FORM

IST-1999-10357

Engineering a Co-operative Inter-Enterprise Management Framework Supporting Dynamic Federated Organisations Management

Document Number:	IST-1999-10357/WIT/WP3/1019
Title of Deliverable:	D9: Final FORM Framework
Deliverable Type: (P/R/L/I)*	R
Nature of the Deliverable: (P/R/S/T/O)**	P
Contractual Date of Delivery to the CEC:	31st January 2002
Actual Date of Delivery to the CEC:	31st January 2002

Workpackage responsible for the Deliverable:	WP3
Editor:	Eric Leray (WIT)
Contributor(s):	Willie Donnelly(WIT), Dave Lewis (UCL), Birgitte Lønvig(LMD), Jens Dyhre Mouritzsen (TDC), Vincent Wade(TCD)
Reviewer(s):	Hervé Karp(ATOS), Niamh Quinn(Broadcom)

ABSTRACT

The FORM project has specified an open development framework for the development of such service applications. The framework focuses on how to mediate between existing open frameworks, rather than being highly prescriptive and thus avoids becoming just another framework competing for the attention of industry stakeholders. The objective of this deliverable is to provide the reader with a detailed specification of the FORM development framework. The final Open Development Framework (ODF) presented here is targeted at service applications designers and developers in an open service market. It supports the concept of software component reuse (based on the concept of building blocks) and focuses on supporting the design and development of such reusable software components.

KEYWORDS

Open Development Framework, Logical Architecture, Methodology, Technology Selection, Reusable Elements, Building Block, Inter-Enterprise Service Provider, Business Model, External Information Model

* Type: P-Public, R-Restricted, L-Limited, I-Internal

** Nature: P-Prototype, R-Report, S-Specification, T-Tool, O-Other

IST-1999-10357

FORM

Deliverable D9

Final FORM Framework

Editor : Eric Leray (WIT)

Status – Version : Final

Date : **30/01/2002**

Distribution : Limited

Code : IST-1999-10357/WIT/WP3/1019

Table of Contents

Executive Summary

The management of telecommunications services and networks has undergone a major paradigm shift over the last number of years. The increased fragmentation of the telecommunications marketplace has spawned multiple service providers and connectivity providers. Hence, the emphasis has shifted to managing network and service QoS in line with service customers' expectations (as defined in the Service Level Agreement).

In such an environment, service delivery requires the integration of service components from customer, network operator and value added service providers. However, from a software perspective the framework must also reflect the relationship between the software system stakeholders such as independent software developers, system integrators and standards. Present day application development platforms are not sufficiently sophisticated to support the required capability. New frameworks are required that incorporate requirements from multiple perspectives and reflect industry best practice.

The FORM project has specified an open development framework for the development of such service applications. The framework focuses on how to mediate between existing open frameworks, rather than being highly prescriptive and thus avoids becoming just another framework competing for the attention of industry stakeholders.

The objective of this deliverable is to provide the reader with a detailed specification of the FORM development framework. The initial step in the approach to developing the framework was to identify the key players involved in the delivery of such services. Through a process of requirements capture, within the industry, the project specified the functionality and interdependencies between each of these players. An initial framework was then developed which was designed to support the needs of the various players. The framework thus adopts and reconciles concepts from different existing frameworks in preference to defining new ones.

Key aspects of the framework were validated through the design and implementation of industry compliant trial subsystems (fulfilment, assurance and billing). The evaluation of these trial systems against a predefined set of evaluation criteria enabled the project to validate and enhance the framework. In addition, the project validated the approach through reviews and feedback from members of the business units within the partner organizations.

The final Open Development Framework (ODF) presented in this deliverable is targeted at service applications designers and developers in an open service market. It supports the concept of software component reuse (based on the concept of building blocks) and focuses on supporting the design and development of reusable software components. The framework enhances the development lifecycle by integrating four key viewpoints (logical and technological architecture, development methodology and reusable elements).

1 Introduction

1.1 Purpose

The purpose of this document is to provide a specification of the FORM Open Development Framework. This framework supports system designers, implementers and integrators, as well as service application developers' needs. It mainly focuses on providing the means for reusable element development, and also addresses the needs of the various players in the service provisioning and development value chains.

1.2 Scope

This deliverable represents the final milestone in the development of the FORM Open Framework within the project lifecycle and constitutes the basis for the elaboration of the ODF public positioning document. It is a complete specification of the framework and its four key viewpoints (logical and technological architecture, development methodology and reusable elements).

1.3 Document Reading Guidelines

This document starts by providing an overview of the ODF, describing genesis and aims of the framework, then presenting the stakeholders, and the way they interact through value chains. It presents the structure used to separate concerns within the ODF, introducing its four major portions: The Logical Architecture, the Development Methodology, the Technology Selection Guidelines and the Reusable Elements.

Section 3 provides details of the Logical Architecture, defining its architectural principles, the benefits they impart to the ODF stakeholders and an overview of the architectural meta-model. It also provides a description of how elements of the meta-model may be applied by users conducting specific development roles within the stakeholders and describes the relationship between the Logical Architecture and other related architectures.

Section 4 provides a brief overview of the Development Methodology portion of the ODF and its relationship to the Logical Architecture. A full description of the Development Methodology is given in FORM deliverable D12 [formD12].

Section 5 describes the Technology Selection Guidelines portion of the ODF. It should be read as set of guidelines for a technology selection process in the FORM framework context.

Section 6 describes the Reusable Elements portion of the ODF, explaining the structure of these elements as developed for the IES application domain addressed in FORM. A full specification of some reusable elements developed in FORM is provided in FORM deliverable D11 [formD11].

Section 7 provides a description of how the ODF was applied in the FORM development trial, illustrated with examples taken from the project's development documentation.

Section 8 describes further details of the Logical Architecture, providing the detailed specification of the meta-model as well as laying out requirements for future revision of this model.

Section 9 provides conclusion on the definition and application of the ODF and is followed by references, acronyms and a glossary.

2 The FORM Open Development Framework

2.1 Genesis of an Open Development Framework

Early open management architectures, such as the IETF's Internet Management and the ITU's Telecommunication Management Network, did not address the modelling of software components, and modelling was limited to the definition of managed objects in languages such as SMI [rfc2578] and GDMO [x722] that were tailored to specific management protocols. One of the earliest attempts to encourage open component-oriented software in the telecommunications domain was Bellcore's OSCA/INA [osca]. This defined modelling concepts for separately deployable software components and the distributed computing services they would require, such as object location, event management, distributed transaction management, amongst others.

These concepts were taken further by the Telecommunication Information Networking Architecture Consortium (TINA-C) which consisted of many of the world major telecommunications operators. TINA took a modelling approach based on ISO's Open Distributed Processing (ODP) [x901] architecture applied to both control and management software. This provided a more co-ordinated approach to the software lifecycle, using separate models for enterprise (business) concerns, information aspects of designs, the computation aspect which identified components and their interfaces, engineering aspects related to the structure of the underlying distributed computing environment and the technology concerns related to use of specific protocol bindings. When applied to service control software, the modelling techniques used became more specialised, using formal modelling techniques with the aim of supporting automated service creation [lucidi]. However, it is pointed out [lodge] that many of these techniques rely on an existing, well defined and relatively narrow functional frameworks within which such formal modelling techniques are cost effective. In the management application domain, where the computing platform is necessarily more heterogeneous, TINA-C modelling concepts were taken by various R&D projects [kande] [lewis99a] and aligned with the emerging industry standard for the modelling of software systems, i.e. the Unified Modelling Language. The EU funded project FlowThru highlighted the need for common modelling and methodological guidance on the modelling of management components [lewis99b] as well as the need to support multiple component integration technologies [wade].

In parallel, the TeleManagement Forum (TM Forum) has moved to a component-oriented approach to management system development. This began with a set of requirements for generic telecommunication management components that was based on the OSCA/INA work and characterised such components as Building Blocks [gb909]. These requirements were later used to inform the TM Forum's New Generation Operating System Support (NGOSS) initiative, which attempted to promote a separation between technology specific and technology neutral architectural concerns, while also supporting contemporary software integration techniques such as workflow, message buses and multi-tiered architectures [tmf053].

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 telecommunications 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]).

2.2 ODF Aims

To help refine the overall goal of the ODF, a set of stakeholders has been defined which represent the main organisation types involved in developing, deploying and maintaining management systems from components and which therefore possess the requirements on any suitable framework. These stakeholders are:

- Standards Bodies, which produce the industry agreements that underpin interoperability and integration of separately sourced software components.
- Independent Software Vendors (ISVs), which produce and market software components.
- System Integrators, which produce management system constructed from separately sourced software components, including ones that are developed internally.
- Service Providers, which possess the business requirements for management systems and operate them.

Considering the needs of these stakeholders, the scope of the ODF is further refined by the following aims:

1. Support common mechanisms for the communication of products (both spec-ware and software) between these stakeholders
2. Ensure the processes for developing the products exchanged between stakeholders converge with industry best practise such as model-based development and the use of UML.
3. Support the management of products, such as specifications and software, once they have been made available by a stakeholder in order to facilitate their later reuse by others.
4. Encourage the separation of techniques for integrating between different technologies from those techniques for integrating between different models. This is key to supporting late binding between models and technologies.

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.

The ODF does not explicitly aim to support automatic model checking or code generation, but does aim to exploit the current capabilities of widespread modelling tools, e.g. Rational Rose. Additionally, it is not an aim to endorse any particular technology, though guidance is given in the issues related to technology selection.

2.3 ODF Stakeholders

The ODF is 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 architectural and modelling principles to be shared between Standards Bodies, Independent Software Vendors (ISVs), System Integrators and system customers, i.e. the Service Providers. 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 ODF. 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 interoperate 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. More detail of the interactions between stakeholders is given in section 3.3.

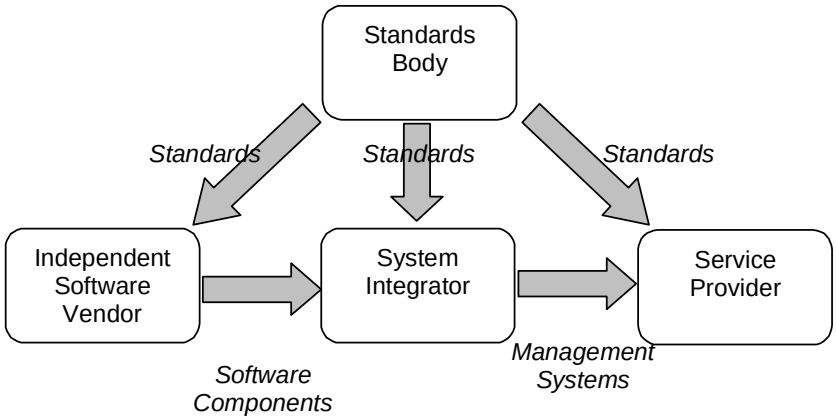

Figure 2-1: Stakeholder Model for FORM Open Development Framework

There may be occasions where organisations take on more than one role represented by these stakeholders, e.g. a large service provider may produce standards for procurement purposes and may have internal divisions producing management system or components. However, these stakeholders provide a relatively simple, but comprehensive, model for the market interactions that the ODF must address. The primary interactions addressed by the ODF are:

- The consumption of standard specifications by Service Providers for procurement, by System Integrators and ISVs for software development and by other Standards Bodies.
- The consumption of software components from one or more ISVs, as well as those developed internally, for management systems development by System Integrators.
- The procurement by a Service Provider from a System Integrator of a management system that must integrate with management systems from other sources, already present in the Service Provider's OSS and those of other Service Providers.

The ODF takes a model-based approach to software development and places an emphasis on developing a limited number of common models that can usefully be used and exchanged by the ODF's stakeholders. 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.

2.4 Overall Structure of ODF

As depicted in the Figure 2-2 the ODF is structured into four portions: a Logical Architecture, a Development Methodology, a Technology Selection Guidelines and a Set of Reusable Elements. This structure follows a pattern observable in other management related frameworks, e.g. ITU-T's Telecommunications Management Network architecture, the OMG's Open Management Architecture etc.

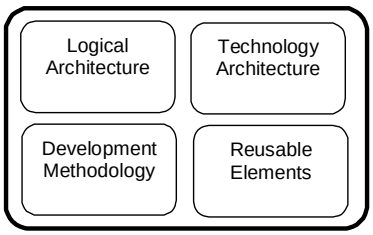

Figure 2-2: Structure of the Open Development Framework

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

- **Logical Architecture:** The Logical Architecture describes the structural concepts of the ODF and their relationships. This is described in terms of a meta-model that is used to ensure consistency of the models generated when applying the guidelines present in the rest of the ODF. 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 via which BBs interact with their environment. Management systems are built primarily from assemblies of BBs. Systems can be modelled at a business level in terms of Business Processes and Business Roles. Reference Points exist between Business Roles and are realised through Contract implementations. To promote their reuse, contract may be described in a technology neutral format which can be transformed to one or more technology specific versions for implementation in BBs. Contract definitions also include specifications of the information passed via the Contract by reference to an explicit information model.
- **Development Methodology:** The Development Methodology describes the processes and notations needed to design Contracts, develop Building Blocks and assemble Management Systems which conform to the ODF. The primary modelling notation used is UML, though XML is used for packaging UML with other model formats in Contract and BB specifications. The methodology integrates 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 partially used as a 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 are mostly assembled from of Building Blocks.
- **Technology Selection Guidelines:** The Technology Selection Guidelines addresses the different criteria at stake while selecting a technology or set of technologies to implement Management System BBs. It diverges from the technology related portion of most other frameworks in that it does not attempt to promote the use of a specific technology or integration technique. Issues related to the selection criteria include functional and non-functional requirements, as well as organizational context issues. Due to the nature and scope of the project this section aims at raising awareness on the importance given to these criteria but does not provide a catalog of patterns for implementing functional or non-functional requirements.
- **Reusable Elements:** This portion of the ODF is the repository for reusable products that result when the ODF 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 Logical Architecture is outlined in Section 3 of this deliverable, with details of its Architectural Model given in Section 8. An overview of the relationship with the Development Methodology is given in section 4, but for detailed guidance on using the Development Methodology the reader is referred to deliverable D12 [formD12]. Aspects of the Technology Selection Guidelines that have been addressed in FORM are presented in section 5.

2.5 Specialisation of ODF

The ODF is intended to be generic and extensible. The ODF has a core generic part, which in FORM is extended with others concerns related to the IES Management problem domain, to form the IES Management Framework. This generic part consists of the Logical Architecture and the Development Methodology. It is expected that other users of the ODF will extend the generic part to provide development frameworks for other domains, e.g. optical network management or mobile service management, and may reuse some of the parts specific to the IES Management Framework.

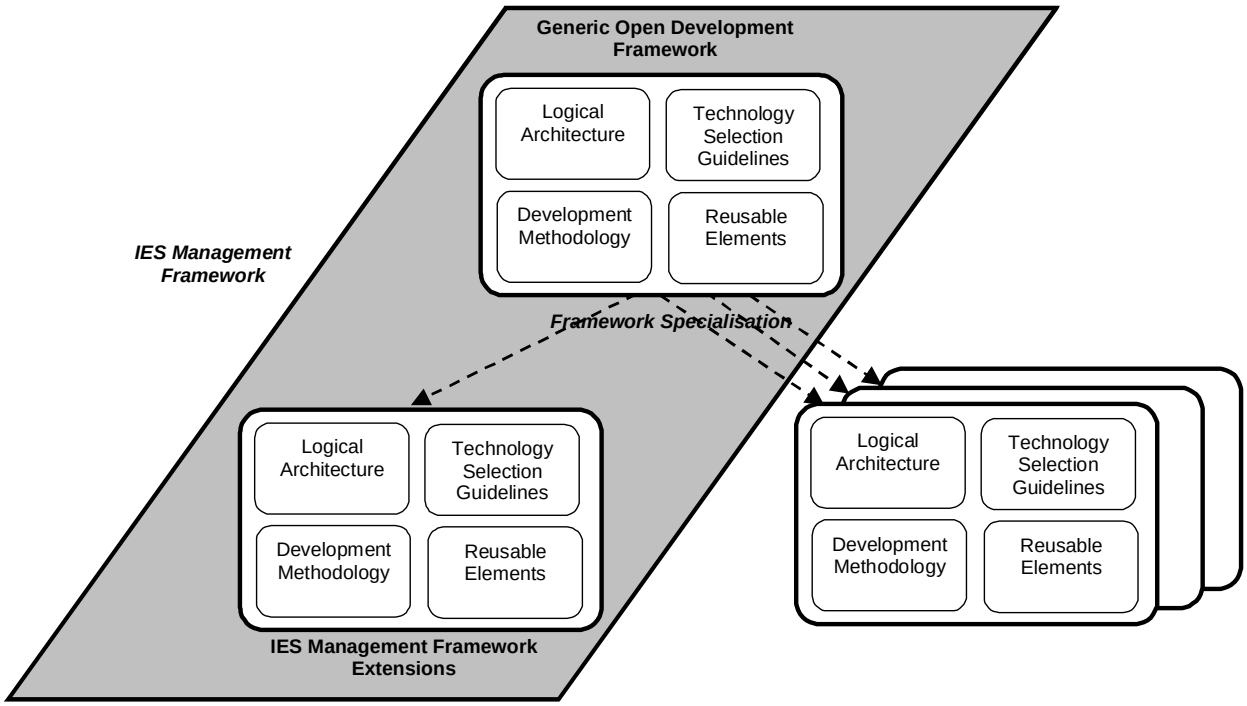

Figure 2-3: Specialisation of the Open Development Framework

The IES Management Framework is characterised by the definition of a service provision chain supporting the provision of an IES. This service provision chain requires different service providers to collaborate through seamless service management in an environment where market structure, network capabilities and customer requirements are changing rapidly. Enterprises must be able to create, reconfigure and dissolve business collaborations at an accelerating pace. The IES scenario examined in FORM addresses how an IES Provider may manage the dynamic relationships and the quality of service of electronic business interactions between groups of enterprises, the Application Service Providers (ASPs) they use and the Internet Service Providers (ISPs) providing communications infrastructure.

Figure 2-4 depicts the overall stakeholder model for the IES Management Framework. The system development value chain represents the more generic set of stakeholders that drive the core set of generic architectural concepts in the ODF. The architectural concepts of the ODF could be applied to any service provision value chain where component-oriented management systems were required. The service provision value chain represents the more specific IES related stakeholders used to exercise the ODF in FORM.

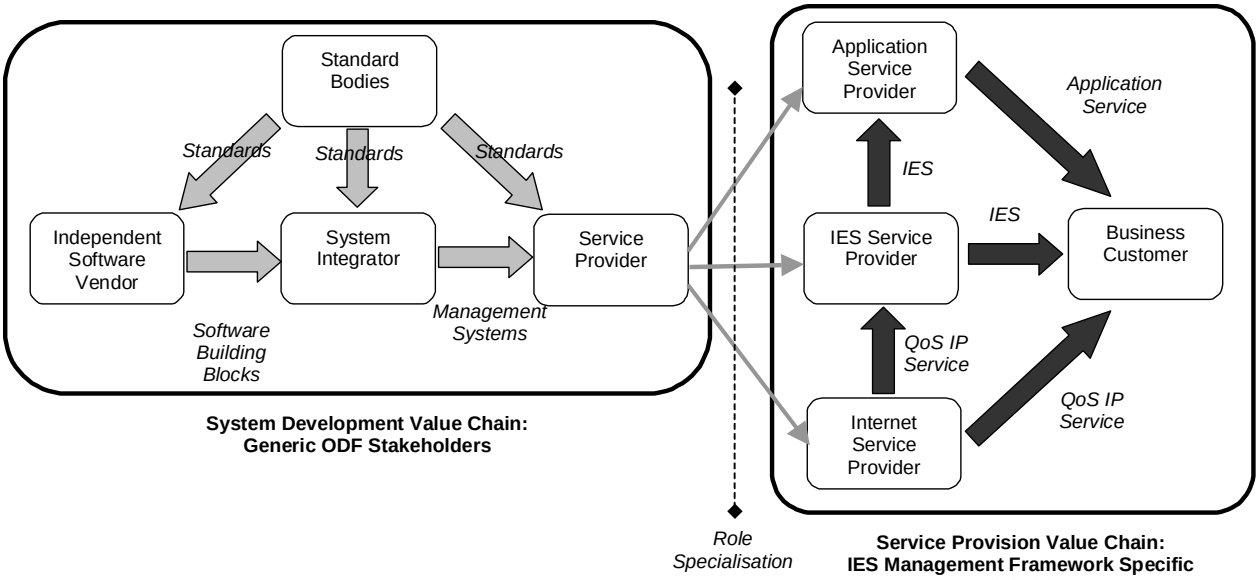

Figure 2-4: Stakeholder Model for the IES Management Framework specialisation of the Generic ODF

3 Overview of 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 consists of the following prescriptive aspects:

- A description of the *architectural principles* upon which the Logical Architecture is based, together with the benefits these principles offer to the ODF stakeholders. This is presented in section 3.1.
- A *meta-model* describing the main structural elements of the Logical Architecture and how they relate to each other - this is referred to as the Architectural Model. An overview is presented here in section 3.2 and presented in more detail in section 8.

