AeroSpace and Defence

Industries Association of Europe

ASD STRATEGIC STANDARDISATION GROUP

MoSSEC Overview

Rev0.3, 22/11/2013

Ref: **ASD/MoSSEC/1.0**

Confidentiality level: **Public document**

Executive Summary

This document introduces the scope of the proposed MoSSEC standard which is designed to provide a capability to share Modeling and Simulation information in a collaborative Systems Engineering Context "MoSSEC". This is to enable full traceability and re-use of modeling and simulation information throughout the product lifecycle and independent of the specific IT applications used across collaborating enterprises. The standard targets to cover a core subset of AP239 systems engineering information content and related information services, that can be readily implemented and deployed to support engineering collaboration.

The MoSSEC project is proposed as an international project, to bring together the works started in Europe by CRESCENDO Consortium [5] to apply AP239/PLCS for this scope, and the similar works in Europe and US motivated by similar goals.

The globalization of aerospace and defence industries drives large volumes of work for modelling and simulation of product performance and behaviour, into geographically distributed teams and into the supply chain. This data is used to justify change decisions and to validate the product throughout development, certification and in-service. The MoSSEC standard enables a proper exchange and sharing of modelling and simulation data with traceability to its systems engineering & PDM context. This enables competitive and robust product development in global teams, where modelling and simulation data is fully traceable to the PDM referential, and enabling quick validation of next design changes, whether in product development or in-service phases.

List of Contributors

	Name	Organisation	Role / Title
Document Leader	Adrian Murton	Airbus	MoSSEC Project Leader
Contributing Author(s)	Peter Coleman	Airbus	CRESCENDO Project Leader
	Judith Crockford	Airbus	
	Steve Allwright	Airbus	
	Phil Spiby	Eurostep	
	Nigel Shaw	Eurostep	
	Yves Baudier	EADS	

Document History

Version	Date	Reason of change
0.1	08/07/2013	Document created
0.2	21/10/2013	New document name, links to CRESCENDO server removed

0.3	22/11/2013	Integration of updates provided by Judith Crockford and Steve Allwright

Content

List of Figures

1 Introduction

For many years, industry sectors like Automotive, Aerospace, Construction, and Nuclear have been investigating new ways to manage their Modelling and Simulation (M&S) assets and to increase the effectiveness of their analysis engineering activities. The maturity of Simulation Data Management (SDM) technology and tools (part of a global approach known as Product Life-cycle Management PLM) has enabled several companies to deploy tools and methods to capture traceability of their simulation processes and related data. These companies now need to connect their SDM environments to M&S partner environments, to operate in a global Extended Enterprise context, while preserving the Intellectual Property (IP) of the models they exchange.

The MoSSEC standard uses a core subset of AP239 to enable a capability to share modelling and simulation information in a collaborative systems engineering context, and to enable full traceability and re-use of this information throughout the product lifecycle and independent of the specific IT applications used across collaborating enterprises for Simulation Data Management (SDM). The MoSSEC standard will provide the structure for the information in the distributed dataset, the services to expose that dataset, and the data exchange specification as illustrated in Figure 1 below.

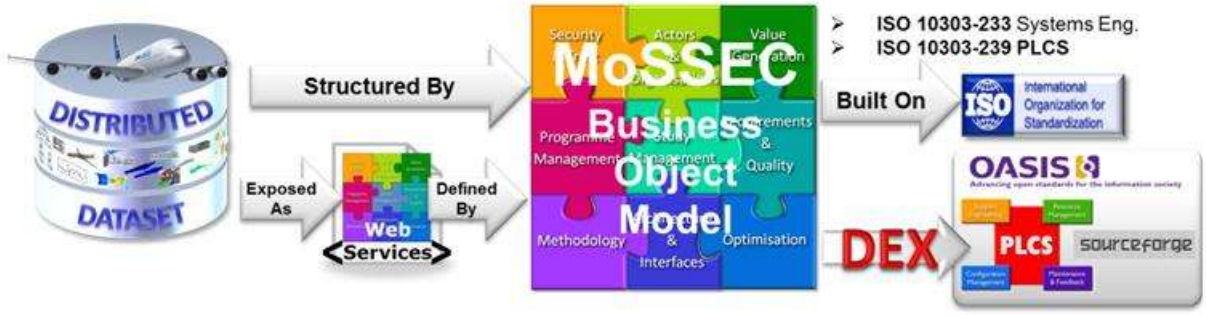


Figure 1 MoSSEC standard for distributed dataset

This MoSSEC Overview is consolidated from the results of the CRESCENDO European project: the Behavioural Digital Aircraft [5]. CRESCENDO was an Aeronautical-domain project, gathering more than 60 Aeronautical companies, research centres and IT vendors. The intent is to share these specifications with other Industrial Sectors, to increase their coverage and maturity and, at the end, ensure their implementation in commercial SDM solutions.

The document is structured with the following chapters:

- Chapter 2 describes the context, including the SDM interoperability paradigm and the related MoSSEC Business Object Model that identifies the MoSSEC objects that need to be shared and exchanged.
- Chapter 3 describes the MoSSEC Data Exchange (DEX) Specifications
- Chapter 4 describes the MoSSEC web services. These are further developed in Annex 2.

2 Context

2.1 SDM Interoperability

Figure 2 shows a scenario of three collaborating companies, an OEM, a Partner and a Supplier, though this is also applicable to collaborating teams within a company. It shows each company or team having their own SDM technology to connect their specialist modelling and simulation methods and tools that support the different behaviours of the digital product. Each company or team can also have their own data repositories for the technical data, and for the collaborative System Engineering context data. There are many different aspects to the collaborative data, and this is represented in the figure as coloured jigsaw pieces making up the whole, and is the concern of the MoSSEC standardisation.

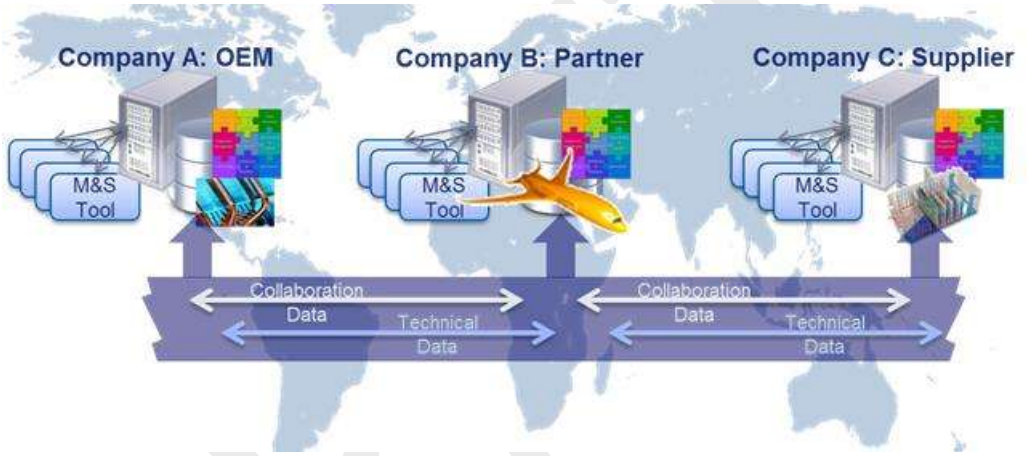


Figure 2: Functional View : Competing and complimentary solutions

The companies agree on the context for their collaboration which is represented by the wide dark blue arrows connecting them. The Collaboration and Technical data is then shared and exchanged within this context (represented by the narrow arrows), which allows the connection and control of the SDM dataset including the global processes. The standard for sharing and exchange of the collaboration data (MoSSEC) is built on the ISO standards 10303-233:2012 (Systems Engineering) and 10303:239:2012 (Product Lifecycle Support). Whereas sharing and exchange of technical data (e.g. finite element files, geometry files and other specialist data) can use existing technical standards such as STEP AP242 or AP209, or can use other representations and formats as agreed for the collaboration context.

