Towards the Technology Neutral Modelling of Management Components

David Lewis, Vincent Wade, Brian Cullen,

{Dave.Lewis|Vincent.Wade|Brian.Cullen@cs.tcd.ie}
Trinity College Dublin, Department of Computer Science

Abstract

This paper outlines an Architectural Model and accompanying modelling notation that addresses on the need to model management component interfaces and their business contexts in a technology neutral manner in order to promote convergence on stable, reusable solutions. The approach combines existing modelling concepts related to component-based and model-driven software development from TINA-C, OMG, DMTF and TM Forum in order to provide guidance on the development of models that need to be exchanged between organisations involved in the development of software components and the management systems in which they are used. The Architectural Model is assessed through application to the management a specific set of e-business support services.

Keywords
Components, Contracts, Information Modelling, XML, UML

1 Introduction and Background

E-businesses aim to automate their business processes including the interaction with the business processes of customers and suppliers. They use process flow enactment technologies, e.g. workflow, and systems assembled from reusable off-the-shelf components with well-defined interfaces, in order to rapidly reconfigure their support systems to changing requirements caused by fluid e-commerce value chains. When applied to the delivery of services over networks with quality of service guarantees, such flexible system development must extend to the management of communication services and value added services, e.g. Virtual Private Network (VPN) service or application services. However, the size and complexity of Operational Support Systems (OSS) used in existing communication service providers and the potential range of organisation types involved means that effective integration cannot rely on the use of components from a single vendor, or even reliance on a single set of component interoperability standards. The systems to be integrated will be developed at different times, using different technologies and with adherence to different standards regimes. Currently, such interface specifications often omit important assumptions about their business context and are expressed in languages tailored to implementation using a specific technology. This reduces the specifications' longevity and breadth of application in maintaining interoperability between separately-sourced components as system requirements, the technology base and component capabilities evolve over time.

The problem of early technology binding extends well beyond the communications management domain and is starting to be widely addressed through the promotion of Model Based Development as exemplified by the Open Management Group's (OMG) standardisation of its Model Driven Architecture (MDA) [1]. The MDA emphasises the generation of technology independent models using the Unified Modelling Language (UML) at different points in the software development lifecycle, e.g. requirements modelling, system analysis modelling, design modelling. This enables the binding to a specific implementation technology to be made as close to the implementation stage as possible. The MDA defines mechanisms for exchanging software models based on a standardised meta-model, the Meta Object Facility (MOF) [2], which therefore allows technology independent models to be exchanged and reused between organisations, independently of the different implementation technologies used.

Flavours of technology independent modelling already have been attempted at different levels of abstraction in the communications management domain. The Telecommunication Information

Networking Architecture Consortium (TINA-C) took a modelling approach based on International Standard Organisation's Open Distributed Processing (ODP) [3] architecture applied to both control and management software. ODP provides explicit separation of concerns such that issues related to the distributed processing technology used are separated from the modelling of business requirements, information structures, component interfaces and component compositions. The ODP separations have been successfully used to model management systems in UML [4][5] as well as for assembling reusable model packages for individual components capturing their requirements capture, analysis, design, implementation and test model [6]. The Distributed Management Task Force (DMTF) is successfully standardising the technology neutral Common Information Model (CIM) for enterprise management interfaces. The CIM specifies managed objects and their associations in the Managed Object Format (MOF) language [7] (not to be confused with the OMG's MOF), which can also be partially rendered as UML class diagrams. Separate standards are then developed for defining how CIM-based interfaces can be implemented using different technologies, such as remote procedure calls in the Distributed Computing Environment, directory objects using the Lightweight Directory Access Protocol or web objects encoded in the eXtensible Markup Language (XML) and transported using the Hyper-Text Transport Protocol. Technology neutral requirements and analysis models have been successfully standardised by the TeleManagement Forum (TM Forum), primarily by capturing generic business process models for telecommunications service provider e-businesses in the Telecoms Operation Map (eTOM). The TM Forum is now in the process of integrating the standardisation of the eTOM with the standardisation of component interfaces and of the information that passes over those interfaces. This standardisation work is being conducted under the Next generation Operations Systems and Software (NGOSS) initiative, however work to date has focussed on collecting high level architectural concepts [8]. Little is presented on a methodological guidance for developing and integrating Building Blocks or on how elements that it allows to be standardised separately, namely Business Processes, Contract Specifications and Shared Information Models can be reconciled with each other.

The work presented in this paper draws on TINA, the DMTF's CIM and, most significantly on the NGOSS architecture. This paper does not aim to define a new framework for component-based management system development. Instead it outlines an structure of models that, in combination with UML-based methodology guidelines, have been used to investigate: the relationships between business models, contract interface models and information models; how these models can be developed in a technology neutral manner and how they may best be structured to facilitate the exchange of models and products between stakeholders in component-based management system development. Experiences gained in these investigations have been fed back to NGOSS and DMTF working groups.