The Logical Architecture also contains a descriptive element, addressed in section 3.3 which is included to explain how the elements of the Architectural Model should be applied to the needs of the ODF stakeholders. This is described by defining a set of Framework User Roles representing potential users of the ODF, which are active within stakeholder organisations at different points in the lifecycle of its main structural elements. This approach illuminates a key feature of a component-based architecture, which is that it addresses a broad range of the software development lifecycle and therefore brings together complex, interacting concerns that occur in the overall software lifecycle.

3.1 Architectural Principles

The Logical Architecture is structured around a set of architectural principles. These principles are heavily influenced by the architectural principles defined for the Technology Neutral Architecture of the TM Forum's NGOSS initiative. However, in comparison to the NGOSS principles, the principles described here place more emphasis on the development and exchange of models and less on the software integration features required. The principles start with the concept of a Building Block, which is used in FORM to represent a reusable software component.

P1: Management Systems are software systems that perform some tasks related to communications or systems management in an operational environment. They are constructed partially or fully from Building Blocks (BB).

Benefits:

- System Integrators are able to reuse the same BB in different Management Systems.
- System Integrators are able to potentially source BBs from multiple ISVs thus stimulating competition in the BB market and reducing costs.

P2: Building Blocks are pieces of software that are atomic units of deployment (one can be replaced in a running system without requiring other BBs to be replaced or modified).

Benefits:

- System Integrators may more smoothly upgrade individual pieces of software with less impact on the overall system.
- Service Providers suffer less system downtime due to software upgrades.

P3: Building Blocks are atomic units of system management.

Benefits:

- System Integrators may define common system management interfaces at the granularity of a BB, which in turn may stimulate the development of a market in system management applications that can exploit such common system management interfaces.

- Service Providers may gain more comprehensive system management capabilities, either packaged with Management Systems bought from System Integrators, or, if common system management interfaces are supported, through third party system management applications.

P4: Building Blocks may support multiple interface types termed Contracts.

Benefits:

- ISVs and System Integrators are both able to hide details of a BB's internal design and to changes to its implementation since only the Contracts are visible to BB users. This help protects the ISVs intellectual property and helps in the management of software upgrades. For System Integrators this hides details of BB implementation from reuser of BBs in a Systems Integrator's organisation.
- ISVs and System Integrators are both able to use multiple interface types to allow separate BB user views to be given separate Contracts, which can then have different access rights. This supports courser BB granularity.

P5: A Contract may support multiple business operations.

Benefits:

- Standards Bodies, ISVs and System Integrators are all able to group related functions into a unit of specification and documentation release.

P6: The Logical Architecture does *not* prescribe the technology to be used in implementing Building Blocks or their Contracts.

Benefits:

- ISVs are not constrained in the technology they use to implement BBs and can better match this to changing market demands.
- Similar constraints are also avoided by System Integrators, which as a result can better match this to the changing needs of Service Providers.
- System Integrators are able to encompass multiple technologies in a single system, but in turn face the problem of needing to address interworking between different Contract technologies. This, however, is seen as an inevitable aspect of Management System development.

P7: Contracts may be defined in a technology neutral form or a technology specific form.

Benefits:

- Standards Bodies can mediate industry agreements on Contract in a technology neutral form without necessarily excluding specific target technologies, thus potentially making the specification more long-live in the face of rapid technology churn.
- ISVs can use the technology neutral form to support the same Contract specification for customers across a range of technologies and over changes in popular technologies.
- Similarly, System Integrators can use the technology neutral form to support the same Contract specification for reuse within its organisation.
- System Integrators are able to compare Contract specification presented in the technology neutral form without being forced into an early, potentially excluding, technology decision and with the knowledge that any implementations obtained may offer stable functionality over technology changes.

P8: A BB implements a Contract in a technology specific form. When mapped from a Contract specification in a technology neutral form, this must be performed using an explicitly described transform.

Benefits:

- ISVs may use the explicit transform between the technology neutral and technology specific forms to allow different technologies (or even different profiles of the same technology) to be used to implement the same Contract to match market requirements.
- Similarly, System Integrators may use such explicit transforms to implement the same Contract to match the different integration technology requirements of different systems.
- System Integrators may use knowledge of the technology transforms used to implement different Contracts in managing the separation of technology interworking from model interworking in any integration solutions used.
- Standards Bodies may specify Contracts that may remain stable over changes in interaction technologies.
- Standards Bodies may be able to mediate agreements on sets of transforms from Contracts specified in a technology neutral form to ones in a range of technology specific forms

P9: Different Contracts may support different interface definition paradigms, though one Contract specification can only support one such paradigm. Interface definition paradigms include, but are not limited to, model-centric, operation-centric and message-centric. Different paradigms are typically suited to specific ranges of technologies.

Benefits:

- Standards Bodies, ISVs and System Integrators are all able to design Contracts that take advantage of the features of a specific interface definition paradigm while still being independent of individual interaction technologies that implement that paradigm. This may help tailor and therefore target a Contract specification at particular range of integration techniques required by various market sectors and, in the System Integrators case, by specific target systems.

P10: The definition of the information that is passed via a Contract should be published separately to the Contract specification.

Benefits:

- Standards Bodies are able to encourage commonality in Contract specifications by both using and publishing industry agreements on information that may be passed via a Contract.
- ISVs and System Integrators are both able to encourage better interworking between the BBs it produces by separately publishing and reusing the information passed over Contracts it designs.
- ISVs and System Integrators are both able to encourage better interworking between the BBs it produces and BB from other sources, by reusing information definitions published by other BB developers or, in particular, by Standards Bodies.
- System Integrators may be able to make quicker, more accurate selections of third party Contract implementations by comparing their separately published information content to information flow requirements identified in systems analyses.

P11: Functionally related Building Blocks can be grouped together for the purpose of software release.

Benefits:

- ISVs gain a level of release that retains the benefit of individual BBs as units of deployment, but which is better matched to the needs of sales and marketing, where a coarser level of functionality than that represented by a BB may be required.

P12: Building Blocks must be released with documentation describing: the related business context in which the BB is intended to operate and the analysis of this context that led to the identification of the BB and the design of the Contracts it uses.

Benefits:

- System Integrators are able to select a BB against a systems business requirements and systems analysis.

P13: The behaviour of a Contract and interactions with the behaviour of other Contracts on the same BB may be modified at deployment- or run-time. Where this feature is offered it should use explicitly defined business-rules.

Benefits:

- ISVs may build user-controlled flexibility into a BB, thus enabling them to address a wider range of customer needs.
- System Integrators may build more flexible behaviour into a BB, thus increasing the opportunity to reuse the same BB implementation in a number of different systems
- System Integrators are more easily able to reconfigure the system they produce to meet changing requirements.
- Service Providers are able to perform some reconfiguration of running systems to meet changing operational requirements.

These broad principles have been used to guide the structuring of the Architectural Model described in the next section.

3.2 Overview of Architectural Model

The Architectural Model is a meta-model defining the structure of and relationships between the different types of models produced and used during the application of the ODF. The main elements of the meta-model describe models at the granularity in which they might typically be generated and exchanged by the ODF stakeholders. These models are:

- **Business Context Model:** This captures the requirements for an area of concern, models its organisational environment and defines a business-level model of its externally observable functionality and internal business processes.
- **Domain Model:** This analyses an area of concern, defining a system analysis level model of its externally observable functionality, process behaviour and logical decomposition
- **Contract Set Specification:** This is a set of interface specifications that may be utilised in designing BB software. It provides the primary means of interoperability in the Architectural Model by detailing sets of interfaces that can be used for interacting with BBs.
- **External Information Model:** This is a technology neutral expression of the information that is potentially passed via a range of Contracts. This provides an important aid to ensuring interoperability between Contracts defined in different technologies
- **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, some portion of which uses Building Blocks.

Though these models are described separately, the building of reusable software and the management systems that use them relies on managing the 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 possible 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 artefacts defined in the Development Methodology portion of the ODF.

Figure 3-1 gives an overview of the meta-model relationships between the primary parts of the Architectural Model, shown as a UML class diagram in order to show the associations and cardinalities between the parts of the meta-model.

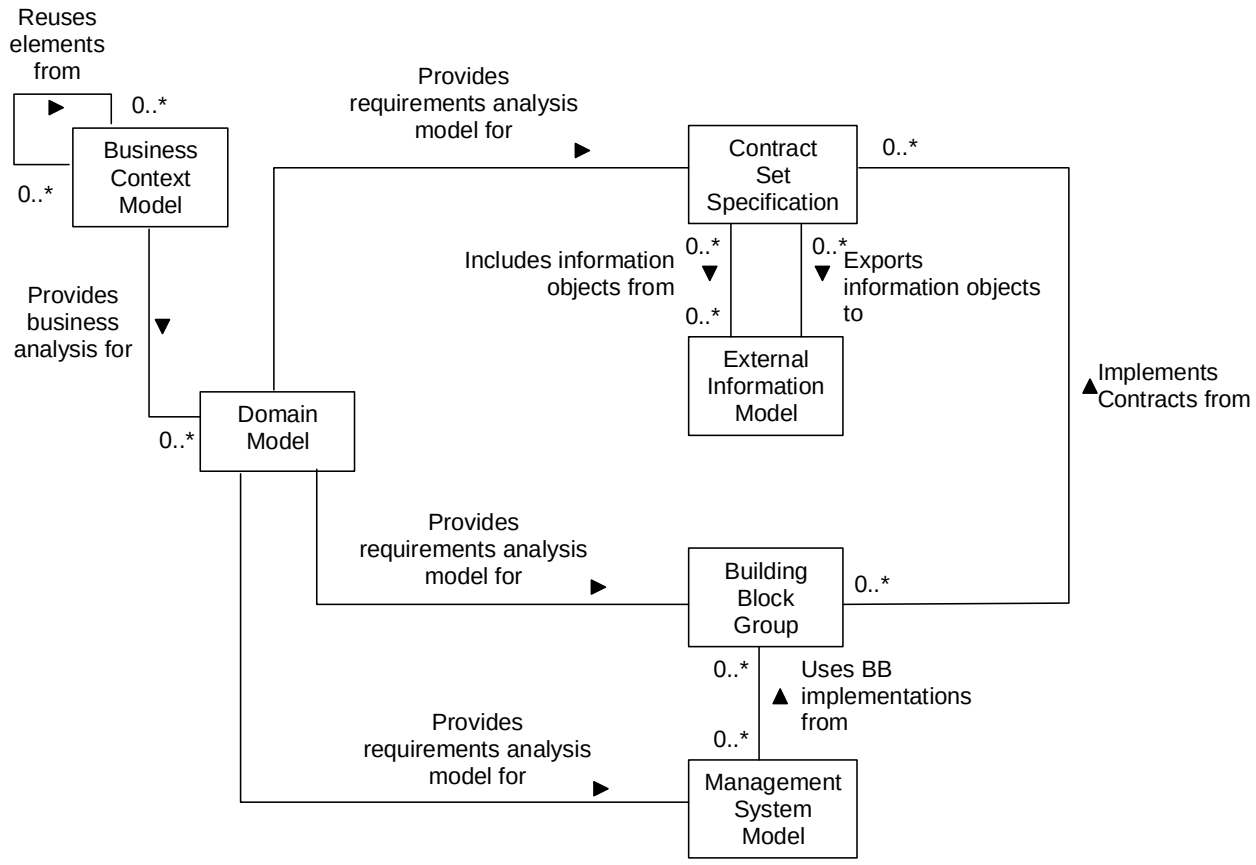

Figure 3-1: Relationships between primary models in the Logical Architecture

The following subsections provide an overview of the reasons why the individual models are included in the architectural model, their structure and their relationships. Section 3.3 provides a description of how these models address the needs of the ODF stakeholders, while section 8 provides more details of the prescriptive aspects of the meta-model.

3.2.1 Business Context Model

The Business Context Model (BCM) captures the requirements upon a domain of interest and models aspects of the business environment in which that domain exists. The domain for which a BCM is generated may be of interest for developing a Management System, developing some Building Blocks, specifying some Contracts or a combination of these. Alternatively, if the domain is broad, a BCM can be generated for the purpose of identifying sub-domains that are then subject to more detailed requirements capture in one or more further BCMs. The BCM is included in the architectural model to facilitate a common understanding and comparison of the requirements and business environment for products produced by separate organisations.

The structure of the BCM is depicted in Figure 3-2 as a UML component diagram.

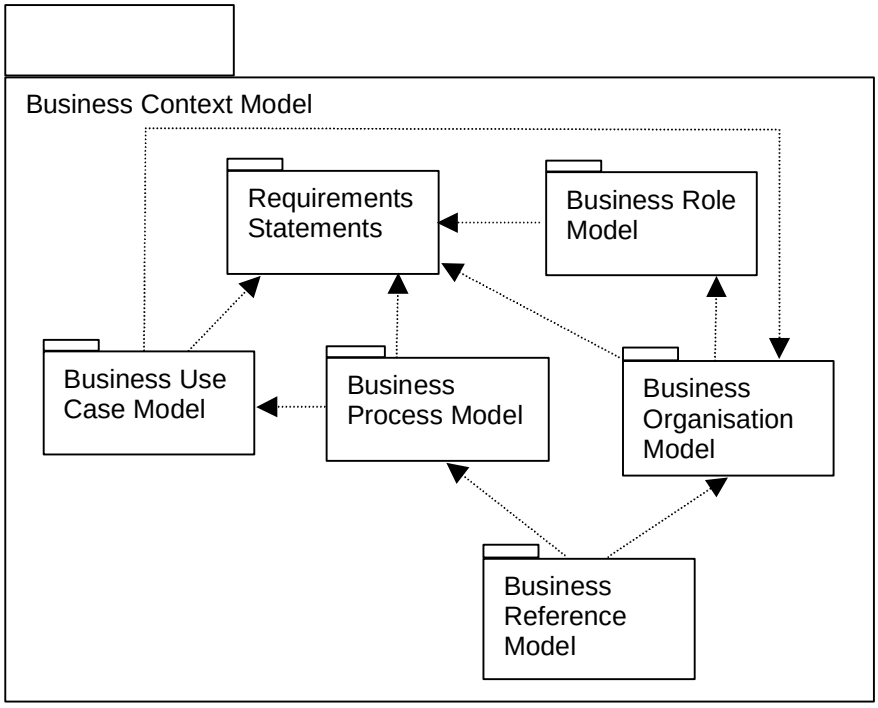

Figure 3-2: Structure of Business Context Model

The constituent models of the BCM are:

- Requirements Statements: a categorised collection of the requirements imposed on the domain of interest.
- Business Role Model: a model of Business Roles representing the different types of organisations present in the domain of interest and its business environment, and Reference Points representing the sets of interactions that may operate between pairs of Business Roles. The Business Role and Reference Points reference any Requirements Statements which contribute to their identification.
- Business Organisation Model: a model of a general business scenario that captures the typical organisations and users that characterise the domain of interest and its environment. Organisations in this model enact one or more Business Roles from the Business Role Model. The Model references any Requirements Statements from which the assumptions underlying the scenario are derived.

- Business Use Case Model: an expression of the functionality observed of the domain of interest by a set of Business Actors, which may include organisations and users from the Business Organisation Model that are external to the domain of interest. The Use Cases reference any Requirements Statements that they address.
- Business Process Model: an expression of the Business Processes that may operate in the domain of interest. Business Processes reference any Use Cases or Requirements Statements from which they are derived.
- Business Reference Model: a mapping between organisations in the Business Organisation Model and the Business Processes operated by those organisations.

3.2.2 Domain Model

The Domain Model (DM) expresses an analysis of the requirements and business environment captured in a single BCM. It therefore provides a more detailed, system-level expression of the environment with which a domain of interest must interact, the functionality it must exhibit to that environment and a breakdown of the logical structure within the domain. A DM provides a requirements analysis for one Contract Set Specification, one Building Block Group, one Management System, or a combination of these. The DM is included in the Architectural Model to provide a common mechanism for tracing between specific elements of a Contract Set Specification, a Building Block Group or a Management System Model and the Business Context Model from which they are derived.

The structure of the Domain Model and its relationship to the Business Context Model is depicted in Figure 3-3.

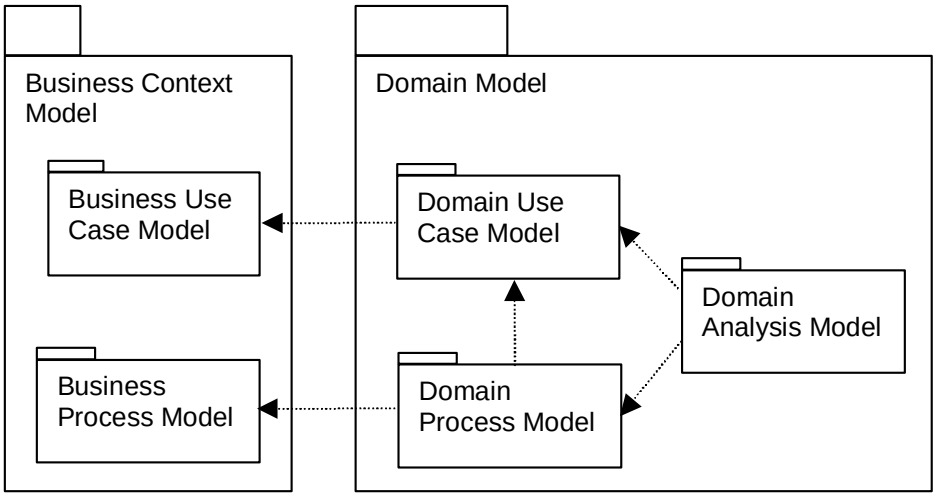

Figure 3-3: Structure of Domain Model

The Domain Model consists of a Domain Use Case Model, a Domain Process Model and a Domain Analysis Model. The Domain Use Case model describes the functionality of the domain as seen by a set of Domain Actors which represent the system-level entities (both human and automated) with which the domain must interact. The Domain use cases refer to Business Use Cases from which they are derived. The Domain Process Model is a refinement of the Business Process Model that reflects the system level concerns of the Domain Use Case Model. The Domain Analysis Model is derived from the Domain Use Case Model and the Domain Process Model. It uses the model-view-controller design pattern to provide static and dynamic models of the domain's logical structure.

3.2.3 Contract Set Specification

The Contract Set Specification (CSS) expresses one or more related interface specifications, termed Contract Specifications that may be used in the implementation of Contracts for Building Blocks. The CSS is included in the Architectural Model to allow Contract Specifications to be generated and published separately to the development of Building Blocks. This encourages the reuse of individual Contract Specifications across separately developed Building Block implementations, thus promoting the quality of interface specifications, agreements on common Contract Specifications and ease of Contract interoperability.

The structure of a CSS model and its relationship to other models is show in Figure 3-4.

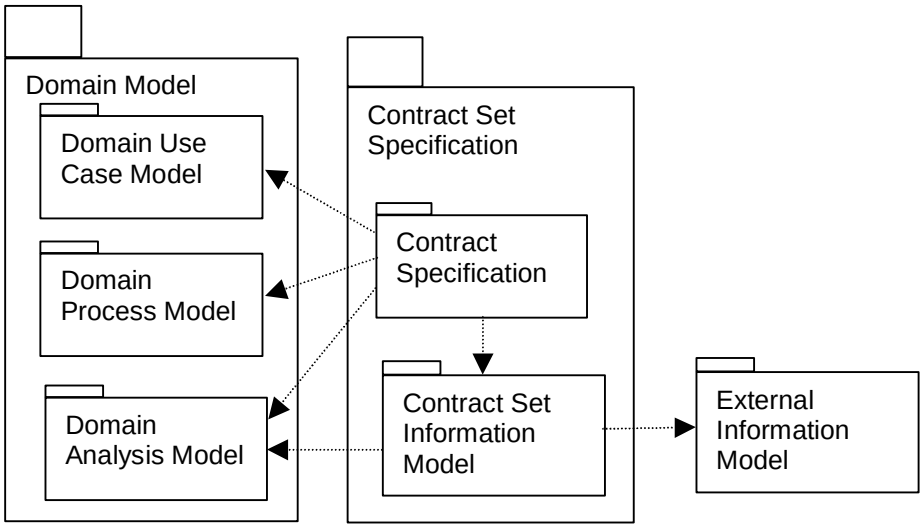

Figure 3-4: Structure and relationships of Contract Set Specification model

A CS contains a number of Contract Specifications and a Contract Set Information Model (CSIM). The Contract Specifications are derived from the elements of the Domain Model. The Contract Set Information Model is the aggregation of the models of information passed via the individual Contract Specifications. The CSIM may reference information objects in an External Information Model and information elements of the relevant Domain Analysis Model.

3.2.4 External information Model

The External Information Model (EIM) expresses information that may be passed via Contracts, but documented in a manner that is separate from the relevant Contract Specifications. The EIM is included in the Architectural Model to encourage the reuse of the information specifications it contains across separate Contract designs. This is performed by the extraction of the information content of Contract Specifications, in a simple common format and the progressive use of this information in consolidation exercises by individual organisations to generate EIMs. As EIMs stabilise, the information content of individual Contract Specifications should increasingly consist of references to EIM elements. EIMs support the following:

- The comparison of EIM elements with information requirements generated, for example, from business process information flows, allows early identification of Contract Specifications as possible candidates for addressing control flow requirements. This is part of the selection process for Building Blocks (or Contract Specifications) and must be coupled with analysis of the Contract's BCM.

