INTEGRATION AUTHORITY

MOD Abbey Wood #2308 Bristol UK BS34 8JH

Implementing the MODAF Enterprise Reference Model

1.0

14 February 2005

IA/02/16-ERMcm02

Prepared by: ……..

Name:	Ian Bailey
Post Title:	IA1dCon5
Phone:	+44 (0)117 913 4045
Mobile:	+44 (0)7768 892 362
Email:	ian.bailey@cornwell.co.uk

Approved by: ……..

Name:	Mr A North
Post Title:	IA1d
Phone:	+44 (0)117 913 4237
Fax:	+44 (0)117 913 4935
Email:	IA1a@dpa.mod.uk

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	14 December 2004	First draft for internal review
Draft 0.2	18 January 2005	Modifications in line with XMI strategy
Draft 0.3	01 February 2005	Minor edits
Release 1.0	14 February 2005	Cosmetic changes, prior to release

Copyright Details

Summary

This paper examines the different implementation approaches that can support data exchange and sharing based on the MOD's Enterprise Reference Model (ERM). The paper also outlines the technologies, XMI & Web Services, that are to be used.

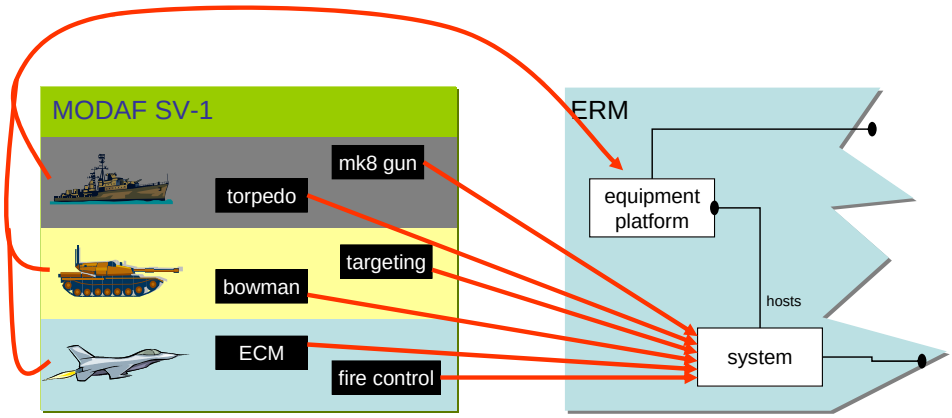
It is proposed to use a two-level modelling approach – conceptual and logical layers – with the ERM taking the role of conceptual model and driving the logical (implementation) models for data exchange and web services. The physical layer is taken care of using a model-driven approach – the XMI and Web Services standards are used to implement the logical models directly.

For file-based data exchange, it is proposed to use XMI – the UML model interchange specification based on XML. The choice of XML is rather obvious, and XMI is also a logical choice given that the goal is to exchange architectural models. To add the necessary rigour for MODAF, the XMI interchange will make use of UML Stereotypes. The stereotypes will be formally specified in the "MODAF meta-model" which will extend the UML and SysML meta-models.

When it comes to data sharing services for the MARS (MOD Architectural Repository System), the decision on implementation approach is less obvious. Web Services and CORBA are both suitable implementation technologies and both can be specified using a model-driven approach. Web Services are the easier technology to implement allow re-use of the XML Schema specified for data exchange, so would seem to the obvious candidate.

Introduction

The MOD Architectural Framework (MODAF) is underpinned by a conceptual information model – the Enterprise Reference Model (ERM). The purpose of this model is to define the nature of the information that makes up an architecture. The model does not seek to capture the layout and graphics, but the meaning behind the architectural elements – i.e. the model deals with systems, organizations, networks, etc. rather than boxes, lines and bitmaps. The ERM is not intended to be implemented directly – i.e. it is a conceptual data model. The scope of the ERM may be greater than that of MODAF, but the entire scope of MODAF is covered by the ERM. The diagram below shows how the elements displayed in a (somewhat crude) MODAF SV1 view map onto elements defined in the ERM:



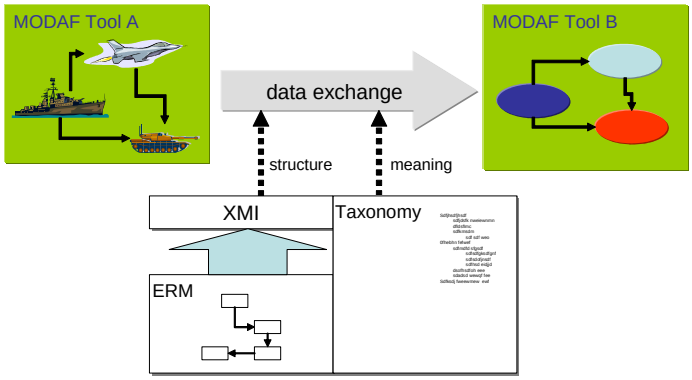
The ERM serves several purposes:

- To provide a semantic underpinning for MODAF.
- To provide the basis for the MODAF meta-model which will be used to specify the data exchange between tools (XMI)
- To drive the requirements for an MOD enterprise architecture repository (MARS).
- To drive the specification of the services exposed by MARS

The ERM has had a number of influences in its development. As well as existing MOD architectural models, the ERM modellers have also considered the DODAF CADM model and in-development standards such as the SysML meta-model (OMG) and AP233 (ISO). The CADM is a very rich and detailed model, with some 600+ entity definitions. Based on vendor feedback on CADM, it was decided that something more manageable was needed for MODAF. Hence, the ERM is meant to be a higher level model than CADM, and relies on the MODAF taxonomy to add richness and detail.

In addition, the MOD has, in consultation with tool vendors, opted to specify the OMG's XMI specification[1] for data exchange between MODAF tools. Any MODAF tool data exchange will use a combination of XMI (with an appropriate UML profile) to define the file structure and the MODAF taxonomy to provide reference data:

[1] See IA document on UML & XMI - IA/02/16-ERMcm03



The ERM is still subject to change, as the MODAF development team (and pilot projects) discover functionality that is not currently covered. The ERM is to be managed using a change process (based on the W3C change process). The traceability from the ERM to the MODAF meta-model and the repository services specification will be maintained in a mapping document that tracks changes in any of the models.

Data Exchange

The key requirement for the ERM is that it must cover all the concepts needed to describe an enterprise architecture. The requirements for a data exchange model are somewhat different. The ERM does not have to concern itself with the everyday problems of tool-to-tool exchange and can act as an undiluted specification of what the business requires. The model that is used to define the data exchange has to take into account several implementation issues, the key one being the requirement to use XMI as the format for data exchange.

XMI is a flexible interchange format that is dependent on the modelling language used – though it's usually used to exchange UML models. For UML XMI (what most people mean when they say "XMI"), the XMI structure is defined by the UML meta-model. In other words, if the UML meta-model is altered, the XMI output changes and will not be readable by standard tools. This places a key restriction on how the model for data exchange is specified. In order to achieve maximum re-use of existing XMI interfaces, the UML meta-model cannot be changed, it can only be extended. The very nature of extending a model places a number of restrictions on how the modeller can work.

The choice of XMI was not arrived at easily. The obvious candidate for a file structure based on a data model is XML Schema. It is relatively easy to derive an XML Schema from an information model – provided that the model takes into account some of the limitations of XML Schema. This would probably be the easiest option for the MODAF team, as it is easier to specify than a MODAF model based on the UML meta-model. However, the tool vendors (most of whom support XMI already) would have to develop XML interfaces from scratch.

XMI is an XML format though, and most of the advantages of the pure XML approach also apply to the XMI approach. There are a number of XML tools in existence, most are very robust implementations, many of which are freely available or even open-source[2]. The other advantage of XML is that most vendors already support it in some form or other and their development teams are familiar with it. The downside to XML is that files tend to be large as the tagging syntax can be somewhat inefficient – the files do compress well using commonly available algorithms such as those used in zip files.

