy Genilloud, Gonzalo Génova

*Departamento de Informática, Universidad Carlos III de Madrid,
avda. Universidad, 30 – 28911 Leganés, Madrid, Spain
guy.genilloud@ie.inf.uc3m.es, ggenova@inf.uc3m.es*

Abstract

The RM-ODP definition of *interaction* is subtle and difficult to understand. On closer inspection, we find that it is in fact invalid. We then attempt to establish the author's original intentions for this concept, and to provide an effective definition. Our investigations lead us to discover some fundamental flaws in the definitions of *interface* and of *behaviour of an object*.

1. Introduction

ISO has performed excellent work with the RM-ODP Foundations [1]. Unfortunately, this work has failed to influence the most popular object-oriented modelling techniques and languages (UML being a case in point), and methodologies. Their creators have either ignored the RM-ODP, or failed to understand it. To their credit, some definitions are excessively subtle or obscure, and the RM-ODP lacks explanations, and references to relevant literature.

In this paper, we analyse one of these subtle definitions: that of the concept of *interaction*. This definition is dependent on that of *environment (of an object)*. While both definitions may seem crisp and clear on a first read, they turn out to be meaningless since based on an apparent circularity. We then investigate whether new definitions, making use of the concept of role, would work. Having hopefully understood, through our analysis, the intentions of the RM-ODP authors, we provide an explanation for the concept of interaction. This analysis allows us to propose alternate definitions. Finally, armed with our new understanding that interactions have roles, we find fundamental flaws in two RM-ODP definitions, those of *interface* and *behaviour (of an object)*.

2. The Environment of an Object

The RM-ODP foundations provide this definition:

8.2 Environment (of an object): *the part of the model which is not part of that object.*

NOTE - *In many specification languages, the environment can be considered to include at least*

one object which is able to participate without constraint in all possible interactions (see 8.3), representing the process of observation. [1]

While its note is unlikely to be helpful to most readers, this definition seems at first to be straightforward. But now, consider the 4th note in the very next definition in the standard, that of action:

8.3 Action: *Something which happens.*

Every action of interest for modelling purposes is associated with at least one object.

The set of actions associated with an object is partitioned into internal actions and interactions.

*An **internal action** always takes place without the participation of the environment of the object. An **interaction** takes place with the participation of the environment of the object.*

NOTES

1 - *"Action" means "action occurrence".*

Depending on context, a specification may express that an action has occurred, is occurring or may occur.

2 - *The granularity of actions is a design choice. An action need not be instantaneous. Actions may overlap in time.*

3 - *Interactions may be labelled in terms of cause and effect relationships between the participating objects. The concepts that support this are discussed in 13.3*

4 - *An object may interact with itself, in which case it is considered to play at least two roles in the interaction, and may be considered, in this context, as being a part of its own environment.*

5 - *Involvement of the environment represents observability. Thus, interactions are observable whereas internal actions are not observable, because of object encapsulation. [1]*

We have 2 contradictory statements: the environment of an object is *"the part of the model which is not part of that object,"* and *"an object can be a part of its own environment."* We assume that it is the second statement, of minor status since in a note, which is incorrect. Peter Linington, one of the fathers of the RM-ODP, confirmed to us that the idea of an object being part of its own

environment is metaphorical (as suggested by the phrase “in this context”). Therefore, an object cannot actually be a part of its own environment.

This metaphorical note is unfortunate, not only because it can confuse readers of the standard, but also because it hides the fact that the concept of interaction is ill-defined. If taking place with the participation of the environment is not a discriminator of an interaction (vs. an internal action), then what is an interaction of an object? We will investigate this question in Section 3.

As for the notion of environment, we believe that it requires further explanations, as we discuss below.

2.1. Observability.

Studying the standard, it is clear that observability was the main concern, if not the only concern, for introducing the notion of environment (“*Involvement of the environment [in an action] represents observability.*” [1, Definition 8.3 Action, Note 5]). But why does the RM-ODP speak of observability, and not of observation?

A possible answer could be that the readers of a specification do not know whether an observer object in the environment actually does observe the action. That is, they do not have full knowledge of the environment of an object – there might be objects in the environment they know nothing about.

But what about specification techniques, such as classical UML [2], in which at most two objects may participate in an interaction? When an object interacts only with itself, no other object may observe this interaction. So why does ODP consider such an action as observable, when in fact it actually cannot be observed?