2 Architectural Model

The approach taken in this paper does not advocate the generation of a single set of standardised models, but assumes that overlapping standards and proprietary models will exist and will need to converge and be integrated over time. It therefore promotes an Architectural Model consisting of a common set of meta-models so that the differences in business contexts and behavioural semantics that present the greatest obstacles to integrating component models from separate sources, can be readily identified and addressed early in the application assembly process. The use of such a commonly understood set of models is also intended to make model assessment and the selection of corresponding products more transparent and consistent with business requirements, and less influenced by technology hype.

The structure of the models defined in the Architectural Model is driven by the analysis of the interactions between the stakeholders which represent the principle organisation types involved in developing management systems from components. 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 software components sourced both internally and from multiple ISVs.
- *Service Providers*, which possess the business requirements for individual management systems and operate them as part of their Operation Support Systems (OSS).

The Architectural Model captures the structure of models that need to be exchanged between these stakeholders in the development of management components, systems and standards. This includes the identification of the relationships between models elements that must be managed from model development, through to their release, selection and reuse. The resulting ability to readily exchange models related to software interoperability and integration in a non-proprietary form is key to enabling an open market in off the shelf components. The Architectural Model is presented here both in terms of the structural meta-model and in an activity model showing how the various models are exchanged between activities conducted by the various stakeholders. The Architectural Model is accompanied by methodological guidelines [9], which use UML and XML notation and are informed by and allow for convergence with industry best practice, principally the OMG's MDA. These guidelines are not presented here, though some indication is given of how individual models may be rendered.

2.1 Structure of Architectural Model

A software component in the Architectural Model is modelled as a Building Block, which offers one or more interfaces called Contracts, following the model defined in [10] a precursor to the NGOSS work. Here, though, Contracts are packaged with related business requirements, system analysis models and an explicit model of the information that is passed by the Contract.

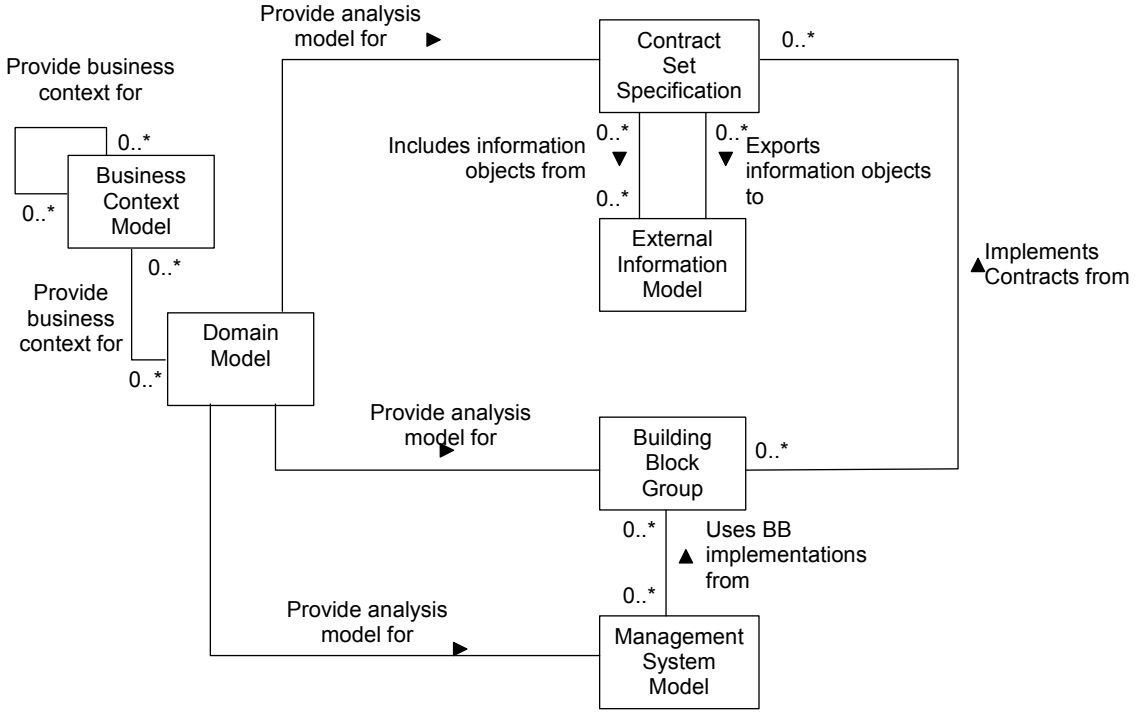


Figure 1: Relationships between main elements of the Architectural Model

Figure 1 gives an overview of the principle models that make up the Architectural Model and the purpose of the links between elements of these models. The following paragraphs provide an overview of the reasons why these are included in the Architectural Model, the structure of their sub-elements, their inter-relationships and their mapping to modelling language artefacts, e.g. UML diagrams. A more detailed description of the Architectural Model is given in [11].

