White Paper:

A Logical Architecture for an Open Development Framework for Component-based Management Systems

Editor : David Lewis (UCL)
Contributors: Vincent Wade, Ingi Thorarensen (LMD)
Reviewers: Colin Harris (TCD), Eric Leray (WIT)
Release: 1.0
Date : 3/10/01
Distribution : Public
Document Number : IST-1999-10357/UCL/WP6/0921-v1

ABSTRACT

The communications management system development industry has long exploited the benefits of open interoperability between systems sourced from different vendors. There is an increasing software engineering trend towards developing software systems that are made up of different components sourced from multiple vendors. This presents the communications management community with the challenge of integrating existing techniques for system interoperability with emerging techniques for component integration. This needs to be done in a way that encourages an open market in off the shelf components applicable to the communications management domain. This paper presents a logical architecture that aims to support an open market in interoperable and integratable communications management components. The architecture combines ideas from the management, distributed systems and software engineering communities in forming a model-driven approach to the selection and reuse of software components. The logical architecture is part of the Open Development Framework developed within the FORM project, which is also described in this paper.

KEYWORDS
software architecture, model structure, open development framework

Table of Contents

1 Introduction

The communications management system development industry has long exploited the benefits of open interoperability between systems sourced from different vendors. This is characterised by information model based standards for Internet Management, Telecommunication Management Network (TMN) and more recently the work of the Distributed Management Taskforce (DMTF). In addition the communication management industry has embraced the interoperability mechanisms of distributed processing technologies, such as CORBA, COM and RMI and WWW based technologies such as HTTP and XML. Increasingly, however, there is an emphasis placed on using standards not only for interoperability between systems but also for the integration of software into systems. This is visible in the trend to develop software systems that are made up of different components sourced from multiple vendors. Such trends present the communications management community with the challenge of integrating existing techniques for system interoperability with emerging techniques for component integration. This needs to be done in a way that encourages an open market in off the shelf components applicable to the communications management domain.

This paper presents a Logical Architecture that aims to support an open market in interoperable and integrateable communications management components. The Architecture combines ideas from the management, distributed systems and software engineering communities in forming a model-driven approach to the selection and reuse of software components. This Architecture is part of an Open Development Framework (ODF) for the emerging market in component-oriented communications management software. The target audience of this Framework is wide, covering service providers, management system integrators, independent software vendors and standards bodies. The ODF aims to allow these stakeholders to comfortably move from their existing software and system development techniques to ones that support semi-formal modelling of systems in converging and inter-transformable formats. The resulting ability to readily exchange models related to software interoperability and integration in a non-proprietary form is regarded as key to enabling an open market in off the shelf components.

This Framework is being developed in the EU-funded project FORM. Within this project, elements of the Framework are evaluated through the development of components and systems for the management of business-to-business services over QoS enabled IP networks, termed Inter-Enterprise Services (IES).

This white paper is structured as follows: the next section provides an overview of the Open Development Framework, the stakeholders it addresses, the challenges involved and the benefits it will provide. Section 3 describes how the Logical Architecture portion of the Framework relates to other existing and on-going work. Section 4 describes the Logical Architecture of the Framework in terms of the abstract roles that will use the Framework, the over-arching architectural principles of the Framework and the architectural model. The relationship of the Logical Architecture to the Development Methodology is given in section 5. This white paper provides a snapshot of the Framework's Architecture midway through the FORM project so section 6 discusses the further work required in refining the Framework before conclusions are drawn in section 7.

2 The Open Development Framework

The Open Development Framework can be characterised as addressing the needs of the system development value chain that exists in the communication management software industry. The system development chain must address the challenges of integrating separately-sourced software to satisfy rapidly changing management system requirements. The software industry is moving towards the (re)use of component-oriented off-the-shelf software and model-driven approaches to the software lifecycle. Applying this to the market for communication management software requires new architectural and modelling principles to be shared between Standards Bodies, Independent Software Vendors (ISVs), System Integrators and System Customers (i.e. the Service Providers). It is assumed that Service Providers operate in service delivery value chains and therefore have interoperability requirements between the systems of different Providers.
Figure 2-1 depicts the major relationships between the stakeholders addressed by the Framework. ISVs provide software components to System Integrators who in turn integrate components from several sources into systems that are developed for Service Providers. These systems have to integrate with the Service Provider's existing systems as well as potentially with those of other service providers in a service delivery chain. Standards bodies play a role in providing industry agreements on interoperable interface specifications between systems and components and on component integration mechanisms.

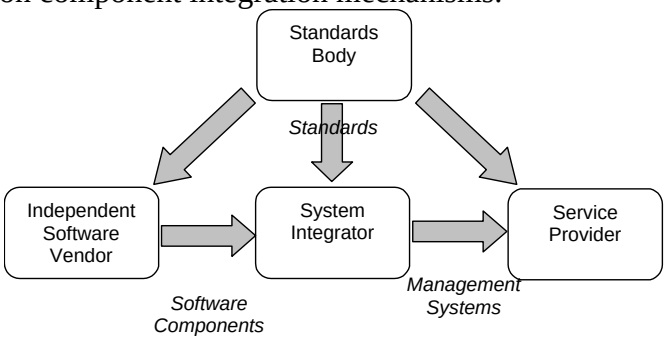

Figure 2-1: Stakeholder Model for FORM Open Development Framework

The integration of separately-sourced software has often relied solely on the expression of well-defined interfaces. However, such interface definitions often omit important contextual assumptions or are optimised for implementation in a specific technology. This makes maintaining interoperability between separately-sourced components increasingly expensive as system requirements, the technology base and component capabilities evolve over time. At the same time, the communications management industry has moved from using just management specific protocols (e.g. SNMP, CMIP, TL1) to encompassing more general distributed system platforms (e.g. CORBA, DCOM) and web technologies (e.g. HTTP, XML). As a result management system development must address the need to interwork between different technologies and to integrate and maintain the models in multiple formats, e.g. SMI, GDMO, IDL etc. The approach taken needs to exploit emerging technologies for integrating separately-sourced software (e.g. EJB, CORBA Components, COM+) and mechanisms for transforming between models (e.g. XML, XSLT [xslt]).
The Open Development Framework is therefore intended to provide common guidance to these industry stakeholders in developing and applying software components and systems for the communications management sector and addressing these challenges. The Framework takes a model-based approach to software development and places an emphasis on developing

models that can be passed usefully between the Framework's users. The use of model-based development aims to allow the exchange of models in a commonly understandable way supporting the publication of models and thus supporting the development of open models that are the result of industry agreement. This is seen as essential to supporting a market in off-the-shelf software components, and the structure of the models is designed to allow exchange of models at points in the software lifecycle most likely to encourage such a market. As depicted in Figure 2-2 the Framework is structured into four portions: a Logical Architecture, a Technology Architecture, a Development Methodology and a Set of Reusable Elements. The Framework is intended to be generic and extensible. The ODF has a core generic part, which in FORM is extended with others the concerns related to the IES Management domain. This generic part consists of the Logical Architecture and core principles of the However it is expected that other users of the Framework will extend the generic part to other domains, e.g. optical network management or mobile service management, and possibly reuse some of the IES Management specific part also.