2.2. Asynchronous interactions.

In a recent paper [3], Peter F. Linington mentions another concern for the environment of an object. As he explains, the RM-ODP enables objects to interact either in a synchronous way (the objects are simultaneously in the interaction, as if they were shaking hands¹), or in an asynchronous way (e.g., the exchange of a message consists of a sequence of synchronous interactions, initiated by the submission of the message, and terminated by its delivery). The notion of environment plays a key role in achieving this capability:

Interactions in ODP are synchronous², but are defined between an object and its environment. If

¹ The notion of synchronicity discussed here has nothing to do with the notion of synchronicity in remote procedure calls.

² P. F. Linington probably meant to write that *atomic* interactions are synchronous. Otherwise, the communication of a message, a composite action, could not be considered to be an interaction...

we then constrain the way objects are composed so that each object binds directly to another object in its environment, the resulting communication is synchronous, but if objects bind to link ends in their environment, the resulting communication is asynchronous. [3]

Such explanations would be interesting and very helpful to the many object-modellers who consider that objects only interact by exchanging messages (the authors of UML among them). Unfortunately, there is not even a hint about this idea in the RM-ODP Foundations. There is not even an explanation that communication links exist in a model (as binding objects), even though they are not specified explicitly (e.g., UML and ESTELLE [4]).

2.3. Are all communications observable?

The idea of asynchronous communication via (implicit) communication links suggests that all asynchronous communications are observable (if an object sends a message to itself, this composite action would be observed by communication links in its environment). However, was this the RM-ODP authors intention?. We are not so sure.

The notion of observation of actions is important with respect to behavioural compatibility³ [1, definition 9.4]: if an object’s behavioural specification is substituted for another, would some environment be able to notice the difference? However, since implicit (unspecified) communication links are unable to tell other objects any differences they may notice, should they count as observers? We think not.

The actions of submitting a message, or delivering it, would then never be (directly) observable (even though asynchronous communications may or may not be observable). Therefore, we would have some actions (e.g., sending a message) which involve participation of the environment, but which are nevertheless unobservable. Or perhaps, the definition of the environment of an object needs being revisited?

The RM-ODP definition of environment is therefore deceptively simple, and perhaps even misleading. The following is a list of questions, which we think the standard should answer:

- Is the environment of an object the same thing as all the other objects in the model?
- Are all the other objects in a model necessarily known to a specifier?
- If an object sends a message to itself, using implicit communication links, is that necessarily an observable action?

³ The RM-ODP should make this point clear, as soon as introduces the notion that some actions are observable.

3. Interactions

3.1. Which actions are interactions?

Intuitively, one would define an interaction as an action in which more than one object participates, and an internal action as an action in which a single object participates. However, this is not the idea that was retained for the RM-ODP. Instead, the participation of the environment was elected to be the discriminating criterion:

*“An **internal action** always takes place without the participation of the environment of the object. An **interaction** takes place with the participation of the environment of the object.” [1, Definition 8.3 Action].*

This definition is unsatisfactory, for at least two reasons:

- 1) Environment is not a crisp and clear concept, as we have seen.
- 2) An object may interact with itself, without the participation of its environment (assuming that an object cannot be part of its own environment).

Regarding the second point, we suppose that the authors' intention was to classify some actions as interactions, independently of the actual involvement of the environment. So, for example, when a Java runtime object invokes one of its own methods, ODP would categorize this action as an interaction rather than as an internal action (because other objects may also invoke the same method). The fact that the RM-ODP considers interactions to be those actions that are observable [1, Definition 8.3 Action, Note 1], rather than those that are actually observed, comforts us in this interpretation.

Involvement of the environment is therefore not a necessary criterion for an action to be considered an interaction of an object⁴.

3.2. Roles in interactions

As we have seen, the RM-ODP authors deliberately attempted to classify some unobserved actions as interactions (observable) rather than as internal actions (unobservable). However, they failed in their attempt of defining a classification.

When being presented with these contradictions, Peter Linington and Jean-Bernard Stefani, two of the most prominent authors of the RM-ODP, proposed to discriminate between interactions and internal actions by referring to the notion of role [5, 6]. However, they did not elaborate much on their respective answers. We will

⁴ In the RM-ODP Overview, Section 7.1.2, it is said explicitly *“Note that these interactions do not necessarily occur with other objects: an object can interact with itself.”*

now examine if such a classification based on roles is indeed possible.

