INTEGRATION AUTHORITY

MOD Abbey Wood #2308 Bristol UK BS34 8JH

XMI, UML & MODAF

1.0

14 February 2005

IA/02/16-ERMcm03

Prepared by: ……..		Approved by: ……..	
Name:	Dr Ian Bailey	Name:	Mr A North
Post Title:	IA1dCon5	Post Title:	IA1d
Phone:	+44 (0)117 913 4045	Phone:	+44 (0)117 913 4237
Mobile:	+44 (0)7768 892 362	Fax:	+44 (0)117 913 4935
Email:	ian.bailey@cornwell.co.uk	Email:	IA1a@dpa.mod.uk

© HM Government Defence Integration Authority

RECORD OF CHANGES

This page will be updated and re-issued with each amendment. It provides an authorisation for the amendment and a checklist to the current amendment number.

Issue No.	Date	Revision Details
Draft 0.1	20 January 2005	First draft for internal review
Release 1.0	14 February 2005	Cosmetic changes, prior to release

Copyright Details

The copyright in this work is vested in HER BRITANNIC MAJESTY'S GOVERNMENT.

Nothing contained herein should be construed as endorsing any particular Technical Solution to any United Kingdom Government Invitation to Tender.

THIS DOCUMENT IS THE PROPERTY OF HER BRITANNIC MAJESTY'S GOVERNMENT, and is issued for the information of such persons only as need to know its contents in the course of their official duties. Any person finding this document should hand it in to a British Forces unit or to a police station for safe return to the Security Officer, Integration Authority, DPA, MoD, Bristol, with particulars of when and how found. THE UNAUTHORISED RETENTION OR DESTRUCTION OF THE DOCUMENT IS AN OFFENCE UNDER THE OFFICIAL SECRETS ACT 1911-1989. (When released to persons outside government service, this document is issued on a personal basis and the recipient to whom it is entrusted in confidence within the provisions of the Official Secrets Act 1911-1989, is personally responsible for its safe custody and for seeing that its contents are disclosed only to authorised persons).

Introduction

In order to facilitate interoperability between MODAF tools, a file format is required. Given the complexity and scope of the MODAF views, some sort of information model will be required to define the structure of a MODAF exchange file. The MOD has suggested that XMI (an OMG standard for model meta-data interchange) could be used as the file format. The purpose of this paper is to examine the use of XMI and other practical options for MODAF tool data interchange.

MOF & the UML Meta-Model

UML is the Unified Modelling Language™, and is an OMG standard. It is used to define software systems architectures – describing systems structure, behaviour, processes (human and computer), data structures and usage scenarios. UML is by far the most widely used modelling language for computer systems.

The UML meta-model is an information model which defines the various constructs used in UML. For each release of UML there is a different meta-model (1.5 is the current release of UML, though 2.0 is soon to be published). The purpose of the UML meta-model is to provide a structured underpinning for the language that can be used to define the structure of a repository for UML. The UML meta-model also defines the structure of XMI – the file format for UML tool interoperability.

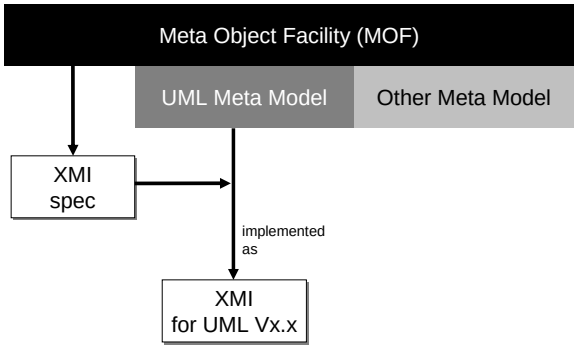
The UML meta-model is based upon another OMG standard – the Meta Object Facility (MOF). The MOF is also used as the basis for other modelling languages (such as IDL, the interface definition language). It defines very high level concepts such as classes, associations and properties. In other words, the MOF defines the basic building blocks that are used by modellers. The UML meta-model classes are instances of MOF elements.