The distinction between these two types of data is illustrated in Figure 3 below.

Figure 3: Sharing and exchange of data using MoSSEC and Technical Standards

Publishing the standards for sharing and exchange of collaboration data provides open interfaces for communication between platforms, and promotes open competition between platform vendors. This allows teams and companies to use the combination of platforms that best suits their range of M&S needs while ensuring a high level of inter-operability to support the high level SE process.

Some examples of platforms involved in CRESCENDO are:

- Generic SDM Solutions:
  - Enovia (Dassault Systèmes)
  - SimManager (MSC)
  - TeamCenter (Siemens)
  - Share-A-space (Eurostep)
  - Virtual Lab (LMS)

- Specialist M&S Tools
  - CADFix (TranscenData)
  - Adams (MSC)
  - Nastran (MSC)
  - Proosis (EcosimPro)
  - CatiaV6 (Dassault Systèmes)
  - NX (Siemens)

Some examples of federated heterogeneous platforms collaborating using early versions of the BDA/MoSSEC concept are shown in Figure 4, and are presented in Refs [5], [6] & [7].

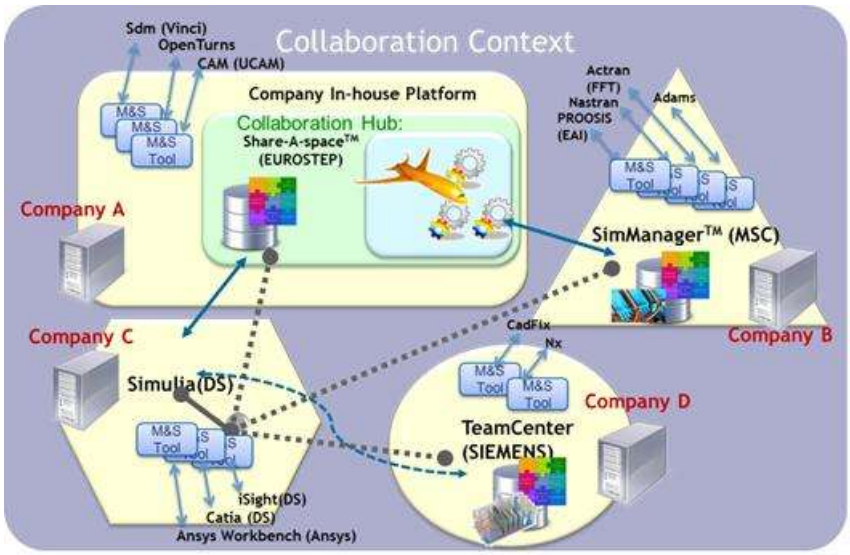


Figure 4: Heterogeneous platforms collaborating using MoSSEC standards

As mentioned above there is a collaboration context (represented by the blue background) within which the trusted organisations are collaborating. "Company A" has an in-house developed platform and is sponsoring the (green) collaboration hub which is provided by Share-A-space™. It has connections to several M&S tools accessible from the Hub. "Company B" is using SimManager as the collaboration platform which communicates with the Collaboration Hub from "Company A". "Company C" and "D" are using other collaborating platforms, and this is represented by using different shapes (triangle, hexagon etc) for their platforms.

The narrow blue arrows represent high/low bandwidth connections in the extended enterprise using the MoSSEC web-services. The grey connections show technical tool integration partnerships.

The data on these platforms together with their ability to communicate with each other allows them together to manage the evolving distributed dataset, as illustrated in Figure 5 below.

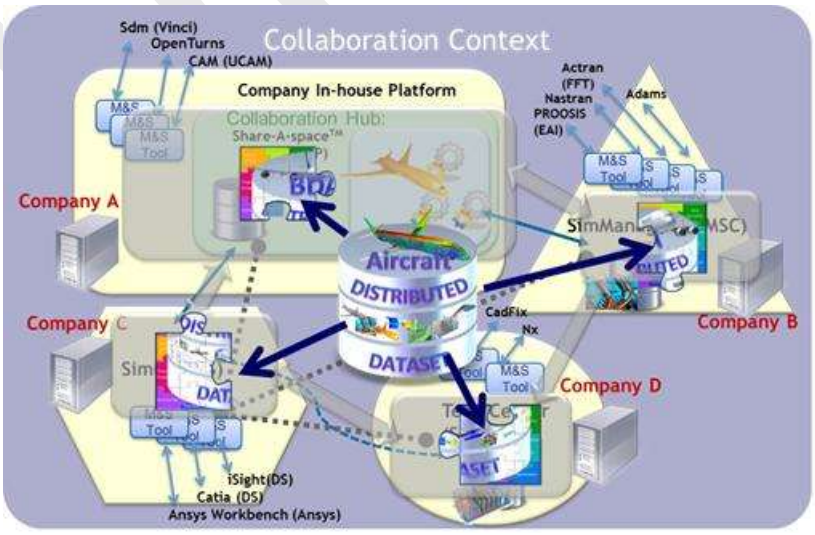


Figure 5: Distributed MoSSEC Dataset

2.2 MoSSEC Business Object Model

The Business Object Model for Modelling and simulation in a collaborative system engineering context (MoSSEC BOM) is a model of objects, properties, relationships and operations that represent MoSSEC domain entities. The basis for the MoSSEC BOM was a deliverable of the CRESCENDO project (also known as "CRESCENDO BDA BOM").

2.2.1 Introduction to the most commonly used objects

Table 1 lists the commonly used objects and shows the images that are used in the following chapters and figures to represent the different objects.

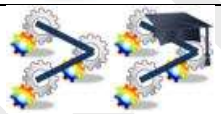
Object Name	Object image
Study	
Associative Model Network [AMN] and Collaborative Model Template [CMT]	
Model Instance and Model Type	
Key Value Instance and Key Value Type	
Requirement, should be satisfied by, and verification	
Method and Tool	
Organisation, Type of Organisation, Person and Type of Person	
Methodology Library	
Approval	
Documents (e.g. managed in a PDM system)	

Table 1 Commonly used objects and corresponding images

2.2.1.1 Study

A collaborative study is package of work that is launched by a Programme to drive the design, modelling, simulation and verification of something. Studies can launch multiple nested (sub)studies allowing complex product/system engineering activities and datasets to be organised and managed.

Studies can have different purposes, e.g. evolving the design, managing change, performing trade-off analysis, investigating sensitivity of a solution to uncertainty, performing optimisation, developing the detail of the design, developing new methods and tools, combining results from other studies into a single baseline.