In fact, the 4th note of the RM-ODP definition of action already makes a reference to the concept of role: *“an object may interact with itself, in which case it is considered to play at least two roles in the interaction”*). This yields an interesting observation: an interaction has multiple *roles*. And indeed, it seems true that every interaction has multiple roles:

- 1) In classical object-orientation, there is only one type of interaction between objects: the exchange of a message between two objects. This is a composite interaction⁵ with two roles, which we may call the sender and the receiver. An object may send a message to itself, in which case it is both the sender and the receiver of that composite interaction.
- 2) A rendezvous between two objects is an example of a synchronous interaction. Such an action serves the purpose of synchronising two activities, the rendezvous being a common action in both activities. There are two roles because two activities are required (otherwise, the rendezvous action would be pointless). Generally, two different objects will participate in a rendezvous, but it is also possible for a single object to play both roles in the rendezvous.

For every interaction type that we may conceive, it seems indeed possible and natural to consider that it has multiple roles. So, we may consider that an action with multiple roles is an interaction, and that an action with just one role, or with no role at all, is an internal action.

However, the question that we must answer is not, given an action, whether this action is an interaction of some object(s). Rather, the question is, given an object and one of its actions, whether this action is an interaction of that object, or one of its internal actions. Answering this second question is not straightforward, because as we will now see, an object can be a composite object.

3.3. Interactions of composite objects

In the RM-ODP, an object can be a composite object, expressed as the composition of more elementary objects.

9.1 Composition:

a) (of objects) A combination of two or more objects yielding a new object, at a different level of abstraction. The characteristics of the new object are determined by the objects being combined and by the way they are combined. The behaviour of a composite object is the corresponding composition of the behaviour of the component objects. [1]

⁵ This interaction is composite since composed of sub-actions, among them submitting the message onto the implicit communications channel, and the message delivery.

Considering the behaviour obtained by pure composition (i.e., without applying any abstraction⁶), all the actions of the *component objects* are also actions of the composite. Obviously, any action internal to one of the components is an internal action of the composite. And any interaction of a component object with the environment of the composite is an interaction of the composite. But what about the interactions of component objects among themselves? Are they interactions of the composite with itself, or are they internal to it?

If we were to use the criterion that an interaction has roles, we would conclude that all the interactions among the components are interactions of the composite. But this conclusion would not reflect what happens in practice. For example, most systems have communications among their components that are not observable by external entities.

But note that definition 9.1 says explicitly that a composite object is defined not only by which objects compose it, but also by how they are combined. Some specification languages enable a specifier to declare, with a *hide* operator, that some interactions among the component objects will be internal actions of the composite (therefore hiding them to the environment of the composite; no actions are suppressed). A similar effect can be obtained by using devices such as interceptors that make it impossible for an external object to access some components of the composite object. In the ODP computational language [7], the behaviour of components might be such that some interface references are not communicated outside of the composite, making it impossible for external objects to have interactions at those interfaces: all interactions at such interfaces shall therefore be considered to be internal actions of the composite.

In [3], Peter Linington explains that the identification of interfaces is a design activity. That is, it is design decisions that allocate interactions to interfaces. The same idea applies with respect to deciding whether an interaction among components is an interaction of the composite, or one of its internal actions. Note that the design decision may not be left entirely to the specifier; it also depends on the specification language that he or she uses.

4. Interactions: an explanation

We have no doubts at all that the RM-ODP authors tried to define interactions and internal actions in a sensible and useful way. It is no accident that they did not define

⁶ We do not understand why the RM-ODP pretends that a composite object is necessarily at a more abstract level of abstraction than its component objects. Surely, composition enables abstraction, but does it always include it?

interaction in the straightforward way, that is, as an action in which two or more objects participate. But they failed to provide clear and unambiguous definitions. Even more annoyingly, we do not even know what were the authors' intentions.

What the authors should have done was to explain the definitions they wrote, and in particular the intentions behind the non-straightforward definitions. We will now try, given our analysis and understanding of the issue, to provide these missing explanations.