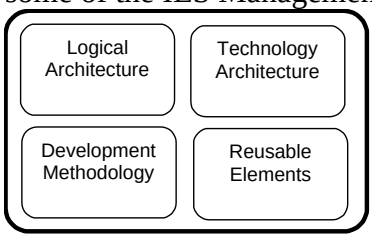

Figure 2-2: Structure of the Open Development Framework

The concerns addressed by the four portions of the Framework are:

- **Logical Architecture:** The Logical Architecture describes the structural concepts of the Framework and their relationships in a manner independent of any implementation technology. The core structural concept is the software Building Block (BB), which is an atomic unit of software deployment and management. A BB implements a number of Contracts that are the sole medium for inter-BB interactions. Systems are built primarily from assemblies of BBs. Systems are analysed in terms of Business Processes and Business Roles. Reference Points (RP) exist between Business Roles. RPs are realised through Contract implementations. To promote reuse of Contract Specifications they are described in a technology neutral format as well as one or more technology specific versions used in BB implementations. Defining Technology Neutral Contracts involves specifying the information passed via a Contract through reference to an External Information Model.
- **Development Methodology:** The Development Methodology provides the processes and notations needed to develop Building Blocks and assemble systems which conform to the Framework. The primary modelling notation used is UML, though the potential of XML for Contract specification is also being examined. The methodology integrated a number of existing modelling techniques such as use case modelling, business process modelling and model-view-controller analysis modelling plus the variety of other modelling approaches supported by UML. The Rational Unified Process (RUP) is used as a partial template to integrate these techniques. The Methodology contains two Guidelines, one for the development of Building Blocks and the other for the development of business processes into management systems that make use of Building Blocks.
- **Technology Architecture:** The Technology Architecture addresses how the concepts expressed in the Logical Architecture can be implemented using a range of technologies. For each technology a single mapping between the technology neutral specification of Contracts to the native specification language of the technology is

sought. Adaptation to allow interoperability between Contracts implemented in different technologies is also addressed.

- **Reusable Elements:** This portion of the Framework is the repository for reusable products that result when the Framework is applied to a particular application domain, e.g. the IES Management domain addressed in FORM. The principle types of reusable entities are: Business Role definitions; Reference Point specifications; Contract specifications and their grouping into BB Specifications and BB implementations. Other types of reusable elements, such as policy and business process definitions are also being investigated.

The generic aspects of the Framework are mostly found in the Logical Architecture, which is the subject of the remainder of this white-paper.

3 Related Work

A basic design goal for the Framework is to minimise the generation of new concepts or techniques. Instead the use of existing architectural models is maximised, where possible using material already subject to industry agreement through standards bodies or industrial fora. Some of the main concepts imported into the Framework are described in the following subsections.

3.1 OMG Model-Driven Architecture

The Object Management Group (OMG) has recently developed its Model-Driven Architecture [ab/2001-02-01]. This builds on the Object Management Architecture, which provided a framework for CORBA standards, and encompass opportunities for improved use of modelling techniques on the software engineering process offered by its standardisation of UML. The FORM Framework places a similar emphasis on modelling at all stages of the management BB and management system development cycle. In addition, UML is used as the primary modelling notation as in the MDA, with RUP providing the skeleton of the development process, an aspect not yet addressed in the MDA.

The Framework also embodies a similar emphasis on developing models that are independent of the implementation technology, or “platform-independent modelling” as it is termed in the MDA. The range of technologies over which such independence will be exerted is limited to ones implementing object-oriented RPC-style interactions (e.g. CORBA etc) and WWW based interactions (e.g. SOAP). However, the Framework needs to address the full range of interoperability technologies used in management systems, including message passing and management-agent paradigms, neither of which are strongly visible in the current MDA roadmap.

The MDA places a strong emphasis on basing its models on a well defined meta-model that is defined as part of the Meta Object Facility (MOF) [ad/97-08-14]. However this is used primarily to ensure that the architecture is well structured and self consistent, and is not exploited directly for model interchange between CASE tools as is the case with the MOF.

3.2 TM Forum Telecoms Operations Map

The TM Forum’s Telecoms Operation Map (TOM) provides a generic, high-level process model for telecommunications management. FORM adopted the notion of using business process modelling for driving system and building block requirements, however it does not advocate a generic process model, but the generation of domain specific models that reuse and refine some of the process definitions of the TOM, e.g. an IES Business Process Model.

3.3 TM Forum Generic Requirements for Telecommunications Management Building Blocks

The TeleManagement Forum has produced document GB909 that is a set of requirements for generic telecommunications building blocks [gb909], which are heavily derived from Telcordia's OSCA/INA work. These requirements were adopted by FORM as an initial set of development requirements for the Framework. The concepts of Building Blocks and Contracts were informed by these requirements. After initial implementation trials conducted in FORM a clearer assessment of these requirements has been formed which has been reflected in the revision of the Framework presented in this white-paper.
GB909 places a number of, sometimes conflicting, requirements on the atomicity and separation of functionality that a Building Block may exhibit. These, therefore, were found difficult to adhere to in FORM's implementation trials. The FORM Framework now therefore relaxes the requirement for Building Blocks to exist strictly within one of three computing tiers (Human Interaction, Process Automation and Enterprise Information tiers). Instead this separation in design is encouraged by adopting a Model-View-Controller pattern for generating analysis models used in the definition of Contracts. This aims to ensure that Contracts are designed with a focus on the separation essential to these three tiers, but allows Building Block developers to breach these separations if practical design or business issues dictate.
GB909 also specifies the Building Block to be an atomic unit of deployment, management, distribution, security, and interoperability. Trial experience in FORM has revised this view such that the Building Block is now simply a unit of deployment and management, the Contract is the unit of interoperability and of security and a Building Block Group is the unit of software Distribution.
Several GB909 requirements were assessed to be more akin to design guidelines, and therefore are being considered for inclusion explicitly in the Framework's Development Methodology. Other requirements that relate to computing platform services supporting trading, transaction and data stewardship are not being addressed directly in FORM.

3.4 TM Forum's NGOSS

The TM Forum's NGOSS initiative builds upon the GB909 requirements described above and aims to produce an architecture for component-based management systems, an aim very similar to that of the FORM Framework. As a result FORM has been closely tracking and contributing to this initiative. NGOSS is work in progress and at the time of writing an initial version of the Technology Neutral Architecture for NGOSS is available for membership comment [tmf053]. However, FORM has included several concepts from NGOSS into its Framework, namely:

- The explicit specification of information passed via Contracts in an externalised information model, termed a Shared Information Model in NGOSS. However, the use of externalised information is restricted to the reuse of models between separately developed Contract specifications. Such models are not necessarily intended in FORM to be used for structuring corporate data repositories, and related issues, such as data stewardship, are not addressed.
- The specification of Contracts (and Shared Information Models) in a technology neutral manner, with mappings being developed to specific interface technologies.
- The separation of component software functions from business logic, with the latter expressed in a form enactable at runtime. However, in NGOSS this focuses on the logic that drives the external invocation of Contracts, in FORM this is extended to

address the coupling of the invocation of a BB's Contract to other aspects of a BB behaviour, e.g. the emission of events or the invocation of another Contract.

FORM differs from the NGOSS approach as follows:

- The Contract in FORM is not specified as a unit of business process as in NGOSS, and may contain several separate operations that may be invoked from process enactment engines, other Building Blocks or legacy systems. NGOSS Contracts include formal pre- and post-conditions that are subject to runtime checking by such process enactment environments. Such run-time checks are not addressed in FORM.
- The FORM Framework does not address the specification of abstract Framework Services of which Building Block implementations make use, as addressed in NGOSS.

3.5 DMTF's CIM

A strong candidate for use in the definition of the External Information Model (EIM) used in a Contract Specification is the Common Information Model (CIM) meta-schema [cim] used by the DMTF, which can be used to express information models in XML. The CIM meta-schema is an object-oriented meta-model underlying the Managed Object Format language. It supports object classes and instances with properties of simple types or arrays of simple types, methods, indications and relationship properties. Class inheritance is supported as are association objects representing relationships between two or more objects. The Schema definition uses qualifiers to characterise its different elements.

The CIM benefits from the following with respect to information modelling:

- It is in the management domain, so existing CIM specifications could be imported for use in EIMs.
- The CIM Schema was designed as a technology neutral modelling language and mappings to several specific technologies including HTTP/XML (Web-Based Enterprise Management – WBEM) [wbem], Remote Procedure Call (Desktop Management Interface – DMI), LDAP-based directories (Directory Enabled Networks - DEN) and CMIS [festor] have been demonstrated.
- Though the CIM Schema is expressed in its own language (MOF), the DMTF have an XML mapping [cim-xml] for it (part of WBEM).
- The DMTF has already demonstrated how the CIM Schema can be used to express and enhance class models expressed in UML.