[2] Open source software is freely available and is provided with the source code. This type of software is usually subject to an open licenses (such as the Gnu Public License) which restricts the re-use of the software without appropriate recognition of the authors.

XML was not the only option considered though – other file formats exist that can be used to implement data models. The simplest and most obvious candidate is comma separated variable (CSV). This format has been used to dump relational database tables and spreadsheets for years. Where CSV falls down is its inability to enforce the rules of a complex data model – relationships can only be represented by key values, and there is no computer interpretable schema to define the columns of data.

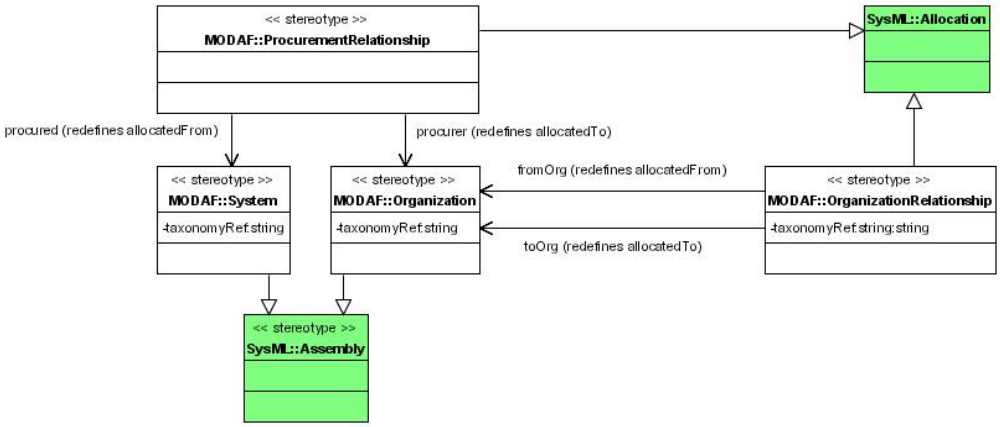
Another solution is to use the specifications developed for ISO10303, the standard for the exchange of product model data. This standard has its own modelling language called EXPRESS (part 11 of the standard) and a file format for exchanging data which conforms to the EXPRESS models (part 21). As the modelling language and the file format were developed together, there is close correspondence which allows detailed file content checking. The standard has been used successfully for years, but industry is beginning to demand XML Schema implementations. As a result, a new part of the standard (part 28) has been developed to provide an XML implementation of the EXPRESS models.

Many other file formats exist, though most text encodings are being superseded by XML. Binary formats, such as ASN1, should be considered if there is a large amount of binary data (e.g. bitmaps, videos, etc.) to be taken care of. This is not the case for the ERM, so XML, despite some of the limitations of XML Schema, seems to offer the best solution for architecture interchange.

The MODAF Meta-Model

To use “vanilla” UML XMI for data exchange requires that UML stereotypes are specified for each of the items that are used in the architectural framework – e.g. systems, capabilities, acquisition clusters, operational nodes, etc. If the information in the exchange file is to be sensible, it also necessary to define stereotypes for all the possible relationships between those items – e.g. procurer-procured, capability parent-child, etc. These stereotypes, when put together, form a UML Profile.

There is a formal way to specify a UML Profile, by extending the UML meta-model. For MODAF it is proposed that this extended model be based on the SysML[3] specification, and be called the MODAF meta-model. The development of this meta-model will be driven by the content of the MODAF views (this is the priority for data exchange) and by the concepts specified in the ERM. The MODAF meta-model is not yet specified, but some early attempts models have been developed for the new MODAF views, such as AcV-1:



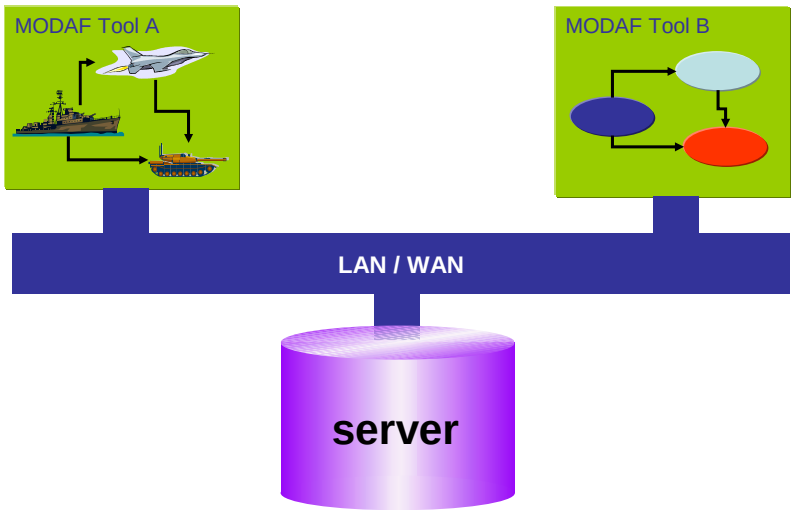
Note: This example is not finalised and is subject to change

[3] See www.sysml.org

Although this is a specification of a UML profile, for all intents and purposes, it is an implementation (logical) data model. As the model extends an existing meta-model, the elements defined tend to be counter-intuitive to the uninitiated. However, the elements defined in the MODAF meta-model can be traced back onto concepts defined in the ERM, enabling better understanding of how the model is used.

Data Sharing

The Enterprise Reference Model will also act as the basis for a set of data sharing services, defining the structure of the information that is shared. A number of approaches exist for data sharing, most of them being based on remote service invocation – i.e. a client makes a request to a server and information is returned:



The information is requested and served using a suitable protocol, running over a local or wide area network. The key design decision is which protocol to use. There are several approaches that suit this kind of architecture, but four main ones are considered here – web services, CORBA, Microsoft COM, and RMI. For MARS, possible services could be:

- Get all systems of a certain type (referring to a taxonomy entry)
- Get all systems that can be connected to a given system
- Get all capabilities that are applicable to a given Epoch
- Add an operational node
- Connect two systems
- etc.

Web services technology provides a mechanism for business-to-business communication, or more accurately machine-to-machine communication. They use an XML protocol for submitting requests and serving information. A web services server publishes the services it is able to offer, defined in WSDL (web services definition language). The services can be anything, but more often than not, they involve some form of data access. The data that is sent and received conforms to an XML schema. As with the data exchange scenario, where the ERM acted as the logical model to drive the physical exchange schema, the web services definitions can be derived from the ERM concepts.

Web services have become extremely popular because of the simplicity of implementation. A server broadcasts what services it has available, and clients can make use of those services provided they have access to the server. Additional levels of security can be added, and the security issues are roughly the same as for an http web server. From a developer's point of view, web services are very easy to work with because they integrate well with most

existing development environments and are hardware and OS neutral – i.e. UNIX, Mac, PC and Linux machines can all communicate with the same web service without need for extra work. The disadvantage with web services is that the protocol used is quite bulky (lots of XML tags) so can be slow when working with large amounts of data. This is not usually apparent on fast local area networks, but can cause performance problems on the internet. Another potential problem with web services is that two “flavours” of implementation have emerged – the J2EE approach favoured by the Java community and the .NET™ approach favoured by Microsoft. Careful definition of the web services can help to mitigate these problems, however.

CORBA stands for “Common Object Request Broker Architecture” and is an OMG standard. CORBA is a heavy duty services architecture best suited to local area network implementation. It is fully object oriented, and closely follows the class structure of UML, which makes CORBA a strong candidate for model driven architectures. It uses a standard protocol (IIOP) for submissions and responses (web services use XML and http). CORBA implementations tend to be robust and large-scale. UML class models can be converted to IDL then implemented as CORBA, often using sets of standard services the OMG has defined. Implementing a full set of CORBA standard services is quite a daunting task, however. The downside to CORBA is that performance can be very slow when the data being shared is composed of many small objects – CORBA works better if “business objects” are defined at a high level, aggregating a lot of data into one place. For the ERM, this would inevitably mean developing a set of new classes with methods to support them. CORBA is used by several large organizations for mission critical data – particularly where the data is complex. For smaller-scale implementations, web services would seem to be flavour of the month, rather than CORBA.