We suppose that the RM-ODP authors were very much concerned with the problem of writing behavioural specifications for objects. We also know that some ODP authors are not always coherent with the standard, as they entertain confusion between a term and the model element to which it refers. In particular, they incorrectly speak of "actions in a specification" (see for example [8]), which does not make sense since an action is "*something which happens*." And of course, a specification is a static entity in which nothing may happen. An "action in a specification" may be explained as a term for an action in a model⁷. Or more precisely, it is a term for one or more actions, as a same term can be interpreted again and again, each time referring to a different action.

In [3], Peter F. Linington provides an example of this situation in the context of ODP. He observes that an object can theoretically participate in infinitely many actions. As he explains, "*most objects will have a behaviour that allows arbitrary repetition of smaller fragments of behaviour associated with some sort of thread or session*." We would say instead that *the term for a fragment of behaviour* (itself composed of terms for actions) can be interpreted again and again to refer to different fragments of behaviour⁸. This property is essential for enabling an object to have an infinite behaviour, since a specification can only have a finite number of terms. More pragmatically, this property is also very important for writing specifications of reasonable length, as any programmer could testify.

Thus, a term of an action generally refers to many actions. Now, consider a term of an action that is related to actions in which it is *always* the case that only one object participates. ODP authors (we suppose) would consider all these actions to be *internal actions of this object*. Consider another term, which in general refers to an action in which this object plays just one role, but which may refer to an action in which this very same object plays all the roles. ODP authors (we suppose) would consider all these actions to be *interactions of this object*.

⁷ As we announced in Section 1, we use the term "model" in its ODP meaning, and not in its UML meaning.

⁸ Strictly speaking, Linington's sentence is incorrect, since a fragment of behaviour, being a collection of actions (occurrences), cannot be repeated.

Our explanation is compatible with [1, definition 8.3], in which it is said that “*An internal action always takes place ...*” Since an action takes place only once, why did the authors of the RM-ODP write “always takes place”? We suppose that they were thinking about the term of the action.

4.1. What it means to be observable

In the RM-ODP foundations, the notion of observability (of actions) is important in the context of another notion, that of behavioural compatibility (between objects):

9.4 Behavioural compatibility: An object is behaviourally compatible with a second object with respect to a set of criteria (see notes) 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. [1]

Since an ODP object is “*the model of an entity*” [1], it serves no purpose to consider two objects, and to wonder whether or not they are behaviourally compatible. The question of behavioural compatibility is in fact about object specifications, on whether one is compatible with the other⁹. More precisely, the problem is to know whether an object’s environment (i.e., the environment of the object in some model) would be able to find out which of two possible specifications applies to this object. Nothing is known about the object’s environment, besides some assumptions (derived from the set of criteria in the above definition). For the environment to observe anything about the object, it must participate in interactions with it.

Therefore, we are talking here about many actions in many models. What we really want to classify is not the actions of an object, but the terms of its actions in a specification. An action term is said to be “observable” if it refers to at least one action, in some model, which may be observed by the environment of the object. Unless an action term can be proven to always refer to unobserved actions of an object, it is to be considered an “observable” action term of the object specification¹⁰.

Thus, our explanation is that ODP considers an action to be observable if it is referred to by an “observable” action

term; otherwise it is non-observable. And the name “interaction” is a synonym for “observable action”.

Thus, in a given model, some interactions of an object are observed while others are not. But they are all considered to be observable, in this particular ODP sense. This is the case even though the environment, in this given model, cannot in fact observe them. On the other hand, no internal actions of the object are ever observed, in any model, and they are therefore non-observable.

4.2. An alternative explanation

In a recent private communication, Peter Linington provided us with an interesting explanation of what it means for an object to interact with itself, and to be in such context a part of its own environment:

“The text about an object contributing to its own environment is in a note and is clearly not the defining occurrence. In fact the phrase “in this context” is a strong hint that the usage is metaphorical. The point being made here is that when an object interacts with itself, there is a degree of decoupling, so that the constraints on the interaction arise from the object playing two roles, and for each role the actions of the other are as if from the environment. This implies that static analysis of the behaviour of an object as a configuration component may indicate liveness that is not guaranteed if the two roles are closely coupled by the internal behaviour of the object; the situation goes from possible liveness to necessary deadlock.”

We do agree with the above explanation, but we are not sure that it is sufficient. We therefore choose to stick to our own explanations, until convinced otherwise.