However, in order to fully satisfy the requirements for information modelling in the Framework several modifications need to be made to the CIM Schema as discussed in section 6.1.

3.6 TINA

Though the TINA Consortium is no longer active, several of its architectural concepts are used in FORM. Elements of the ODL language evaluated in TINA and now standardised by the ITU [itu-old] have informed the FORM Building Block specification schema, in particular the inclusion of references to Contracts that are required by a Building Block.

In addition, FORM has adopted the TINA business modelling concepts [mulder] of business roles and reference points between roles. The segmentation of reference points as used in TINA is also being considered, but has not yet been included in the Architecture. The FORM Framework maps Reference Points onto sets of Contract Specifications. Unlike TINA, FORM does not define a generic business model but advocates the use of domain-specific business models, which can then be mapped onto business process models for the same domain.

4 Logical Architecture

This section describes the Logical Architecture of the FORM Open Development Framework, which together with the Development Methodology makes up the core of the generic portion of the Framework. The Logical Architecture is described in terms of the following:

- A set of abstract roles representing potential users of the Framework at different points in the lifecycle of its main structural elements.
- A set of over-arching principles governing the structure of specifications and software that conforms to the Framework
- A set of models describing the main structural elements of the Logical Architecture and how they relate to each other

This approach is taken because a key feature of a component-based architecture is that it addresses a broad range of the software development lifecycle and therefore brings together concerns that are only relevant to certain roles in the overall software lifecycle.

4.1 Framework User Roles

The Framework User roles defined in this section are used to aid in the description of the Logical Architecture. As with any component-based architecture, the FORM architecture addresses concerns over a broad range of the general software lifecycle. Therefore parts of the architectural model that are relevant in one phase of the lifecycle, e.g. requirements analysis, are not relevant to another stage, e.g. software deployment. Therefore these User Roles are introduced to allow readers of this white-paper to understand the relevance of the various parts of the architecture to their particular sphere of interest.

Domain Analyst: Produces requirements and analysis models addressing a scoped domain of management functionality. The structure of such models is not prescribed here but it is assumed that it contains some high-level analysis models of services and information structures relevant to the domain. These would be presented with traces to the original requirement statements (typically via a use case analysis). This role may be performed by a variety of business stakeholders. It could be performed by a team within a Service Provider (SP) which wishes to specify a set of management system functionality the SP wishes to put out to tender. The role could be performed by product development analysts in an Independent Software Vendor (ISV) company, who are attempting to analyse requirements from a broad range of customers and from internal domain specialists in order to develop a set reusable component products. The role could also be taken by business modelling groups in an industrial forum, e.g. the TM Forum's Business Aware Contract Team (BACT).

- **Contract Designer:** Produces Technology Neutral Contract Specifications and External Information Models. This role may also be performed by a number of stakeholders. It may be performed by designers in an ISV in order to define interoperable interfaces to component-oriented products they are implementing. It may be performed by designers in a System Integrator (SI) or a SP as part of the specification of component-oriented interfaces to existing systems or prior to a component development or procurement activity. It may also be performed by modelling teams in an industry forum, e.g. TM Forum's BACT, in an attempt to provide open interface specifications against which components can be implemented.
- **Building Block Developer:** Produces software Building Blocks that support technology specific bindings of technology neutral contract specifications. This role may be performed by ISVs, or the internal development departments of SI or SPs.

- **System Builder:** Produces and deploys software systems that consist largely of Building Blocks. This role will typically be performed by product development teams of a SI or within a SP.
- **System Administrator:** This role is responsible for monitoring and managing a system containing Building Blocks in order to ensure it operates within required operational parameters. This role is not, however, explored further in this paper.

The FORM Framework defines the architectural concepts in which each of these roles has an interest and how these concepts are used in interactions between these roles. It is important to note, however, that these are *abstract* User Roles introduced here purely to aid in understanding the lifecycle of concepts in the Logical Architecture. Real-life developers may therefore take on several of these abstract Roles. The Framework's methodology guidelines are based on the needs of such real-life developers.

4.2 Architectural Principles

The Framework is structured around the following architectural principles:

- Management Systems are software systems that perform some task related to communications management in an operational environment. They are constructed partially or fully from Building Blocks (BB).
- Building Blocks are pieces of software.
- Building Blocks are atomic units of Deployment (one can be replaced in a running system without requiring other BBs to be replaced).
- Building Blocks are atomic units of system management.
- Building Blocks may support multiple interfaces types and multiple instances of those types. These interface types are termed Contracts.
- The Logical Architecture does not prescribe the technology used to implement Building Blocks or their Contracts.
- Contracts may be defined in a technology neutral form called a Technology Neutral Contract Specification (TNCS).
- A BB software release implements Contracts in a technology specific form, termed a Technology Specific Contract Specification (TSCS), such as an IDL specification. Explicitly declared transforms are required to map a TNCS to a TSCS.
- A Contract can support multiple business transaction operations.
- A Contract type can support only one interaction paradigm, e.g. RPC, Manager-agent, DB query, publish/subscribe and others (perhaps more fundamental).
- The information passed via a Contract is expressed through reference to an External Information Model (EIM).
- Building Blocks can be grouped into Building Block Groups. This is the typical unit of purchase from a Software Vendor (though sale of individual BBs is possible as a BB Group with a single member).
- Building Block Groups must be released with documentation giving the accompanying business context, use cases, analysis model, TNCS and an External Information Model related to the BB and TSCS designs in that Group.
- Building Blocks possess behaviour linking interactions over different contracts. BB design may allow its behaviour to be modified at deployment or runtime. Where this feature is offered it should use a common, policy-based mechanism.
- Reference Points identify boundaries over which interactions occur between two Business Roles identified for a Business Domain (e.g. IES). Reference Points can be mapped to one or more Contract Specifications in order to define how business level interfaces may be implemented using BBs.

- TNSC and EIM elements should be traceable to specific business requirements and vice-versa.
- The behaviour of BBs or groups of BBs should be modifiable at run-time through the modification of business rules.

These broad principles have been used to structure the architectural model described in the next section.

4.3 Architectural Model

The main structural concepts present in the Framework are grouped in terms of the models generated by the Framework user roles and thus used in exchanging information between those roles. These models are:

Business Analysis Model: This scopes an area of concern, provides its business context and defines a high-level analysis model of its externally observable functionality and logical decomposition.

External Information Model: This is a technology neutral expression of the information that is potentially passed via a set of Contracts. This provides the primary mechanism for ensuring interoperability between Contracts defined in different technologies.

Contract Set Specification: This is a technology neutral set of interface specifications that may be utilised in designing BB software. It provides the primary means of interoperability by detailing sets of interfaces that can be used in interacting with Building Blocks.

Building Block Group: This is a set of Building Block implementations and its accompanying documentation.

Management System Model: This is the description of a Management System's design, which uses Building Blocks to some extent.

Though these models are described separately, the building of reusable software and management systems that use them relies on managing links between elements of these models. For instance a Management System Model will refer to elements of one or more Building Block Group models, which in turn will reference one or more Contract Set Specifications. The structure of the above models and the links between elements in the different models are captured as a set of linked meta-models. Elements of this meta-model are the basis for the modelling artefact defined in the Development Methodology portion of the Framework.

Figure 4-1 gives an overview of the meta-model relationships between the primary parts of the Architectural Model.