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 two or more derived 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 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 that represent the different types of organisations present in the domain of interest and its business environment, and of Reference Points, each of which represents the possible sets of interactions that may exist between a pair of Business Roles. This model is based on similar concepts used in TINA [12]. The Business Roles and Reference Points reference any Requirements Statements that contribute to their identification. Roles may be represented in a UML class diagram as object classes, and reference points are represented as associations between those object classes.
- *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, such that the business Organisations may be regarded as instances of the Business Roles in the Business Role Model. As such the model may be represented as an UML object instance diagram. 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 in the same manner as taken in the TM Forum's eTOM. The model identifies business process areas and end-to-end business process flows, modelled as a UML activity diagrams, identifying specific activities in different process areas that are chained together to perform a use case. Business Processes reference the Use Cases or Requirements Statements from which they are derived. Business Processes can be mapped to Role instances in the Business Organisational Model to provide a multi-domain process map for a specific scenario.

Though the BCM uses modelling concepts from the eTOM and the TINA Business Role Model it is not intended to prescribe single standardised version of such models as the related business requirements are too volatile to generalise. Instead this BCM provides a tool for different organisations to represent business requirements in a common manner that may then be easily accessed by users of related models.

Domain Model

The *Domain Model (DM)* expresses an analysis of the requirements and business environment captured in a single BCM. It therefore provides a 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 primarily to provide a common mechanism

for tracing between instances of these models and the BCM(s) from which they are derived. The DM consists of the following sub-elements:

- The *Domain Use Case Model* and the *Domain Process Model* are refinements of the corresponding models in the BCM, which address system level concerns.
- 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 both static and dynamic models of the domain's logical structure. These are modelled using UML class diagrams with boundary, control and entity class stereotypes, supplemented by interactions diagrams showing the interactions between instances of these classes needs to accomplish specific Domain Use Cases.

Contract Set Specification

The *Contract Set Specification (CSS)* expresses one or more related interface 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 thus encouraging the reuse of individual Contract Specifications across separately developed Building Block implementations.

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 may be derived from entity objects in the relevant Domain Analysis Model.

Ideally, the Contract Specification captures the binding between technology neutral models and technology specific ones used in actual software implementations, i.e. Building Blocks and Management Systems. There is a requirement therefore for a language for defining Technology Neutral Contract Specifications (TNCS), i.e. which captures the models needed for interoperability, but in a technology neutral manner. This language should support transformation to a Technology Specific Contract Specification (TSCS) using a deterministic language transform for a specific interface technology, resulting, for example in IDL or GDMO specifications. The selection of a suitable TNCS language is addresses under further work in Section 5.

As a first step toward designing a full TNCS language, the Architectural Model has defined a technology neutral format for the information model part of a Contract, which may be used for the CSIM. This was derived from the meta-schema defined for the DMTF's CIM, but simplified for use in defining sharable information models in UML. It uses the same attribute type system as the CIM (simple and array types only), the same use of different association types and of constraints on attribute value ranges and conditions on associations. From the CIM meta-schema it excludes; support for object instance specifications, the use of qualifiers, class methods, and notification/event definitions, the latter two being related to the functional aspects of a contract definition and therefore delegated to the full TNCS language.

External Information Model

The *External Information Model (EIM)* expresses information that may be passed via Contracts, but documented separately from the relevant Contract Specifications so as 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. In this way a federated population of EIMs from different sources may emerge, allowing their comparison and, where appropriate, encouraging their integration. As EIMs stabilise, the information content of individual Contract Specifications should increasingly consist of references to EIM elements. 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.

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, and the externally visible behaviour of the BB. 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 by ISVs to System Integrators. The BBG also supports better the situations where there are functional dependencies between BB that must be maintained, i.e. when BBs have a close design coupling.

Within a BBG, each BB is represented by a Building Block Descriptor that 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. The Descriptor may also reference other CSSs for Contract Specifications that are used by the BB. 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. As the Building Block descriptor is largly a container for reference to other models, UML package diagrams may be used, though an XML document may also be appropriate, especially for publishing the descriptor.

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. 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 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, though they lack the component features, such as acting as a unit of distribution, that allow them to be 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 operational environment.

The structure and relationship of these sub-elements of the Architectural Model are summarised in figure 2.

2.2 Use of Architectural Model

This section examines how the elements from the Architectural Model are used within the stakeholders and for exchange of models between them. To fully understand the use of the Architectural Model, a clear picture is needed of the relationships between the Architectural Model Stakeholders. Over time, 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 analysis of 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.