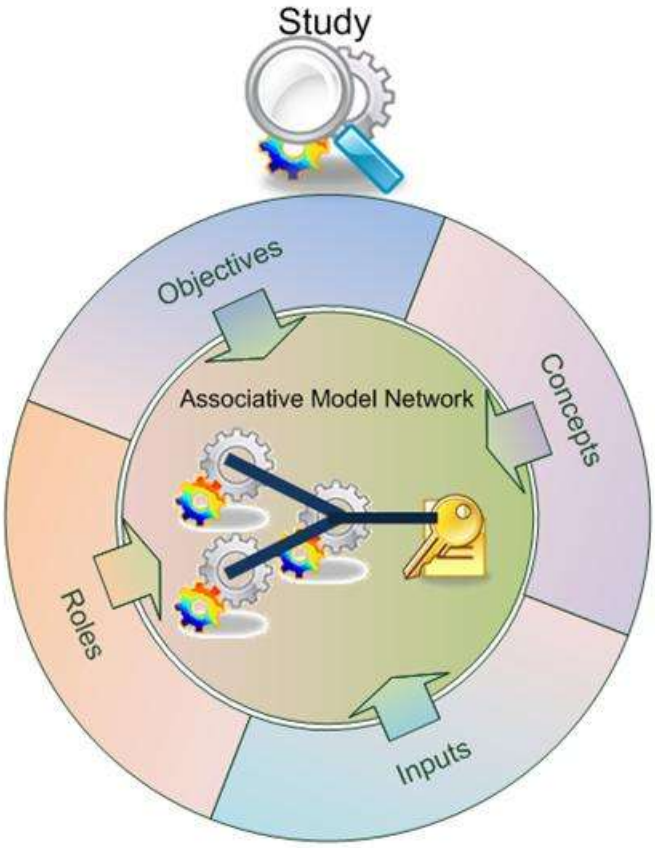


Figure 6 Illustration of a study with related information

As illustrated in Figure 6 above, a study has:

- **Concepts** to be investigated. These can be models or documents that describe the concepts to be investigated : e.g. Technology Standards, Design Concepts, Material Specifications etc.
- **Objectives** to be met. These are a description of what is to be achieved by the study, or references to formally managed product requirements and targets, i.e. to references to systems engineering context.
- **Inputs** to start from. These are existing designs and results from earlier studies that are to be evolved during the course of this study, i.e. product information context. They can also be methodology objects to use as templates (see below).
- **Roles** of people and organisations. This allows the work in a study to be assigned or delegated to other people, teams or organisations.
- The **package of work** to be controlled and managed. This inter-related set of activities and models is recognised as an Associative Model Network (see below) that defines the models to be created and their associativity to each other and to the product/system engineering context. It can also include the proposed process, method and tool to use for each model.

The relationships are flexible to support the different purposes of studies, and to support proper traceability of design information as it is progressively matured and integrated from successive studies.

2.2.1.2 Associative Model Network [AMN]

An Associative Model Network [AMN] is a container that identifies all the elements that together represent the set of activities and results for a study. The associativity between these elements firstly represents the evolving plan of what is to be done, and finally represents the audit-trail of what has been done. Each element in the AMN has an understanding of its dependencies (what it is derived from), and so together they make up a network of associative models. These allow to plan and record the "who", "what", "where", "when", "how" and "why". The dependencies can link to elements outside of the AMN, so enabling proper interconnection to previous results and to systems engineering context.

The high-level representation of both the work-package activities and results combined in a single Associative Model Network has a number of advantages

- efficiency : the single AMN is both the plan of the high-level process and is also the plan of the information asset to be delivered by the study. It implicitly supports basic systems engineering practices and ensures an SE compliant audit trail and robust decision basis. The detailed workflow at individual model level, whether automated or manual, is delegated to the accountable organisation and actor at that next level. The network provides the proper framework to manage collaboration with actions and results always managed together.
- flexibility : the AMN view supports Boolean operations, allowing direct simplification of process when certain information results are not demanded, and most importantly, supporting integration of the overall product information set and verification processes during the ascending leg of the V-cycle.
- agility : the AMN view links design, model, simulation and verification data to requirements and targets ensuring rapid and complete traceability to manage the challenge, evolution and validation of overall product requirements and targets. This allows the animation of high-level processes to combine both “state-based” and “process-based” triggers that provides the basis to ensure both agile and appropriate response to change in context.

As illustrated in Figure 7 below, the information from the managing study is linked to the Associative Model Network as follows:

- Concepts: These are linked to the models and key values that implement the concept: for example it could be a new material specification that is to be investigated for its thermal behaviour in an existing design.
- Objectives : These are linked to models, key values or methods that will be used as evidence that the objective or requirement is satisfied (or not)
- Inputs : These are existing models, key values and requirements that give the product information context for the new activity. The new models in the associative model network will link back to this input data.

- Roles : All studies and models are linked to the roles, organisations and actors that perform the work, from planning through to the audit trail record, e.g. who is to manage, who is to execute, who wants to watch, who actually did it, who is to approve, who is to authorise release, etc.

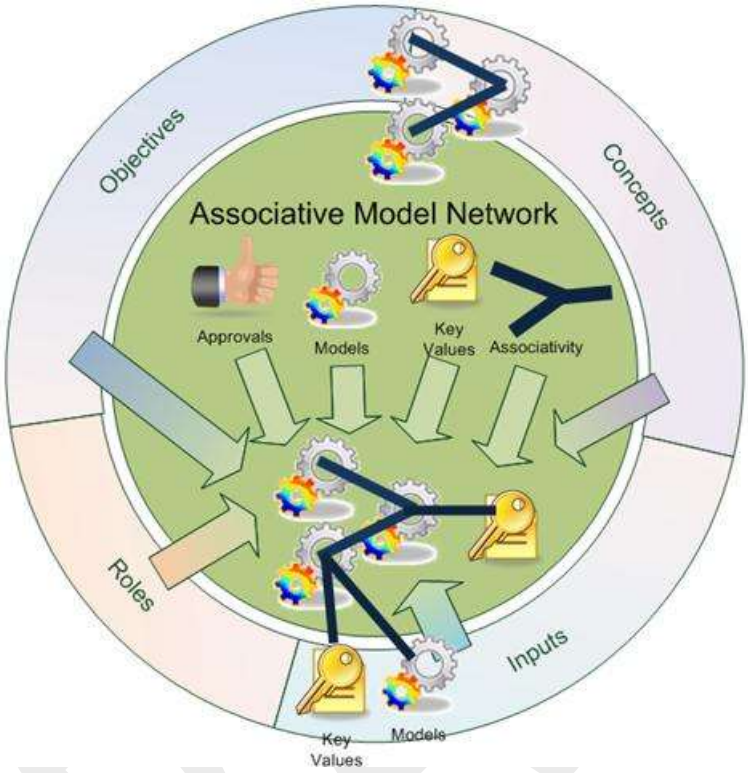


Figure 7: Associative Model Network content with links to study information

A template version of the Associative Model Network is called a Collaborative Model Template. This allows key collaborative processes to be managed as a methodology resource for repeated and systematic deployment (it is described in Chapter 2.2.1.5 below).

2.2.1.3 Model Instance

A model instance is the primary object that is used to identify a specific activity and its resulting product information e.g. numerical data, behavioural data, geometrical data etc. A model instance has a lifecycle, starting as an entity to plan the work to be done and concluding when the resulting model data is associated to the model instance using documents.

The information content of this model can be at any level of granularity according to the purpose of that activity in the overall collaborative process, for example:

- a simple data file comprising single parameters, or lists and arrays of data
- a model representing the product and how it behaves physically
- a simulation results file detailing technical performance and behaviour data
- a report summarising the technical performance vis-a-vis the objective and requirements

The format of the document can be standards based or any other format that is fit for purpose.

The definition form of the model instance is a model type described in Chapter 2.2.1.5 below.

A model instance belongs to an Associative Model Network (AMN) as described in chapter 2.2.1.2 above. It is derived from other model instances and key value instances (note these do not have to belong to the same AMN). It can also identify the model instance from which it is evolved. This evolution is similar to versioning, but allows an evolution tree to be constructed (as illustrated in Figure 8 below) rather than linear evolution implied by versioning.