In analysing the applicability of the UML meta-model to MODAF, it is worth considering the trends in military architectural framework usage. DODAF is the longest established and most used military architectural framework. DODAF volume II suggests some UML approaches for representing the various views and a number of DODAF users believe that UML (or at least specialist form of UML, such as SysML) is ideal for representing most of the DODAF views. Opinion is divided on the suitability of UML for all the DODAF views, but the benefits of using a standard notation for architectural diagrams is clear – cleaner tool interoperability and reduced ambiguity in the human interpretation of the diagrams.

The jury is out on whether UML will prevail as *the* notation for DODAF. The majority of DODAF architectures that have been produced have not used UML, or if they have, it has been applied specifically to the systems views. One reason for this is that versions of UML upto 1.5 have not been particularly rigid specifications, which opened up endless possibilities for ambiguity. However, version 2.0 of UML is a much more thorough specification, and the upcoming SysML standard introduces a systems engineering rigour to UML.

XMI – XML Metadata Interchange

XMI is another OMG standard. Contrary to popular belief, XMI is not a file format, it is a way of producing a file format for a modelling language. The XMI specification defines how a meta-model (which must be based on the MOF) can be translated into an XML specification. This means that there is not just one XMI file format. For each release of UML, there is usually a different meta-model (based on the MOF), and therefore the XMI specification for each version of UML is different.



To claim XMI conformance, therefore, one must first decide which meta-model is to be used in producing the XMI. Historically, the XMI specification has only been used to generate XMI for the different versions of UML. Standards which alter the UML meta-model (most profiles do not do this) therefore result in an XMI definition which may not be compatible with the core UML XMI.

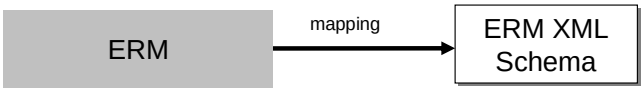
It is possible to use XMI without using UML, and two options are possible. The first is to create a meta-model (based on the MOF) for a specific purpose, then generate the XMI that is needed. The other is to use the UML XMI definition and define how the non-UML elements map onto the various XMI elements.

Practical Approaches to MODAF Tool Interoperability

The following approaches are not an exhaustive set of alternatives. Instead, they represent what are considered practical approaches for MODAF.

Option 1 - ERM Projection to XML

In this approach, the ERM information model is translated directly to an XML Schema. Such a translation is relatively simple provided the restrictions associated with XML Schema are considered at the time of developing the ERM.



The advantages of this approach are:

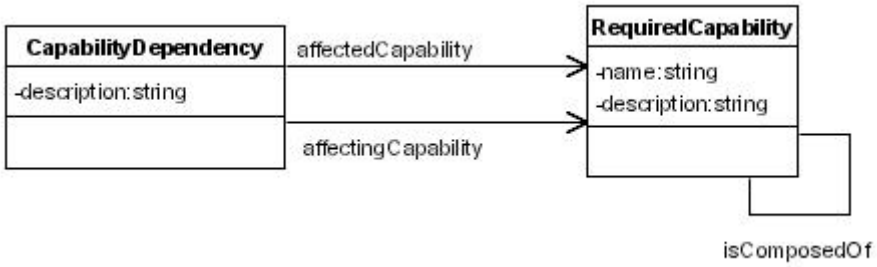
- The ERM has been designed specifically with MOD's enterprise architecture and MODAF in mind.
- The file format will define the data structure in a way which is obvious to the user – i.e. there will be XML elements called "system", "organization", etc.
- The XML Schema can be automatically generated from the ERM model – some work is needed to implement the generator, but after that, any changes to the model can be instantly reflected in the XML Schema.
- The ERM has been shown to the vendor community. Its existence is likely to cause confusion amongst the vendor community if it plays no part in the data exchange process

The disadvantages are:

- Vendors have to develop a new import/export interface to their tools
- Vendors have to become familiar with the ERM

- UML tool vendors (and some other tool vendors) already have XMI interfaces in place and would prefer to use the existing interfaces.

