Proposal for a Model Driven Approach to Creating a Tool to Support the RM-ODP

D.H.Akehurst
University of Kent
D.H.Akehurst@kent.ac.uk

Abstract

The potential for revising the RM-ODP standards is an excellent opportunity to move the given specifications into a form that is more amenable to the provision of tools that support the standard. Adoption of the approach advocated by the RM-ODP would be greatly increased if tools to support the approach were more readily available. This paper proposes an approach to generating such tools, directly, from a model based approach to specifying the RM-ODP viewpoint languages and correspondences. In particular this paper highlights certain requirements that the specification approach would need to meet if the production of such tools were to be achievable.

1. Introduction

The Reference Model for Open Distributed Processing (RM-ODP) [11] belonging to the International Standards Organisation (ISO) was published about ten years ago. Although originally developed with respect to the telecommunications industry, as we move into a situation where more and more software systems are essentially distributed, this reference model becomes similarly more and more relevant to software development.

Over the last few years there has been more focus on supporting distribution and an increase in the number of different technologies to support distributed software systems; originally CORBA, COM, DCOM and lately Web Service based approaches such as .NET and J2EE.

Even though these distribution technologies have been proposed, the take up and use of the RM-ODP has not been as common place as its relevance and potential usefulness would lead us to expect. Contrast this with the outputs from the OMG, such as the UML and latterly their MDA approach, which appear to be widely used and are certainly commonly discussed.

There could be a number of reasons for this, marketing and publicity being one possible candidate. However, another reason is the accessibility of the OMG outputs which are nearly always supported by tools (even if they are not considered to be the best possible tools), these give practitioners a tangible artefact by which to evaluate and try out the proposed technologies.

Where are the easy to use graphical language based tools to support the RM-ODP? Perhaps such a tool cannot be created and at the same time be technically sound. However, it would be useful to have one that provides the easy accessibility offered by the numerous UML tools, which, it could be argued, are not always so technically sound but do provide an easy way into the world of OMG-Oriented Modelling.

Development of a simple graphical tool based on the RM-ODP is not quite as straight forward as it is for languages such as UML. Firstly, the RM-ODP proposes a five viewpoints approach to system design involving at least as many separate but related sets of concepts but specifically does not prescribe any particular language with which to write design specifications. Secondly, the concepts recommended for each viewpoint are described using English text (as opposed to the OMG's approach of modelling the language concepts – known as metamodelling).

Thus, before we can build a tool that supports the design of systems using the RM-ODP approach, we need to address these two issues. This position paper proposes using the OMG's approach to solving the second issue, by forming metamodels that define the concepts described in the ODP Standard documents, as has been done for some of the viewpoints already [2, 9, 10] The first issue, that of viewpoints and languages, we can address by firstly defining notations that map to the metamodel concepts and secondly by specifying the inter viewpoint relationships as relations between metamodel concepts. Both of these (inter viewpoint correspondences, and notations) can be defined using the relation based transformation specification technique taken from the MDA initiative [8].

In the following sections we first discus the five viewpoint language concepts and their corresponding metamodels; secondly we propose a technique for defining notations for each viewpoint; thirdly we illustrate an approach for specifying inter-viewpoint consistency relationships. The paper ends with a discussion about the

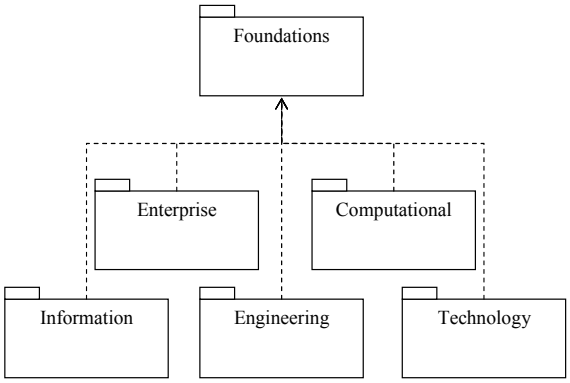

Figure 1 Viewpoint Metamodels

issues and problems to be addressed and a future direction for the work.