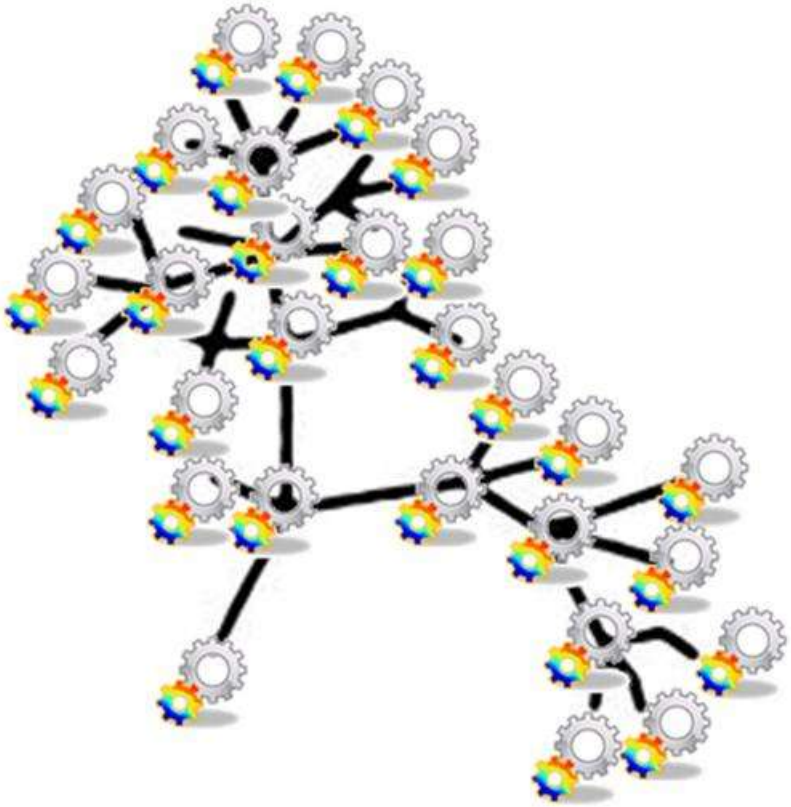


Figure 8 Model evolution is used rather than versioning to allow an evolution tree

2.2.1.4 Key Value Instance

Key value instances are significant values that need to be exposed in an Associative Model Network, e.g. for display on dashboards, or for monitoring and control of the network. These are not properties of models because flexibility is needed over which models they are extracted from, so allowing different studies to compute the same key value with different levels of fidelity, or maturity. For example a component "weight" value can be computed using an approximation model in a concept phase study, and another instance of the same component weight can be computed in a detail design phase using a complex geometric model.

As with Model Instances, the key values do not have versions, but can record their evolution.

The definition of the key value instance is a key value type described in Chapter 2.2.1.5 below.

2.2.1.5 Methodology Objects (Collaborative Model Template, Model Type, Key Value Type)

Methodology objects can be thought of as templates or definitions. Unlike the instances, they have contextual versioning.

The main objects are:

- Collaborative Model Template [CMT]

This can be thought of as a template for an Associative Model Network. It is the context for associativity between Model Types and Key Value Types. It can have documentation for explaining the intended use of the template.

CMTs can be composed of other CMTs. This is useful where templates from several disciplines, teams, or organisations are being combined into a bigger package. The disciplines/teams/ organisations can retain control of their own templates, and then these can be consolidated into a combined template.

- Model Type

This is the definition or template for a Model Instance. It has documentation to describe the purpose and information content of the model, and can include examples or templates for creating instances. The content of this model-type definition can be as detailed as needed. The different methods to create the models might also be described and linked to the model type however we do not describe this in detail in this document.

A Model Type can inherit from another Model Type. This allows the reuse of definition information when a specialisation is created (e.g. Mesh > Thermal Mesh > Wing Thermal Mesh)

A Model Type can belong to more than one Collaborative Model Template, and so can have different associativity in the context of the different templates.

- Key Value Type

A Key value Type is the definition of a Key Value Instance. As with Model Types, it can have documentation, can belong to more than one CMT, and so have alternative associativity.

In the diagram below, there are two CMTs that contain the same Model Type that is derived in different ways, and has different Key Value Types extracted from it. I.e. a single Model Type's associativity is in the context of the different CMTs.

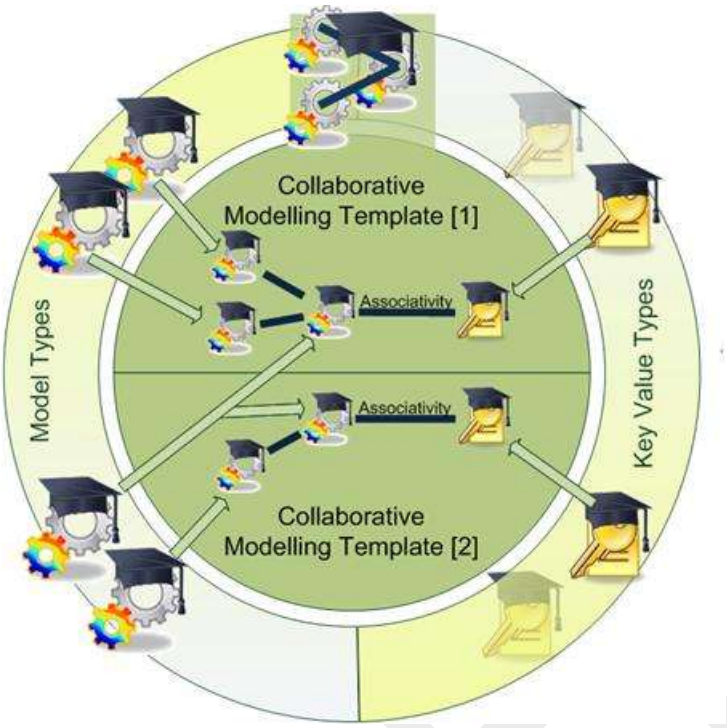


Figure 9: Methodology Objects

The links between all the objects once set up are retained. So when a package of work is set up for a study, then all the definitions supplied by the Types are available as specifications for what is expected to be delivered.

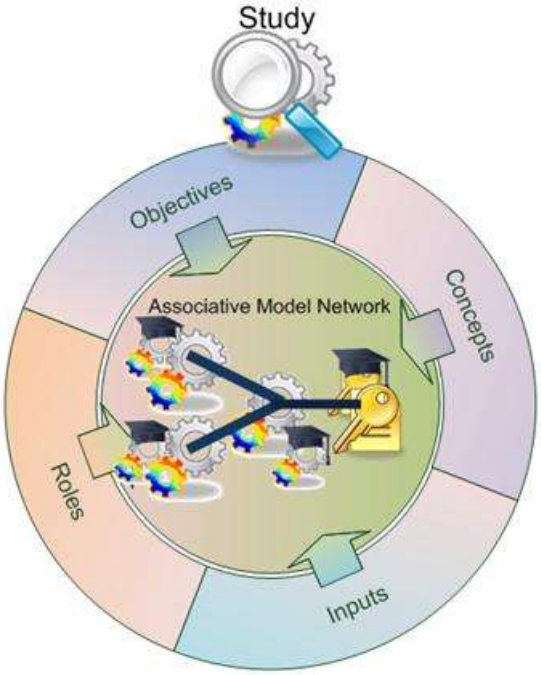


Figure 10: Study with Associative Model Network and Methodology Objects

2.2.1.6 Requirements and VV&A

Requirements are usually managed in official requirements management platforms (such as DOORs from IBM). These platforms support strict procedures and business processes according to the needs of the organisation and regulatory requirements. It is not the intention to replace these platforms, but instead to hold Requirement objects that reference to the requirements stored in the platforms.

However in order to use the requirements effectively in a collaborative environment, sufficient of the source information needs to be made visible. Therefore the requirements object can have its textual definition, and optional associated documents and properties. Relationships (e.g. Contradiction, Tracing, and Decomposition) between requirements can also be defined

There are two parts to the verification of requirements:

- Identification of the item that the requirement should be satisfied by
- identification of the evidence that will show (or has shown) that it has satisfied the requirement

In the planning phase these can be identified even though the work has not been done. So it is possible to declare up front how a requirement is to be verified, and so this information can be used in planning. The values of the verification "status" property indicate whether this is yet to be verified, and after verification whether it has passed of failed etc.