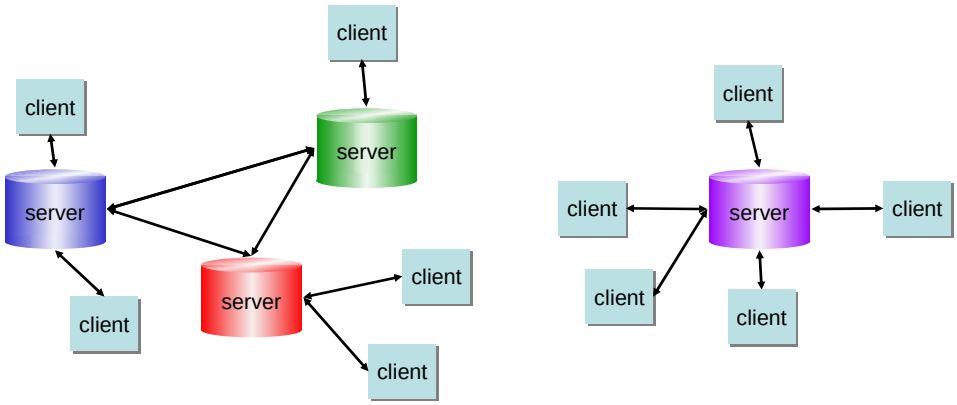
COM stands for “component object model” and is the basis for most Microsoft Windows™ applications. There is a distributed COM technology (DCOM) which provides similar functionality to CORBA. Whereas CORBA and Web Services are platform independent, DCOM is firmly in the Microsoft domain (though there have been attempts to develop COM bridges to other technologies). Microsoft have made COM very easy to implement and use, and most programmers have had to work with COM libraries at some point in their career. Performance is good on local area networks, and DCOM comes with less “baggage” than CORBA. DCOM tends not to be used over the internet though, for performance and security reasons – however, its sibling, *ActiveX*, is widely used on the web.

RMI standards for Remote Method Invocation and is a Java technology allowing objects on distributed systems to invoke methods remotely – effectively exposing the underlying Java services to other Java objects. RMI is widely used, but is a Java-only technology (again, some attempts have been made at implementing bridges to other technologies). RMI tends to be used on small-scale implementations, on local area networks. Performance is very good, and implementation is relatively easy for most experienced programmers.

The target implementation is MARS – the MOD Architectural Repository System. MARS acts as an integrated, shared data environment for enterprise architecture information. The scope of data is probably greater for MARS than for MODAF tools – e.g. MARS will have to manage multiple versions of data elements, ownership, and time stamping issues. For most enterprise architecture usage, the data traffic will be relatively small, and will mostly be text-based or numeric data (i.e. little or no binary data). The only time that large amounts of data will be transferred will be when a complete architecture is requested, which would probably best served as an exchange file anyway.

The requirement for platform independence is not altogether clear, but it is apparent that the various enterprise architecture tools are implemented on different platforms. Given its ease of implementation, Web Services seems the most obvious candidate for MARS.