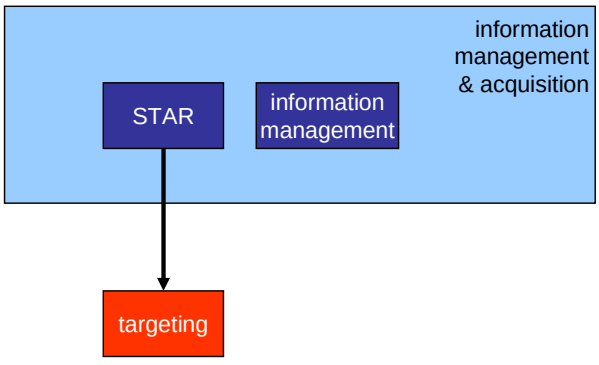
Using a very simple UML to XML mapping, the following ERM-like data model:



Would result in XML like:

```
<?xml version="1.0"?>
<MODAF>
   <RequiredCapability modaf-id="x1">
      <name>information management and acquisition</name>
      <description>the collection and analysis of information</description>
      <isComposedOf>x2 x3</isComposedOf>
   </RequiredCapability>
   <RequiredCapability modaf-id="x2">
      <name>STAR</name>
      <description>surveillance targeting analysis and reconnaissance</description>
   </RequiredCapability>
   <RequiredCapability modaf-id="x3">
      <name>information management</name>
      <description>collating and analyzing information</description>
   </RequiredCapability>
   <RequiredCapability modaf-id="x4">
      <name>targeting</name>
      <description>identifying, confirming and locating a target</description>
   </RequiredCapability>
   <CapabilityDependency modaf-id="x5">
      <description>targeting requires input from STAR</description>
      <affectedCapability>x5</affectedCapability>
      <affectingCapability>x2</affectingCapability>
   </CapabilityDependency>
</MODAF>
```

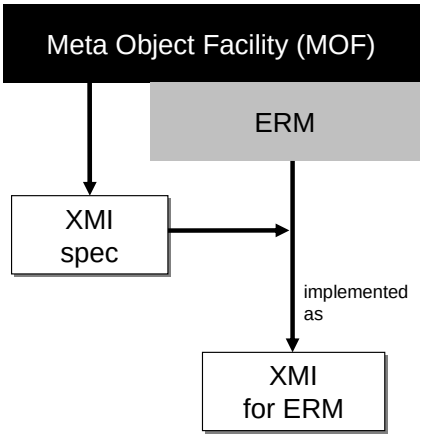
For a MODAF StV-4 view showing capability clusters and dependencies as follows:



Option 2 - XMI from ERM

This approach is very similar option 1, in that the ERM is used to generate an XML specification for data exchange. The first difference is that the MOF is introduced as the basis for the ERM – i.e. the various ERM data elements are instances of the appropriate modelling constructs defined in the MOF. The second difference is that instead of generating

an XML schema using an arbitrary mapping, the XMI specification is used to generate an XMI definition for the ERM.

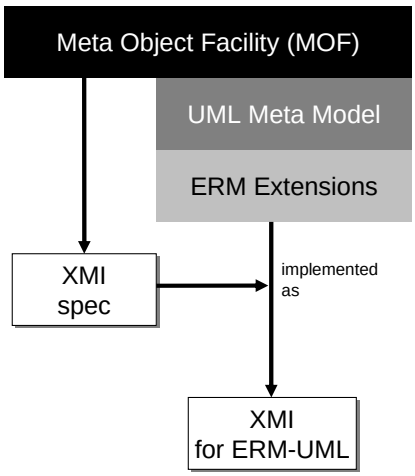


The advantages and disadvantages of this approach are the same as for option 1, with the added advantage that the tool vendors should be able to re-use at least some parts of their existing XMI interfaces. It could also be claimed that the use of the XMI mapping specification in generating the file format means that a standard approach has been used instead of the arbitrary mapping to XML used in the previous approach.

Without carrying out the MOF-ERM integration, it is difficult to predict accurately how the resulting XMI would look, so no example is possible.