Depending on the requirement, different types of objects can be used to satisfy it. For example:

- Requirement that all items in the equipment bay must not exceed a surface temperature of x degrees
    - Satisfied by: Components (or their representative model instances) in the equipment bay.
    - Supporting Verification Evidence: Thermal Analysis model instances for all components in the equipment bay
- Requirement that virtual test data is within tolerance of actual test data.
    - Satisfied by: Method.
    - Supporting Verification Evidence: the comparison of the virtual to actual test data. This could be a Model Instance or a Key Value Instance of the tolerance which is extracted from the comparison Model Instance

These examples are illustrated in Figure 11 below.

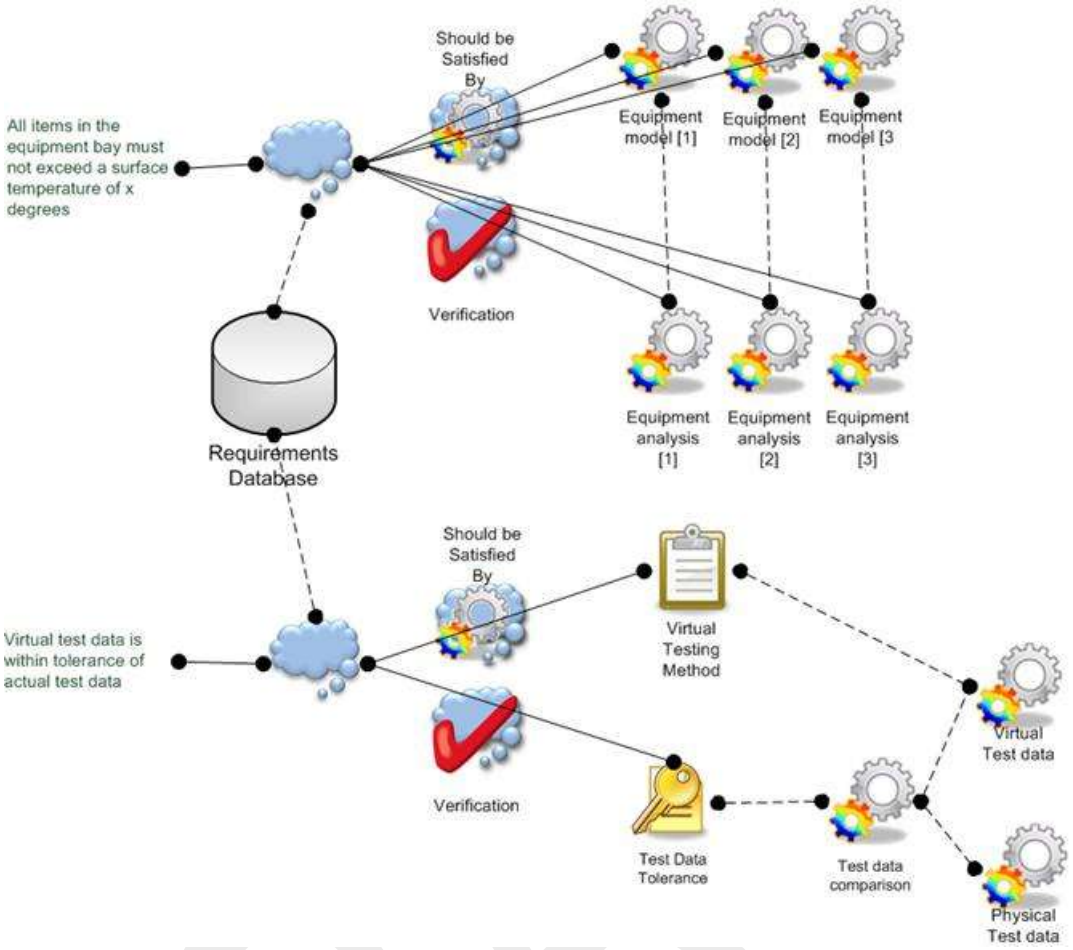


Figure 11: Requirement Verification examples

2.2.1.7 Common Objects combined

Figure 12 below combines Figure 7 to Figure 11 showing

- How a study can use the results of other studies as inputs, or as concepts to be investigated.
- How models in the study are used for requirement verification.
- How templates can be inputs to the study for guiding what type of models and their associations to use
- The use of Organisations, teams and people in the roles within the study.

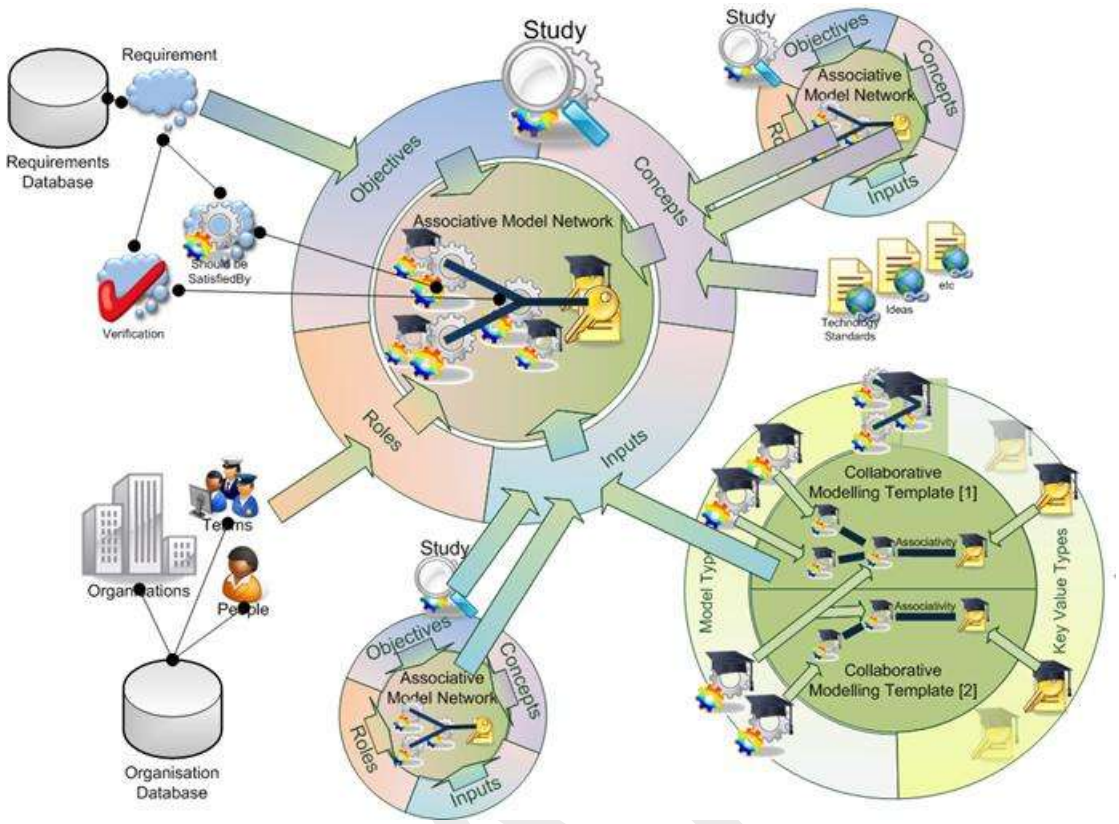


Figure 12: Schematic view of the most commonly used objects

2.2.2 MoSSEC Business Object Model structure

Figure 13 below is a UML diagram of the most commonly used objects. The diagram uses the same images to represent the classes as in chapter 2.2.1 above. It has been simplified for clarity, omitting the properties, operations and some of the relationships.

The Business Object Model is structured into logical packages. A first group of packages are identified as essential to carry out a “basic” form of MoSSEC, while a further group of packages have been identified as necessary to carry out more “advanced” forms of MoSSEC, for example including formal representation of value and risk.