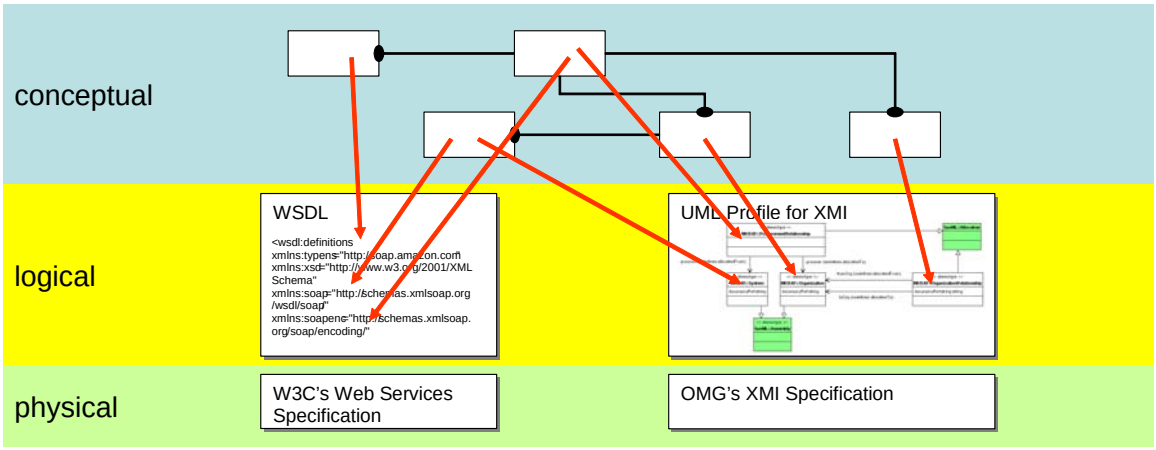
As a final point, it is worth considering the architecture of MARS – initial proposals have concentrated on the idea of one centralized repository and multiple clients. However, a services-based architecture would allow for multiple, federated repositories which together act as one virtual repository:



In the example on the left, a given client uses the services on its local server. However, the data it is manipulating may be on another peer server located elsewhere. The collection of servers provides a connected, federated data source. In the example on the right, all data is stored at the central server. Note that if the repository services are defined to allow it, both approaches are possible.

Layered Model Approach & MDA

It is common practice in software engineering to define data structures at three levels[4]. The conceptual model is used to capture the business concepts and the relationships between them. A conceptual model usually has no attributes, or only those attributes considered key to the business. The logical model is a referentially correct model which considers matters of cardinality, existence dependence, etc. to define the structure of the information. The physical model is a computer-interpretable specification based on the logical model which takes into account implementation aspects such as file format, relational database tables, etc. Traditionally, the physical model has been developed from the logical model by hand, but the drive towards standards-based implementation means that physical models are increasingly generated automatically from the logical model (model driven architecture).



[4] See http://www.1keydata.com/datawarehousing/data-modeling-levels.html

The Object Management Group (OMG) has trademarked the term “model driven architecture” and MDA™ when applied to software systems. Although the term is trademarked, the principles behind it are older than the OMG itself – data exchange initiatives such as ISO STEP and PLIB have used a model driven approach for years. The basic idea of MDA™ is that software implementation is automatically derived from a logical model (in this case, a UML model). In most MDA™ implementations, this means a UML class model being used as the basis for database structure, an exchange file / message protocol, and an API. So, from one model it is possible to derive all the data and interface aspects of an implementation. More extreme applications of MDA™ take the behavioural aspects of UML and derive functional software from this.

The ERM is a clear example of a conceptual model – it is a statement of the information requirements for enterprise architecture in the MOD. To implement the ERM, an intermediate (logical) model will be required, and a method for generating the physical schema. For data exchange, the physical schema definition is already in place; XMI. The purpose of the logical model in this case is to constrain how the XMI is used. For web services, it may be possible to use the same logical model, though the UML meta-model aspects may make it unsuitable for service definitions.

Conclusions

The ERM is not a suitable model for direct implementation. In addition, there is a stated requirement for MODAF to use XMI for tool interchange. In order for this to be done properly, a UML profile for MODAF will be required in order to constrain the XMI content. The profile is to be specified by extending the UML meta-model – effectively developing a MODAF meta-model. In the tradition of three-layer model development, the ERM is to act as a conceptual model, guiding the specification of the (logical) MODAF meta-model. It will also drive the development of web services specifications for MARS.

The choice of XMI creates some issues for the modellers working on the MODAF meta-model. It is not a trivial task to extend the UML meta-model. However, the ERM provides guidance for what information is to be handled, and the MODAF view specifications provide a clear scope for that information.

When it comes to services implementation, Web Services probably have the edge for the MARS repository, though CORBA is equally well suited. In the end, the recommendation has to be Web Services because of their sheer simplicity of use and popularity with implementers.

Acknowledgements

Thanks go to the reviewers:

- Andy North, IA
- Fariba Hozhabrafkan, Cornwell Consulting