- Contract Specification standardisation and BB releases by different organisations may establish a growing, easily accessible body of accompanying 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 may be 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.

The introduction of the EIM therefore potentially improves the similarity between the models of information used in separately developed Contracts and thus eases any transformation required for future interoperability. Additionally, EIMs may be used to locate Contract Specifications relevant to a particular problem domain by comparison to the domain's information requirements presented in the relevant BCM and DM.

The information objects in the EIM may be exported from specific Contract Sets and may be referenced by other Contract Sets.

3.2.5 Building Block Group

The Building Block Group (BBG) expresses for each of one or more Building Blocks; the Contracts supported by the BB, other Contracts upon which the BB relies, the externally visible behaviour of the BB and the means for expressing runtime changes to the this behaviour. It also includes the deployable software for each BB. The BBG is included in the Architectural Model to provide a means for conveying the usage of the constituent BBs for the purposes of reusing them in developing a Management System. The BBG is used for this, rather than individual BBs, in order to support the needs of packaging software for sale between ISVs and System Integrators. The BBG also supports better the situations where there are functional dependencies between BB that must be maintained.

The structure of a Building Block Group model and its relationship to other models is show in Figure 3-5.

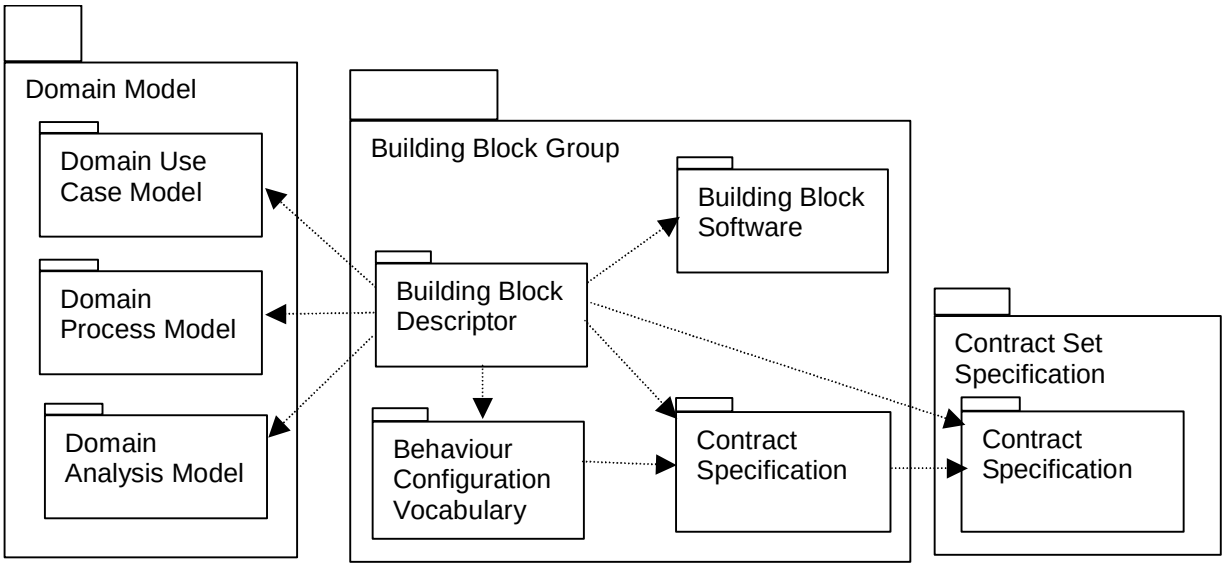

Figure 3-5: Structure and Relationships of Building Block Group model

Within a BBG, each BB is represented by a Building Block Descriptor. The Building Block Descriptor provides information needed to deploy the BB such as platform dependencies and run-time behaviour modification capabilities. It references the actual BB software and the Contract Specifications that the BB supports, which in turn may reference a Contract Specification from a CSS. The Descriptor may also reference other CSSs for Contract Specifications that are used by the BB. The Building Block Descriptor also references the Behaviour Configuration Vocabulary, which is a set of interpretable language elements that can be passed to the BB at runtime in order to modify its externally visible behaviour. The Building Block Descriptor, therefore, captures the aspects of the BB design that are visible to its users, and as such also references elements of the relevant Domain Model that drives that design.

3.2.6 Management System Model

The Management System Model (MSM) expresses the design and implementation of a software system performing some operational support activities. A MSM is the result of a single development project by a System Integrator. An OSS, therefore, typically consists of several Management Systems developed at different times. The MSM is included in the Architectural Model in order to model the use of BBs in different application domains and to assist in the design and implementation of non-BB software needed to satisfy the domain's requirements.

The structure of the MSM and its relationship to other models in Figure 3-6.

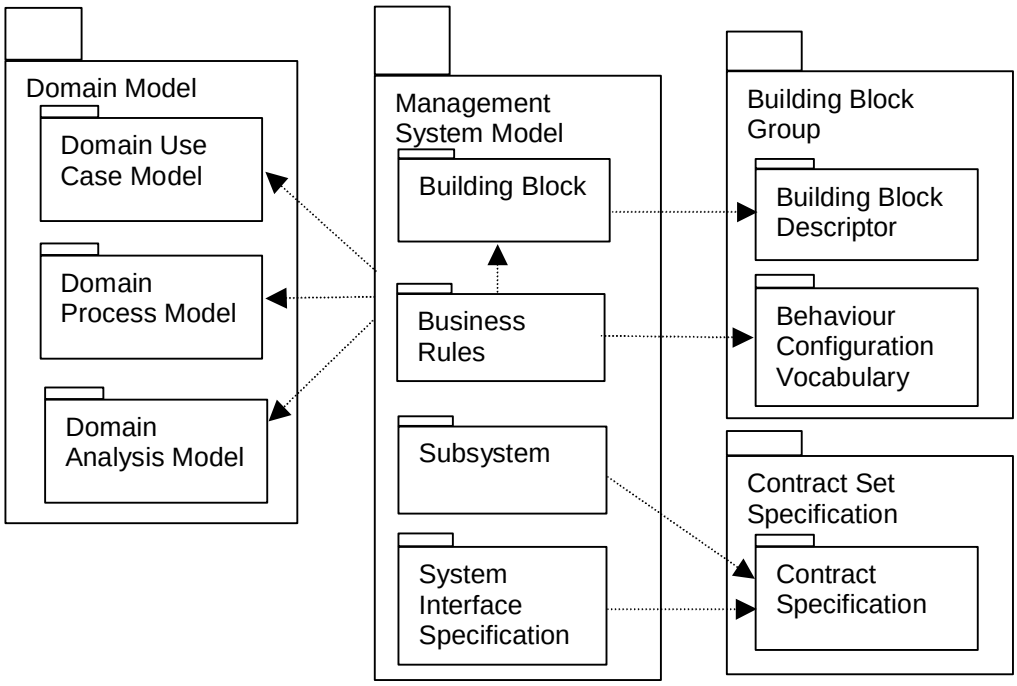

Figure 3-6: Structure and relationships for the Management System Model

The overall structure of the MSM is derived from a single DM. Its constituent BBs come from BBGs that are referenced by the MSM. The behaviour of Building Blocks, in a particular Management System deployment, is defined by the Business Rules expressed in the MSM. These Business Rules use language elements from Behaviour Configuration Vocabularies in relevant BBGs. The MSM also supports Subsystems, which are software elements not composed of BBs, i.e. where suitable BBs are not available or not viable to develop for the required functionality. Subsystems may support and use existing contract Specifications, even though lack of other features, e.g.: distribution or management unity, preclude them from being characterised as a BB. The MSM must also have a complete set of System Interface Specifications that define the interfaces via which the System interacts with its environment, which may include legacy systems and other Service Providers' systems. Some System Interface Specifications may be mapped to existing CSSs, either ones implemented by BBs in the System or ones representing existing or future BBs which form part of the System's environment.

3.3 Usage of Architectural Model

To support the description of how the Architectural Model delivers the anticipated benefits to the ODF stakeholder, this section defines a set of Framework User Roles and describes how they use and exchange the main elements of the Model defined in the previous sections.. As with any component-based architecture, e.g. J2EE, 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 directly visible at another stage, e.g. software deployment. Therefore these User Roles are introduced to allow readers of this document to understand the relevance of the various parts of the Architectural Model to their particular sphere of interest.

The Framework User Roles are:

- **Requirements Analyst:** Collects business requirements from requirements holders and analyses them to produce Business Context Models.
- **Domain Analyst:** Produces Domain Models addressing a scoped domain of management functionality.
- **Contract Designer:** Produces Contract Specifications and External Information Models.
- **Building Block Developer:** Produces software Building Blocks.
- **System Builder:** Produces and deploys Management Systems and their associated models.
- **System Administrator:** Monitors and manages a Management System in order to ensure it operates within required operational parameters.

This section defines elements of the Architectural Model in which each of these Roles has an interest and how these elements 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 Architectural Model. Real-life developers may therefore take on several of these abstract Roles. The ODF Development Methodology's guidelines are based on the needs of such real-life developers.

To fully understand the relationship between the User Roles and the elements of the Architectural Model, a clear picture is needed of the relationships between the ODF stakeholders on behalf of which the User Roles are conducted. Figure 3-7 provides a UML class diagram representing these relationships.

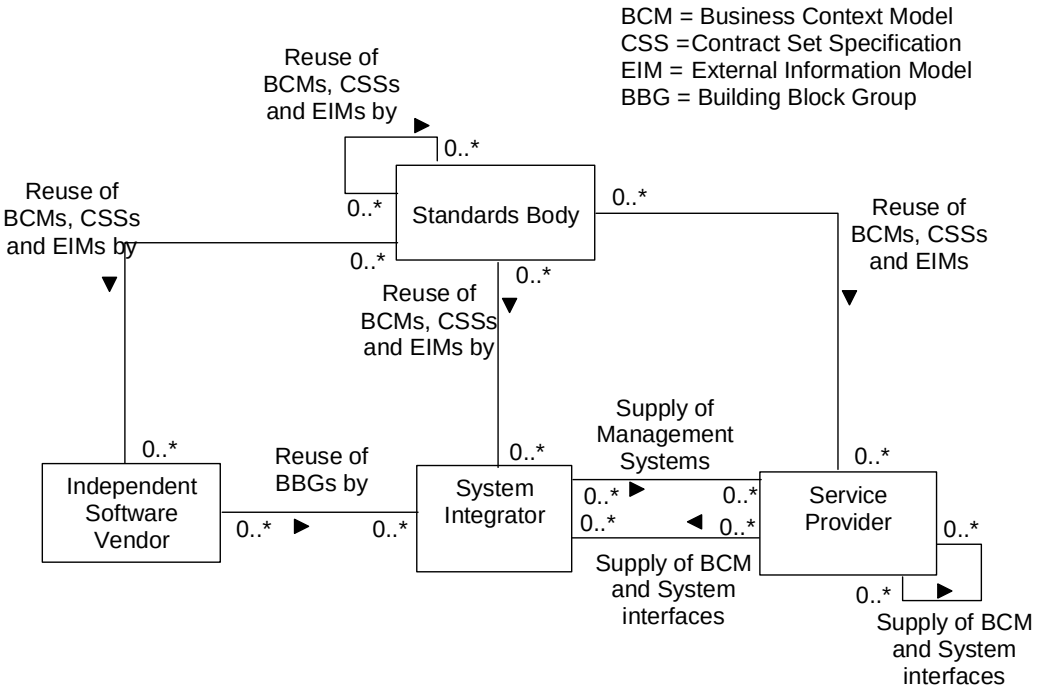

Figure 3-7: Model of relationships between ODF stakeholders

Over time, Service Providers will be supplied with Management Systems from a number of different System Integrators, as value for money and a specific range of capabilities are sought. Service Providers will need to provide Business Requirements (in the form of a BCM) and any specific System Interface Specifications to the relevant System Integrator in order to do this. The Service Provider may also opt to express its business requirements in terms derived from BCMs available from Standards Bodies in order to improve the understanding of the requirements and to facilitate matches to existing solutions derived from the same BCMs. System Interface Specifications may also be taken from ones available from Standards Bodies, which may be expressed as Contracts or with reference to EIMs, in order to improve the stability of the System within a changing OSS environment. A Service Provider will need to exchange Business requirements and System Interface Specifications with other Service Providers when assembling the requirements for a Management System that interacts with those of other Service Providers.

System Integrators will aim to provide Management Systems to a number of Service Providers in order to maximise sales. In doing so they will make use of BBs obtained from a number of ISVs, in order to obtain the required functionality at optimal price and quality. System Integrators may attempt to align requirements supplied from Service Providers procuring Management Systems with BCMs made available by Standards Bodies. This assists in aligning possible design solutions with corresponding Contract Specifications and EIMs from Standards Bodies and thus increases the chances of finding a match with BBs from ISVs that implement those standards.

ISVs will aim to provide BBGs to as wide a range of System Integrators as possible in order to maximise sales. ISVs may also attempt to make use of BCMs made available by Standards Bodies in the development of new BBGs in order that they provide good matches to standardised Contracts and EIMs and thus encourage their reuse.

Standards Bodies may use more general BCMs from other Standards Bodies to provide grounding for a more focussed standardisation effort. Equally, Standards Bodies may wish to use elements of Contract specifications and EIMs from other Standards Bodies, where appropriate to their domain of interest, in order to prevent proliferation of unnecessarily dissimilar models, and thus encouraging take-up of the models that are standardised.

The motivation for this is described in more detail in the following paragraphs by detailing how the User Roles active within each stakeholder generate, use and exchange the main elements of the Architectural Model. Each of the User Roles is played out by personnel in one or more of the ODF stakeholders. By having the same User Roles operating in more than one stakeholder, the commonality in the information passed between the stakeholders is more easily analysed and identified. Figure 3-8 depicts how the main elements of the Architectural Model are generated by the different User Roles working in the various stakeholders. The figure also shows how those User Roles may use Model elements generated by other User Roles (though the use of models referenced from other models is omitted for clarity).

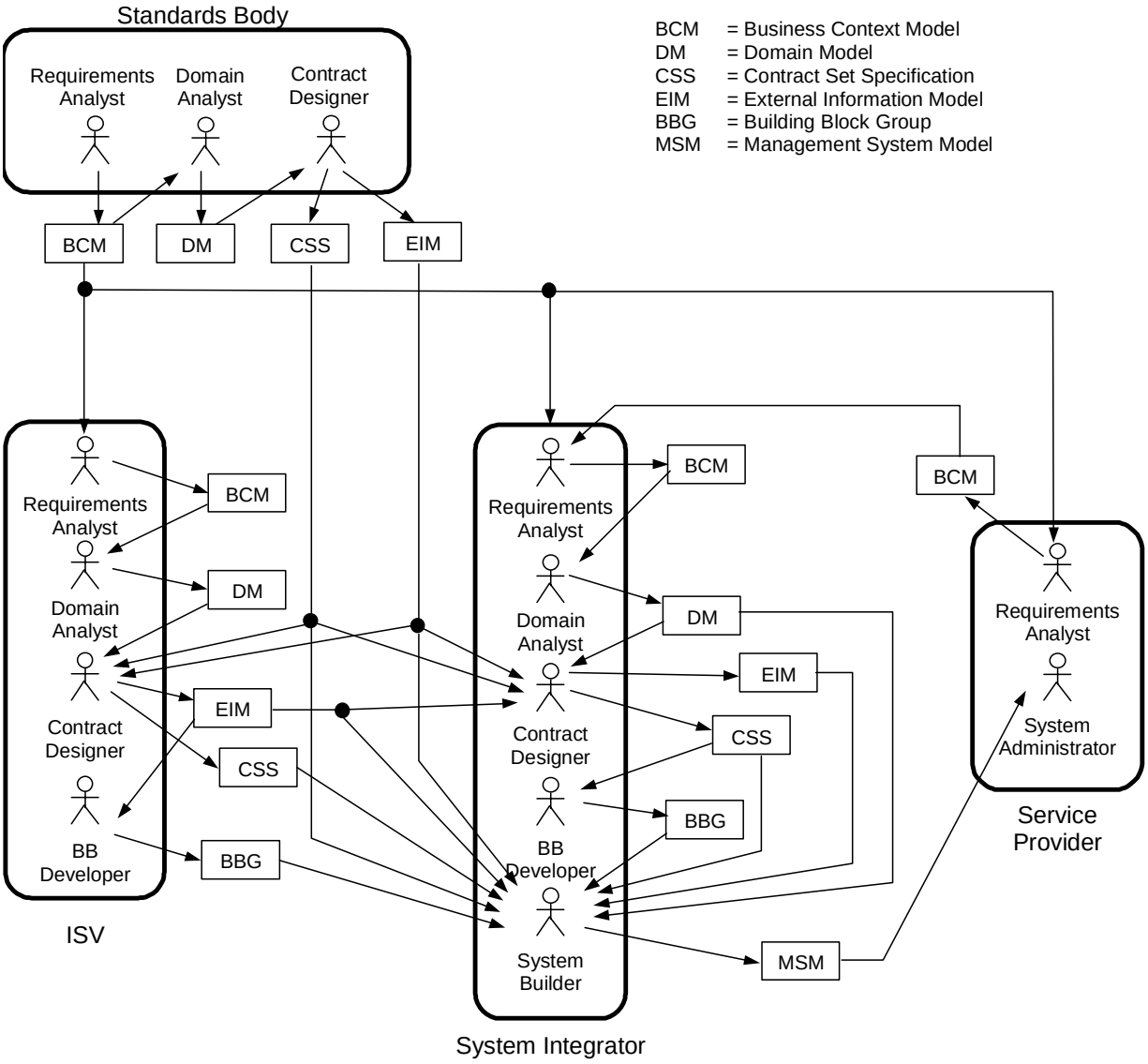

Figure 3-8: Exchange of main Architectural Model elements between User Roles

Within the Standards Body stakeholder the User Roles may make the following use of the main Architectural Model elements:

- The Requirements Analyst may use Business Context Models (BCM) from one or more other Standards Bodies when developing its own BCMs, in order to improve consistency between standards.

- The Domain Analyst Role uses a BCM generated by the Standards Body's Requirements Analyst in developing a Domain Model (DM) for a particular area where open interfaces are to be developed in the form of a Contract Set (CS).
- The Contract Designer uses a DM generated by the Standard Body's Domain Analyst in order to guide the design of a standardised Contract Set. It may use the traces available within the Domain Model to reference relevant parts of the BCM, where this may be needed to resolve design issues. The Contract Designer may reuse elements of Contract Sets and may reuse information elements from External Information Models (EIMs), where these are made available from its own or other Standards Bodies. Information elements developed for the new Contract Set may be published as part of an EIM.

Within the ISV stakeholder the User Roles may make the following use of the main Architectural Model elements:

- The Requirements Analyst may use BCMs from one or more Standards Bodies in developing its own BCM for a planned Building Block Group (BBG) in order exploit existing well-understood business model elements, which will aid in the later promotion of the BBG. The Requirements Analyst may maintain an overall BCM, from which BCMs for separate BBGs are derived, in order to maintain consistency of requirements across its product range.
- The Domain Analyst Role uses a BCM generated by the ISV's Requirements Analyst in developing a DM for a particular BBG.
- The Contract Designer uses a DM generated by the ISV's Domain Analyst in order to guide the design of a Contract Set Specification (CSS) for use in development of a BBG. It may use the traces available within the Domain Model to reference relevant parts of the BCM where this may be needed to resolve design issues. The Contract Designer may reuse elements of CSSs and may reuse information elements from EIMs, where these are made available from Standards Bodies or internally from previous Contract design work. Information elements developed for the new CSS may be published as part of an EIM to be used in the development of other BBGs, in order to encourage information consistency across its product range.
- The Building Block Developer uses one or more CSSs to provide the specifications of Building Blocks (BBs) in developing a BBG.

Within the System Integrator stakeholder the User Roles may make the following use of the main Architectural Model elements:

- The Requirements Analyst will use elements of a BCM from the Service Provider that is procuring the Management System as the driving requirements for the BCM for the System. The Requirements Analyst may also use BCMs from one or more Standards Bodies in developing Management System's BCM in order exploit existing, well-understood business model elements, which will aid in the later reuse of elements of the System, including internally developed BBs. The Requirements Analyst may maintain an overall BCM, from which BCMs for individual Management Systems are derived, in order to maintain consistency of requirements across its Management System product range and internally developed BBs.
- The Domain Analyst Role uses a BCM generated by the System Integrator's Requirements Analyst in developing a DM for a particular Management System. This DM may also form the DM for relevant BBs that may be developed internally for this System but which are deemed useful for reuse in future Management System development projects.