It is planned to review and update the definition of these logical packages in the MoSSEC project, and to agree the essential elements for a first version of the MoSSEC standard. The packages are introduced below and are detailed in Reference 4.

The first set of packages are those necessary to carry out "basic collaboration" :

- Study Management
  Studies and the direct relationships to other objects
- Models Management
  Networks and instances, used to identify the work to be performed and the models that result from a study.

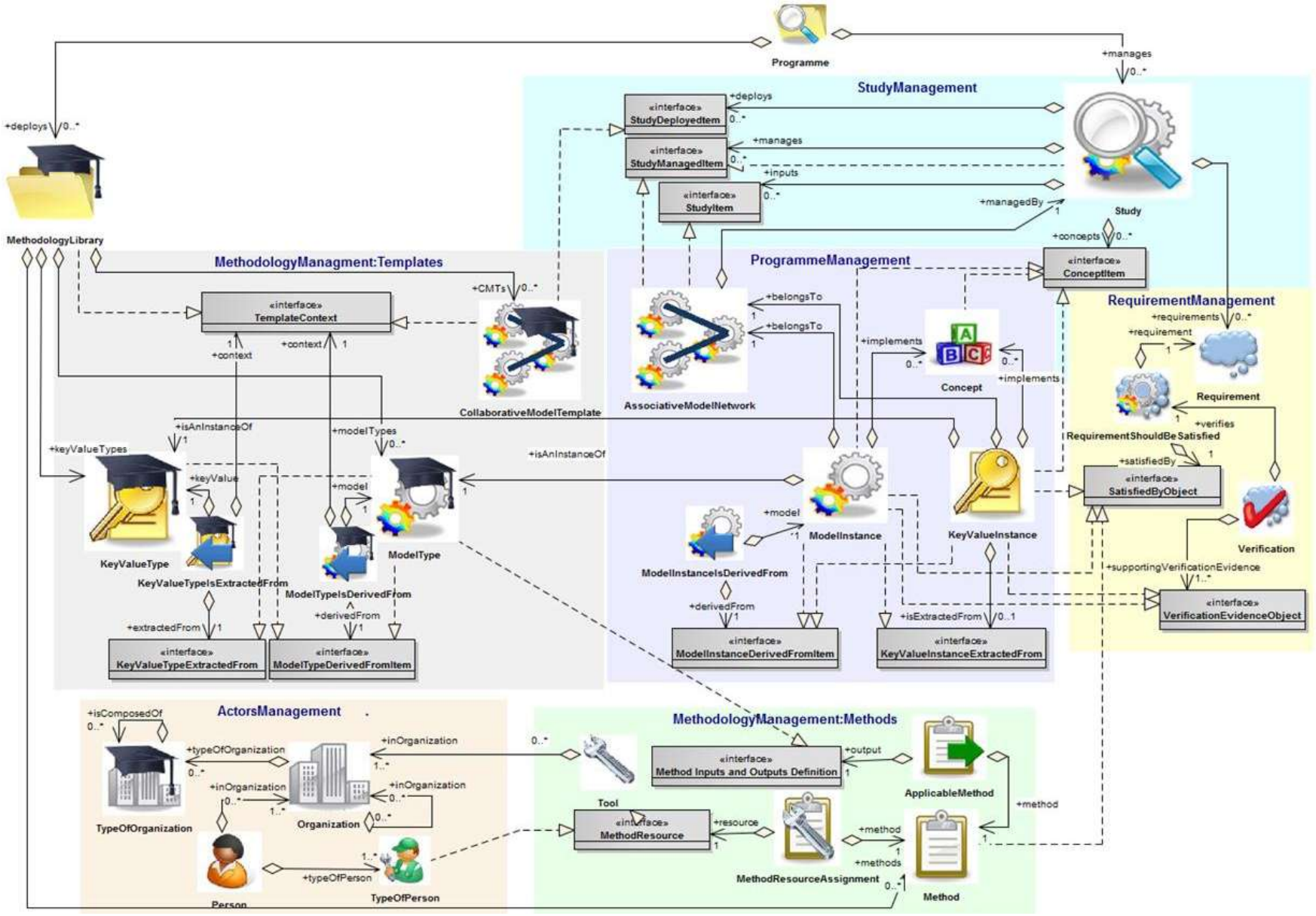


Figure 13 Simplified UML diagram of the most commonly used objects on context

- Actors Management
  Types of person, individual actors, organisation information
- Methodology Management
  Templates and Type definitions for deployment in studies
- Requirement Management
  Requirements and their direct relationships to other objects. Includes ShouldBeSatisfiedBy and Verification
- Collaborative Process Management
  The Objects used to manage the work. Requests, Work Orders and Activities
- Document Management
  Documents and Digital files
- Base Objects
  The common (mainly abstract) objects that all other objects are built from (inherit)

These are then followed by a series of more specialist packages that would be needed for more advanced "value and risk managed collaboration" (however these logical packages also include some entities now considered necessary for basic collaboration) :

- Breakdown Management
  Generic representation of breakdowns (e.g. Product Breakdown, Functional Breakdown). Showing which objects can be included in the breakdown structure. It includes Components and component usage which are described in more detail in Interfaces and Components.
- Interface Management
  Interfaces, Ports and Connections with the relationships to other objects, particularly links to requirements and representations. Also the use of interface specifications, hierarchies, and constructed and unconstructed interfaces.
  See also: Interfaces and Components
- Security and Trust Extensions
  Contracts, Information Rights and Security Classifications, and how they can be applied to the other objects.
- Quality Extensions
  - Approvals
  - Justifications and Assumptions
  - Decisions and Argumentation (see Decision Capture)
  - Distributions
  - Meetings
  - Quality Gate Management (see Quality Gates)
  - Quality Reports
  - Uncertainty Management

  See also: Quality Domains
- Value Modelling Extensions
  Extending mainly the Requirements and Study Management package, this adds objects to handle StakeholderExpectations, StakeholderNeeds, and ValueCreationStrategies with the associated QuantifiedObjectives and Targets to fullfil the StakeholderNeeds of the strategy.
- Simulation Extensions

This extends the Methodology Management package, adding extra relationships and properties to ModelType and Software

- Geometry Extensions
  These are extensions to ModelType and ModelInstance to allow the identification or relative coordinate spaces between geometry models. This is particularly important when using representations in the Interface Management package
- Optimisation Extensions
  This allows the creation of optimisation template that identifies, which are the variables and constraints and their distribution, which are the objective(s) with their target and resolution, and also which are the constants and additional things to observe during the optimisation.

2.2.3 MoSSEC standardisation

In order to ensure an open and modular architecture, the MoSSEC Business Object Model is mapped to a combination of the existing information standards ISO standards 10303-233 Systems Engineering [AP233] and 10303-239 Product Life Cycle Support [AP239]. This has the advantage that there are already a number of tools and implementations available that are based on these standards. It also provides a communication mechanism to systems that are not BDA enabled, and an archive format for MoSSEC data.

Templates, corresponding to MoSSEC Business Objects, are defined to support the specification of a data exchange specification [DEX] see chapter 3. These templates, as part of their formal specification, are mapped to an OMG SysML representation of ISO 10303-239:2011 Product Lifecycle Support [PLCS PSM] see Ref [3].

The mapping is done using OMG SysML and is based upon the SysML Parametric Diagram approach that decomposes blocks into their internal parts to identify the target objects in the AP233/239 data structures. Binding connectors are then used to relate public properties to these parts.

An example of the Model Instance SysML Parametric Diagram is shown in Figure 14 below.