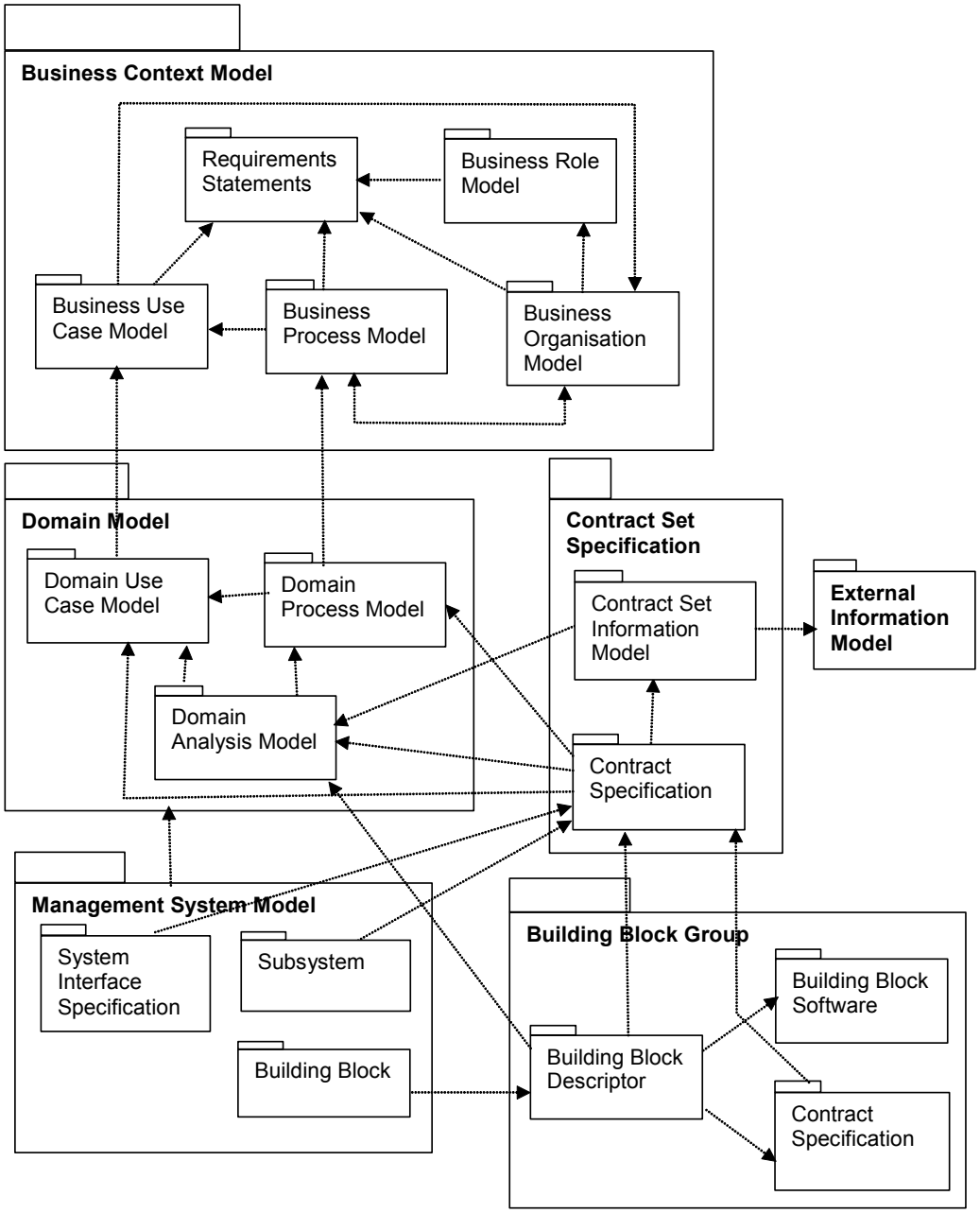


Figure 2: Structure and relationships of sub-elements of Architectural Model

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 by the 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 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 which may be used in BB products. ISVs may publish their CCSs and EIMS to aid and encourage the selection of their BB products. Where ISVs have popular product ranges, this publication may encourage smaller ISVs to market compatible niche products.

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.

By having the same activities operating in more than one stakeholder, the commonality in the information passed between the stakeholders is more easily analysed and identified. Figure 3 depicts how the main elements of the Architectural Model are generated by the different activities operating in the various stakeholders. The activities shown may be specialised and integrated with other development activities to suit the internal methodologies of the organisation concerned.