- The Contract Designer uses a DM generated by the System Integrator's Domain Analyst in order to guide the design of a CSS for use in development of any constituent BB developed for use in the Management System. It may use the traces available within the Domain Model to reference relevant part of the BCM, where this may be needed to resolve design issues. The Contract Designer may reuse elements of CSSs and may reuse information elements from EIMs made available from Standards Bodies, to encourage future reuse of the Contract Specifications within the System Integrator. Elements of CSSs and EIMs that resulted from previous internal BB development may also be reused in order to benefit from development and maintenance cost reductions across the range of Management Systems developed. Information elements developed for the new Contract Set may be published as part of an EIM to be used to aid the selection for reuse of the resulting BBs in future Management System development, in the development of other internal BBs and in order to encourage information consistency across its product range.
- The Building Block Developer uses one or more CSSs to provide the specifications of BBs in developing a BBG, which may then be used in future Management System development projects, in addition to its use in the Management System for which it was developed.
- The System Builder uses a DM generated by the System Integrator's Domain Analyst in order to guide the design of a Management System. It may use the traces available within the DM to reference relevant parts of the BCM where this may be needed to resolve design issues. The System Builder may reuse elements of CSSs and may reuse information elements from EIMs made available from Standards Bodies, to support interworking with the Management System. The System Builder will use EIMs and Contract Specification related to both internally developed BBs and BBs from ISVs in order to select those needed to implement the required functionality for the Management System.

Within the Service Provider stakeholder the User Roles may make the following use of the main Architectural Model elements:

- The Requirements Analyst may use BCMs from one or more Standards Bodies in developing Management System's requirements, in order exploit existing well-understood business model elements and thus ease the communication of requirements to the System Integrator.
- The System Administrator uses the Management System Model to configure and deploy the Management System.

The above description of how the User Role make use of the main elements of the Architectural Model should highlight how this single set of models can be used to meet the internal development requirements of the separate stakeholder as well as assisting the exchange of models between stakeholders needed to support an open market in management components.

3.4 Relationship to Existing Architectures

A basic design goal for the ODF 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 ODF are described in the following subsections.

3.4.1 OMG Model-Driven Architecture

The Object Management Group (OMG) has recently developed its Model-Driven Architecture (MDA) [ab/2001-02-01]. This builds on the Object Management Architecture, which provides a framework for CORBA standards, and encompasses opportunities for improved use of modelling techniques on the software engineering process offered by its standardisation of UML. The ODF places a similar emphasis on modelling at all stages of the development cycles for Contracts, BBs and Management Systems. 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 covered in the MDA.

The ODF 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 ODF needs to address the full range of interoperability technologies used in management systems, including message passing and management-agent paradigms, neither of which are well addressed in the current MDA roadmap.

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

3.4.2 TM Forum Telecoms Operations Map

The TM Forum's Telecoms Operation Map (TOM) provides a generic, high-level process model for telecommunications management. The ODF has adopted the notion of using business process modelling as part of business requirements modelling, however it does not advocate a single generic process model, but the generation of domain specific Business Process Models by competent organisations, e.g. the Business Process Model developed for the IES Framework. The most recent version of the TOM, termed the eTOM [gb921], also includes the definition of generic business roles and reference points between them, similar to the Business Role Model of the ODF's BCM. This can be mapped to business processes in a similar manner to the Business Reference Model supported in the BCM, as first suggested in [lewis99b]. The eTOM presents a specific model containing business roles and the reference points between them that may then be applied to any application of the TOM. In comparison, the ODF uses the Business Role Model to capture potential Business Role networks for specific application domains and does not propose universally applicable Business Roles, which may be rapidly out-dated in this fast moving sector.

3.4.3 TM Forum Generic Requirements for Telecommunications Management Building Blocks

The TeleManagement Forum has produced document, GB909, which 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 ODF. 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 architectural principles presented in section 3.1.

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 development trials. For instance, the ODF now 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 is encouraged by adopting a Model-View-Controller design pattern for expressing the Domain Analysis Model, in the ODF's Domain Model. This ensures that Contract designs derived from such Domain Analysis Models are encouraged to exhibit the functional separations essential to these three tiers, while allowing Contract Designers 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 ODF's Development Methodology. Other requirements that relate to computing platform services supporting trading, transaction and data stewardship are not being addressed directly in FORM.

3.4.4 TM Forum's New Generation Operating System Support

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 ODF. 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 (TNA) for NGOSS is available for membership comment [tmf053]. However, FORM has included several concepts from NGOSS into the ODF to varying extents, namely:

- The TNA identifies the use of a Shared Information Model to define common models for information to be transferred between systems with different internal information models and to guide the structure of information used in new NGOSS components performing information handling. A similar approach is taken in the ODF (see section 3.1, principle P10), however, the use of such common information modelling is restricted to the exchange and reuse of models between separately developed Contract Specifications via the medium of EIMs. Such models are not necessarily intended in the ODF to be used for structuring corporate data repositories, and related issues, such as data stewardship, are not addressed.
- The principle of specifying Contracts and Information Models in a technology neutral manner, with mappings being developed to specific interface technologies, is adopted in the ODF (principles P6-P9), though technology neutral Contract modelling is not mandatory
- The separation of component software functions from business logic, with the latter expressed in a form dictated at runtime is adopted in the ODF (principle P13). However, in NGOSS this focuses on the logic that drives the external invocation of Contracts, in the ODF 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.

The ODF differs from the NGOSS approach as follows:

- The Contract in the ODF 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 (see principle P5). 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 the ODF.
- The ODF does not address the specification of abstract Framework Services of which Building Block implementations make use, as addressed in NGOSS.

3.4.5 Distributed Management Taskforce's Common Information Model

The Common Information Model (CIM) meta-schema [cim] used by the Distributed Management Taskforce (DMTF), which can be used to express information models in XML, is a strong candidate for expressing the EIM. The CIM meta-schema is an object-oriented meta-model underlying the Managed Object Format language (MOF – not to be confused with the OMG's Meta Object Facility), which is used for defining the DMTF's information models. 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 meta-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 meta-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 meta-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, thus supporting its alignment with the use of UML in the ODF's Development Methodology.

However, in order to fully satisfy the requirements for information modelling in the ODF several modifications need to be made to the CIM Schema as discussed in section 8.3.

3.4.6 Telecommunication Information Network Architecture

Though the Telecommunication Information Networking Architecture Consortium (TINA-C) is no longer active, several of its architectural concepts are used in the ODF. Elements of the ODL language evaluated in TINA and now standardised by the ITU-T [itu-odl] have informed the ODF Building Block structure, in particular the inclusion of references to Contracts that are required by a Building Block.

In addition, the ODF has adopted the TINA business modelling concepts [mulder] of business roles and reference points between roles. The ODF maps Reference Points onto sets of Contract Specifications as suggested in TINA. Unlike TINA, the ODF 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 Development Methodology

4.1 Overview

The methodology tackles the twin challenges of designing reusable components and the construction of business processes driven management systems. The methodology recognises these two different 'development' workflows – namely component design, and systems construction using (existing) components. Typically, such development workflows is carried out by different types of organisations or roles within an organisation e.g. component developer and systems integrator. Thus the methodology is presented as two *development guidelines* which specify the 'what', 'how' and 'when' of the development workflows. Each development guideline consists of a co-ordinated set of development workflows. Each guideline prescribes the modelling notations and artefacts to be produced by its development workflows. However, although the methodology is presented as two separate guidelines, compatibility of the artefacts, notations and models developed by each are ensured. The two development guidelines are called the ***Building Block Development Guideline*** and the ***Business Process Driven System Development Guideline***. [formD12]

4.1.1 Objectives and Scope of Building Block Development Guideline

The objective of Building Block Development Guideline is the development of re-usable management Building Blocks Contracts and Building Blocks. The Guideline not only provides advice as to how to model Building Blocks Contracts, and prescribes how such Building Blocks Contracts should be represented so as to ensure that the contracts could be reusable by other actors (i.e. actors not involved in the development of the Building Block Contract).

More specifically, the objectives of the guidelines are to:

- Guide the design activities in developing Building Blocks Contracts & Building Blocks.
- Specify the development workflows required to design the Building Block Contract.
- Identify modelling notations and the models to be developed during each development workflow. Indicate the traceability of artefacts developed across the development workflows.
- Prescribe sets of artefacts[1] to characterise and communicate usage of Building Block Contracts.

The guideline focuses primarily on the development of the Building Block Contract. Thus the guideline does not fully prescribe implementation and testing workflows.

4.1.2 Objectives and Scope of Business Process Driven System Development Guideline

The objectives of this Business Process Development Guideline are

- To provide support for a 'Business Process Driven' approach to management system construction from re-usable Building Blocks Contracts
- To provide a development guideline which will allow management systems integrators to construct management solutions from Building Block Contracts.

[1] An artifact is a piece of information that is created, changed and used by actors when performing development activities. An artifact can be a model, a model element or a document [jacobson2000].

4.1.3 Overview of Building Block Development Guideline

This guideline adopts a Business Model/Use Case Driven approach to represent the management functional areas of interest (e.g. Fulfilment, Assurance, Billing etc.). The development activities and workflows to be carried out are identified below. The principle development workflows are listed below.

1. ***Perform Business Modelling Workflow*** – This process workflow facilitates the definition of business model(s) based upon management business processes. This involves identifying Business Roles, Business information entities, Business Use Cases and Organisational Units and Business Workers.
2. ***Define Reference Architecture Workflow*** – This workflow specifies the development of a Reference Architecture that identifies reference points between organisational boundaries, the placement of process areas within these boundaries and the relationships between these process areas across organisational boundaries.
3. ***Perform Requirements Analysis Workflow*** – This involves such development activities as performing requirements analysis, development of use cases and supplementary requirements specification, modelling of activity graphs (diagrams) to represent the various control and data flows in the use cases.
4. ***Develop Analysis Object Models Workflow*** – This involves the development of analysis objects and development of analysis collaboration models.
5. ***Model (Re)Organisation*** – This workflow guides the re-organise the development classes and models developed in the previous workflow and advises on their potential groupings. This workflow specifies the modelling and specification of Building Blocks Contracts. The Building Block Contracts are specified using XML base description.
6. ***Implement Building Blocks and perform testing.*** This workflow describes the development activities required for Building Block implementation and testing.

The approach taken in developing the Building Block Development Guideline was to use best practice in software development and add new workflows, model, artefacts and specifications where required. The Building Block Development Process was attempted to be implemented using Rational Unified Process. Although generally useful, RUP does not support key modelling artefacts and design activities, which are fundamental to the guideline. Therefore the Build Block development guidelines added new workflows, artefacts and descriptions to RUP which aim to support Building Block design. Other changes to RUP involved the phases of development and templates used to describe the models.

4.1.4 Overview of Business Process Driven System Development Guideline

The goal of the Business Process System Development guideline is to facilitate management systems integrators to construct management solutions from Building Block Contracts. The Guidelines takes a ‘Business Process Driven’ approach to management system construction from re-usable Building Blocks by explicitly modelling the required system processes and their constituent system activities. The guideline uses these system activities to determine the Building Block Contracts needed to implement these processes.

The Guideline itself is divided into eight process workflows. Each workflow has a specific objective and produces or refines model(s) or artefacts. The workflows iterate the classic development activities from business modelling, requirements capture & management, system analysis and design modelling, implementation and testing [fowler97].

The guideline specifies the mapping of the management system activities (in the management system processes) to Building Block Contracts. This supports the reuse of existing Building Block Contracts in implementing the management system processes. This mapping is at the heart of the reuse of Building Block Contracts in the implementation of management processes. A second part of the mapping of management activities to Building Block Contracts is the reconciliation of External Information Model(s) of the Building Blocks Contracts to the information flows in the system processes.

Figure 4-1 identifies the principle workflows involved in the Guideline. These involve:

(i) *Performing Business Modelling:* This workflow facilitates the definition of Business Roles, business use cases and organisational units. The key results of this workflow are the development of Business Use Case Model(s), Business Model (representing the business roles and organisational units), and a refinement of the Reference Model for the ODF framework (i.e. a specialisation of the ODF reference model indicating the management processes and reference points to be used).

(ii) *Define Requirement Analysis:* This workflow facilitates the identification of candidate behaviour of the management (business) processes, software requirement specifications and supplementary specification. The key result of this workflow is the system use cases and supplementary use case specifications

(iii) *Perform System Process and System Information Modelling*: In this workflow the required system process(s) are represented as system activity diagrams. Thus this workflow facilitates the modelling of system activities, their control flow, and their information flows. The key results of this workflow are system activity diagram(s) representing the system processes to be implemented.

(iv) *Re-Model System Processes and Map to Building Block Contracts:* This workflow allows the mapping of system activities (and information flows) to be mapped to Building Block contracts. This is one of the most important workflows in the guideline. In this workflow the system activities are decomposed or aggregated to match, as closely as possible, available Building Block Contract interface specification. This involves matching the BB Contract interface function(s) as well as their information requirements. Where matching is possible, the system activities are annotated with the Building Block Contract, which support it. Where the matched Building Block Contract requires extra information, these extra information objects have to be included in the system process. Where matching is not possible, the system activities will be modelled as bespoke system objects, which require separate design and implementation. This matching of system activities and Building Block Contracts also has an informational aspect.

The key result of this workflow is the system process modelled as activity diagrams, with the (some of the) activities annotated with the Building Block Contract associated with them. The information objects in the activities diagrams are a combination of information objects drawn from the Building Block Boundary Information models and Information Objects developed specially for the process (i.e. bespoke information objects).

(v) *Model Missing Objects and Information Workflows:* This workflow supports the modelling of system objects and information, which is not supported by the chosen Building Block Contracts. This workflow concentrates on the bespoke development of management system functionality/objects, which have to be developed, as there is no appropriate Building Block Contracts to readily support it. The system objects are modelled as use cases, activity diagrams, and class diagrams. Where a system activity involves the design of several system objects, they are grouped together in a subsystem package.

(vi) *Implement Building Block Integration:* This workflow facilitates the implementation of the integration of the Building Blocks so that the intended system processes are executable. This step may be automated if the design tool (used for modelling the system activity diagrams) is

capable of generating an XMI description of the system activity model. Such an XMI description can be used to determine the control flow, data flow and appropriate Building Block Contract invocations. Depending on the integration implementation approach, such an XMI description could be used to populate a Business (Logic) Object capable of making the necessary decisions and invocations on the appropriate Building Contracts (interfaces), or could be used to populate workflow specifications for a workflow execution environment. The key result of this workflow is the implementation of the control and information flow for the business process.

(vii) *Map Building Block Contracts to Building Blocks and deploy BBs:* This workflow facilitates the selection of Building Blocks to be deployed in the system (if they had not yet been previously deployed). This workflow can also involve identification and placement of technology and data gateways where building blocks are implemented using different technologies (to be Business (Logic) object or workflow integration engine). This would be needed if the Building Block Contracts were technology neutral (i.e. the interface descriptions of the Building Block Contracts were not described in a particular distributed implementation technology). In this way the technological or information representation heterogeneity of the Building Blocks can be hidden from the integration business logic. Where the Building Block Contracts are specified using technology specific interfaces, there is no need for these gateways as the Business (Logic) Object or workflow engine would be generated with the required technology specific invocations.

(viii) *Perform testing and Deployment:* This workflow defines and executes the testing necessary for the management process execution. This involves generating test plans and execution of those plans.

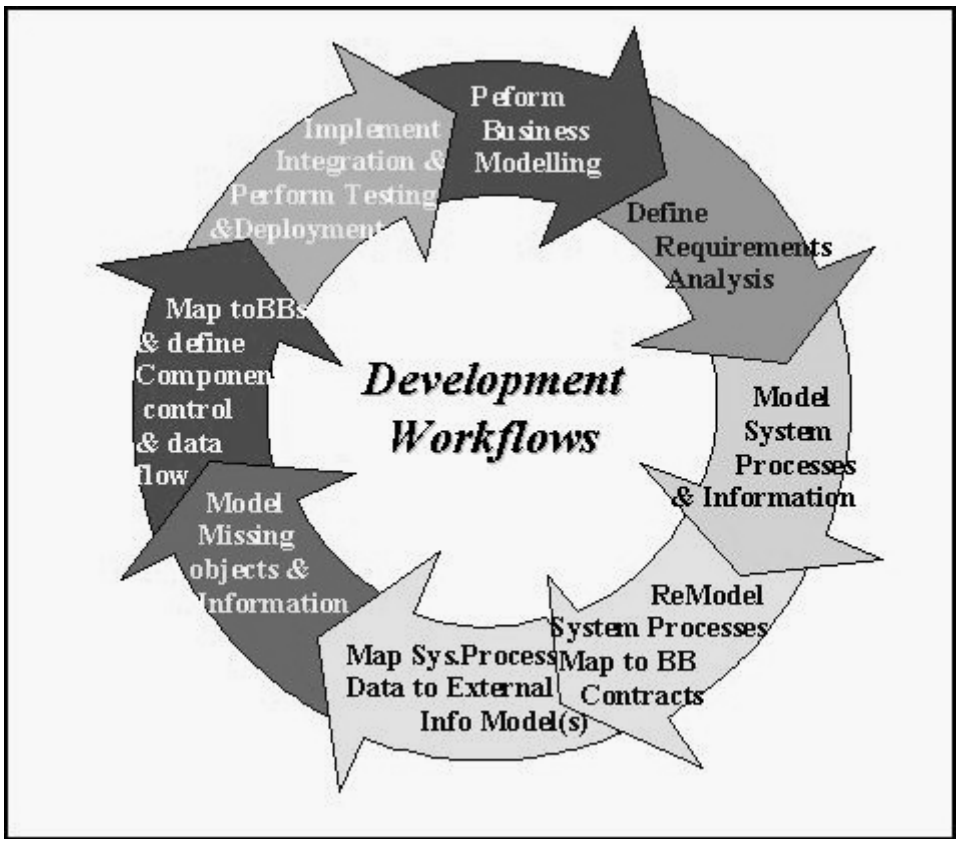

Figure 4-1: Overview of Process Driven System Development Guideline

4.2 Relationship to Logical Architecture

The Logical Architecture can support a large number of methodological guidelines tailored to specific needs to organisation representing the organisation stakeholders in specific application domains. The two Guidelines developed as part of the Development Methodology in FORM address needs relevant to stakeholders thought most likely to immediately benefit from the ODF, namely ISV's and System Integrator developing Building Blocks (served by the Building Block Development Guideline) and System Integrators developing management system using Building Blocks (served by the the Business Process Driven Management System Development Guideline). These Guidelines focus on the internal development needs of these stakeholders as this is the first place the ODF must be accepted before the benefits of using the ODF in exchanging products between stakeholders can be realised, initially though using common models for Building Blocks, their Contracts and EIMs. Further methodological guidance, such as for the development of Contract Set Specification by Standards Bodies or for locating and interworking between models using the EIMs, will also be required to support wider acceptance of the ODF.

Each of the two Guidelines in the Development Methodology follows processes that are structured as a set of Process Workflows. Each Workflow both uses and generates modelling artefacts.

The following paragraphs lists the Workflows for each of the Guidelines, the UML modelling artefacts they produce and the mapping of those artefact to elements of the Architectural Model. These mapping are the idealised mapping of the methodological artefact to the architectural elements, rather than necessarily the ones performed in the FORM Development Trials. This is because not all aspects of the Architectural Model were exercised in these Trials, e.g. the collection of Contract's into Contract Set Specifications. Details of what architectural elements were actually generated as part of the Trials is detailed in section 8.

Key UML Models and Artifacts Produced by Methodology Guidelines	**Building Block Development Guideline** **Workflows**	**Business Process Driven System Development Guideline** **Workflows**	**ODF** **Meta-Model Elements**
Business Use Case Model, Business Use Case Realisation Model, Business Activity Model	Developed by the *Perform Business Modelling Workflow* and *Define Reference Architecture Workflow*	Developed by the *Perform Business Modelling (and Reference Architecture Refinement Workflow)*	Business Context Model
System Use Case Models, (System) *Activity Models*, *Supplementary Requirements Specifications*	Developed by the *Define Requirements Analysis Model Workflow* and the *Develop Analysis Models Workflow.* Used in *Reorganise Analysis Model Workflow*	Developed by *Define Requirements Analysis* in combination with *Model System Processes and Information Workflow.*	Domain Analysis Model
Building Block Contract Specification Model,	Constituent Contracts developed as part of the *Reorganise Analysis Model Workflow*	Used in *Re-Model System Processes and Map to Building Back Contracts Workflow*	Contract Set Specification
External Information Model	Developed as part of the *Reorganise Analysis Model Workflow*	Used in *Re-Model System Processes and Map to Building Back Contracts Workflow*	External Information Model
Building Block	Developed as part of the	Used in *Map Building Block*	Building

Descriptions	*Reorganise Analysis Model Workflow and Implement Building Blocks Workflow*	*Contracts to Building Blocks and Define Component's Control and Data Flow Workflow*	Block Group
System Process Model (including *annotation of Building Block Contracts* and *Information Models*)	Not applicable	Developed as part of *Model Missing Objects and Information Workflow*, and as part of *Implement Integration and Map Building Block Contracts to Building Blocks*	Management System Model

However, one limitation currently of the ODF Logical Architecture is that a clear definition of a Building Block Contract Interface which was technology neutral was not fully defined. The difficulty is to define at what level of abstract is an interface deemed to be technology neutral. For example, there is no clearly defined, widely agreed taxonomy of interface paradigms e.g. RPC, Message Based, Event Driven. Rather than arbitrarily chose such a taxonomy, the Methodological Guidelines allow the specification of technology specific Building Block Contracts if the developers so wish.