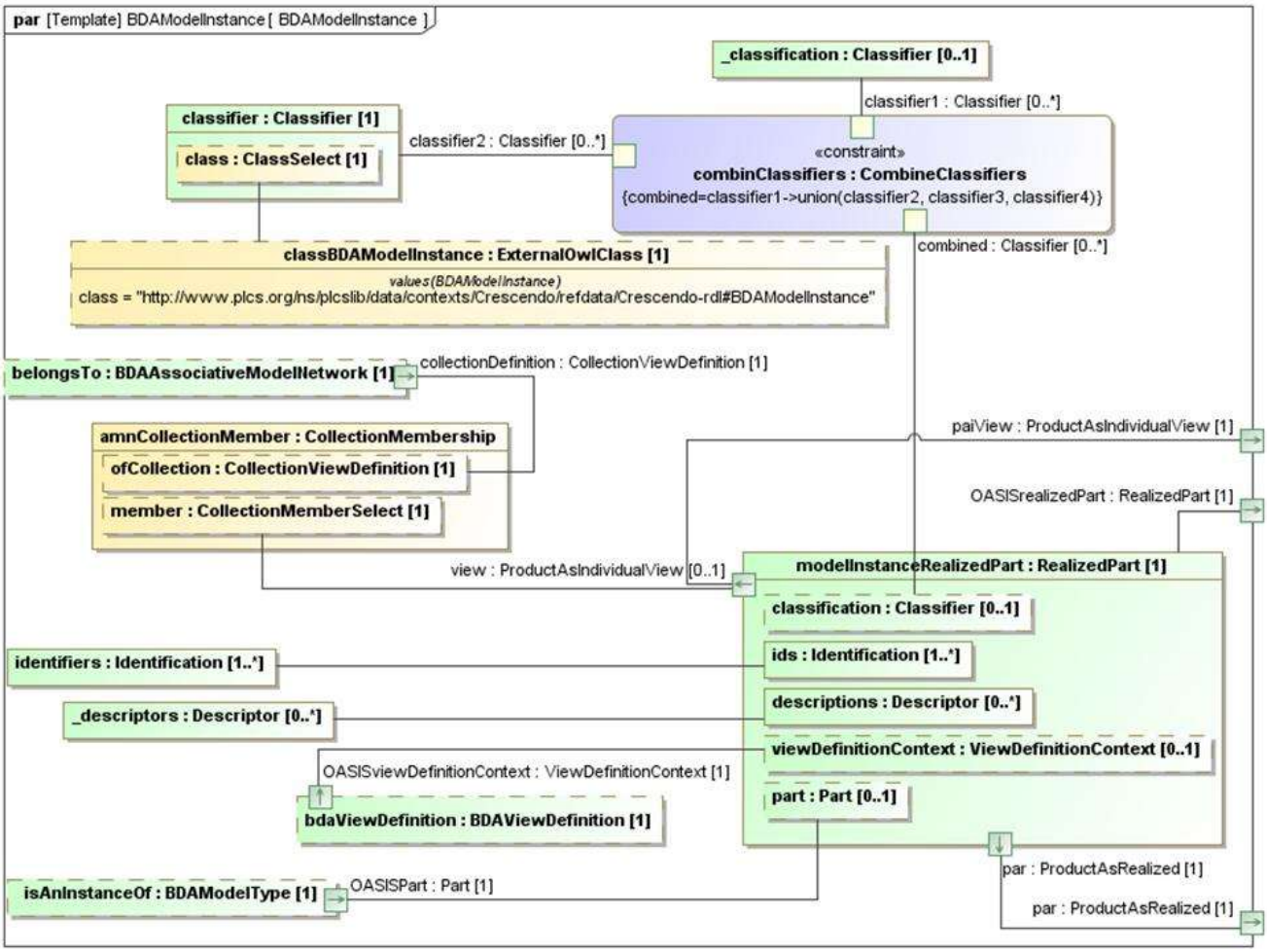


Figure 14 Model Instance parametric block diagram mapping to AP233/239

In addition to the standard for data exchange based on the object specification, there is also a standard for data sharing based on the specification for the objects and associated operations. These are independent of implementation, but are usually implemented with web services. For more information see chapter 4.

3 MoSSEC DEX Specification

Templates, corresponding to MoSSEC Business Objects, are defined to support the specification of the data exchange standard [DEX]. These templates, as part of their formal specification, are mapped to an OMG SysML representation of ISO 10303-239:2011 Product Lifecycle Support [PLCS PSM] see Ref [3].

The mapping uses OMG SysML Parametric Diagrams to decompose the Templates (representing the Business Objects) into OASIS templates or PLCS PSM Data Model elements. Binding connectors are used to specify how attributes or properties of the Business Objects are represented in the Data Elements.

Two DEXs were identified for initial implementations, these were "ModelInstance" and "AssociativeModelNetwork". The details can be found at PLCSlib MoSSEC Context and DEXs, Ref [3]

3.1 Business Overview

3.1.1 Business process

The business process for which the DEXs provide information is summarized in the SysML activity diagram in Figure 15below.

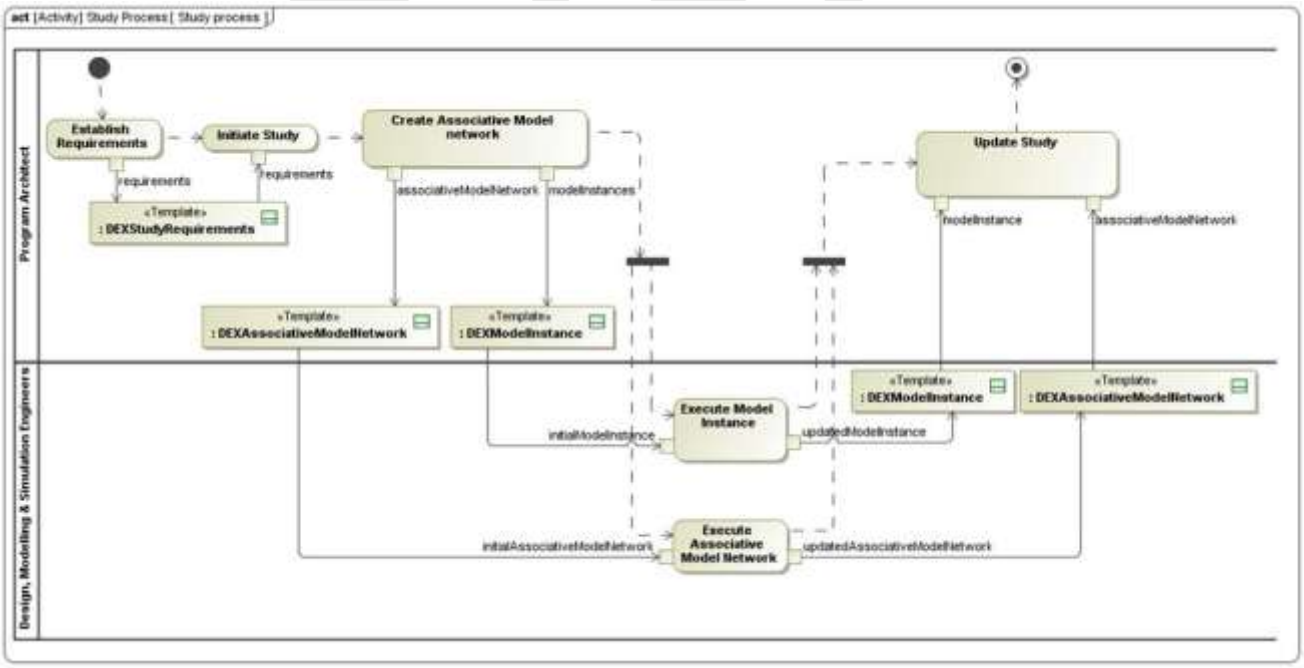


Figure 15 Business Process showing the points for data exchange

3.1.2 Business Information Overview

This section provides a high level overview of the business information that can be represented by this DEX. A more detailed overview of the information is described in Associative Model Network: Business Information Model, and a detailed definition of how the information is represented using the PLCS PSM is provided in Associative Model Network: PLCS PSM representation.