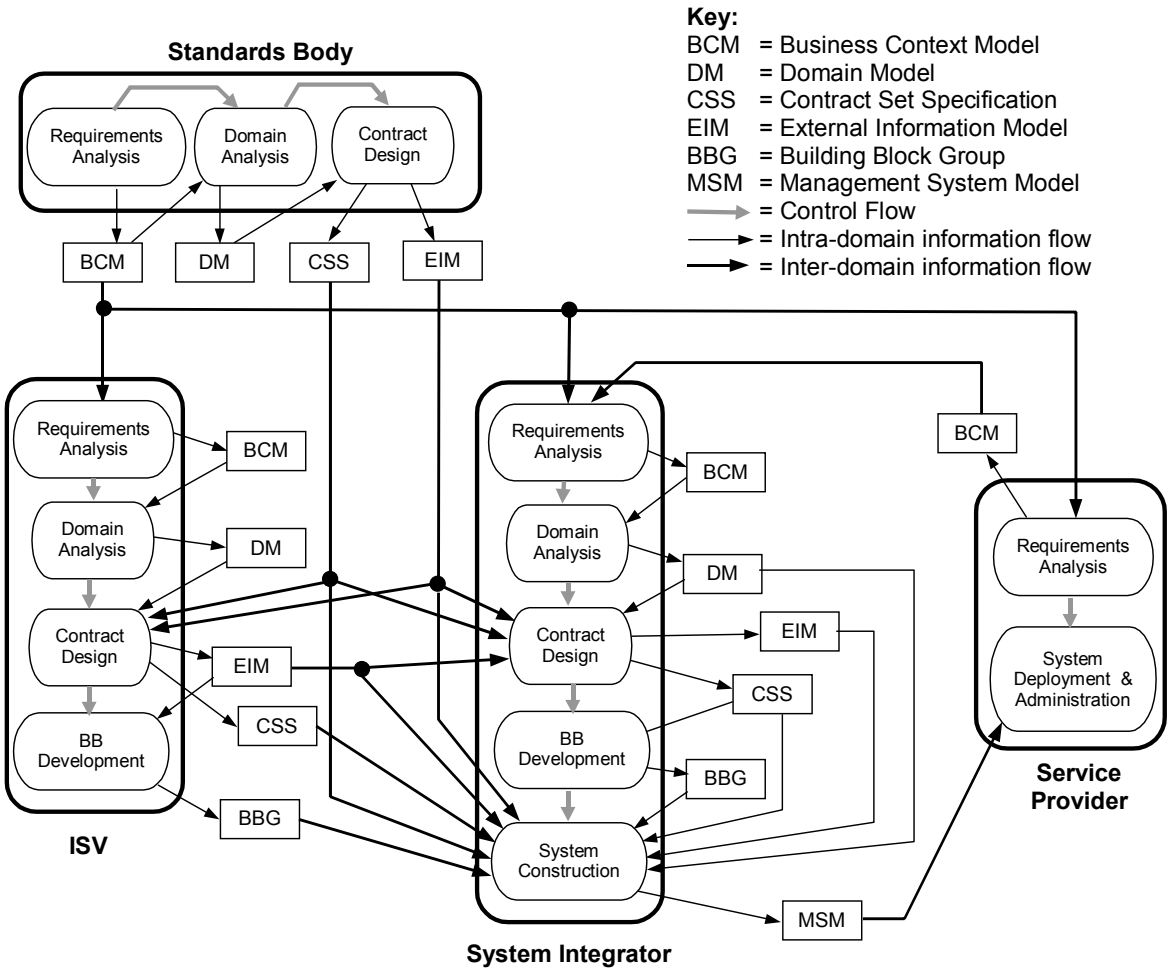


Figure 3: Activity diagram showing the exchange of models between stakeholder types