5 Technology Selection Guidelines

5.1 Introduction

The technology selection guildeines/guidelines section of the framework is focused on providing mechanisms for selecting one or more technologies for the implementation of specific applications for management systems. Selecting a technology is not a trivial task and will constitute a strategic decision for most organisations. The objective of this section is not to present and compare technologies but rather to raise awareness about the different criteria to be considered in the technology selection process. A major mistake in selecting a technology is to look solely at a set of functions or features that satisfy the business processes needs and that would be acceptable to the users. Although this will constitute a major step, the precognition of a Building Block based system architecture means that a number of quality requirements must be met. These requirements, expressed mostly by the TMForum Application Component Team, play an important role in the selection process. Criteria for the technology selection process can be divided into three different categories. The first criterion is purely functional and corresponds to the set of functionalities (business processes) that the management system is to fulfil. The second criterion is associated with the FORM Architecture principle and relates essentially to non-functional system requirements. Finally the last criterion in the technology selection process is more external to the technology features and focuses on corporate strategy, developer's training, technology trends, etc…

5.2 Addressing the Functional Requirement Criterion

The system's functionality is the most obvious evaluation criterion. Simply put, this evaluates a technology's features against the functions to be delivered by a system. The definition of a management system's functionalities is the result first of the analysis high-level modelling work done in specifying the logical architecture models. Although the FORM framework does not provide for "internal" Building Block detailed design guidelines, it becomes also important within the technology selection process to address the set of functions a building block will "internally" achieve to fulfil its external role.

System functionalities are usually defined in one of two ways:

- **Functional groupings:** within enterprise management technologies, these categories could include enterprise management console functionality, user administration, asset and inventory tracking, electronic software distribution, server management and monitoring, job scheduling, backup and recovery, enterprise service desk, etc...
- **Process groupings:** within user administration for example, this could include profile development, profile administration, profile distribution and profile manipulation.

Defining all functionalities of a management system is an important step to take before selecting a technology. A particular care must be taken in matching these functionalities with the features that such or such technology can offer. For instance, when selecting a technology for implementing an administration user interfaces one will be very careful not to opt for an implementation environment that does not support such GUI development (such as prolog for example).

In a telecom management system context, functionalities will often be constrained by communication constraints. Hence in this context, the distribution facilities a technology can offer will be of greater importance.

In the past, Building blocks might have been implemented without the benefit of components or object-oriented languages. But since a building block enforces its object encapsulation by non-advertisement of its internal interfaces and by proper access control mechanisms; containers provide better encapsulation and many implementation conveniences, such as simplified access to DPE services [gb909]. The concept of containers is present in new component technologies such as J2EE, CCM [ccm], or COM+. This container concept also has an impact on how non-functional requirements can be fulfilled as presented in the next section.

5.3 Addressing the Non-Functional Requirement (NFR) Criterion

Non-functional requirement Definition: *a description of the features, characteristics, and attributes of the system as well as any constraints that may limit the boundaries of the proposed solution.*

In an environment where software quality importance is ever increasing, it is no more enough to solely evaluate the technology functionality allowance. Characteristics such as accuracy, security and performance are quite important concerns for assessing the quality of a system. Quite appropriately, these concerns, so called non-functional requirement (NFR), have been recognised as playing a crucial role in requirements engineering and therefore their role and impact in the technology selection process must be addressed.

Standards bodies, such as the TMForum, have addressed a more "business" non-functional set of requirements. The TMForum Application Component Team while detailing the general requirements of Building Blocks has defined a set of non-functional requirements that a BB should aim at conforming to. From security to recoverability and interoperability, a wide range of non-functional requirements is covered. Each of them should have an impact on the technology section.

The following Building Blocks concerns expressed in [gb909] have been recognised has important for the impact they have on the technology selection process:

- Enable a BB to detect if a service or system functionality it requires is unavailable. (GB909, GR-III.021)
- Enable a BB to inform the system managing the BB in case of abnormal BB behaviour e.g. caused by a required service or system functionality being unavailable. (P3, GB909 GR-III.037)
- Enable a BB to validate the users of its services. (GB909, GR-II.31)
- Enable a BB to inform the system managing the BB in case of security violations. (GB909, GR-II.35)
- Enable a BB to participate in transaction based services. (GB909, GR-II.024)
- Enable a BB to recover e.g. rollback to a pre-transaction state in case of an aborted transaction. (GB909, GR-III.027)
- Enable a BB to recover with minimal impact on BB functionality in case of a transaction failure. (GB909, GR-III.028)
- Enable BB to share an identical interface to provide general application management capability to enterprise systems management facilities. (GB909, GR-V.36)

Overall, the "Generic Requirements for Telecommunications Management Building Blocks" [gb909] document expresses number of constraints that will help in selecting an appropriate technology to realise Building Block based systems implementation. However, although the document is helpful it does not cover entirely the non-functional requirements (NFR) aspect of the technology selection process. NFRs are often associated with a rich, but potentially confusing set of concepts (sorts), which can be organised as by Chung, Nixon [chung,nixon] into a hierarchy to guide the developer.

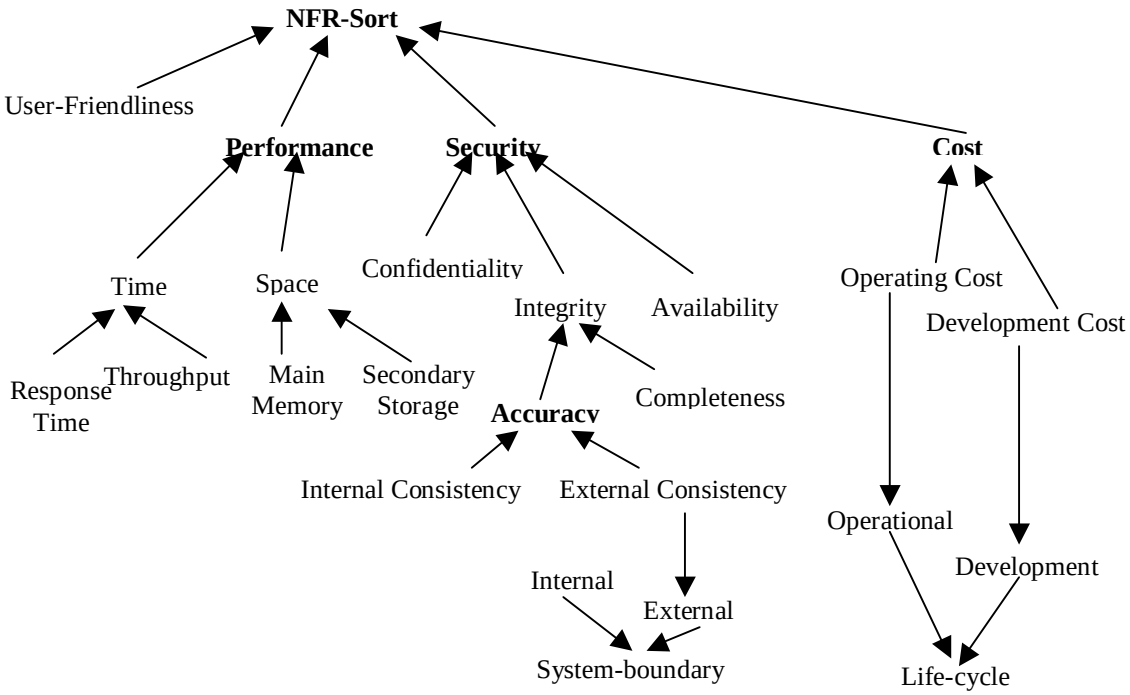

Figure 5-1: Sort hierarchy [chung,nixon]

The NFR that the Architecture principles (section 3.1) and the GB909 document bring on should help in defining the "system technology". A system technology defines the technological environment in which one or more Building Block can operate successfully in accordance with pre-defined requirements. This will depends partly on applications architectures, software reusability, easiness of administration and platform and database support. Relative to other criteria, best practice selections place a lower relative importance on the system technology criterion. This can be seen as quite deceptive because the system technology criterion usually houses the majority of an organisation's mandatory criteria, which include server, client, protocol and database support, application scalability and other architectural capabilities. Often, the expression of NFR as a mandatory criterion within allows to quickly narrow the long list of potential vendors to a short list of applicable solutions.

5.4 Addressing the Non Technology Feature-based Criterion

The features a technology allows for will play an important role in the selection process, however, they should not be seen as the only reason why a technology should be selected. Awareness of the organisational context is a very important factor that influences the technological selection. Elements of this organisational context are among others, the organisation's corporate strategies and enterprise resource plans (ERP) as these define the (implementation) staff competences/training perspectives, and these elements should play an important part in the selection process itself. An illustration of the importance of the organisational context would be to make the decision to select a component-based technology, such as EJB, without having the BB developer trained on Java technologies. This would obviously generate a high risk factor in the development cycle and put at stake the realisation of the product. At stake, strategic plans for technology will be influenced largely by the various technology trends. Currently, the software industry is heavily influenced by Component based development and many organisations have already invested in staff training on technologies such as EJB, CORBA or COM, but since the FORM architecture principles are technology independent, it is essential for future trends in software industry to be monitored and assessed against the potential benefits they could bring in fulfilling functional and non-functional requirements.

5.5 Summary

The purpose of the technology selection guildeines part of the framework is not to dictate the use of specific technologies but rather to help in defining the basis for an improved selection process. As discussed, the need for a BB to meet certain generic requirements such as unity, release and resource independence, recoverability and others, generates a set of mandatory non-functional requirements that should help in narrowing the technology choices. However, an organisation should be aware that technology decisions must be part of a broader and long-term strategic plan. Performing a selection for implementation and integration technologies requires an organisation to marry a number of internal business requirements with a myriad of vendor attributes that relate to both performance as well as the ability to effectively provide long-term value to the organisation.

6 Reusable Elements

This section describes the Reusable Elements portion of the ODF, explaining the structure of these elements as developed for the IES application domain addressed in FORM. A full specification of these elements is provided in FORM deliverable D11 [formD11].

The Logical Architecture defines the main models of the ODF in a way intended to clearly explain the main separations in the ODF and to highlight the primary reusable elements of the model. However, to reuse the models as part of the Development Methodology the elements of reuse are models that actually span the main elements of the Architectural Model through resolving traces between them.

The reusable elements from the system models are presented as a series of catalogues:

- **Reference Architecture**. The reference architecture is the base business scenario for the prototype design and implementation within FORM. It contains reusable networks of Business Roles and Referneces Point and their mapping to Business Processes
- **Reference Point Catalogue**. The reference point catalogue is a list of reference points from the reference architecture, which are addressed by the FORM prototype implementation.
- **BB Contract Catalogue**. Contains a summary of the Contracts, which have been defined as part of the prototype development within FORM. Contracts are implemented as actual BBs.
- **BB Catalogue**. This is a list of the BBs designed and implemented in FORM with indications of the amount of reuse per BB.

To achieve these reusable elements the FORM development methodology has been applied to the IES application domain. This application was conducted to exercise and evaluate concepts of the ODF.

The Reference Architecture showing the functional decomposition of management processes and the reference points is defined as the result of Business Modelling (ODF terminology: Business Context Modelling). The Business Models consist of:

- Business Processes in terms of Business Use Cases.
- Organisations, business roles and reference points in Business Object Models
- The Reference Architecture

The BB contract catalogue and the BB catalogue constitute the outcome from Systems Modelling (ODF Terminology: Domain Modelling). In systems modelling the functional requirements are described in use case models, these use cases are realised in analysis models, which are used as basis for mapping functionality to BBs. Further the interfaces for BBs are described in contracts. Systems modelling then have the following outcome:

- Functional requirements in use case models
- The realisation of use cases in Analysis Object Models
- BB and BB groupings
- BB contracts

BB Contracts are assembled into one catalogue, Each system contributes to a BB catalogue, each system can produce one or more BB Group depending on the separation needs. In addition, each Contract specification specifies which, if any, of the Reference Points from the Reference Architecture it supports. Therefore, a catalogue of Reference Points can also be generated that contains for each Reference Point the Contract Specifications, which are supported by that Reference Point. The figure below gives an overview of the Contracts which address Reference Points as discussed in [formD10]. The Contracts and square brackets are ones that have not been implemented in FORM but which have been suggested as further Contracts which address that Reference Point.

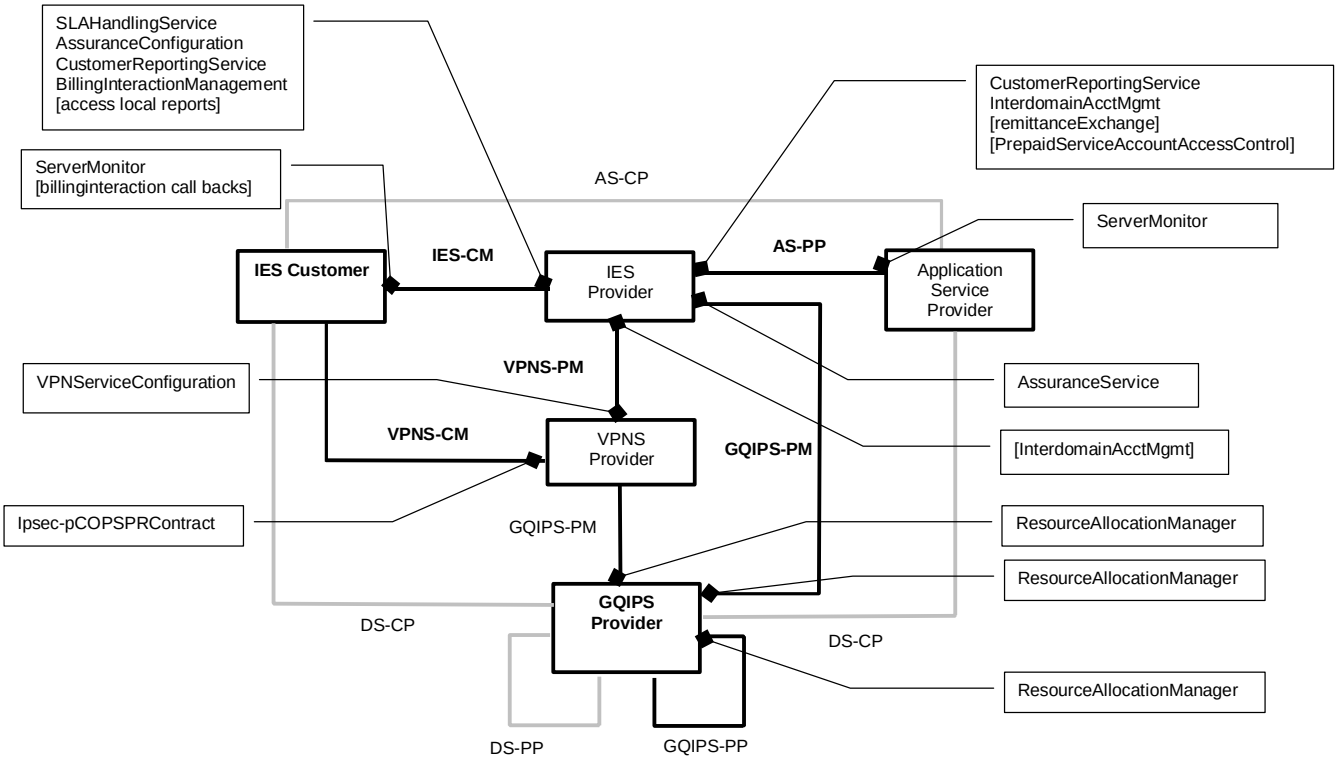

Figure 6-1: Overview of mapping between Contracts and Reference Points

Though Building Block Groups and Contract Set Specifications were not explicitly addressed in the trial development, suggestions for such models had been made as part of the trial 2 evaluation in [formD10].

The full specification of the reusable elements is provided in FORM deliverable D11 [formD11]. D11 consists of a main document applying the FORM methodology “Business Process Driven System Development Guideline”, and four Annexes applying the FORM methodology “Building Block Development Guideline”. The assessments of the reusable elements are provided in FORM deliverable D10 [formD10].

The entire FORM Contract Catalogue can be accessed on-line at the FORM website http://www.ist-form.org/ContractCatalogue/index.html

7 Application of ODF – Case Study

The model elements in the meta-model have been mapped to modelling artefacts that make up the Development Methodology. These modelling artefacts are what have actually been produced by developers of the FORM Trial 2 systems and the components from which they are contracted. These models and the experiences of developers in producing them have been used for evaluating the ODF in this document. This situation is summarised in Figure 7-1.

The development of the FORM Trial 2 systems and the associated systems and component-related models reflects the guidance given to developers through the Development Methodology Guidelines. The Trial 2 development experiences, therefore, only indirectly reflect an application of the architectural principles and the related meta-model from the Logical Architecture.

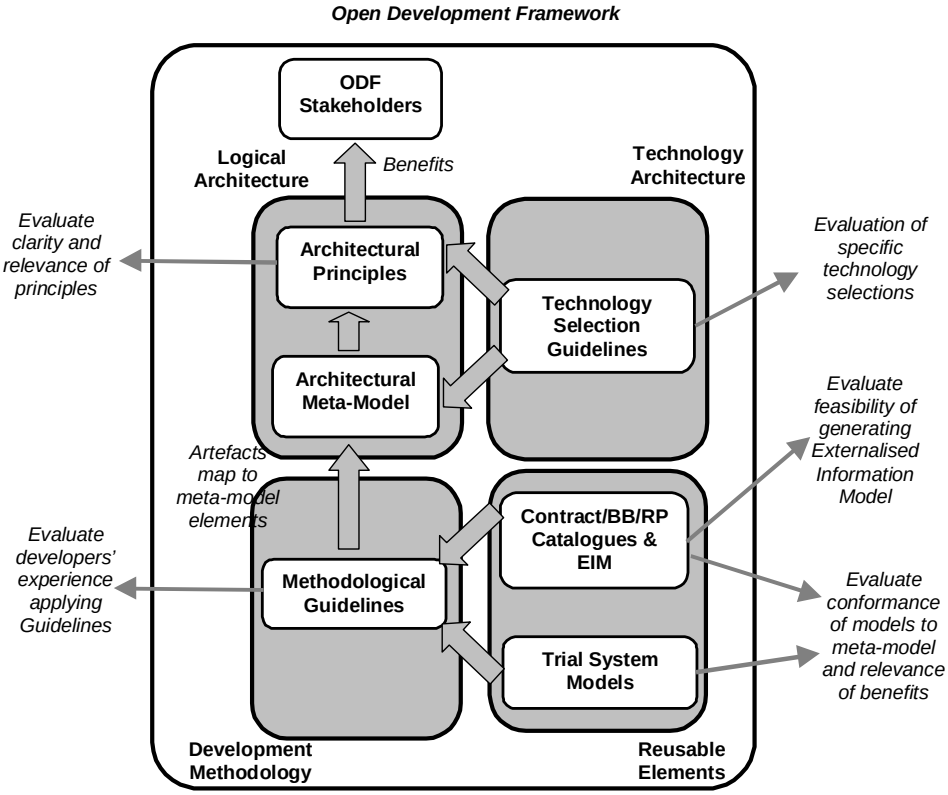

Figure 7-1: Summary of ODF element related to Trial 2 evaluations

7.1 Application of Architectural Model: Case Study

The development work performed in the FORM implementation trials was conducted in order to exercise and evaluate concepts of the ODF. Expressed in terms of the architectural meta-model the following paragraphs describe the architectural elements that were exercised during trial development and illustrate the mapping to certain methodological artefacts that were used in modelling them. The examples of methodological artefacts are taken from the modelling documentation for the development Trial, a detailed version of which can be found in deliverable D11 [formD11].

A single overall Business Context Model was developed for the IES Management Framework. This consisted of a Business Role Model, a Business Organisation Model, a Business Use Case Model and a Business Reference Model, which provided a mapping of the Business Roles to Business Processes in a typical organisational model for the IES domain. Therefore the Business Process Model in this BCM only contained static definitions of process areas and no models of Business Process Flows.

The Business Role Model was captured as a UML class diagram (see Figure 7-2) with classes representing the Business Roles, and associations between them representing the Reference Points and the cardinalities they imposed between the Business Roles. This was supplemented with textual descriptions of the Business Role and Reference Points.

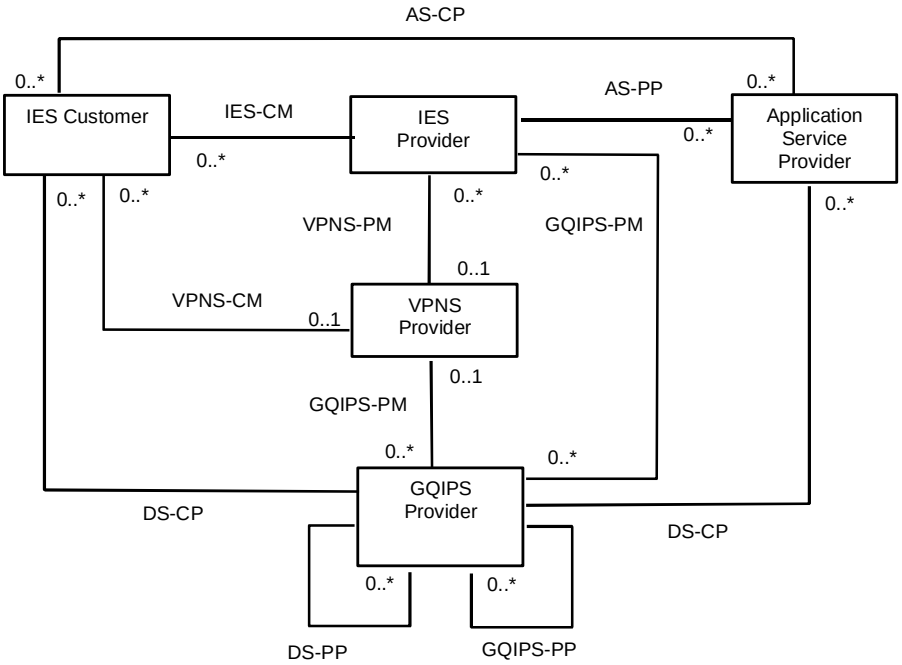

Figure 7-2: Business Role Model from overall BCM for IES Problem Area

The Business Organisational Model defined an organisational scenario that was used as a basis for all the individual organisational scenario addressed by the different workgroups in FORM. To fully exercise the Business Role model, each Role was taken by at least one Organisation. This allowed the Business Organisation Model to be initially represented as a UML object instance model, with Organisations being object instance of classes (see figure below). This form of diagram does not support situations where an Organisation takes on more than one Business Role, and does not support identification of Human Users within the Organisation, which in the BCM were identifies separately.