2. Viewpoint Metamodels

A number of papers have proposed metamodels for different RM-ODP viewpoints; [10] proposes an Enterprise Viewpoint metamodel; [9] and [2] discus Computational Viewpoint metamodels; and [7] describes a metamodel for the RM-ODP Foundation concepts.

A RM-ODP tool could be based on a set of five such metamodels, with the addition of (at least) a common 'Foundations' metamodel. Using an OMG MOF like language we can illustrate the approach as shown in Figures 1-3. Figure 1 shows an overview of six packages containing the metamodels for the five viewpoints and a metamodel for the common Foundations. Figure 2 and Figure 3 show example segments of possible Computational and Engineering Viewpoint metamodels.

The specification of viewpoint (or language) concepts in this manner makes it very easy to generate core parts of tools that support the viewpoints. Tools such as the Eclipse Modelling Framework (EMF) [5] and Kent Modelling Frameworks (KMF) [6] easily generate basic repositories that can form the core of a design tool from such models of a language.

3. Viewpoint Notations

In addition to repository functions a useful design tool requires a mechanism for populating the repository using

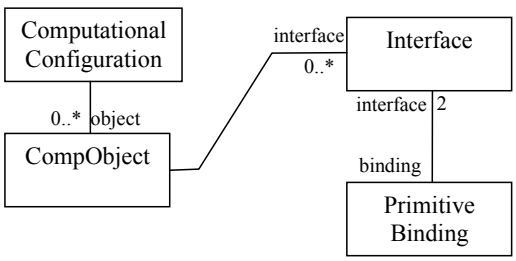

Figure 2 Computational Objects

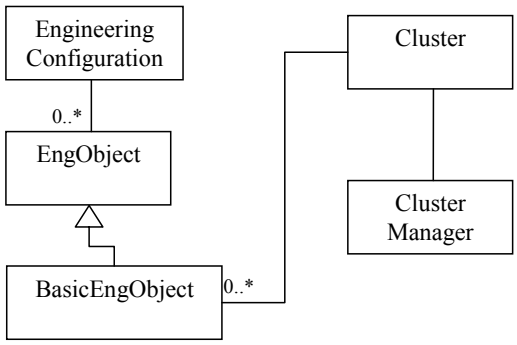

Figure 3 Engineering Objects

appropriate notations (or concrete syntax). The ODP ISO standards along with books such as [Blair/Stefani] make use of notations for describing various viewpoint designs. However although clearly understandable the notations are informally defined. If we can define these notations more formally, then there is scope for auto-generation of tools to support the viewpoint languages.

In [1] and [4] a modelling approach to defining visual languages is described. The approach is to define the concrete syntax of the notation as a model and subsequently specify a model transformation (via a set of relations) between the concrete syntax model and the concepts metamodel.

Using notation from the latest submission to the OMG's QVT RFP [8] and the technique defined in [3], an example for part of a Computational Viewpoint language specification is illustrated in Figure 4. It shows the specification of relations between Computational Objects and Circles, and between Interfaces and Solid rectangles (or bars); an example of the notation is shown in Figure 5.

The detail of the relations must be added to the graphical view of them, defining the domain and range of the relation and a matching condition expression that specifies which elements from one side (domain or range) of the relation are mapped to elements from the other. It is also necessary to define characteristics of the relation such as whether it is functional, total, bijective, etc. (The detail of the relations is shown in an Appendix.)