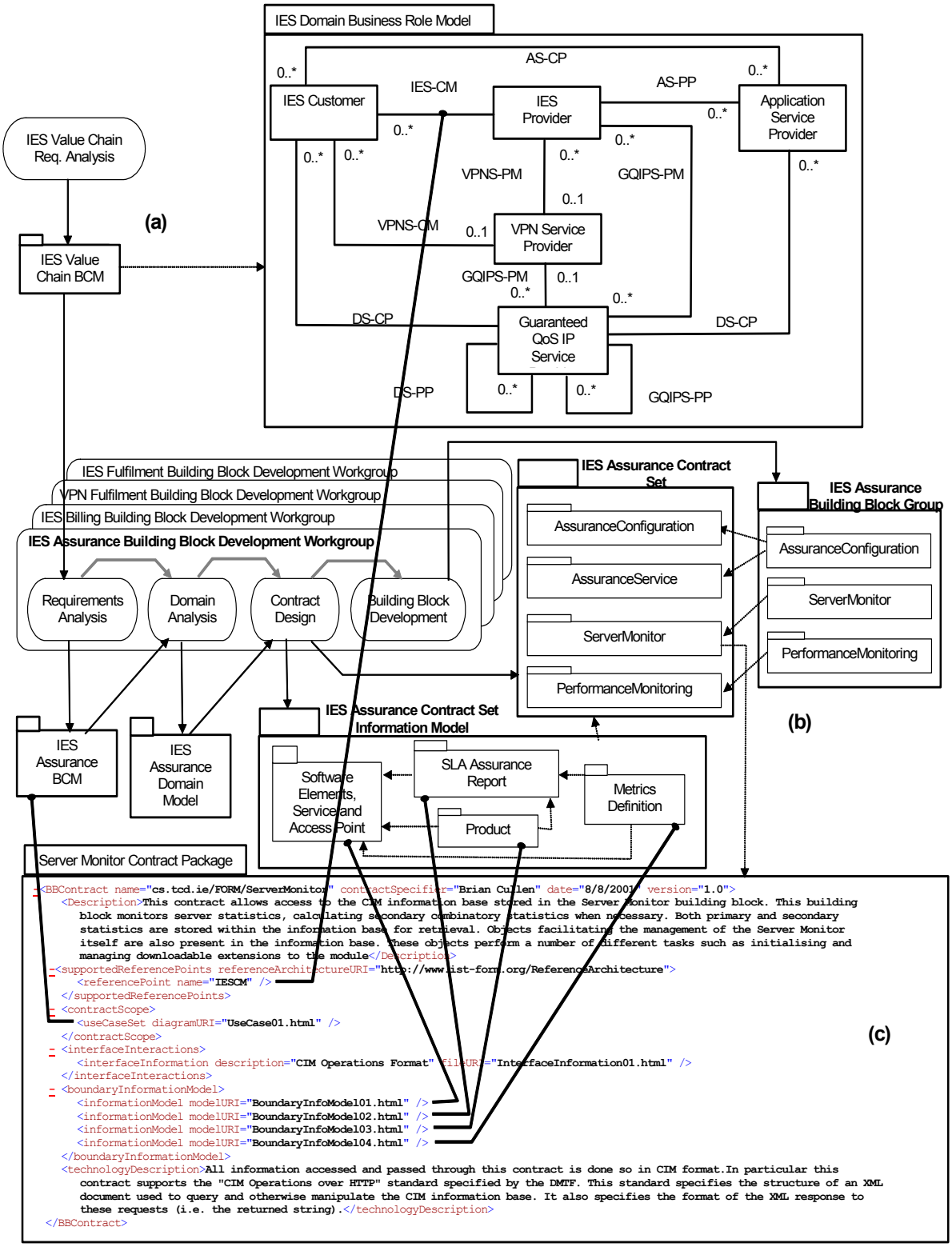
3 Trial Application

The Architectural Model has been partially applied and evaluated in the EU funded project FORM. This project has developed prototype management systems targeted at the management of service chains between e-businesses. The scenario developed is of a service chain built around an *Inter-Enterprise Service* (IES), which enables enterprises to create, reconfigure and dissolve business

collaborations rapidly and flexibly. The IES domain addresses how an IES Provider may manage the dynamic relationships and the quality of service (QoS) of electronic business interactions between groups of Customer enterprises, the Application Service Providers (ASPs) that they use and the Internet Service Providers (ISPs) which provide communications infrastructure. The IES exploits the use of end-to-end network quality of service and security management service offered by a VPN service provider.

The aim of this trial development was to test the compatibility of elements of the Architectural Model with the development of practical components and the use of existing system development techniques. The developers involved in the trial used the Architectural Model in terms of development steps and modelling notations presented in the accompanying methodological guidelines. The two guidelines used in this assessment address the development of Building Blocks by ISVs or System Integrators and the development of Management Systems using Building Blocks by System Integrators. These guidelines focussed on the internal development processes and notations needed by these stakeholders as it was here that the Architectural Model must prove practical before the benefits of its use in exchanging products between stakeholders can be realised. The primary modelling notation used was UML, though XML was used for packaging UML diagrams with other model formats in Contract and BB specifications. The Rational Unified Process was used as a partial template to integrate these techniques.