Option 3 – XMI from Extending the UML Meta-Model

In this approach, the UML meta-model (probably of version 2.0) is extended to include the concepts defined in the ERM. Specialising the UML meta-model has the effect of defining which UML diagram elements are used to represent the various MODAF concepts. The XMI that would be generated would be specific to the hybrid model produced:



This approach has a small advantage over option 2, in that the XMI produced will be closer to the regular UML XMI. The vendors will, however, have to still extend their import export interfaces significantly. The main advantage is that the UML and ERM concepts are captured in one integrated structure.

The disadvantages of this approach are:

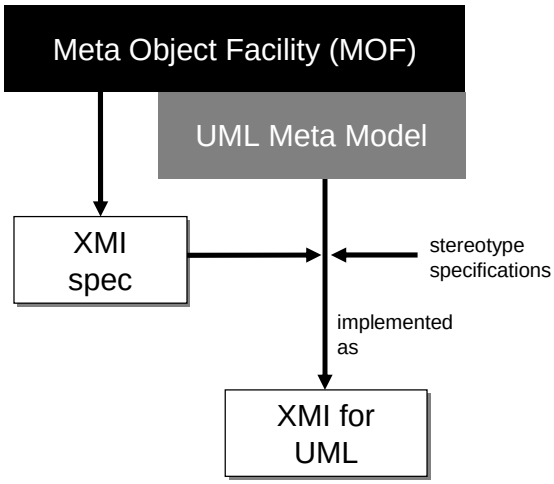
- The UML meta-model defines modelling constructs such as class, attribute, specialisation relationship, etc. The ERM defines more business oriented constructs. To specialise the UML constructs to ERM constructs is possible, but probably not ideal.
- The resulting XMI specification is likely to be convoluted due to some of the design work-arounds that will be necessary to specialise the UML meta-model into the ERM
- A great deal of work will be required to map the models, and will result in an XMI format which is still not compatible with existing XMI interfaces.

Without carrying out the ERM integration with the UML meta-model, it is difficult to predict accurately how the resulting XMI would look, so no example is possible.

Option 4 – UML Stereotypes

UML profiles (such as SysML) make extensive use of the UML stereotype concept. A stereotype is a way of using specialised UML constructs (classes, assemblies, activities, etc.) without having to alter the underlying meta-model. In other words, stereotypes are user-defined ways to make special use of the existing UML constructs – as opposed to something defined at the meta-model level.

As the use of stereotypes does not alter the meta-model, the resulting XMI specification is not altered from the regular UML XMI. However, the *way* the XMI is used does change. Standard XMI for UML includes a mechanism for representing stereotypes, as one would expect as any user can create their own stereotypes.



To use this approach would require that an appropriate stereotype is defined for most elements shown in all the MODAF views (activity model and logical data model views could probably be regular UML) . So, for example a <<capability>> stereotype could be created for class. When shown in XMI, this is reflected as follows:

```
<UML:Class xmi.id = 'sm$1eed0fb:10152804774:-7ff5' name = 'MyCapability'
  visibility = 'public' isSpecification = 'false' isRoot = 'false' isLeaf = 'false'
  isAbstract = 'false' isActive = 'false'>
  <UML:ModelElement.stereotype>
    <UML:Stereotype xmi.idref = 'sm$1eed0fb:10152804774:-7ff4'/>
  </UML:ModelElement.stereotype>
</UML:Class>
<UML:Class xmi.id = 'sm$1eed0fb:10152804774:-7fd4' name = 'MyOtherCapability'
  visibility = 'public' isSpecification = 'false' isRoot = 'false' isLeaf = 'false'
  isAbstract = 'false' isActive = 'false'>
  <UML:ModelElement.stereotype>
    <UML:Stereotype xmi.idref = 'sm$1eed0fb:10152804774:-7ff4'/>
  </UML:ModelElement.stereotype>
```

```
</UML:Class>
<UML:Stereotype xmi.id = 'sm$1eed0fb:10152804774:-7ff4' name = 'capability'
  visibility = 'public' isSpecification = 'false' isRoot = 'false' isLeaf = 'false'
  isAbstract = 'false'>
  <UML:Stereotype.baseClass>Class</UML:Stereotype.baseClass>
</UML:Stereotype>
```

The classes “MyCapability” and “MyOtherCapability” refer to a stereotype definition. The UML diagram for this would be:

The tool that exported the XMI need not be a UML tool, it need only be able to export XMI. It is clear, however, that this approach is best suited to UML tools.