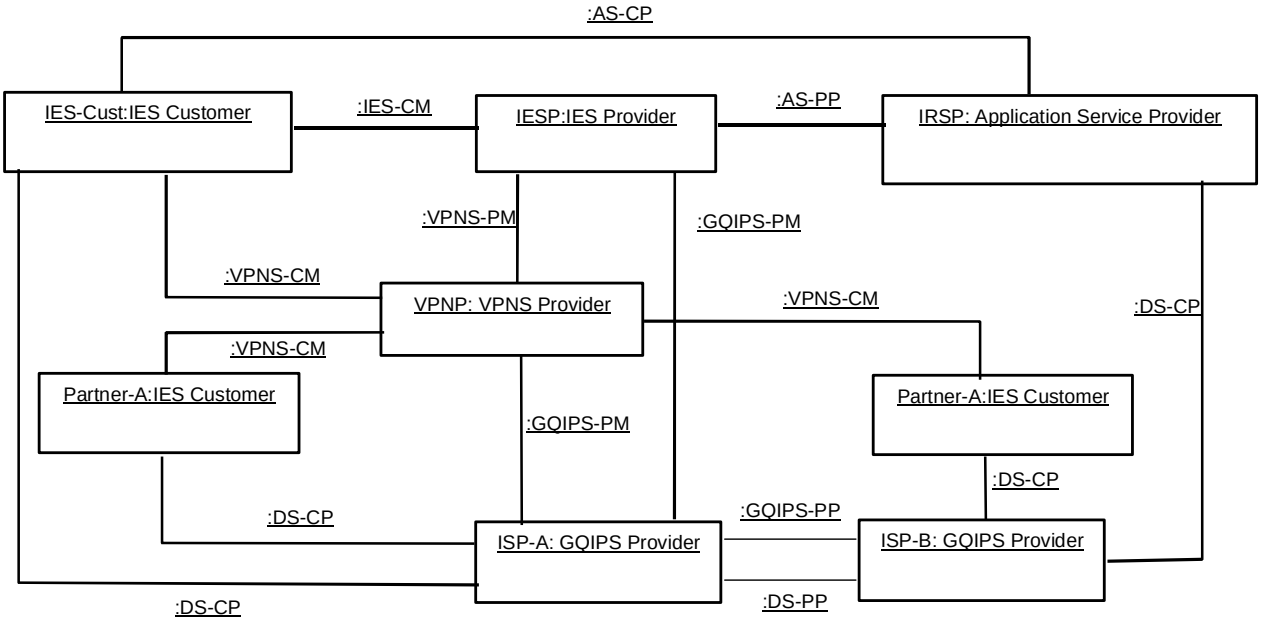

Figure 7-3: Business Organisation Model for the overall BCM for the IES Problem Area

The Business Reference Model was captured in a methodological artefact termed the Reference Architecture due to the important role it played in capturing the business architecture of the problem domain. For this a custom diagram was used, as no UML diagram was able to clearly and concisely capture this model.

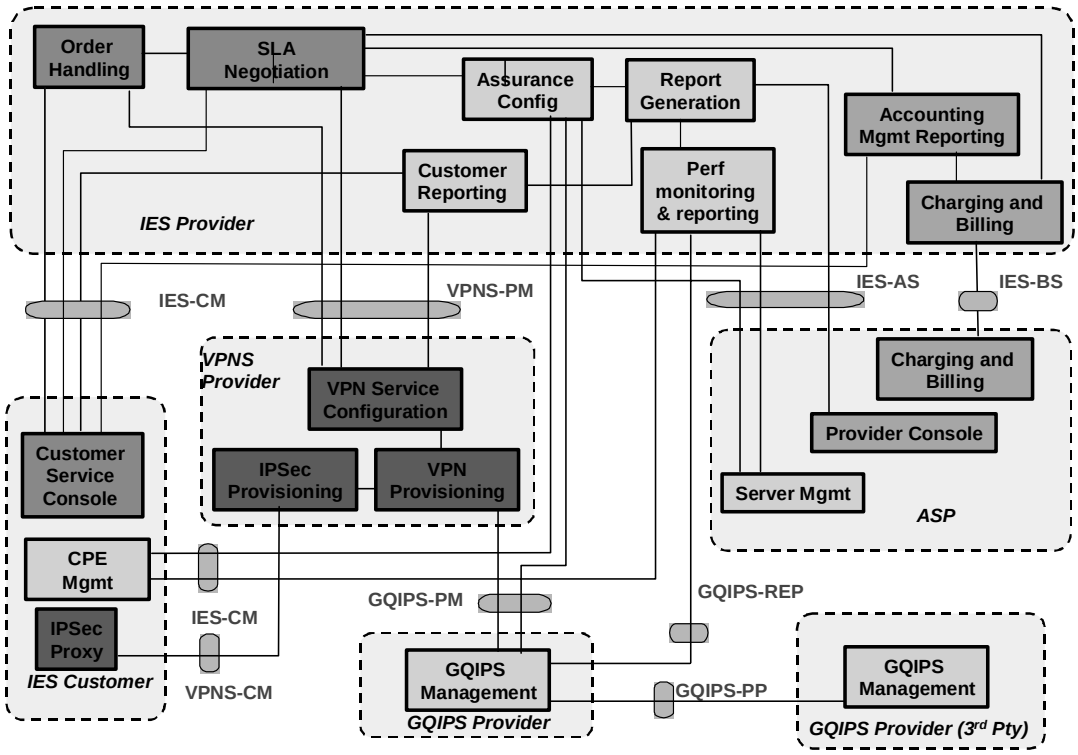

Figure 7-4: Reference Architecture diagram representing the Business Reference model for the overall BCM for the IES Problem Area

A very basic Business Use Case Model was defined for the overall IES BCM. This contained just one Use Case describing the overall functionality involved in ***fulfilling, assuring and billing for an IES service***. Seven Business Context Models were then established to address different business process flows within the context of the single common Business Context Model. Each of these Business Context Models was developed by a separate workgroup. Four of these workgroups were ***Trial workgroups*** that had the aim of developing Building Block based Management Systems based on mutually exclusive process flows. The other three workgroups performed paper-based studies examining process flows that spanned the process flow areas addressed by the Trial development workgroups. All seven Business Context Models included Business Organisation Models, Business Process Models and Business Use Case Models based on specific subsets of the common Business Reference Model.

Partial Domain Analysis Models were developed by the Trial development workgroups. The project plan required these workgroups to develop both the Management Systems and the Building Blocks from which they were composed, in effect acting as System Integrators which were developing Building Blocks for later reuse. Therefore, the Domain Analysis Models generated by each workgroup supported both the design the Building Blocks and the design of the Management System in which they reside. Each Domain Analysis Model defined an analysis object model with analysis objects representing individual Building Blocks. In some cases, the use cases in the Domain Analysis Model merely reflected the ones used in the corresponding Business Context Models.

Contracts were designed as needed for each working group, and were not developed explicitly as members of Contract Sets. Equally, Building Blocks were designed to suit the needs of the Domain Analysis Models shared with the Trial Management Systems, rather than as parts of a Building Block Group. Contracts were not specified in a fully technology neutral form since the interface of the Contract signature was specified using whatever language was suitable to the contract implementation technology used for the Building Blocks as developed for the Management System concerned. However, the information that could be exchanged by each Contract was explicitly modelled in a common format. Within some workgroups this information model was co-ordinated across the different Contracts, while some key piece of information, such as that specifying Service Level Agreements, was co-ordinated across different workgroups with the aid of insight gained from the inter-workgroup process flow modelling performed by the other groups. The information models from all the workgroups' Contracts has been collectively analysed with the aim of generating an External Information Model covering all the contract developed for the IES Management Framework.

Management System designs consisted of both Building Blocks and their Contracts as well as non-Building Block subsystems. Several Building Blocks deployed in these Management Systems had behaviours which were controlled by Business Rules, though no common format was used for these rules.

8 Detailed Architectural Model

The following subsections describe the details of the meta-models for the principle elements of the Architectural Model identified in Section 3. Some types of models, such as Use Case Models, Information Models and Process Flow Models are present in more than one part of the meta-model and are therefore only explained in detail the first time they are encountered. To aid understanding of the structure of the models, their elements are written with initial capital letters. UML class diagrams are used to precisely define the relationships between elements within the models.

8.1 Structure of Business Context Model

The Business Context Model (BCM) is the description of the business requirements related to a specific problem area. A wide range of mechanisms exist for capturing business requirements and this model does not aim to restrict which ones are used, however the core elements defined in the BCM have been specified as the minimum needed to ensure consistency with the rest of the Architectural Model. The following subsections give details of the elements required within each of the BCM's constituent models, plus an overview of the relationships that should be maintained between the models. It is important to note that one BCM can be used to provide a context for more detail requirements analyses in one or more other BCMs.

8.1.1 Requirements Statements

This enumerates and categorises existing requirements statements to allow traces to be made to them from the rest of the BCM and other derived models. Categorisation may be between functional, non-functional (e.g. adherence to existing standards), exception-related and information related requirements.

8.1.2 Business Role Model

The Business Role Model defines generic Business Roles relevant to a problem area. 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.

A Reference Point may be mapped to one or more Contract Specifications once a stable understanding of the Business Role Model is reached and a suitable population of Contracts exists.

The Business Role Model may represent a subset of Roles and Reference Points from a Business Role Model in another BCM or may be the combination of ones taken from several BCMs.

8.1.3 Business Organisation Model

The Business Organisation Model can be thought of as an instance of the Business Role Model constructed to illuminate realistic potential business scenarios and thus drive more detail requirements capture. Several Business Organisation Models could be generated from one Business Reference Model capturing the different possible configurations of Roles, e.g. where several organisations simultaneously play the same Role or Organisations take on several Roles. A Business Organisation Model may focus on some useful subset of the Roles in the Business Role Model.

The Business Organisation Model contains one or more organisations, each of which may support zero or more different types of Human Users. Each organisation must enact one or more roles from the Business Role Model and each Human User must be allocated to one Role. Potential interaction between organisations can only exist where relevant Reference Points are supported in the Business Role Model between Roles taken by the Organisation involved.

Provided it is consistent with the Business Role Model in the same BCM, the Business Organisation Model may represent a subset of Organisations and Human Users from a Business Organisation Model in another BCM or may be the combination of ones taken from several Business Organisation Models.

8.1.4 Business Use Case Model

The Business Use Case Model captures the required functionality that is to be observed in a Business Scenario represented by a Business Organisation Model. This functionally is described from the point of view of the Human Users identified in the Business Organisation Model. .

The Business Use Case Model is a set of Use Cases that defines the full set of interactions that the problem area has with Use Case Actors which represent those imposing requirements on the problem area. Various types of Use Case Actors may be used in Use Cases including the following:

- *Human Actors* that represent human system users which will interact with a software system that wholly or partially implements the requirements being analysed. Each Human Actor represents a different Human User identified in the Business Organisation Model
- *System Actors* that represent software systems that will interact with a system resulting from the requirements analysis. This may represent an existing system, or a planned system that is subject to another BCM defined in parallel to or subsequently to this BCM. Existing systems already may offer well-defined interfaces via which interactions with that actor must be conducted, either expressed as references to Contracts Specifications or to other standards
- *Process Actors* that represents Business Processes that are part of the problem area's environment and which are taken from the parent BCM to the one in which this Use Case Model resides.
- *Role Actors* that represent Business Roles that are part of the problem area's environment and which are taken from the parent BCM to the one in which this Use Case Model resides.

Each Use Case takes one Actor as its primary actor, though any number of other Actors may be secondary Actors. 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.

8.1.5 Business Process Model

The Business Process Model defines a set of business processes operating within the problem area and the flow of control and information between them. Business Processes enact the functionality of the problem area identified in the Business Use Case Model. At the highest level of abstraction the Business Processes may be identified and statically grouped, e.g. by TMN logical layers or according to the TM Forum Business Process Model. At a more detailed level, Business Process Flows are defined that capture the end-to-end flow of information and flow of control across the problem area.

A Business Process Flow identifies Activities that address specific parts of a Business Process. Activities themselves can be decomposed if necessary into finer grained Activities. Activities have specific starting and terminating conditions, which control the flow of Events between Activities and between Activities and Event Sinks and Event Sources. Activities may also exchange Information Objects and exchange information with Information Sources and Information Sinks. A Use Case from the Business Use Case Model will drive one or more Business Process Flows. Actors from such Use Cases will map to Event Sources and Sinks and Information Sources and Sinks.

8.1.6 Business Reference Model

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

- A Business Process from the Business Process Model may be mapped onto a Business Role as positioned in an Organisation in the Business Organisation Model. The same Business Process may be replicated in more than one Business Role or Organisation, but a single Business Process cannot span more than one Business Role.
- Business Process Flows from the Business Process Model may be mapped to Reference Points from the Business Role Model as positioned in the Business Organisation Model.

These mappings allow consistency checks to be performed. This may ensure that the control and information flows between two Business Processes residing in different Business Roles are 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 control and information flows between Business Processes that pass across that Reference Point.

8.2 Structure of Domain Model

The Domain Model represents an analysis of a domain of interest and of its interaction with its environment. This analysis is based upon, and traceable back to, the requirements and business modelling elements from a single Business Context Model. The Domain Model represents a logical model of the software system that satisfies the requirements for a problem area expressed in the Business Context Model. The Domain Model consists of the following three models:

- *Domain Use Case Model*: This is a set of Use Cases that defines the full set of interactions that the domain has with a set of Domain Actors. These Use Cases and Domain Actors are derived respectively from the Use Cases and Actors of Business Use Case Model of the originating BCM.
- *Domain Process Model*: This is a set of Process Flows derived from Process Flows from the Business Process Model of the originating BCM.
- *Domain Analysis Model*: This is a logical 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 Domain Use Case Actor.
  - Entity AO: This represents information within the domain
  - Control AO: This models functional, algorithmic or process oriented aspects of the domain

Figure 8-1 shows the relationships between the constituent elements of the Domain Use Case Model, Domain Process Model and Domain Analysis Model.

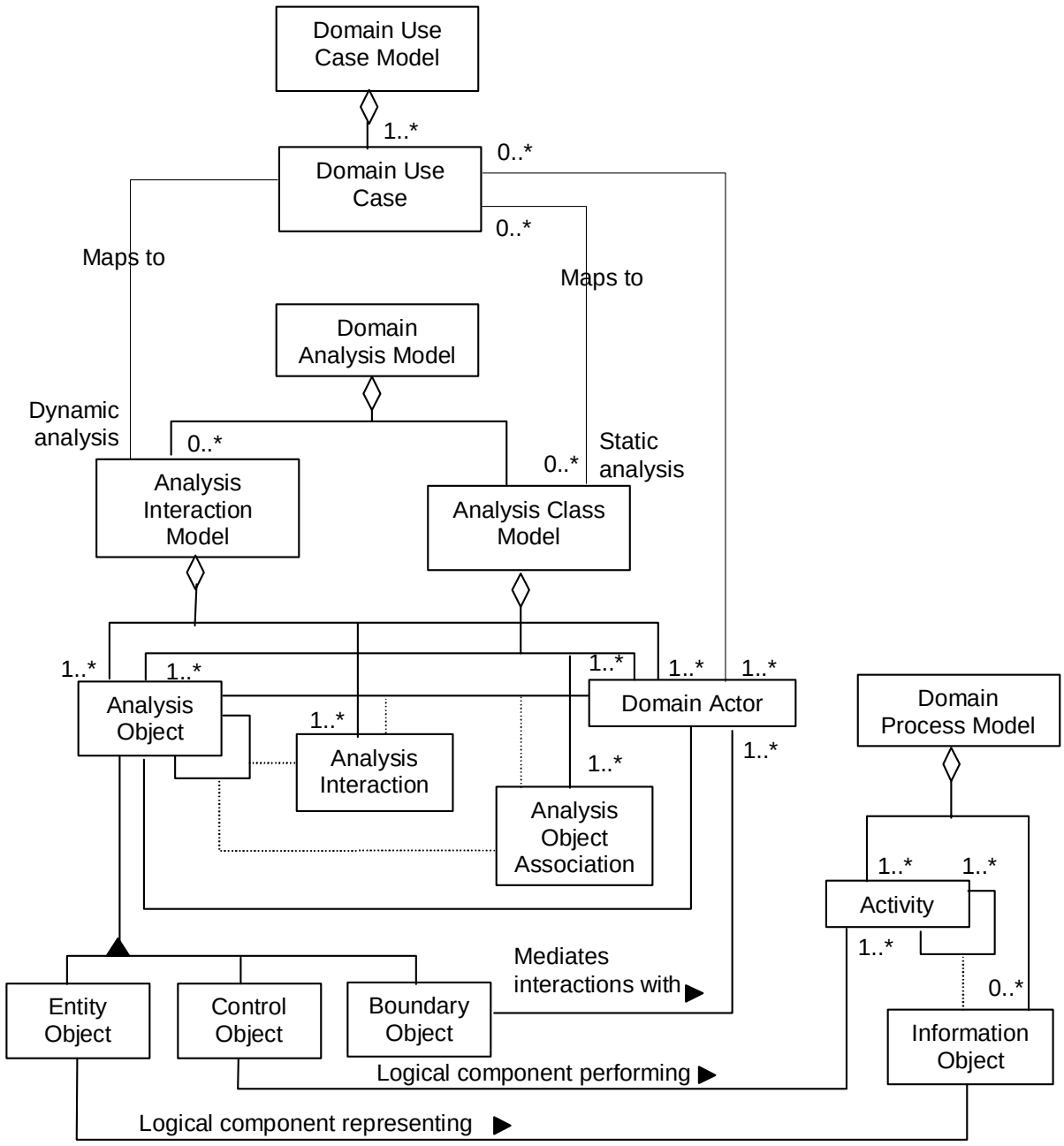

Figure 8-1: Meta-model mapping between sub-models of the Domain Model

The Domain Model must exhibit consistency between its constituent models. Therefore, though the Domain Process Flow Model and the Domain Use Case Model are derived from a BCM's Business Process Model and Business Use Case Model, they need to be adapted so they are consistent with the Domain Analysis Model. This requires consistency of the following relationships between the constituent models:

- Each information object passed between Activities in the Process Flow Model maps to one Entity Object.
- Each Activity must be enacted by only one Control Object.
- Each Event Sink and Source and Information Sink and Source must be mapped to a Domain Actor.
- The interactions of a Domain Actor with the logical system represented by the Analysis Objects should be via a single Boundary Object.

8.3 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 that, therefore, create difficulties in comparing the information content of interfaces, especially when different languages are used, and to design suitable information gateways. Information models placed in an EIM are made available for use in subsequently developed Contract Set Specifications to support comparison of information content with other models.

An EIM is a specialisation of an Information Model. An Information Model, as defined in the ODF consists of Information Objects (IOs). An IO has a class name and a number of attributes. IOs can be inherited from other IOs, which involve 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 from one Information Model may be included in another Information Model. Included IOs may be imported, in which case they are effectively copied into the administrative scope of the new 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 8-2 provides an overview of the structure of the Information Model used in the Architectural Model.