The trial involved 14 developers from 10 different companies and educational institutes across Europe, who were allocated to various development teams. A consensus was first developed across the group on a BCM (see Figure 4a) for the overall IES value chain, identifying Business Roles for an IES Customer, an IES Provider, an ASP, a VPN Service and a Guaranteed QoS Internet Service Provider. A business scenario was captured with a Business Organisation Model defining a value chain with at least one organisation operating as each one of the Business Roles and a high-level Business Use Case describing how they interact in providing a web-based multimedia service from an ASP, via several ISPs which were coordinated by the IES Provider, with QoS guarantees and security concerns addressed by a VPN service provider. Business processes to enact this use case where captured in a Business Process Model, based loosely on the TM Forum's eTOM. This Business Process Model was used to subdivide the scenario for the attention of four separate Building Block development groups, namely one addressing fulfilment of orders to the IES provider, one addressing fulfilment of orders to the VPN Service Provider, one addressing assurance of the multi-media service over the value chain and one addressing the billing for this service based on usage record collected along the service chain. These workgroups followed the Building Block Development Guidelines to develop Contracts, Information Models and Building Blocks for their allotted Business Process Flow. Figure 4b gives some indication of the Contract, Information Model and Building Block Models developed as part of the IES Assurance Process Flow. Figure 4c gives an example of one the XML Contract Specifications highlighting the links back to BCM elements and information models. These XML Contract Specification were transformed to HTML and published on the FORM website (www.ist-form.org) where they became a useful tool in the subsequent Management System Development conducted by the workgroups. Contracts were not specified in a full technology neutral form due to the lack of a suitable TNCS language, however the information exchanged by each Contract was explicitly modelled in the recommended technology neutral information modelling format. As Contracts were modelled individually, rather than in sets, so the information models pertained to individual Contracts rather than Contract Sets. Contracts were implemented in a range of technologies, including XML/HTTP message passing and Java RMI and CORBA interfaces, some of which were simply passing XML documents.




```
<BBContract name="cs.tcd.ie/FORM/ServerMonitor" contractSpecifier="Brian Cullen" date="8/8/2001" version="1.0">
  <Description>This contract allows access to the CIM information base stored in the Server Monitor building block. This building
    block monitors server statistics, calculating secondary combinatory statistics when necessary. Both primary and secondary
    statistics are stored within the information base for retrieval. Objects facilitating the management of the Server Monitor
    itself are also present in the information base. These objects perform a number of different tasks such as initialising and
    managing downloadable extensions to the module</Description>
  <supportedReferencePoints referenceArchitectureURI="http://www.ist-form.org/ReferenceArchitecture">
    <referencePoint name="IESCM" />
  </supportedReferencePoints>
  <contractScope>
    <useCaseSet diagramURI="UseCase01.html" />
  </contractScope>
  <interfaceInteractions>
    <interfaceInformation description="CIM Operations Format" fileURI="InterfaceInformation01.html" />
  </interfaceInteractions>
  <boundaryInformationModel>
    <informationModel modelURI="BoundaryInfoModel01.html" />
    <informationModel modelURI="BoundaryInfoModel02.html" />
    <informationModel modelURI="BoundaryInfoModel03.html" />
    <informationModel modelURI="BoundaryInfoModel04.html" />
  </boundaryInformationModel>
  <technologyDescription>All information accessed and passed through this contract is done so in CIM format.In particular this
    contract supports the "CIM Operations over HTTP" standard specified by the DMTF. This standard specifies the structure of an XML
    document used to query and otherwise manipulate the CIM information base. It also specifies the format of the XML response to
    these requests (i.e. the returned string).</technologyDescription>
</BBContract>
```

Figure 4: Models involved in developing IES Assurance Building Blocks

In parallel to this Building Block development exercise, three teams performed a paper analysis of a further three process flows derived from the overall IES BCM. These were deliberately selected to

intersect the process flows being addressed by the BB development teams, e.g. addressing the passing of Service Level Agreement (SLA) terms from IES fulfilment processes to IES assurance processes which would monitor them, or passing information about a SLA violations from the IES assurance processes to IES Billing processes where an appropriate customer discount could be calculated. This exercise provided some convergence between the Contract information models used within each of the BB development groups, especially on common information such as SLAs.

The Management Systems were developed as demonstration prototypes for exercising the components, and were also modelled in accordance with the development guidelines. There was therefore no attempt to reuse the BBs as third party components or to select them through analysis of their Contract Specifications or associated EIMs. Management System designs consisted both of Building Blocks and their Contracts as well as non-Building Block subsystems. The Management Systems developed all had the majority of their functionally implemented through BBs, 80% on average. Non-BB subsystems typically related to user interaction or service delivery components.

In addition to these trial developments, an analysis was conducted to assess the extent to which a common External Information Model could be constructed from the Contract Information Models developed within the separate trial working groups. This observed that within the models from the different trial workgroups the Contracts developed all shared information content, ranging from common definitions of specific information objects, such as SLA, to common definitions for all information objects exchanged via their Contracts. Some resistance was observed, however, to proving information models conforming to common modelling rules, especially when information models could be readily extracted from the technology specific Contract specifications, resulting in complex attribute types and object methods being included. Little use was made of constraints applied to associations between classes or to values ranges for attributes. Some differences were noted between developers in representing essentially the same information with different sets of information objects. The differences centre on how explicitly the associations between information objects, including aggregations, are modelled. This seemed to be influenced by the technology specific specification from which the information model was derived, with mappings from XML documents resulting in explicit ‘container’ classes compared to those derived from manage object definitions where containment is implied. It is therefore recommended that where possible, information object that merely represent relationships between other objects should not be used in UML class diagrams, with the association being labels with a suitable identifier instead.