This approach can be used to define notations for all of the viewpoint concepts giving us a set of models and

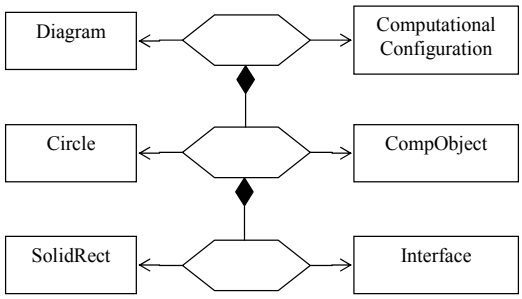

Figure 4 Syntax-CompVP Relations

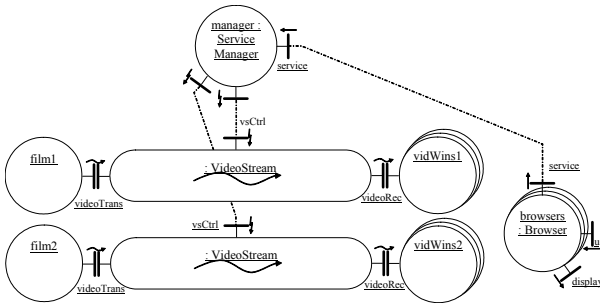

Figure 5 A Computational Configuration

relations from which it is possible to generate a tool that facilitates drawing designs in each of the viewpoint languages, including the necessary features of such a tool offered by a model repository.

The KMF tool developed at the University of Kent [6] has been used to generate parts of such tools for visual languages from this type of specification.

4. Inter-Viewpoint Consistency

A key part of the RM-ODP is the inter-viewpoint consistency specifications that tie the information from the five viewpoints into a consistent design from which an implementation can be produced.

Similarly to the specifications linking concrete syntax to metamodel concepts, we can define relations that link concepts between the different viewpoints. However, for some of the consistency relationships it is not possible to define them at the meta-level. It is necessary for the connections to be made at the design stage, for this we need another language (or at least a tool mechanism).

Figure 6 shows inter viewpoint consistency relations between concepts from the Computational Viewpoint and the Engineering viewpoint. This relations model the following correspondence:

"Each computational object which is not a binding object corresponds to a set of one or more basic engineering objects (and any channels which connect them). All the basic engineering objects in the set correspond only to that computational object."

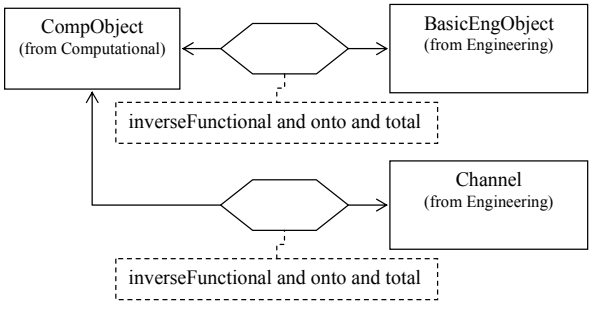

Figure 6 A Comp-Eng Correspondence

The relations are characterised as 'inverseFunctional' (or 1-to-many), each 'single' CompObject maps to 'many' BasicEngObjects (or Channels), but each BasicEngObject (or Channel) maps to a single CompObject. The relations are also defined as 'onto' and 'total' specifying that every CompObject and BasicEngObject (or Channel) in the domain and range of the relations must be part of the relation.

It is not feasible to define a matching condition for these relations (i.e. an expression that would define which objects from domain and range are mapped to each other) because there is no way at the meta-level to determine which objects should be related. The correspondences must be set up by the designer on a per design basis; however, the specification of these relations will enable a generated tool to indicate whether on not the correspondences have been set up.

5. Conclusion, Issues and Future Work

The previous sections have given an idea of how model based specifications of language concepts and relations between them can be given to define aspects from the RM-ODP standard. Using code generation techniques these types of specification can be used to generate tools that support designing a system from the different ODP viewpoints.

This paper has illustrated the ideas using languages provided by the OMG However, OMG languages are not essential, any precise approach to defining the languages and relationships would provide the necessary starting point for generating tools. In particular the following specifications should be given:

1. Precise definitions of the viewpoint language concepts – beyond the textual descriptions currently given – i.e. metamodels.
2. Precise specification of the correspondences between concepts in each viewpoint, along with suggested mechanisms for specifying these – i.e. a Correspondence Specification Viewpoint.
3. Precise specification of *example* notations for each viewpoint. Perhaps both graphical and textual.