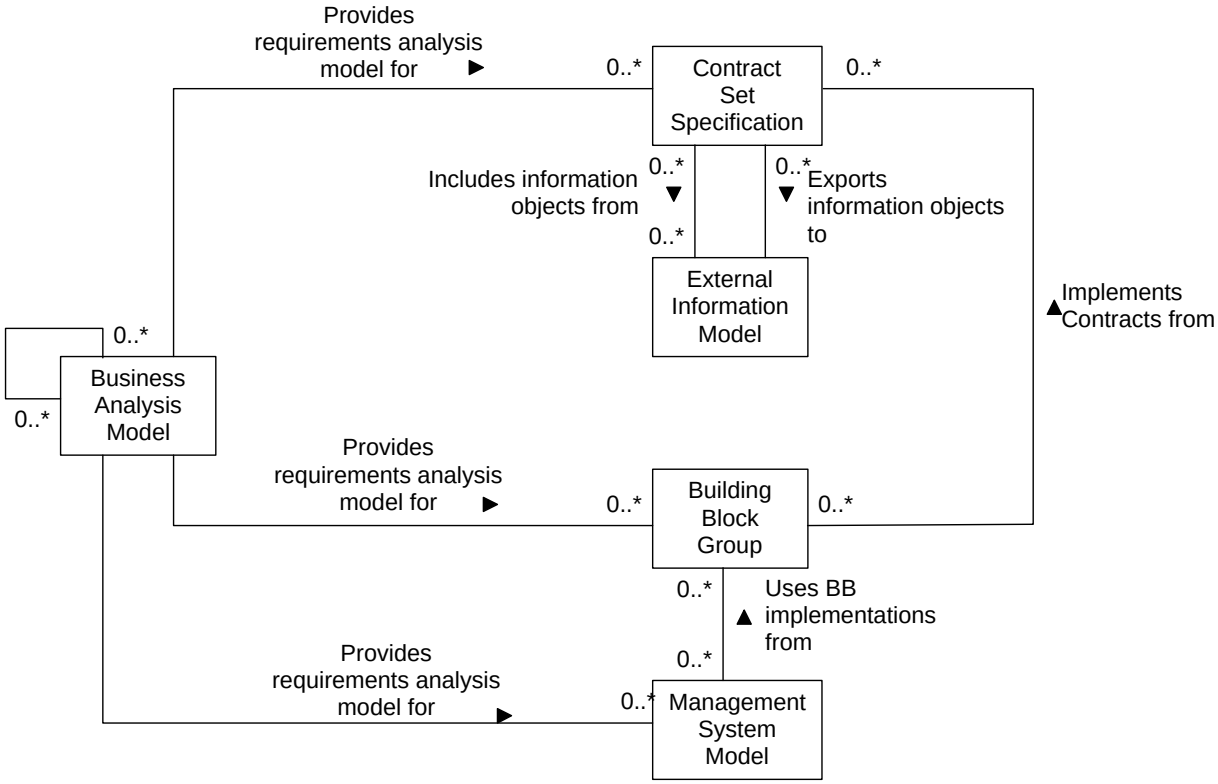

Figure 4-1: Relationships between primary models in the Logical Architecture

The primary models of the Logical Architecture are each produced by a specific architecture user role and used by one or more other roles as follows:

- The Business Analysis Model is generated by the Domain Analysis role and used by other Domain Analysts as well as by the Contract Designer, Building Block Developer and System Builder roles.
- The Contract Set Specification is generated by the Contract Designer role and used by other Contract Designers as well as by the Building Block Developer, System Builder and System Administrator roles.
- The External Information Model is generated by the Contract Designer role and used by other Contract Designers as well as by the Building Block Developer, System Builder and System Administrator roles.
- The Building Block Group is generated by the Building Block Developer role and used by other Building Block Developers as well as by the System Builder and System Administrator roles.
- The Management System Model is generated by the System Builder role and used by other System Builders as well as by the System Administrator role.

The following subsections describe the structure of the individual models in more detail.

4.3.1 Structure of Business Analysis Model

A Business Analysis Model captures the business requirements of a domain of interest and analyses then to provide a logical model of the domain's interfaces and behaviour. It is intended that the elements of this analysis are maintained with traceable links to the requirements-oriented parts of the Model. These traceable links will support the overall traceability between requirements and EIM and TNCS elements.

A Business Analysis Model may be specialised from one which addresses a relatively general domain to one that represents a more specific domain.
A Business Analysis Model has the following constituent models:

- A Business Context Model that captures the requirements of the domain of interest and models its business environment and business structure.
- A Domain Model that expresses in detail the boundary of domain of interest, the behaviour of the domain viewed from outside this boundary and an analysis of the potential decomposition within the domain.

These are described in more detail in the following subsections.

4.3.1.1 Business Context Model

This is the description of the business requirements related to the domain of interest. A wide range of mechanisms exist for capturing business requirements and this model does not aim to restrict which ones are used, however the following have been found useful in FORM:

Categorised Requirements Statements: This enumerates and categorises existing requirements statements to allow traces to be made to them from the rest of the Domain Analysis Model. Categorisation may be between functional, non-functional (e.g. adherence to existing standards), exception-related and information related requirements.

Business Role Model: This defines generic Business Roles relevant to a domain. Reference Points can be identified between pairs of roles and used to collect a set of interactions that may occur between those roles. Zero or more Reference Points may be identified between a pair of Business Roles.

Business Process Model: This defines a set of business processes relevant to the domain. Business processes may be linked using modelling techniques that show the flow of events and information between different processes and process users. At the highest level of abstraction the business process may be identified and possibly grouped, e.g. by TMN logical layers. Business process flows may then be defined capturing the end-to-end flow of information and flow of control between business processes. Business process flows need to also identify starting and terminating events.

It is possible to map elements of the Business Process Model onto elements of the Business Role Model. This is termed a Reference Architecture Model and consists of the following mappings:

- A Business Process from the Business Process Model may be mapped onto a Business Role from the Business Role Model. The same Business Process may be replicated in more than one Business Role, but a single Business Process cannot span more than one role.
- Business Process flows from the Business Process Model are mapped to Reference Points from the Business Role Model.

These mappings allow consistency checks to be performed. This may ens ure that an information flow between two business processes residing in different Business Roles is supported by the presence of a Reference Point between those two Business Roles. Similarly the need for and requirements upon a Reference Point is clarified by understanding the information flows between Business Processes that pass across that Reference Point.

4.3.1.2 Domain Model

The Domain Model represents a semi-formal analysis of the domain of interest and its interaction with its environment. This analysis is based upon and traceable back to the requirements and business modelling elements from the Business Context Model. It is

represented in a way that is independent of any issues concerning the implementation technology used for software that implements interfaces and behaviour of the domain. The Domain Model consists of the following models:

- Domain Boundary Model: This defines the domain boundary expressed by describing the Domain Actors that form the environment in which the domain exists. Different types of Domain Actors may be used, taken from the Business Context. The following actor types are possible:
    - Human Actors: human system users that will interact with a software system that wholly or partially implements the requirements analysed by the domain analysis.
    - System Actors: software systems that will interact with a system that results from the domain analysis. This may represent an existing system, or a planned system that is subject to another domain analysis in parallel or subsequent to this domain analysis. Existing systems already may offer well-defined interfaces via which interactions with that actor must be conducted.
    - Process Actors: process descriptions typically used when the domain is defined as part of a business process model that is to undergo detailed analysis.
    - Role Actors: represent business roles that are external to the domain.
- Domain Use Case Model: This is a set of use cases that defines the full set of interactions that the domain has with Domain Actors. Each Domain Use Case takes one Domain Actor as its primary actor, though any number of other Domain Actors may be secondary use case actors. Domain Use Cases should have pre-conditions and post-conditions, and these may be bound between use cases so that certain post-conditions from one Use Case in the Model may form pre-conditions to another use case.
- Domain Analysis Model: This is an object-oriented model of the Domain. It consists of Analysis Objects (AOs), class definitions and definitions of the static associations and dynamic interactions between them. AOs take one of three stereotypes that reflect the Model-View-Controller design pattern:
    - Boundary AO: This models the interactions between the Domain and a human Domain Actor.
    - Entity AO: This represents information within the domain
    - Control AO: This models functional, algorithmic or process oriented aspects of the domain

Figure 4-2 shows the relationships between the constituent elements of the Domain Use Case, Domain Boundary and Domain Analysis Models.