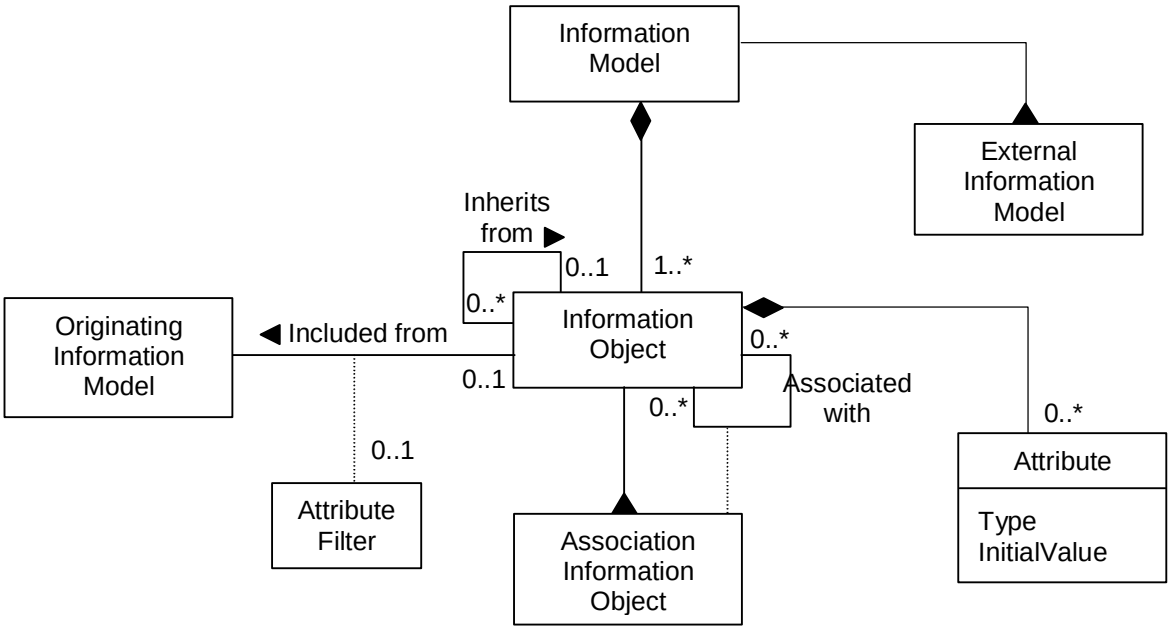

Figure 8-2: Structure of Information Models

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. 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 Specification are explained in detail in the next section.

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

8.4 Structure of Contract Set Specification Model

A Contract Set Specification (CSS) is the main mechanism within the Architectural Model for specifying interoperability. A CSS primarily consists of one or more individual Contract Specifications together with a Contract Set Information Model (CSIM) that contains definitions of the Information Objects (IOs) describing the information passed in individual Contract operations. The IOs may be defined specifically for use in the CSS or they may be included from an EIM. An IO in the CSIM referenced from a specific Contract Specification 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 CSS by inheriting from the original IO (which may be included from the EIM).

A Contract Specification within a CSS may be included as a reference to a Contract Specification in another CSS. A Contract Specification included by reference must therefore reflect revisions to the referenced Contract Specification. In addition, any IOs used by a Contract Specification included by reference must be included in the CSS as a reference to the relevant IOs in the CSS containing the referenced Contract Specification.

Two type of Contract Specification may be used, though only one type may be used for all Contracts in a single CSS:

- A Technology Neutral Contract Specification (TNCS) which is specified in form that is independent of any implementation technology. TNCSs may, however, 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 Technology Specific Contract Specification (TSCS) which is expressed in a form that is specific to an implementation technology, using an interface definition language such as GDMO or IDL.

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 generation of a TSCS from a TNCS may follow a standardised or a proprietary Technology Transform. References to existing Transforms and details of any bespoke Transforms used are recorded in the TSCS.

The meta-model for Contract Sets is outlined in Figure 8-3.

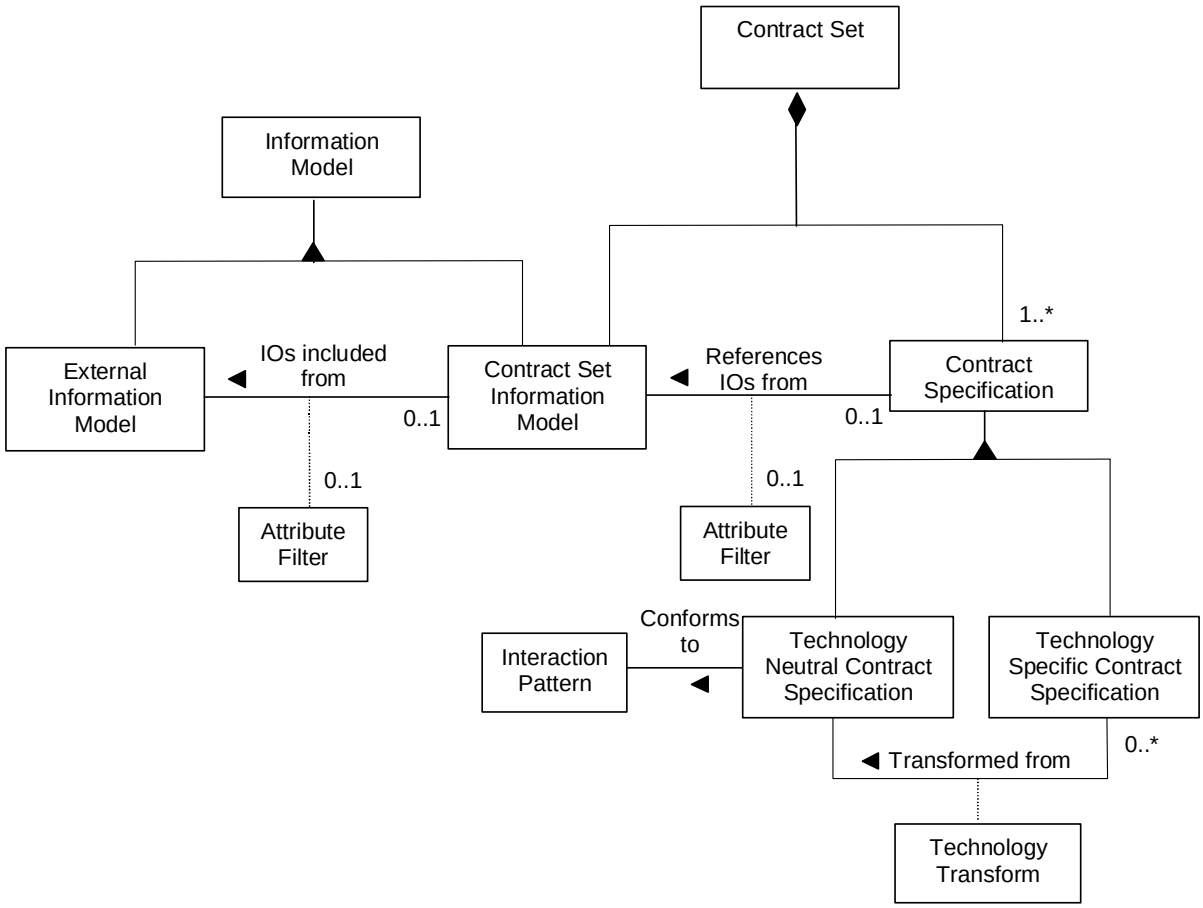

Figure 8-3: Structure of Contract Set Specification Model

A CSS is based on the requirements analysis from one Domain Model.

The lifecycle of a CSS and the Contract Specifications it contains and the lifecycle of an EIM are essentially separate with the relationships between them managed by the inclusion of references in a CSS. 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 CSSs in order to construct an EIM common to a range of CSSs.

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, a model adaptation tool 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.

8.5 Structure of Building Block Group Model

A Building Block Group is a model containing a collection of Building Block Descriptors and accompanying implementations. A Building Block Descriptor contains one or more TSCSs. Each TSCSs can either be taken from a CSS containing TSCSs or can be a mapping from a TNCS using a Technology Transform, which must be recorded in the BB descriptor. It is possible for a BB to implement different (or even the same) TNCSs in different technologies. The BB Descriptor also contains a Behaviour Configuration Vocabulary for the BB, which provides the language from which Business Rules can be constructed to modify the behaviour of BB at runtime.

A BBG is ready for release once each of its BB's has a complete set of TSCSs 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. Some of the BB's internal logic may be 'soft-coded', i.e. specified by a Behaviour Configuration Vocabulary that may be used to express at deployment or run time Business Rules that control the BB's behaviour at run-time.

Figure 8-4 represents the meta-model for Building Block Groups.

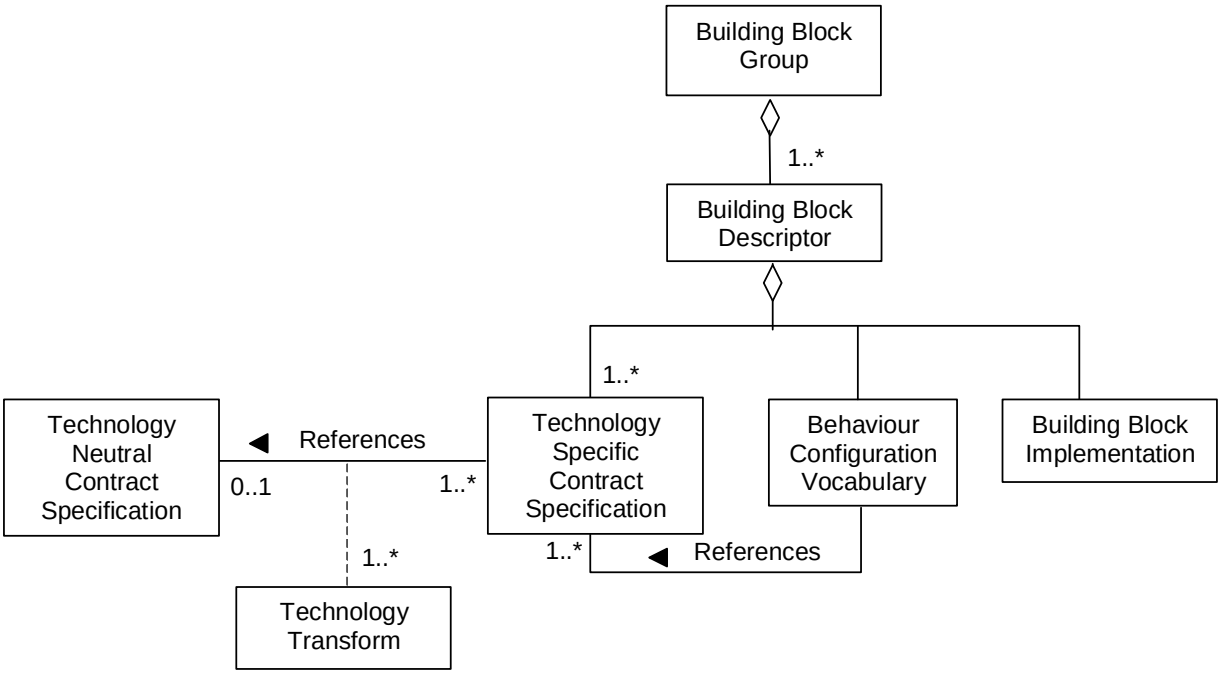

Figure 8-4: Structure for Building Block Group Model

8.6 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 components 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 using that Building Block's Behaviour Configuration Vocabulary. 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 that determine the order in which Contracts and System Interfaces are called across a system. The target BB for such rules therefore typically plays a co-ordinating 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.

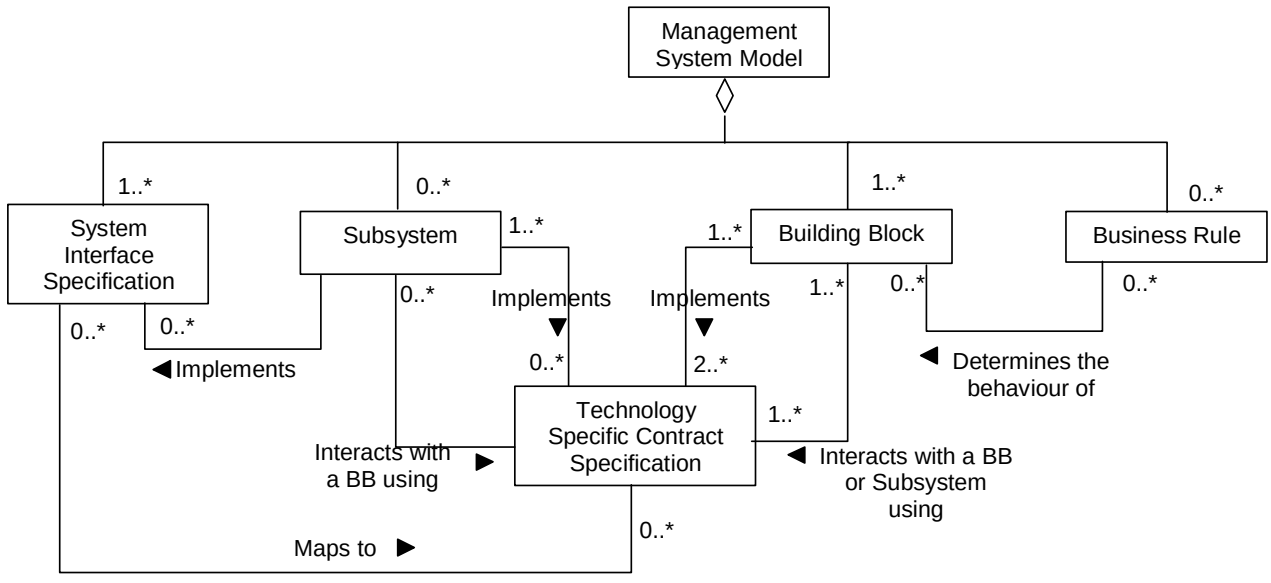

Figure 8-5: Structure of Management System Model

The requirements capture and analysis that typically precedes the generation of a Management System Model should be expressed in or traced through a Domain Model. In such a case, the System Interface specification may be derived, wholly or in part, from a Reference Point definition.

8.7 Business Rule Issues

8.7.1.1 Workflow

A workflow is a description of a sequence of tasks that need to be carried out to achieve an objective. These tasks may be carried out manually (i.e. by a person) or automated by a computer system. A Workflow Management System (WFMS) manages an organisations set of workflows (also termed business processes). Typical workflows in an organisation may describe the sequence of tasks necessary to process an insurance claim, fulfil a book order or handle a customer complaint. The WFMS streamlines these workflows by automating tasks that can be competently completed by the system, leaving the worker free to handle more challenging tasks [wfmc]. A process definition is a complete system specification of a business process that can be executed at runtime in the WFMS. The process definition defines the control flow, data flow and actors involved in the business process.

The benefits of a workflow approach in constructing enterprise business systems are rapid implementation of business processes with little or no coding, built-in process administration and monitoring tools, and a tight coupling between the business domain process model and the actual process definition implementation. It is also noted that these business processes are the most transient part of an enterprise system; the WFMS enables rapid modification/ updating of these transient processes. However the downside of current enterprise workflow systems is that they are still very proprietary. An organisation wanting to adopt this approach will have to buy into the workflow vendor's complete solution. The biggest drawback currently is their use of proprietary graphical process models, proprietary process definition languages and lack of extension mechanisms for handling more complex control structures [wfPattern].

Approach taken in FORM

FORM was interested in using this WFMS approach to manage its telecom management system business processes, however we wanted to avoid the use of a proprietary solution and thereby increase its potential of uptake within industry. We define our approach, as a Business Process Management Framework (BPMF), which attempts to provide a solution that is standards based, open and configurable to suit the particular organisations development style and needs.

The BPMF uses the outputs from the Business Process Driven System Development Guideline (i.e. UML activity diagram and class diagrams) as its process model. These standard UML models are automatically mapped, using XMI [omgXMI], to the BPMF process definition. The process control and data flow is defined as a structured collection of Condition – Action rules. An example rule *(defrule activity:processBill condition:(process_start) => action:(invoke processBill))* defines that Activity *processBill* is invoked on Condition *process_start*. A generic Rule Interpreter interface enables a wide range of rule languages and rule engine implementations to be used for implementation of the process flow. A process executes by coordinating invocation of its activities (FORM Building Block methods) according to its defined process flow (rules). This rule approach is a more flexible means of implementing control and data flow than the more common petri-net and state transition techniques [knolmayer].

The BPMF approach enables an organisation to use standard UML output from it's UML modelling tool and also choose their preferred rule language for the process flow definition. The UML activity diagram has the basic set of process control structures, i.e. sequencing, branching, synchronisation, iteration. However, more complex structures are often needed, for example the Discriminator pattern [wfPattern]. The UML activity diagram can be extended in a standard manor (i.e. stereotyping) for specification of additional control structures and the rule based process definition is flexible enough to enable implementation of these more complex control structures.

The core components of the BPMF consist of Definitions to store defined processes, Instances to store process state, a Process Interpreter (rule engine) to execute process definitions and Adaptors to map conceptual process activities to specific Building Block implementations. An example business process in FORM is the Order Handling of a Customers request for an e-Learning Service over VPN (this scenario is described in detail in Section 6 of Deliverable 11).

Relation to policies

The workflow-based BPMF in FORM defines process control and data flow as a structured collection of Condition-Action rules. The generic Rule Interpreter interface enables use of any common rule engine and its associated rule language to implement this process control and data flow. Similarly, policies consist of a collection of Event-Condition-Action rules. One of the approaches to policy in FORM [UCL] also couples the policy language and policy interpreter together, enabling use of different policy engine implementations to implement the policies.

However, workflow defines a deterministic sequence of actions (via the structured collection of rules) to produce an objective. It lies at the higher levels of process abstraction, integrating Building Block Contracts together, dealing with both control flow and data flow issues. It also provides additional process management functionality such as runtime administration and monitoring. It is suitable for high-level business processes such as Customer Ordering/ Fulfilment. When the business processes become fine grained the overhead of runtime process interpretation outweighs the benefits of easy process management. In FORM the policies work at a lower level of granularity. They are used inside Building Blocks for dynamic configuration, and complex condition processing. Examples of this usage are: dynamic configuration of the Server Monitor building block, dynamic configuration of the Virtual Topology by the VPN Provisioning building block and complex condition logic inside the SLA Negotiation Building Block.

A more detailed comparison of workflow and policies will be given in the white paper W22, *Workflow for component integration*.

8.7.1.2 Policy

Several policy models and policy languages are available at the moment. These languages mostly relate to specific application areas.

The joint effort by IETF/DMTF aims at developing policy based systems for controlling network resources in order to guarantee an agreed QoS. The Ponder language that has been developed by Imperial College in London provide the means to describe certain rights and permissions that manager roles can have on a set of managed objects primarily, but it intends to cover other application areas as well.

A number of European IST project have also been looking at policy languages and policy-based systems: Tequila, Aquila, Cadenus and others. Most of this work is in progress.

A special TMF Policy Group has been active and the first results are expected in due course.

Despite the research effort in policy-based systems up to now a significant list of open issues remains:

- Policy taxonomy
- Generic policy language
- High/Low level policy organisation
- Transformation between high and low level policies
- Policy validation

- Conflict resolution
- Policy domains
- Security concerns

A policy rule as used in FORM have the overall format: ‘when <event(s)>, if <condition> then <action>’ with sub-elements defined as follows:

- Event: This is what triggers the evaluation of a condition in a policy rule
- Condition: If a policy rule condition evaluates to be true an action will take place
- Action: A process that is executed depending on the evaluation of a condition in a policy rule

Approach Taken in FORM

As described in [formD10], policies were used in 5 different BB designs, but no single approach was taken to applying policies, and different policy languages were adopted. The SLA Negotiation Engine BB used a custom XML-based policy language, POLML, which is a superset of the IETF/DMTF languages, designed to allow for better design time grouping of policy rules. The VPN Provisioning BB used the DROOLS policy language (see http://drools.org/). The IPSEC-P BB used the policy language from the IETF, and in particular work from IPSEC working group. The IPSEC-P BB used the policy elements from the IETF Drafts "IPSec Configuration Model" and "IPSec Policy Information Base", for the Policy Objects to be exchanged between IPSec-P and VPN-P (i.e. the BB on the layer above). The Server Monitor and Performance Monitor BBs made use of the DMTF Policy Model. Its policies were generated at run-time based on terms obtained from an SLA, so they made little use of existing rules, policies or conditions, expect for the DMTF representation for a time period condition.

The POLML approach took the broadest architectural view in applying policies, and this is used below in related policies to the ODF’s Architectural Model.

Relation to Architectural Model

The design of POLML supported specifying a number of policy principles and a policy language. The objective was that the policy language is generic enough so that it can be used for policies in any application area. Underlying POLML is the model depicted in the following figure. This shows the breakdown described above for a Policy Rule, but also identifies the possibility of a rule identifying vectors pointing to the particular entity that may, either handle events, interpret conditions and actions and validate conditions and actions. This allows POLML to express the relationship of a specific Policy Rule to elements of software architectures, such as specific Contracts.