In addition, a number of **full** example system designs should be provided illustrating intended use of the framework.

The approach to generating tools presented in the paper is an initial idea, a number of issues and problems are likely to make it difficult. Some of these are discussed below:

- Metamodelling – What concepts should be used to define the metamodels? Sections 6 and 7 of Part 2 of the standard informally define some language definition concepts, is it possible (or even a good idea!) to extend these to give a full and sufficient metamodelling language.

- Relations – Is the proposed technique for specifying relations expressive enough for the proposed task. Some correspondences are not at all easy to define! How useful or possible is it to define them all as relations?
- Correspondences - we need a mechanism for a designer to specify correspondences, i.e. we need a viewpoint for defining inter-viewpoint correspondences.
- Technology Viewpoint – the concepts for a metamodel of this is not obvious, in fact the concepts currently in the standard are minimal. Is it necessary to have something more and if so what should be in it?

Extending the idea of a supporting tool; it would be very useful to provide MDA like support for generation of system implementations (or simulation/analysis models) from the given designs. To facilitate this we certainly need good definitions of Technology models.

Generating an implementation could be achieved using an MDA like approach of transforming information drawn from designs given in the five viewpoints and providing a set of implementation source code, configuration, and deployment files.

6. References

[1] Akehurst D. H., "An OO Visual Language Definition Approach Supporting Multiple Views," in proceedings VL2000, IEEE Symposium on Visual Languages, September 2000.

[2] Akehurst D. H., Derrick J., and Waters A. G., "Addressing Computational Viewpoint Design," in proceedings Enterprise Distributed Object Computing Conference, EDOC 2003, Brisbane, Australia, pp. 147, September 2003.

[3] Akehurst D. H., Kent S., and Patrascoiu O., "A relational approach to defining and implementing transformations between metamodels," *Journal on Software and Systems Modeling*, vol. 2, pp. 215, November 2003.

[4] Clark A., Evans A., and Kent S., "Engineering modelling languages: A precise meta-modelling approach," in proceedings ETAPS 02 FASE Conference, LNCS, Springer, April 2002.

[5] IBM, "Eclipse Modeling Framework," http://www.eclipse.org/emf/, 2003.

[6] KMF-team, "Kent Modelling Framework (KMF)," 2002, www.cs.kent.ac.uk/projects/kmf

[7] Naumenko A., Wegmann A., Genilloud G., and Frank W. F., "Proposal for a formal foundation of RM-ODP concepts," in proceedings ICEIS 2001, Workshop On Open Distributed Processing - WOODPECKER`2001, Setúbal, Portugal, pp. 81-97, July 2001.

[8] OMG, "Request for Proposal: MOF 2.0 Query / Views / Transformations RFP," Object Management Group, ad/2002-04-10, April 2002.

[9] Romero R. and Vallecillo A., "Formalizing ODP Computational Viewpoint Specifications in Maude," in proceedings EDOC 2004, Monterey, California, September 2004.

[10] Steen M. and Derrick J., "ODP Enterprise Viewpoint Specification," *Computer Standards and Interfaces*, vol. 22, pp. 165-189, September 2000.

[11] X.901-5, "Information Technology - Open Distributed Processing - Reference Model: All Parts," ITU-T Recommendation, 1996-99.

Appendix A

The detail of a relation specification needs to define a matching condition that indicates which elements of domain and range should be related. In addition the links between relations should define the contents of the domain and range sets for the sub relations. The following specifications indicate the approach for the relations given in Figure 4. Depending on the level of difference there is between the related components, the complexity of the expressions in the relation specification will vary.

```
relation {
  domainType : Diagram
  rangeType : ComputationalConfiguration
  matchCond : true
  subRel : { CircleRelCompObj(
                  diagram.circles,
                  config.objects ) }
}

relation {
  domainType : Circle
  rangeType : CompObject
  matchCond : compObj.name = circle.label.text
  subRel : { RectangleRelInterface(
                  circle.connections,
                  object.interfaces ) }
}
```