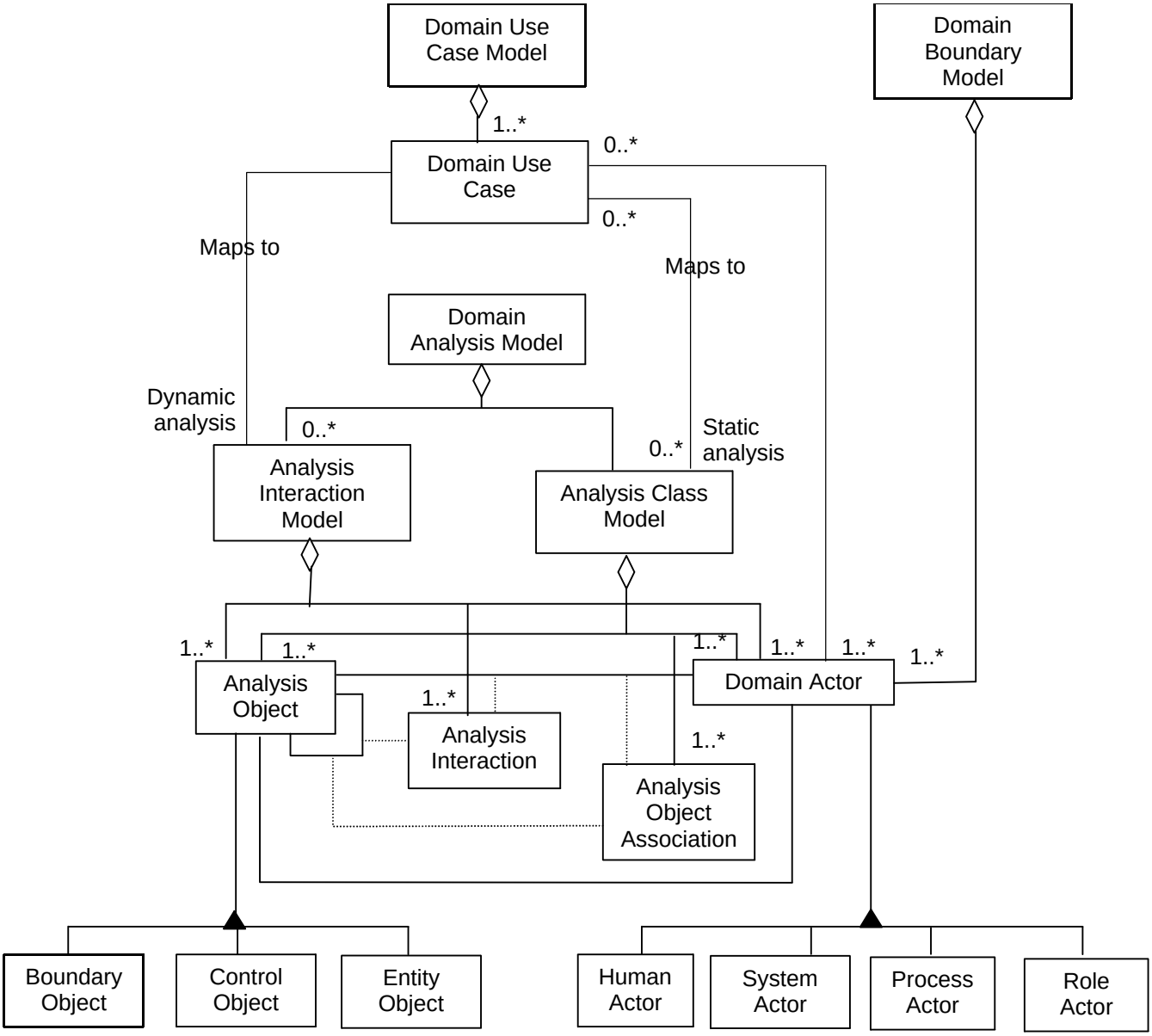

Figure 4-2: Meta-model mapping between Use Case Model and Analysis Model Elements

4.3.1.3 Relationships in the Business Analysis Model

As indicated in Figure 4-1, a Business Analysis Model may be related to one or more other Business Analysis Models. This allows a Business Analysis Model to be broken down into more manageable parts and it also allows one organisation to use and build upon elements from a Business Analysis Model from another organisation. Hence the relationship between Business Analysis Models involves including a subset of elements from one model into another. If the elements included are from the Business Context Model of the originating Business Analysis Model then elements from the Domain Model, (e.g. Domain Actors, Domain Use Cases and Domain Analysis Object definitions and interactions) which can be traced to those Business context Model elements may be automatically included.
To support traceability between a Business Analysis Model and a Domain Model the following mappings between elements in the two models may exist:

- A human user of the domain of interest, identified in either a Business Process Model or a Requirements Statement may be represented as a Human Actor in the Domain Boundary Model.
- A Business Process identified in a Business Process Model, which is external to the domain of interest, but which interacts with a Business Process within the domain may be represented as a Process Actor in the Domain Boundary Model.
- A Business Role identified in a Business Role Model, which is external to the domain of interest, but which interacts with a Business Role within the domain may be represented as a Role Actor in the Domain Boundary Model.
- A requirement from the Business Analysis Model for the domain of interest to interact with an existing system external to the domain or to be able to interact with an external system via a given interoperability specification can be represented in part as a System Actor in the domain Boundary Model.

4.3.2 Structure of External Information Model

An External Information Model (EIM) externalises the information content of one or more Contract Specifications. Such models are often embedded in interface definition languages. Information models placed in an EIM are made available for use in subsequently developed Contract Set Specifications.
The ODF advocates the externalisation in a simple common format of the information content of Contract Specifications and the progressive use of this information in consolidation exercises by individual organisations to generate External Information Models (we use this term instead of “Shared Information Model”). As EIMs stabilise, the information content of individual Contract Specifications should increasingly consist of references to EIM elements. External Information Models accomplish the following:

- Comparison of EIM elements to information requirements generated, for example, from business process flows, allows Contract Specifications to be identified as possible candidates for addressing control flow requirements. This is part of the selection process for Building Blocks (or even unimplemented contract Specifications) and must be coupled with analysis of the Contract’s Business Context.
- Contract Specification standardisation and BB release by different organisations may establish a growing body of accessible, easy to understand EIMs, which in turn removes barriers to the reuse of existing information models in the development of new Contracts.
- The risks of developing adaptation functions between different Contract Specifications are more easily assessed though identification of common EIM references, comparison of Contract Set Information Models and knowledge of the number of other Contract Specifications referencing an EIM element

An EIM is a specialisation of an Information Model. An Information Model as defined in the FORM Framework consists of Information Objects (IOs). An IO has a class name and a number of attributes. IOs can be inherited from other IOs, which involves all of the attributes of the latter IO being present in the inherited IO. The term Information Object is used here to refer to object classes describing both the structure of data and the associations between items of data. Therefore, a specialised type of IO called an association object, represents associations between other IOs and has properties that represent the roles played by other IOs in the association.
IOs may be included in an Information Model from another Information Model. Included IOs may be imported, in which case they are effectively copied into the administrative scope of

the Information Model and will not reflect future revisions to the IO in the originating Information Model (though they must contain a reference to the source IO). Alternatively, included IOs may be expressed as a reference to one in another Information Model, in which case its expression in the importing Information Model will reflect revisions to the IO in the originating Information Model.
Regardless of whether an IO is imported or referenced, it may be subject to an Attribute Filter within the Information Model that allows attributes in the original IO to be excluded from the Information Model.
Figure 4-3 provides a UML description of the meta-model for the Information Model in the FORM Framework.