The analysis revealed no specific inconsistencies between information objects in different Contract information models. However there are cases where this is due to weak typing in one information model compared to another. In particular, the SLA definition used in the IES Fulfilment group is aimed to be very flexible, since it must accommodate SLA terms that cannot be known in advance. This is handled by supporting later binding of additional SLA template and SLA negotiation policy information in the SLA Negotiation BB, where the SLA information is defined. This SLA therefore uses name-type-value triples to support these terms at the SLA negotiation related Contracts. Other Contracts that use terms from the SLA, however, have more strongly typed definitions. Though these are not inconsistent with the SLA definition, due to the latter's weak typing, they must be reflected in business rule models accompanying the SLA negotiation BBs.

4 Trial Results

The trial gave a number of insights into the problems of developing and exchanging models between different development teams. A total of 18 Contracts were defined, with an average number of information object per contract of approximately 11 and the average number of operation supported per contract at approximately 5. This gives a rough guide to the complexity of Contracts that resulted from the approach taken and granularity of the corresponding BBs. However, these figures should be used with caution as further discussion with developers revealed the interpretation of these figures depended on the developer’s approach to information modelling and their interpretation of what an ‘operation’ was. Concerning information modelling, the non-conformant use of complex types hid a large number of Information Objects than would have resulted if the information guidelines had been

followed more closely. Equally, even fundamentally identical information models were open to representation with varying numbers of UML class objects. Concerning the definition of Contract 'operations', this was not clearly defined in the Architectural Model and was thus interpreted differently. For instance, a Contract interface signature may have a single method called something like 'process message' but then the message structure contains elements, e.g. message name, for which different values imply different operations at a business level. A clearer definition of a Contract operation is required that ties it to individual Domain Model use cases.

The use of a CIM-based technology neutral model for capturing the information elements of contracts was found useful in achieving some consistency between the technology specific contracts developed using different technologies by different organisations. However, extracting the information models from the technology specific contract specifications was regarded as an overhead by developers and the models obtained sometimes diverged from the modelling guidelines or had structures that that traded simplicity for ease of extraction from the technology specific language. This points to the need for a clearer analysis of the return on investment obtained through this technology neutral information modelling approach and a more effective set of modelling guidelines and accompanying quality assurance techniques.

5 Conclusions and Further Work

The Architectural Model and associated notations presented in this paper represents a step towards a common, technology neutral approach to the modelling of interoperable management components. It aims to provide a simple, minimum set of modelling guidelines that supports the exchange and comprehension of models between organisations involved in the development of integrated component-based OSS.

Trial application of the Architectural Model has shown that the modelling of the business context of a product and its linking to the design model via system analysis level models (i.e. the Domain Model) can be performed with existing, technology neutral models using UML. The models used take their lead from existing industry best-practice using business process, use case and model-view-controller modelling techniques.

This work also helps refine requirements for further work needed to achieve full technology neutral modelling, in particular a modelling language needs to be defined for expressing TNCSs. The Web Service Description Language (WSDL) being standardised by the World Wide Web consortium offers a promising solution. It uses XML to define interfaces separately to the mapping of the interface to a specific communication protocol. However, though WSDL clearly support remote procedure call and message passing communication paradigms, it use for manager-agent style interfaces in untried. This may require a more structured approach to defining the information content of WSDL interfaces, currently expressed using the XML Schema language, which would need to be compatible with the object-oriented expression of information models in the Architectural Model.

The Architectural Model is deficient in not addressing test models, which is needed for support the full reuse lifecycle of Building Blocks. In addition, no support is given to the deployment-time configuration of a Building Block to meet security, system management, performance and failure recovery policies applied to the Management System in which it is applied. The integrating of a standardised expression of policies, e.g. [13], with Building Block models therefore requires further investigation.

Acknowledgements

This work has benefited from insightful comments from many members of the FORM consortium (www.ist-form.org), as well as from Tony Richardson, Declan O'Sullivan, Paul Doyle, Karl-Heinz Weiss and the paper's anonymous reviewers.

References

[1] Model Driven Architecture, A Technical Perspective, OMG Architecture Board, Review Draft, Doc number ab/2001-02-01, 14th February 2001,

[2] Meta Object Facility, revised submission, ad/97-08-14, OMG, August 1997

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

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

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

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

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

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

[9] V. Wade (ed), D12 - Guidelines for Co-operative Inter-Enterprise Management, IST-1999-10357/TCD/WP3/012, February 2002.

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

[11] Eric Leray (ed), D9 - Final FORM Framework, IST-1999-10357/WIT/WP3/1019, March 2002

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

[13] B. Moore, L. Rafalow, Y. Ramberg, Y. Snir, J. Strassner, A. Westerinen, R. Chadha, M. Brunner, R. Cohen, Policy Core Information Model Extensions, Internet Draft <draft-ietf-policy-pcim-ext-01.txt>, April 2001.