5. Proposed new definitions

Having (hopefully) understood the original ODP authors’ intentions, we are now in a position to propose some alternate definitions of the concepts of environment and interaction. But first of all, we would like to stress that what matters first and foremost is the message we want to communicate to the readers of the standard. It would be better to leave the RM-ODP Foundations with its current deficient definitions, but to add proper explanations, than to provide revised but obscure (or excessively subtle) definitions. And after all, the new definitions might prove to be defective again, in which case the explanations would be sorely missing.

⁹ Whereas object is a basic modelling concept in the RM-ODP, behavioural compatibility is a specification concept.

¹⁰ It is in general an undecidable problem to determine which action terms of an object specification may refer to (in some interpretation) an observed action. For that reason, the principle of precaution applies: in case of doubt, an action term of an object specification is considered to be “observable.” This matters, for example, when trying to determine whether an object specification is equivalent to another.

5.1. Environment, and communications

The following is our proposed revised definition for the environment of an object:

8.2 Environment (of an object): all the objects in the model which are not part of that object.

NOTES

0- An object is never a part of its own environment.

1 - In many specification languages, the environment can be considered to include at least one object which is able to participate without constraint in all possible interactions (see 8.3), representing the process of observation.

2- In some specification languages, an object can be considered to include, as parts of itself, outbox and inbox objects, even though those are not explicitly specified. An internal outbox object enables a sender object to submit a communication (see 8.9) with its environment, without that action being observed directly (the communication as a whole is observed, but not its initiation). Likewise, an internal inbox object enables the delivery of a communication, without that action being observed by the environment of the receiver object.

Note 0 is provided here to stress the change in this new definition from what is said in the original RM-ODP Foundations standard.

Note 1 may need to be suppressed or amended, since it explains the process of observation in a narrow way, valid only in the case of some specification languages. But it has the merit to introduce a reference to the notions of interaction and observation, which are the very reasons behind the definition of environment. And they also help with Note 2. But whether or not this remains, it is essential for the revised standard to explain the very reasons behind the concepts of environment and observability.

Note 2 may need to refer to complementary explanations regarding the exchange of asynchronous communications between objects. In particular that a communications channel consists of a configuration of objects, among which an outbox object (or inbox object) which is part of the object itself, and other binding objects which are parts of the object's environment. The standard should explain that inbox and outbox objects are internal to an object for reasons of evaluating whether two object specifications are behaviourally compatible or not. Since implicit communication channels are beyond the control of any specifier, it is not necessary, but preferable, to consider that submitting a communication or receiving it are internal actions of the sender and the receiver, respectively.

5.2. Interactions

We do not see a way to define interaction without referring to specifications and terms. We first need this definition:

Observable action term (of an object): An action term is **observable** in an object specification, if it refers to at least one action, in some model, which may be observed by the environment of the object. Otherwise, this action term is **non-observable**.

NOTE: In case of doubt, an action term is observable: if an action term cannot be proven to be non-observable, it is to be considered as being observable.

Internal action (of an object): an action of an object which is referred to by a *non-observable action term* in the specification of that object.

NOTE: Not a single internal action of an object is observed by its environment.

Interaction (of an object): an action of an object which is referred to by an *observable action term* in the specification of that object.

NOTES:

1- An object may play all the roles of an interaction, in which case this action is unobserved (by this object's environment). This action is still categorized as an interaction of this object because it is referred to by an observable action term in the object's specification.

2- Some specification techniques are such that an action is referred to by several action terms. In this case, it is sufficient for one of these action terms to be deemed observable for the action to be an interaction.

Since observable in ODP has a non-obvious, counter-intuitive meaning, we propose to deprecate the use of the phrase "observable action".

6. Other issues in the RM-ODP

In our investigation, we have realized that interactions are characterized by having multiple roles. This fact was not taken into account when writing some of the definitions of the RM-ODP Foundations, which as a result are flawed.

6.1. Interface

The definition of interface needs revisiting:

8.4 Interface: An abstraction of the behaviour of an object that consists of a subset of the interactions of that object together with a set of constraints on

when they may occur.

Each interaction of an object belongs to a unique interface. Thus the interfaces of an object form a partition of the interactions of that object.

NOTES

1 - An interface constitutes the part of an object behaviour that is obtained by considering only the interactions of that interface and by hiding all other interactions. Hiding interactions of other interfaces will generally introduce non-determinism as far as the interface being considered is concerned.