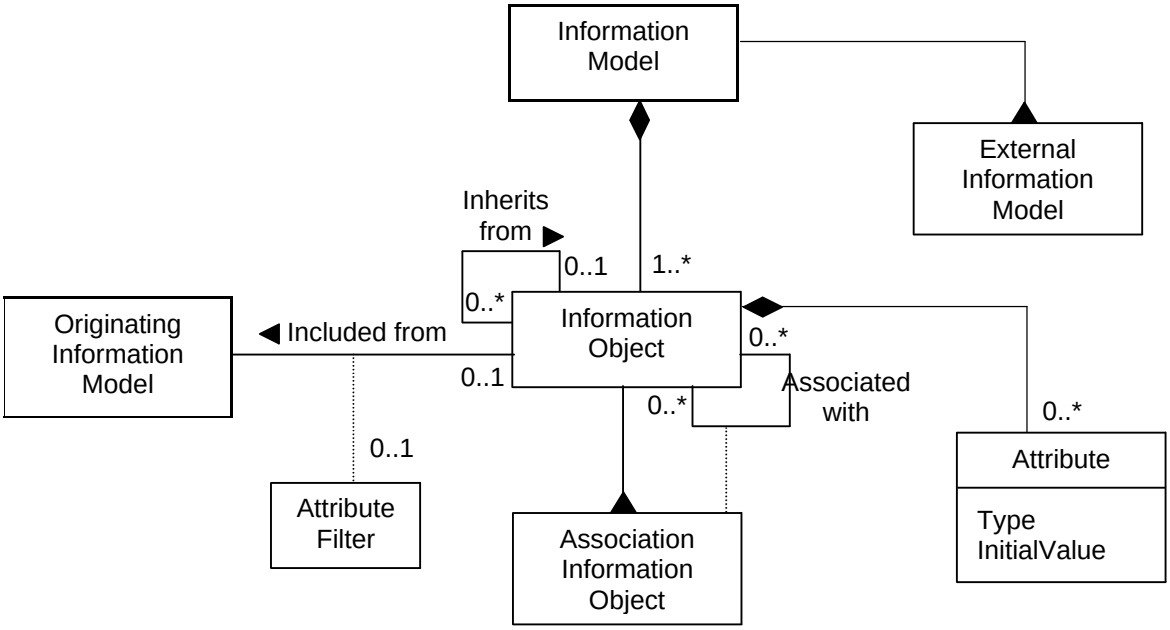

Figure 4-3: Structure of Information Models

The following rules should be followed in generating Information Models:

- The objects in an Information Model may only represent information that is passed via a Contract. They should not represent information internal to any software that provides an implementation of a Contract but which is not visible via a Contract.
- An Information Model should contain class elements and associations between classes
- Classes may only possess a class name and attributes, classes may not possess methods.
- Class attributes may only be simple type or arrays of simple types. Currently the set of simple types is char, string, real, integer, boolean, though this may be expanded in future (perhaps aligned with the DMTF CIM type set). Complex types must be broken down into classes.
- Class attributes can be given initial values
- Classes may be inherited from other classes in the model.
- Associations must be named. The classes at each end of the association should be given a name indicating their role in the association. If no name is given, the name of the class is used to indicate their role in the association.
- The cardinality of each end of an association should be given.

The relationship of EIM elements to elements of a Contract Set Model are explained in detail in the next section.

The relationships between an EIM and a Business Analysis Model consist of mappings between Entity Object classes in the Domain Analysis Model of the latter and IO classes in the former.

4.3.3 Structure of Contract Set Specification Model

A Contract Set Specification is the main mechanism within the Architecture for specifying interoperability. A Contract Set primarily consists of one or more individual Technology Neutral Contract Specifications (TNCS) together with a Contract Set Information Model (CSIM) that contains definitions of the Information Objects (IO) describing the information passed in individual TNCS operations. The IOs may be defined specifically for use in the Contract Set or they may be included from an External Information Model. An IO in the CSIM referenced from a specific TNCS may be done so via an Attribute Filter. Where a Contract Set needs an IO that is similar to one in a EIM, but which requires additional attributes, a new IO may be derived within the scope of the Contract Set Specification by inheriting from the original IO (which may be included from the EIM).
Though specifications in a Contract Set are independent of specific implementation technologies, individual TNCSs may be expressed in a style that is bound to a particular Interaction Pattern. An Interaction Pattern captures a style of interface specification that may be common to a range of interface implementation technologies. Interactions Patterns are differentiated primarily on the basis of how individual interactions are expressed, e.g. RPC operations, notification, message sequences, and how this expression makes use of an information model.
A TNCS within a Contract Set may be included as a reference to a TNCS in another Contract Set. A TNCS included by reference must therefore reflect revisions to the referenced TNCS. In addition, any IOs used by a TNCS included by reference must be included in the Contract Set as reference to the relevant IOs in the Contract Set containing the referenced TNCS.
The meta-model for Contract Sets is outlined in Figure 4-4.

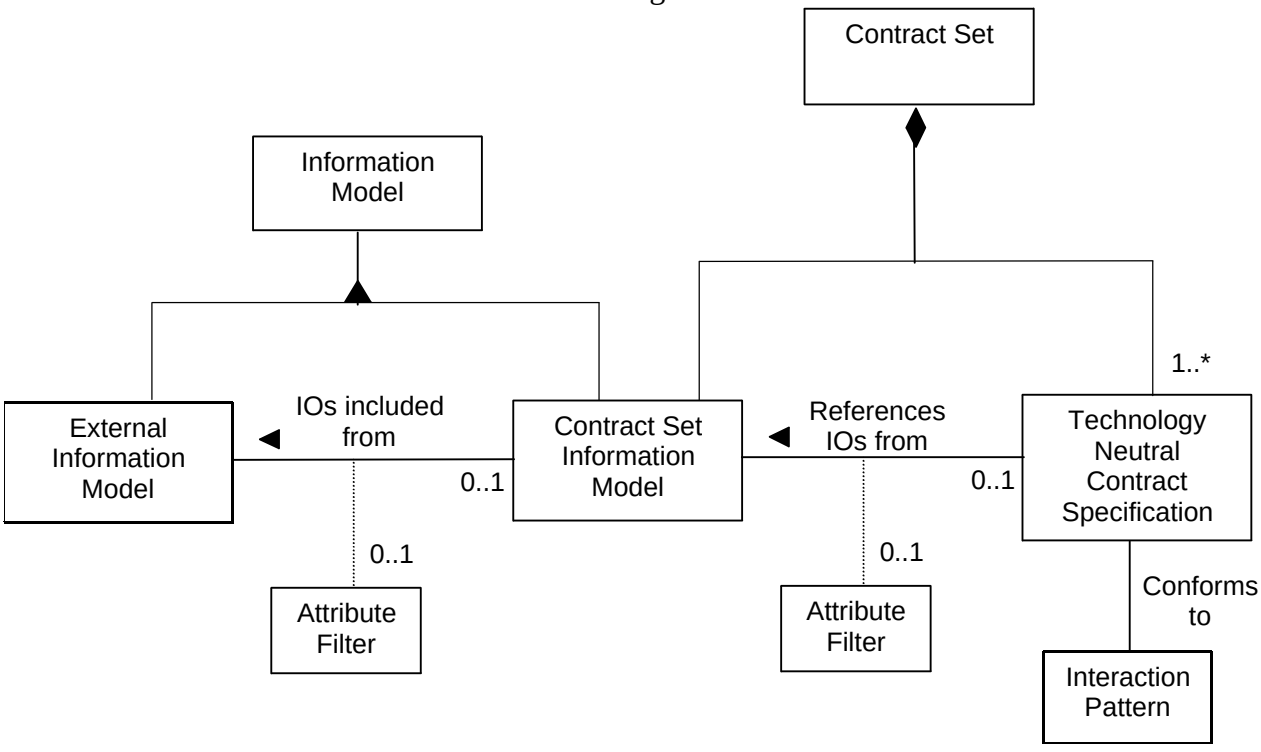

Figure 4-4: Structure of Contract Sets

A Contract Set Specification is based on the requirements and analysis from one or more Business Analysis Models.
The lifecycle of a Contract Set and the TNCS it contains and the lifecycle of an EIM are essentially separate with the relationships between them managed by the inclusion of tags in a Contract Set Specification. However, a Contract Specification may export IOs into a new or existing EIM. This may be performed in concert with the IOs exported from other Contract Sets in order to construct an EIM common to a range of Contract Sets.
When an included IO or a derivative of an included IO is exported to an EIM, the inclusion tag referring to the originating EIM IO and any inclusion filter must also be included. In this way model adaptation tools can quickly identify similarities between IOs that come from common design paths, as well as details of differences in terms of derivation and filtering information.