Option 5 – Hybrid Approach

In this approach, the ERM takes care of the architectural framework semantics whilst the UML meta-model is used to represent how that semantic data is to be represented as UML classes. Effectively, this combines either option 1 or 2 with option 4. To achieve this hybrid approach, the ERM and UML meta models could be combined, with appropriate relationships between the architectural element definitions in the ERM and the UML constructs that are used to represent them in the UML meta-model – e.g.

```
<MODAF:Capability modaf.id = 'x1'>
   <name>ISTAR<name>
   <description>surveillance targeting analysis and reconnaissance</description>
   <representation xmi.idref = 'sm$1eed0fb:10152804774:-7ff5'/>
</MODAF:Capability>
<UML:Class xmi.id = 'sm$1eed0fb:10152804774:-7ff5' name = 'MyCapability' visibility = 'public'
    isSpecification = 'false' isRoot = 'false' isLeaf = 'false' isAbstract = 'false'
    isActive = 'false'>
</UML:Class>
```

Alternatively, the XMI information could be embedded inside the XML tags instead of referred to – e.g.

```
<MODAF:Capability modaf.id = 'x1'>
   <name>ISTAR<name>
   <description>surveillance targeting analysis and reconnaissance</description>
   <representation>
       <UML:Class xmi.id = 'sm$1eed0fb:10152804774:-7ff5' name = 'MyCapability'
          visibility = 'public' isSpecification = 'false' isRoot = 'false' isLeaf = 'false'
          isAbstract = 'false' isActive = 'false'>
       </UML:Class>
   </representation>
</MODAF:Capability>
```

Either way would work, and either approach allows the XMI aspect to be optional – i.e. tools could opt to simply export the architectural information without any reference to the XMI representation. With the first approach, the ERM XML and XMI to be in separate files, allowing a complete XMI specification – complete with stereotypes.

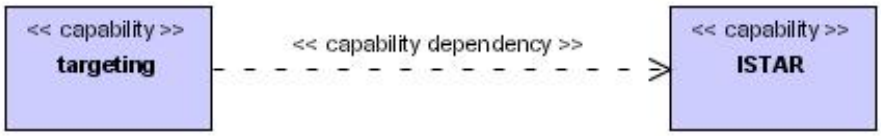
Way Forward

In order to fulfil the MOD’s stated goal of using XMI as the exchange format for MODAF tools, option 4 is the only realistic solution. This provides MODAF with a means to exchange data using “vanilla” XMI. This means that tool developers who already support XMI do not need to modify their import / export interfaces. For those tool vendors who do not offer XMI

support, there is the added advantage that the XMI interface they develop for MODAF can be re-used when communicating with other UML tools for non-MODAF applications.

For XMI to be a useful exchange standard for MODAF, there is a requirement to constrain how the XMI will be used, and types of information that can appear in an XMI file. To do this, it is necessary to develop a formal UML profile[1] for MODAF, specifying a set of UML stereotypes[2] which match the MODAF elements. This does not mean that UML need be used to display the MODAF views, simply that a profile is the most formal way to specify the allowable stereotypes. A profile also constrains the potential content of the XMI file in terms of specifying which stereotypes can be related – e.g. a "capability dependency" element can only relate two capabilities.