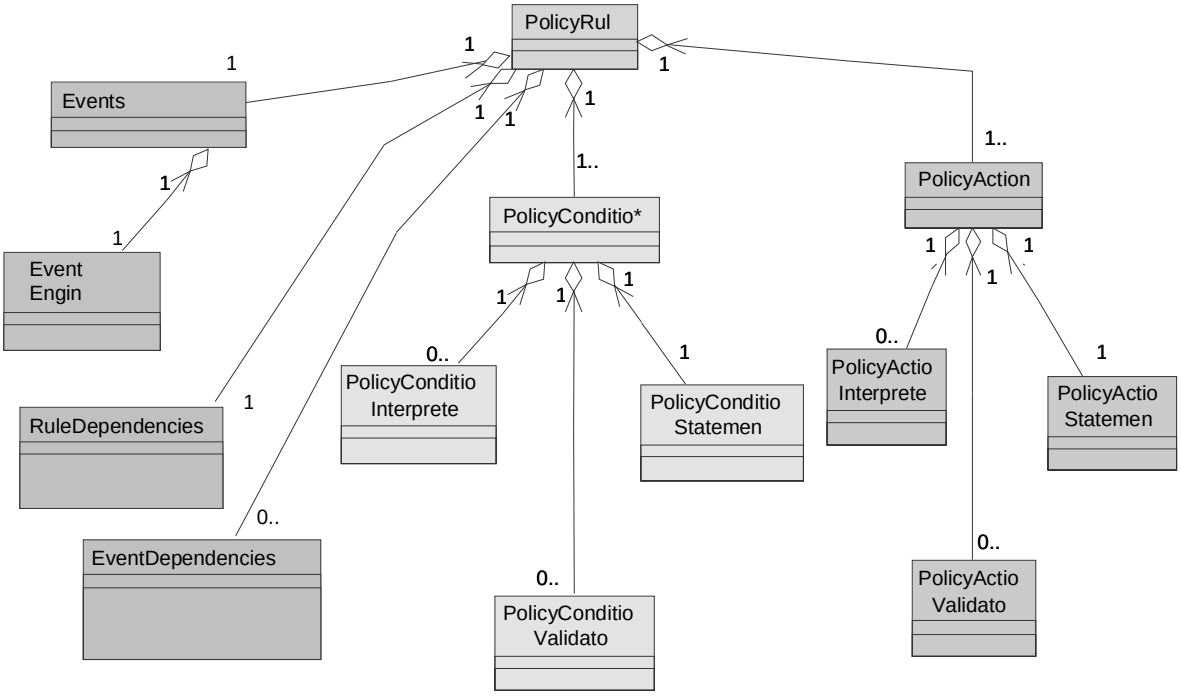

Figure 8-6: Underlying model for POLML

While this model of policy rules allows a high degree of flexibility in expressing how a Rule may be processed by software elements, a more concrete structure is needed to provide more practical guidance on applying policies. Therefore a policy execution environment has been defined with the following components:

- The Policy Manager: is the component that provides the interface between the human user/administrator and the policy-based system. It allows the administrator to load policies: root policies, policy groups or policy rules. It also allows them to enable/disable these policies and to manage the events that trigger them.

- The Policy Execution Engine: is the component that when an event is received it will be asked to "evaluate" or "execute" the policy (root, group or rule). For a policy rule it can check if the conditions are true and it can execute actions. For policy groups it can determine the order of evaluation of the corresponding policy rules (evaluation pattern) and perform it. Similarly, for root policies it can determine the order of evaluation of policy groups and perform it.

- The Event Engine: is the component is in a position to determine that an event occurred. When an event happens, the event engine will notify the policy execution engine.

- The Policy Store: is the component in which policies are stored. These policies can be enabled or disabled. Only enabled policies can be evaluated.

- The Clock: is the component that provides a reliable clock. This is important for time-related events.

The following diagram shows the Policy Execution Environment components and their relationships:

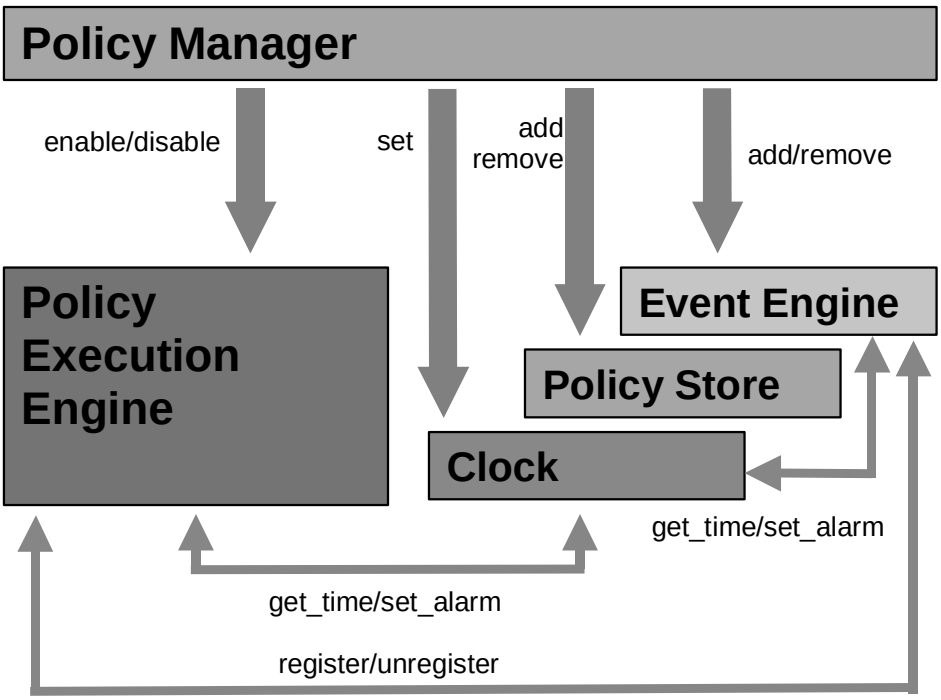

Figure 8-7: Policy Management and Execution Environment

The following diagram shows how these components work together for the following scenario:

1. Add a policy group and a number of policy rules to the policy execution environment.
2. Add the relevant events.
3. Enable the added policy groups/rules.
4. Register the enabled policies with the event engine.
5. Event occurs.
6. Evaluation of relevant policies

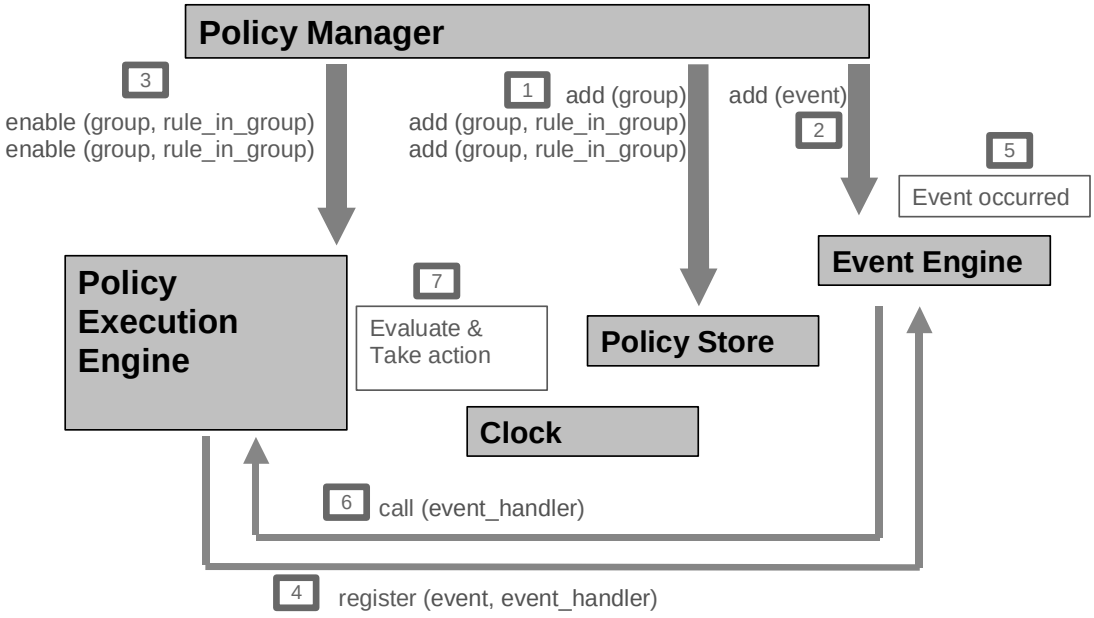

Figure 8-8: Policy Management and Execution Scenario

This environment would need to be supported in a suitable component-software platform, such an EJB container, though an additional service API. The policy execution engine would use the binding between event engine, validators for conditions and actions and interpreters for conditions and actions to manage the communication between different BBs via their Contracts, without those BBs being aware of those bindings. In this was the policy execution environment acts similarly to a workflow execution environment, albeit with difference in the expressive power of the 'Business Rule' language used.

9 Further Work

The ODF developed and evaluated in FORM contains much useful guidance that can be taken up immediately by the targeted stakeholders. Included in this is the Architectural Model presented in this document. However, the architectural principles laid out for the Logical Architecture are ambitious and not all aspects have been realised in detail in the Architectural model, nor have they therefore been mapped to development guidelines and assessed in the Development Trials. The following is a list of issues related to the Architectural Model that require further definition and evaluation work:

- Use of EIM: Though good progress was made is using a CIM-based information modelling schema for extracting information content from Contract Specifications, this has not been used to actually select Contracts or to develop model interworking gateways. The potential of the EIM for this needs to be evaluated, ideally using third party EIMs.
- Technology Neutral Contract Specifications (TNCS) Language: Only Technology Specific Contract Specifications were captured during the FORM Development Trials as no suitable language was identified for expressing TNCSs. For this, a clear set of Interaction Patterns need to be identified and characterised for the ODF and then a language needs to be selected and possibly adapted. The essence of such a language seems to be in flexibly binding the definition of Contract operations to its information content. WSDL is an example of such a language since it can be used to define both RPC-style services and message-passing style services. This was considered insufficient for the ODF as it did not capture the rich, managed object based interactions needed for manager-agent style interactions, support of which is essential for the ODF's management application domain. However, the use of XML for WSDL is seen as a sensible direction, with elements of XSLT being used to bind operations to information model also represented in XML (for which existing CIM-XML mapping from the DMTF could be used). In addition, any TNCS language would need to readily support the application of Technology Transforms to existing technology specific languages such as IDL, GDMO etc. Here the capabilities of XSL should be investigated.
- Behaviour Configuration Vocabulary: No language is identified for this Vocabulary and their use in defining Business Rules. The use of both policies and workflow languages has been assessed in FORM, however these are yet to be reconciled into a common model. In addition any such common model needs to be clearly bound to elements of a TNCS language.
- Maintenance of model: The models generated in the FORM development Trial were mostly first generation model, so no experience has been gained on the modification and maintenance of these models and the impact this may have on existing reusers of the models.
- Tool support: Existing UML case tools and XML editing tools were used in the FORM Development Trial. However, beyond the definition of DTD for XML rendered models, little effort was made to customise these tools to more directly support the Architectural Model and the publication and exchange of its elements. Such tools customisation and the generation of suitable transformation and rendering tools exploiting the capabilities of XMI and XSLT should be investigated.
- Platform Services: The Technology selection guildeines touches on the considerations to be made in technology selection, and for many platforms this involves the identification of common services provided to component-based software. Currently however, the use of such services is not addressed in the Architectural Model. An assessment needs to be made if such services can be represented in a technology neutral manner and if so then investigation into a suitable language needs to be conducted. Such a language would need o be accommodated in the BB descriptor so that service related deployment information may be included. The use of policy languages may assist here in managing technological diversity, and ongoing work in the TM Forum's NGOSS initiative may also provide useful input in the definition of platform services.

10 Conclusions

This deliverable has presented a detailed specification of the FORM Open Development Framework. The framework is comprehensive and represents the results of a major technical effort. It is based on industry best practice and incorporates the knowledge and expertise of a group of leading system designers, architects and software developers. The work in developing the framework has been carried out over a two year period with constant modifications and enhancements based on the output of the project trials and validation activities, as well as comments and feedback from the broader telecommunications community.

The strengths of this approach is that it reflects not only the telecommunications business model, defining roles and relations, but also the software stakeholder model which incorporates the roles of independent software vendors and system integrators. An additional strength of is the inclusion of emerging standards from a broad range of industrial fora (TMForum, TINA, DMTF etc) as well as the incorporation of technological drivers (off-the-shelf software, XML, wedbased services etc).

The framework has been structured into 4 viewpoints, each enabling the system designer/development to focus on key aspects of the overall system in a more manageable way. The logical architecture describes the logical architectures and their relationships. The development of a Meta model ensures the consistency of the overall system models. This is further supported through describing systems in terms of their business process models. At the core of the logical architecture is the concept of a software building block (BB) that is an atomic unit of software deployment and management. A second important aspect of the logical architecture is the concept of contracts. It is through contracts that building blocks interact. The logical architecture enables one to capture and model the functionality of the systems without having to address the implementation specific aspects of the systems. This is addressed in the technological architecture. The technological architecture addresses how the concepts developed in the logical architecture can be implemented using individual technologies. The key link between the two viewpoints is the development methodology that addresses the design of the overall physical system. It describes the processes and notations needed to design contracts and develop building blocks and assemble management systems. The final part of the framework is the reusable elements that provide a repository for reusable products that results when the ODF is applied to a particular application domain.

However, the real strength of the framework is in its ability to produce frameworks' specifications for different telecommunications domains (optical, mobile). The core strength of the approach is the availability of a repository of reusable building blocks, which will support the movement to a telecommunications software development environment capable of supporting rapid and dynamic system development and deployment.

11 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

[chung,nixon] Dealing with non functional requirements: three experimental studies of a Process-Oriented approach, L. Chung, B. Nixon, Proceeding of the 17th International Conference on Software Engineering, Seattle, April 1995.

[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.

[formD10] D10 - Validation of Inter-Enterprise Management Framework, Niamh Quinn, IST-1999-10357/BRI/WP5/02xx, due February 2002.

[formD11] D11 - Final Inter-Enterprise Management System Model, Birgitte Lønvig, IST-1999-10357/LMD/WP4/0522, due March 2002.

[formD12] D12 - Guidelines for Co-operative Inter-Enterprise Management, Vincent Wade, IST-1999-10357/TCD/WP3/012, due February 2002.

[fowler97] UML Distilled - applying the standard object modelling language, M Fowler, K Scott, Addison Wesley, 1997, ISBN 0-201-32563-2.

[gb909] Generic Requirements for Telecommunications Management Building Blocks: Part I of the Technology Integration Map - GB909 v3.0, TeleManagement Forum, January 2001.

[gb921] eTom - The Business Process Framework for the Information and Communication Services Industry, GB921 v2.5, TeleManagement Forum, October 2001.

[jacobson2000] The Unified Software Development Process, I Jacobson, G Booch, J Rumbaugh, Addison Wesley, ISBN 0-201-57169-2.

[kande] Applying UML to Design an Intrer-Domain Service Management Application, Kande, M.M., Mazahaer, S., Prajat, O., Sacks, L., Wittig, M., Proceedings of UML'98 Conference, Mulhouse, France, pp173-182, OMG, Jun 1998.

[knolmayer] Modeling Processes and Workflows by Business Rules, Knolmayer G., Endl R., Pfahrer M., Business Process Management, pp 16-29, LNCS 1806, Springer February 2000.

[lewis99a] The Development of Integrated Inter and Intra Domain Management Services, Lewis, D., Wade, V., Bracht, R., Integrated Network Management VI: Proceedings of the Sixth IFIP/IEEE International Symposium on Integrated Network Management, Boston, USA, pp279-292, Addison-Wesley, May 1999

[lewis99b] A Development Framework for Open Management Systems, Lewis, D., Journal of Interoprable Communication Networks, vol. 2/1, pp11-30, Mar 1999

[lodge] Alignment of the TOSCA and SCREEN Approaches to Service Creation, Lodge, F., Kimbler, K., Hubert, M., Proceedings of the 6th International Conference on Intelligence in Services and Networks, Barcelona, Spain,, pp277-290, Springer-Verlag, Apr 1999

[lucidi] Development of TINA-like Systems: The DOLMAN Methodology, Lucidi, F., Idzenga, H., Batistatos, S., Proceedings of the 5th International Conference on Intelligence in Services and Networks, Antwerp, Belgium, pp379-392, Springer-Verlag, May 1998

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

[omgWf] OMG, Workflow Management Facility Specification, v1.2, April 2000.

[omgUML] OMG, Unified Modelling Language 2.0 Proposal, ad/2001-10-01.

[omgXMI] OMG, XML Metadata Interchange, V1.2, formal/2002/01/01

[osca] Telecordia TR-STS-000915, The Bellcore OSCA Architecture, Issue 1, 10/92

[rfc2578] McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M. and S. Waldbusser, "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, IETF, April 1999.

[tmf053] NGOSS Architecture, Technology Neutral Specification, TMF053, Membership Evaluation Version 2, TeleManagement Forum, July 2001

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

[wade] Three Keys to Developing and Integrating Telecommunications Service Management Systems, Wade, V., Lewis, D., IEEE Communications Magazine, vol. 37, no. 5, pp140-146, 1999

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

[wfmc] Workflow Handbook, 2001, Lawrence P. (ed), Workflow Management Coalition, Wiley.

[wfPattern] Workflow Patterns, Van der Alst W., Hofstede A., Kiepuszewski B., Barros A., http://tmitwww.tm.tue.nl/research/patterns/

[wfUML] UML Activity Diagrams as a Workflow Specification Language, Dumas M., Hofstede A., <<UML>> 2001 – The Unified Modelling Language, 4th International Conference, pp 76-90, LNCS 2185, Springer October 2001.

[x722] Information Technology - Open Systems Interconnection -Structure of management information: Guidelines for the Definition of Managed Objects, ITU-T Recommendation X.722, 1992

[x901] Open Distributed Processing- Reference Model: Part 1: Overview and Guide to Use, ITU-T Recommendation X.901/ ISO/IEC International Standard 10746-1, 1995

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

12 Acronyms

Acronym	Definition
AOs	Analysis Objects
ASP	Application Service Providers
BB	Building Block
BBG	Building Block Group
BCM	Business Context Model
BOM	Business Organisation Model
BPM	Business Process Model
BPMF	Business Process Management Framework
BRM	Business Role Model
BUCM	Business Use Case Model
CASE	Computer Aided Software Engineering
CCM	CORBA Component Model
CIM	Common Information Model
CIM	Common Information Model
CMIP	Common Management Information Protocol
CS	Contract Set
CSIM	Contract Set Information Model
CSIM	Contract Set Information Model
CSS	Contract Set Specification
DM	Domain Model
DMTF	Distributed Management Taskforce
DPE	Distributed Processing Environment
EIM	External Information Model
EIM	External Information Model
GDMO	Guidelines for Definition of Managed Objects
GUI	Graphic User Interface
HTML	HyperText Markup Language
IDL	Interface Definition Language
IES	Inter-Enterprise Services
IOs	Information Objects
ISV	Independent Software Vendor
ISV	Independent Software Vendors
J2EE	Java 2 Enterprise Edition

LDAP	Light Data Access Protocol
MDA	Model-Driven Architecture
MOF	Meta Object Facility
MSM	Management System Model
NFR	Non-Functional Requirement
NGOSS	New Generation Operating System Support
ODF	Open Development Framework
ODL	Object Definition Language
ODP	Open Distributed Processing
OMG	Object Management Group
OSSs	Operational Support Systems
QoS	Quality of Service
RPC	Remote Procedure Call
RUP	Rational Unified Process
SMI	Structure of Management Information
SNMP	Simple Network Management Protocol
SOAP	Simple Object Access Protocol
TL1	Transaction Language 1
TMN	Telecommunication Management Network
TNA	Technology Neutral Architecture
TNCS	Technology Neutral Contract Specification
TNSC	Technology Neutral Contract Specification
TOM	Telecoms Operation Map
TSCS	Technology Specific Contract Specification
TSCS	Technology Specific Contract Specification
UML	Unified Modelling Language
WBEM	Web-Based Enterprise Management
WFMS	Workflow Management System
WWW	World Wide Web
XML	eXtensible Markup Language
XSLT	Extensible Stylesheet Language Transformation

13 Glossary

Term	*Definition*
Attribute Filter	A filter that masks attributes from an IO in an information model from use in a version of that IO that is included in another information model
Building Block	A piece of software that represents a unit of deployment and system management.
Building Block Developer	An architectural user role concerned with the design, implementation and release of Building Blocks
Building Block Group	A set of Building Blocks groups for the purpose of software release
Business Analysis Model	A type of model within the ODF capturing the requirements and the analysis of those requirements for a domain of interest.
Business Context Model	A model that captures requirements and business context for a problem domain.
Business Process Model	A model that captures the structure and process flow of a set of business processes for a particular domain of interest.
Business Role	A representation of a notional organisational agent that has a set of business relationships with other Business Roles
Business Role Model	A model that defines the relationship between a set of Business Roles in terms of the Reference Points that exist between them.
Contract	A general term used to describe an interface to a Building Block
Contract Designer	A notional user role of the ODF responsible for generating contract Specifications and extracting their information content into an External Information Model
Domain Actor	A representation of an element of the environment within which a domain resides
Domain Analysis Model	A model representing
External Information Model	An information model containing IOs from one or more Constract Specifications.
Information Model	A collection of information objects, the associations between them and the constraints that exist on those associations.
Information Object	An information definition within an information model, which exhibits characteristics of attributes and inheritance, but not methods
Interaction Pattern	A definition that characterises a class or style of interaction that may be used in interaction with Contracts and which has an impact on the most convenient form used by in the relevant Contract Specification
Logical Architecture	The portion of the ODF that expresses the structure of the models used throughout the rest of the Framework.
Management System Model	A model that defines the software structure of a management system in terms of Building Blocks, Business Rules, external interfaces, non-BB software and the relationships between them.
Business Reference Model	A model within a Business Context Model that maps elements from a Business Process Model and a Business Role Model form the same

	Business Context Model. The mapping is of Business Processes onto Business Role and of aggregate Business Process Flows onto Reference Points
System Builder	A notional ODF user with responsibility for building management systems and documenting their design in a Management System Model.
Technology Specific Contract Specification (TSCS)	A mapping of a TNCS to a format that may be implemented directly in a specific technology.