4.3.4 Structure of Building Block Group Model

A Building Block Group is a model containing a collection of Building Block Specifications and accompanying implementations. A Building Block Specification contains one or more TSCSs. A TSCS references a single TNSC and binds it to a technology transform that maps the TNCS format to a specific interface implementation technology. The set of TNCSs referenced by the TSCSs of a Building Block can be taken from one or more Contract Sets.
It is possible for a BB to implement different (or even the same) TNCSs in different technologies. The generation of a TSCS from a TNCS will follow a standardised or proprietary technology transform. References to existing mappings and details of any bespoke mapping used are recorded in the TSCS.
A BBG is ready for release once each of its BB's has a complete set of TNCSs defined and their implementations integrated with the BB's hard-coded internal logic and assembled into a single unit of deployable software, i.e. a BB. Note that some of the BB's internal logic may be 'soft-coded', i.e. specified by business rules that are bound to the BB at deployment or run time and evaluated at run-time.
Figure 4-5 represents the meta-model for Building Block Groups.

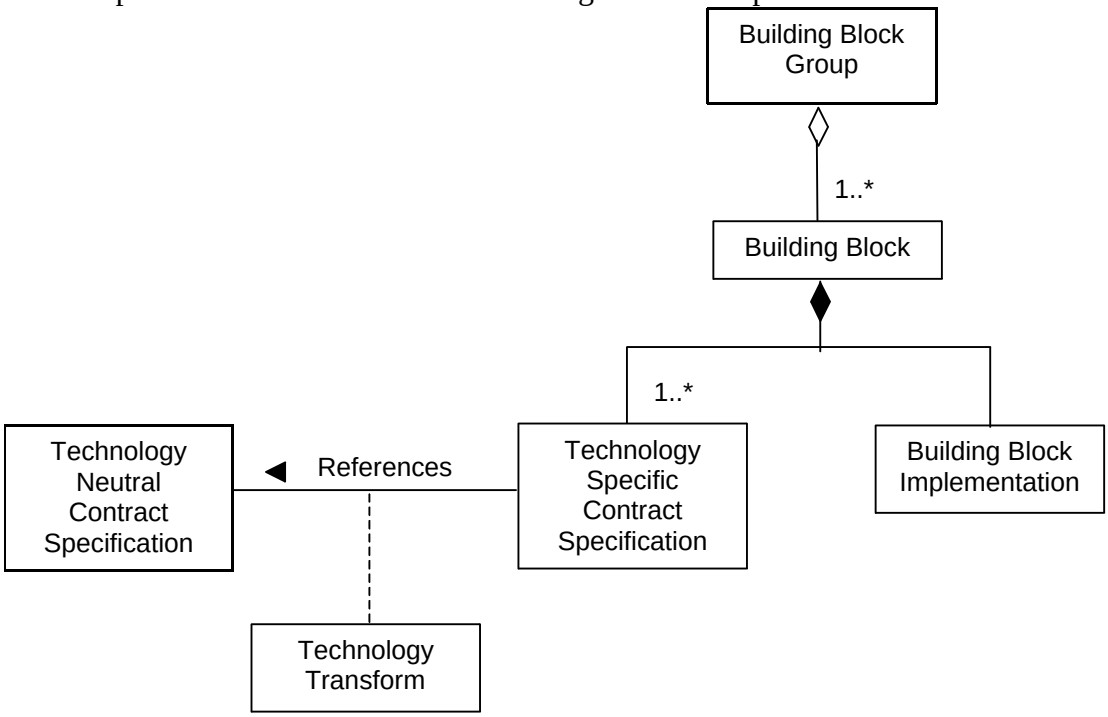

Figure 4-5: Structure for Building Block Groups

A Building Block Group may be related to one or more Contract Set Specifications through implementing TSCSs that are generated via technology transforms from TNCSs from a Contract Set Specification.

4.3.5 Structure for Management System Model

The Management System Model is a representation of the design of a software system intended to perform some set of management-related tasks in an operational context.
The model consists of the following:

- One or more System Interface Specifications via which the Management System will interact with its environment. The architecture imposes no restrictions on how a System Interface is structured or represented.
- One or more Building Blocks taken from one or more Building Block Groups.
- Zero or more Subsystems, which are software modules that do not conform to the required structure of a Building Block. Typically these may be legacy components or glue code needed to integrate Building Blocks. The architecture imposes no restrictions on how a Subsystem is structured or represented.
- Zero or more Business Rules which express aspects of a Building Block's behaviour. Business Rules can typically be changed after the Building Block has been implemented.

Building Blocks communicate with other entities using implementations of the TSCS to which they are bound. A subsystem that interacts with a Building Block must use or implement, as appropriate, the relevant TSCS.
A System Interface Specification can be implemented directly by a Subsystem, bound to one or more TSCSs or a combination of both.
Business Rules determine how a Building Block behaves, as observed via the Contracts it offers and the Contracts it uses. The conditions that determine the triggering and outcome of a Rule evaluation are derived from one the following:

- the result of an interaction by an external entity with one of the BB's Contracts,
- the result of the BB interacting with the Contracts of other BBs
- specific system events such as timeouts and system errors.

The format of Business Rules is not yet fully established in the ODF, however it is intended that it should support at least two types of rule expression. The first type is Process Flow rules which determine the order in which contracts and system Interfaces are called across a system. The target BB for such rules therefore typically plays a coordinating role in the pattern of interaction between different BBs and Subsystems, in a manner similar to a workflow engine. Such a BB is therefore designed in a very generic way, and expected to use a wide range of Contract and System Interface types not known at implementation time. The second type is Policy Rules. These are intended to provide some flexibility in the existing behaviour of a BB, but not to support interaction with arbitrary Contracts on other BBs.

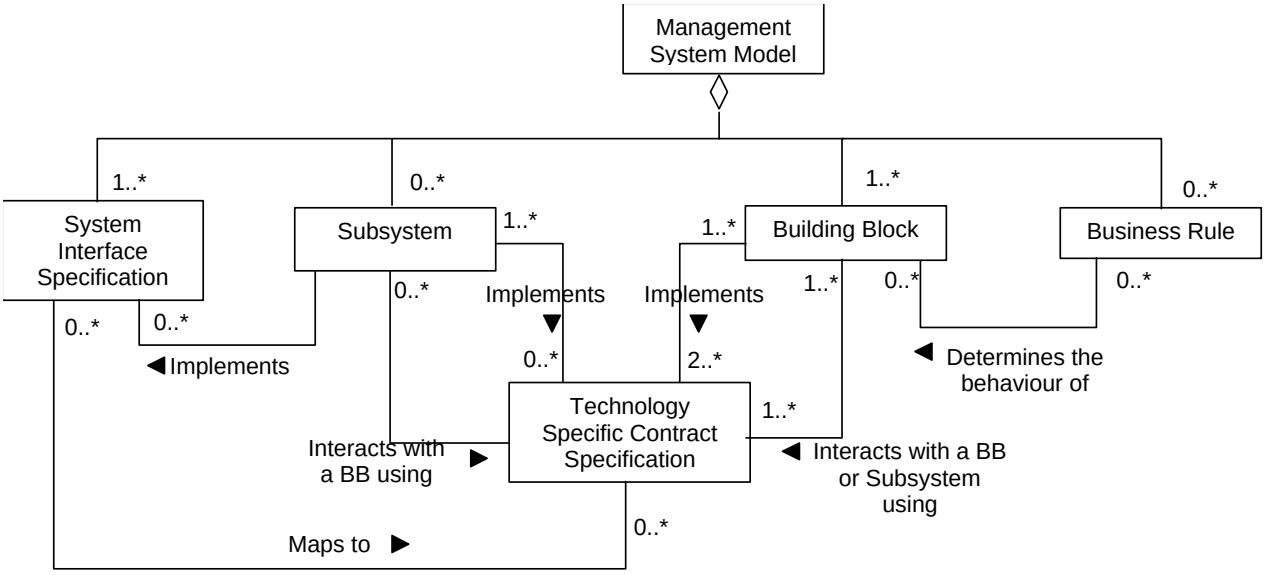

Figure 4-6: Structure of Management System Model

The requirements capture and analysis that typically precedes the generation of a Management System Model should be expressed in a Business Analysis Model. In such a case, the System Interface specification may be derived, wholly or in part, from a Reference Point definition. The Building Blocks in a Management System Model may be taken from one or more Building Block Groups.