The example StV-4 diagram below shows two capabilities, with a dependency relationship between them:



In this case, UML has been used to show the capabilities, but the presentation format is not restricted only to UML. Note that "targeting" and "ISTAR" are both instances of capabilities, and capability is a stereotyped construct – the same goes for the relationship between them. The XMI for this MODAF data would look something like:

```
<UML:Class xmi.id="cap1" name="targeting">
  <UML:ModelElement.stereotype>
    <UML:Stereotype xmi.idref="st1"/>
  </UML:ModelElement.stereotype>
</UML:Class>
<UML:Class xmi.id="cap2" name="ISTAR">
  <UML:ModelElement.stereotype>
    <UML:Stereotype xmi.idref="st1"/>
  </UML:ModelElement.stereotype>
</UML:Class>
<UML:Dependency xmi.id="dep1" isSpecification="false">
  <UML:ModelElement.stereotype>
    <UML:Stereotype xmi.idref="st2"/>
  </UML:ModelElement.stereotype>
  …
</UML:Dependency>
<UML:Stereotype xmi.id="st1" name="capability">
  <UML:Stereotype.baseClass>Class</UML:Stereotype.baseClass>
</UML:Stereotype>
<UML:Stereotype xmi.id="st2" name="capability dependency">
  <UML:Stereotype.baseClass>Dependency</UML:Stereotype.baseClass>
</UML:Stereotype>
```

[1] A UML profile specifies a way of using UML for a specialist purpose (e.g. enterprise architecture, systems engineering, etc.). The profile consists of constraints about how the UML modelling elements can be used, and a set of stereotypes[2] suitable for the purpose of the profile.

[2] UML stereotypes are a way of extending existing UML modelling elements for a specific purpose – for example, one may wish to create a model element called "system" for systems engineering purposes which extends the base "class" concept in UML. UML stereotypes are "run time" in the sense that an out-of-the-box UML tool can support them – this also means that XMI data exchange of models which use stereotypes can be achieved without modifying the XMI structure.

Note that the stereotype definition is exchanged with the instances of the capabilities. This is the way that stereotyped information can be exchanged without changing the XMI specification – i.e. the exchange file is self-defining and self-contained.

Developing a UML profile is not a small undertaking. It requires that each architectural element used in MODAF, and any kind of relationship between elements, are profiled. There is a standard modelling notation for developing UML profiles which involves extending elements in the UML meta-model:

The other potential approach is to take an existing profile which is a close fit (e.g. SysML in this case) and specialise from the UML extensions it has already established:

Note the different notation in the two diagrams – the filled arrow in UML indicates that capability is an extension of the class concept. In the second diagram, the SysML assembly is already an extension of a UML meta model element, so capability is a *specialization* of assembly. This is a narrow distinction, but worth bearing in mind when examining a UML profile, because if specialization (the hollow arrow head) were used from the UML meta model, it would result in the XMI structure and tags being changed. As SysML has already extended (solid arrow head) the UML meta model, it is OK to use specialization from SysML meta-model elements.

In developing a UML profile, it also necessary to identify the relationships between elements and extend / specialise them. The notation is the same as the previous example:

This includes the obvious cases where lines connect elements in a MODAF view, and the less obvious cases where elements are related, such as when one element is contained within another. The MODAF StV-4 view, for example, is made up of capability elements embedded within each other which infers a parent-child relationship, which must also be specified in the meta-model extensions.

Using the UML meta-model as a basis for the data exchange model places some restrictions on the modelling style that can be used in developing the profile. It is possible to use the ERM to inform and guide the development of the profile, but not to use it in the profile. The ERM will act as a conceptual model, specifying the information requirements at a business level – the UML profile will be a more convoluted model, which is not suitable for the majority of MODAF users. However, it is important that the UML profile classes are traced back to the classes defined in the ERM.

Conclusions

It is possible to use XMI for MODAF tool exchange. However, in order to use standard XMI for UML (which is already supported by many tools), it is necessary to control how the XMI is used. It is also important that the MODAF concepts (defined in the ERM) are not lost in the exchange. The use of stereotypes (as part of a UML profile) offers an opportunity to use a standard XMI format, but also ensure that the semantics of the architecture are exchanged.

By the time MODAF is complete, the UML 2.0 spec and the SysML spec are scheduled for completion. Hence it makes sense to base the profile on these standards. In addition, the XMI 2.0 specification is due for release in a similar timescale, and offers significant improvements over XMI 1.x.

Acknowledgements

Thanks go to the reviewers:

- Andy North, IA
- Fariba Hozhabrafkan, Cornwell Consulting
- Feedback from potential MODAF tool/repository vendors