2 - The phrase "an interface between objects" is used to refer to the binding (see 13.4.2) between interfaces of the objects concerned.[1]

The second paragraph, which associates an interaction to a single interface, is wrong. It would be true in the ANSA architecture [9], in which a binding establishes a shared interface between several objects. The RM-ODP was largely inspired from ANSA, but its concept of interface is different: an ODP interface belongs to a single object, and an ODP binding binds two or more interfaces (see [1, definition 13.4.2] and [7, clause 7.2.3.2]). Therefore, each ODP interaction is necessarily associated with two or more interfaces.

It is also wrong to pretend that an interaction is necessarily associated with a single interface of a participant object. Applied in the context of the ODP computational language, such a statement would disallow an object to bind two of its own interfaces, client and server, and therefore to interact with itself. Peter Linington and Jean-Bernard Stefani confirmed that these two statements do not reflect the RM-ODP authors' intention [5, 6].

We propose to rewrite these two statements as follows: "Each interaction role played by an object belongs to a unique interface of that object. Thus the interfaces of an object form a partition of the interaction roles played by that object." The first note of the definition would need to be revised accordingly, since the roles at other interfaces must be hidden as well.

6.2. Behaviour

Interaction roles are not mentioned at all in the definition of behaviour in the RM-ODP, of which we reproduce here its first sentences:

8.6 Behaviour (of an object): A collection of actions with a set of constraints on when they may occur.

The specification language in use determines the constraints which may be expressed. Constraints may include for example sequentiality, non-determinism, concurrency or real-time constraints.

[1]

Knowing which role an object plays in an interaction is critical to knowing its behaviour. For example, asking a question is not quite the same thing as answering it. But the current definition of behaviour implies that it is sufficient to say that an object participates in an interrogation...

7. Summary and conclusions

In writing this paper, we spent a lot of time and efforts trying to understand just one concept, that of interaction. We indeed discovered that it was not really defined in the RM-ODP. In doing so, we had to go past some subtle statements and metaphors, and second-guess the author's original intentions behind them. Our work would have been much simpler, and more effective, had these intentions been documented in the standard itself, or in a companion document.

Our investigations lead us to discover some fundamental flaws in the definitions of *Interface* and of *Behaviour of an object*. These flaws have remained unnoticed (or at least unreported) for more than ten years. However, when one is armed with the understanding that interactions have roles, these flaws are fairly obvious.

While performing our research, we came to better appreciate the considerable difficulty of writing the RM-ODP Foundations: the definitions must be notation independent (even syntax independent), grounded in semantics, very general and widely applicable, and yet precise. Compared with other works of the same nature, ISO has performed excellent work. Unfortunately for the object and software engineering communities, the RM-ODP Foundations have not had yet the impact that they deserve. One reason is that the standard lacks proper explanations and references. We invite ISO experts to provide these missing explanations, rather than just fix the flawed definitions. After all, the new definitions might prove to be defective again, in which case the explanations would be sorely missing.

8. References

- [1] ISO/IEC and ITU-T, "Open Distributed Processing - Basic Reference Model - Part 2: Foundations," in *Standard 10746-2, Recommendation X.902*, 1995.
- [2] OMG, "OMG Unified Modeling Language Specification v. 1.5," OMG, OMG Standard March 2003.
- [3] P. F. Linington, "What Foundations does the RM-ODP Need?," presented at Workshop on ODP for Enterprise Computing (WODPEC 2004), Monterey, California, 2004.

- [4] ISO/IEC, "Information processing systems - Open systems interconnection - Estelle : a formal description technique based on an extended state transition model," in *Standard 9074*, 1989.
- [5] P. F. Linington, "Personal email communication, Aug. 31," 2000.
- [6] J.-B. Stefani, "Personal email communication, Sept. 22," 2000.
- [7] ISO/IEC and ITU-T, "Open Distributed Processing - Basic Reference Model - Part 3: Architecture," in *Standard 10746-3, Recommendation X.903*, 1995.
- [8] P. F. Linington and W. F. Frank, "Specification and Implementation in ODP," presented at 1st Workshop on Open Distributed Processing: Enterprise, Computation, Knowledge, Engineering and Realisation (WOODPECKER'2001), Setubal, Portugal, 2001.
- [9] A. J. Herbert, "An ANSA Overview," *IEEE Network*, pp. 18--23, 1994.