5 Relationship to Development Methodology

The Logical Architecture provides the concepts that are used to structure and guide the use of the other portions of the ODF. This section describes how the structural elements described in section 4.3 are used in the Development Methodology.

The Development Methodology portion of the ODF specifies notations and processes for the development of models and software in accordance to the Framework. A number of specific methodologies may be generated for the ODF depending on the target audience. Two specific set of methodological guidelines are being generated within the FORM project:

1. *The Building Block Development Guideline*: This is aimed at those such as Software Vendors developing management Building Blocks for reuse by others
2. *The Business Process Driven Management System Development Guideline*: This is aimed at System Integrators who are developing management systems based on business processes analysis techniques and wish to make best use of off-the-shelf management Building Blocks.

These guidelines are detailed in separate white-papers.

Though the Logical Architecture is separate to the development methodology, it provides the meta-model for the languages used to express the models generated when following the methodological guidelines. To aid in the generation, processing and comprehension of these models industry standard notations have been used wherever possible. The principal notation used is the Unified Modelling Language (UML), though the eXtensible Markup Language (XML) is also used for packaging models of differing notations in a form readily publishable on the WWW. Both UML and XML are very general languages and have been profiled within

the methodological guidelines to define more precisely how these notations are used in the required contexts.
In addition, however, prior experience in using notations such as UML in the management domain has influenced the form of parts of the Logical Architecture. This was to ensure that the structural concepts presented in the Logical Architecture can be readily expressed using the chosen notations in an easily understandable and usable way, making maximum use of available CASE and document processing tools.

6 Further Work

Several architectural issues are under on-going investigation in the FORM project, with results expected to impact on revision to the Architectural Model. The following subsections outline a few of these issues.

6.1 Schema for Information Model

Though the CIM schema is used for the structure of EIMs, several additional issues need to be resolved including:

- A mechanism for addressing Information Objects in other Information Models, This may use a URL based identifier as used in CIM currently, or something based on X-Path.
- A Schema for Attribute Filters.
- A mechanism for describing IO behaviour. For this purpose method call could be retained, though other additional mechanisms such as state-transition models or policy model may be appropriate.
- A mapping from a UML representation of the model to an XML version to aid readability of the XML model.

6.2 Schema for Technology Neutral Contract Specification

An appropriate schema for the TNCS must be flexible enough to support different contract formats according to different interaction patterns. This must allow different forms of representing interaction as is supported by WSDL, which allows both procedure call or asynchronous message based mappings from a single model. However, in the management domain a manager-agent interaction pattern is also required, so a WSDL schema would need to be adapted to support this. As with the Information Model schema a mechanism is required to link to other model elements, IOs or other TNCSs.

6.3 Schema Building Block Specification

The schema for Building Block specifications needs to be defined further in addition to the use of TSCS. The two main components for this are

- A BB descriptor that brings together the specification elements and the software elements of the building blocks. For this the EJB or CCM [ccm] descriptor may provide a suitable basis.
- An additional specification providing a mechanism for flexible behaviour modification. For this the DMTF Policy Schema may provide a basis together with the generic policy enactment model being developed in FORM, which may allow policy condition and actions to be bound to elements of Contract specifications.

6.4 Technology Transfer Schema

A schema is required to be able to transfer from TNSC to specific TSCS languages such as Java, IDL, GDMO, WBEM etc. The aim, with TNCSs specified in XML is to use XSLT to perform this task, with each XSLT script representing a separate technology mapping.

6.5 DMTF/IETF Policies

Policies are used in various places in the initial FORM implementation trials, based on the policy model shared by the IETF and the DMTF. This includes using variations of existing policies defined for security and QoS management. In addition FORM is examining the use of a generic policy model for binding policy expressions to the software entities that will enact them. It is hoped this will result in a mechanism for binding policies to contract elements.

6.6 WSDL

The Web Service Definition Language (WSDL) was originally defined by Microsoft and is now undergoing standardisation in the W3C. It provides an XML vocabulary for specifying abstract services and then mapping them to a specific communications protocol, e.g. SOAP/HTTP. Its structure allows specification of service in terms of messages or RPC style operations.
WSDL operations are grouped into portTypes that may potentially form the basis of a technology neutral Contract Specification format. The information aspect of WSDL is expressed as xsd types that are used for the message parts that make up input and output fields of operations. This is, in a sense, a technology specific type set and would need to be aligned with the CIM-based XML schema for defining Externalised Information suggested above. Though WSDL represents a good candidate for a technology neutral contract specification language it requires further study before being adopted within the Framework.

7 Conclusions

FORM is developing an Open Development Framework to guide the developers of component-based management systems. This Framework is based on experienced garnered from previous EU projects and standardisation work on component-oriented or management-related architectures, including J2EE, the work of TM Forum's Application Component Team and NGOSS initiative, the DMTF's CIM and TINA-C's Service and Business Models, amongst others.
FORM is taking a build-and-learn approach to evaluating and refining this Framework, by applying it to the challenges presented by assembling value chains in the Inter-Enterprise Services domain. As a result, the presentation of the architectural and methodological concepts of the Open Development Framework will be made in concert with examples and demonstrations of their application in modelling and implementing the IES BB Sets.

8 References

[ad/97-08-14] Meta Objet Facility, revised submission, ad/97-08-14, OMG, Aug 1997

[ab/2001-02-01] Model Driven Architecture, A Technical Perspective, OMG Architecture Board, Review Draft, 14th February 2001, Doc number ab/2001-02-01, http://www.omg.org/cgi-bin/doc?ormsc/2001-07-01

[ccm] CORBA Component Model, OMG 1999: http://www.omg.org/cgi-bin/doc?orbos/99-07-01

[cim] Common Information Model v2.5, DMTF 2000:
http://www.dmtf.org/spec/cim_schema_v25.html

[cim-xml] CIM XML Mapping v2.0, DMTF 1999:
http://www.dmtf.org/download/spec/xmls/CIM_XML_Mapping20.htm

[festor] Integration of WBEM-based Management Agents in the OSI Framework, O. Festor, P. Festor, N.B. Youssef, L. Andrey, Proceedings of the Sixth IFIP/IEEE International Symposium on Integrated Network Management, Boston, USA, pp 49-64, IEEE, May 1999.

[gb909] Generic Requirements for Telecommunications Management Building Blocks, GB909, Member Evaluation Version 2.0, TeleManagement Forum, September 1999

[mulder] TINA Business Model and Reference Points, v4.0, Mulder, H. (ed), TINA baseline document, TINA-C, May 1997

[tmf053] NGOSS Architecture, Technology Neutral Specification, Membership Evaluation Version 1.51, TeleManagement Forum, July 2001

[itu-odl] ITU - Object Definition Language (ITU-ODL), ITU-T, Jan 1998

[wbem] Web-Based Enterprise Management, DMTF 1999: http://www.dmtf.org/spec/wbem.html

[xslt] XSL Transformations v1.0, W3C 1999: http://www.w3.org/TR/xslt