Figure 16 below provides a high level summary of Associative Model Network and Model Instance data exchanged.

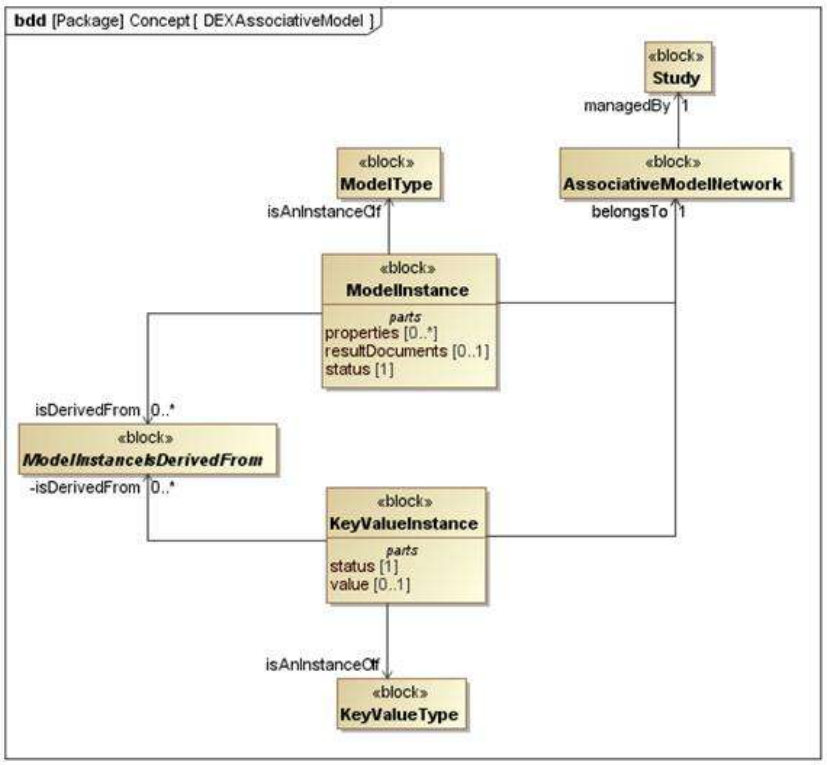


Figure 16 Associative Model Network and Model Instance data exchanged

3.1.3 Business Information Model

This section provides a high level overview of the business information that can be represented by the DEXs, and is shown in the SysML block diagram shown in Figure 17 below. Note, image is intended to illustrate the type of information that can be represented by the DEX. It does not imply that all exchanges must contain all this information.

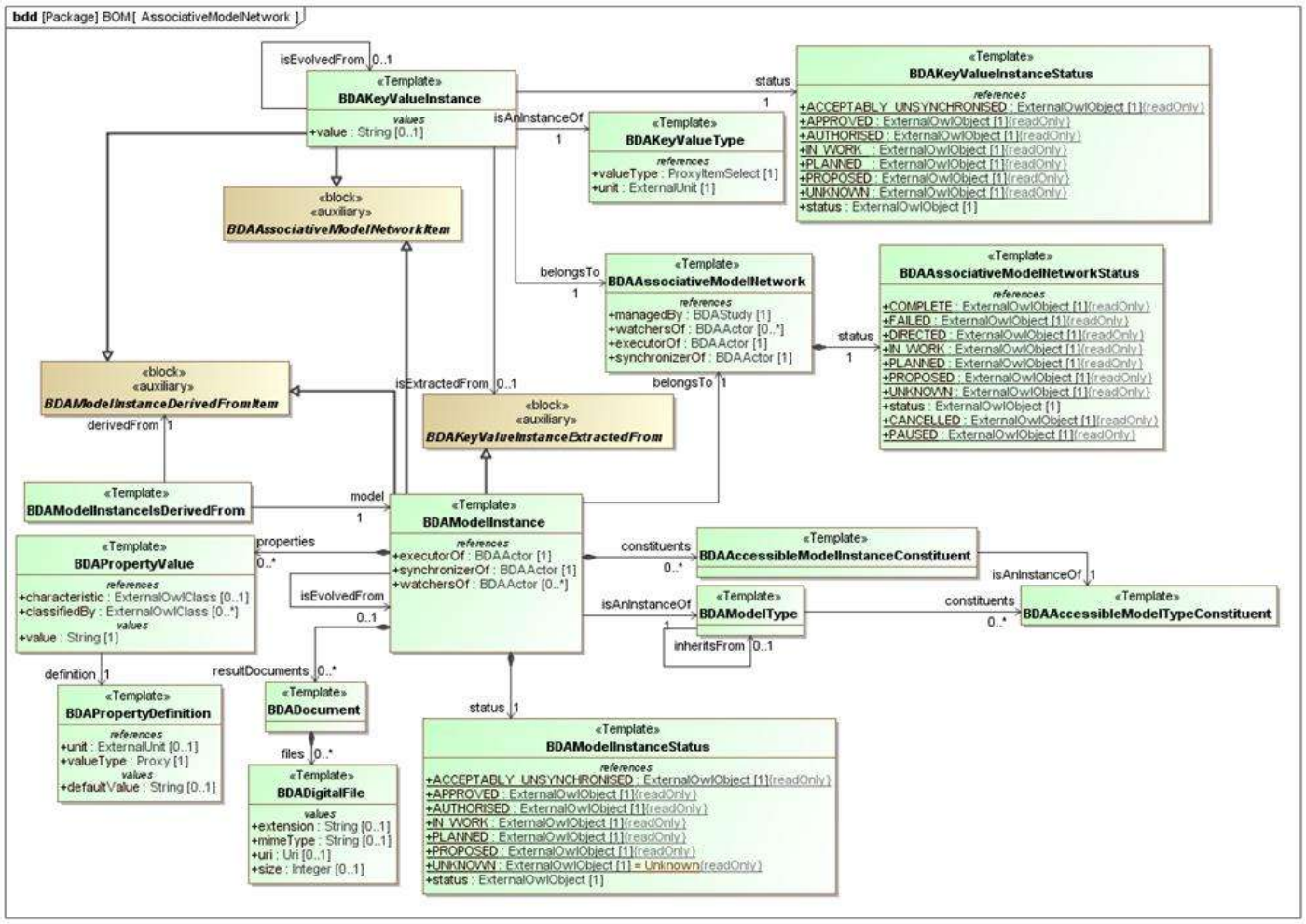


Figure 17 SysML block definition diagram showing all the information that the DEXs support

3.2 Associative Model Network DEX

This DEX enables the exchange of an Associative Model Network between the systems used by an architect to manage a study and the systems used by M&S engineers to evaluate network and associated model instances.

The DEX is first used to send an Associative Model Network to M&S engineers for evaluation. The results of the evaluation are then returned to the architect using the same DEX

The DEX specifies how to use the PLCS PSM to exchange Associative Model Networks (AMNs). Such an AMN represents a related set of simulation information and may be fully or partially evaluated. The DEX represents information such as:

- Model Instances
- Key Value Instances .

This DEX is related to the Model Instance DEX described in chapter 3.3.

3.2.1 Scope

The following are within the scope of this Data Exchange Specification (DEX):

- Associative model network
- Model Instances

The following are outside the scope of this Data Exchange Specification (DEX):

- Collaborative Model Templates;
- Requirements;
- Studies.

3.2.2 PLCS PSM Representation

The necessary information to be exchanged by the DEX is shown in the SysML Parametric Diagram in Figure 18 below.

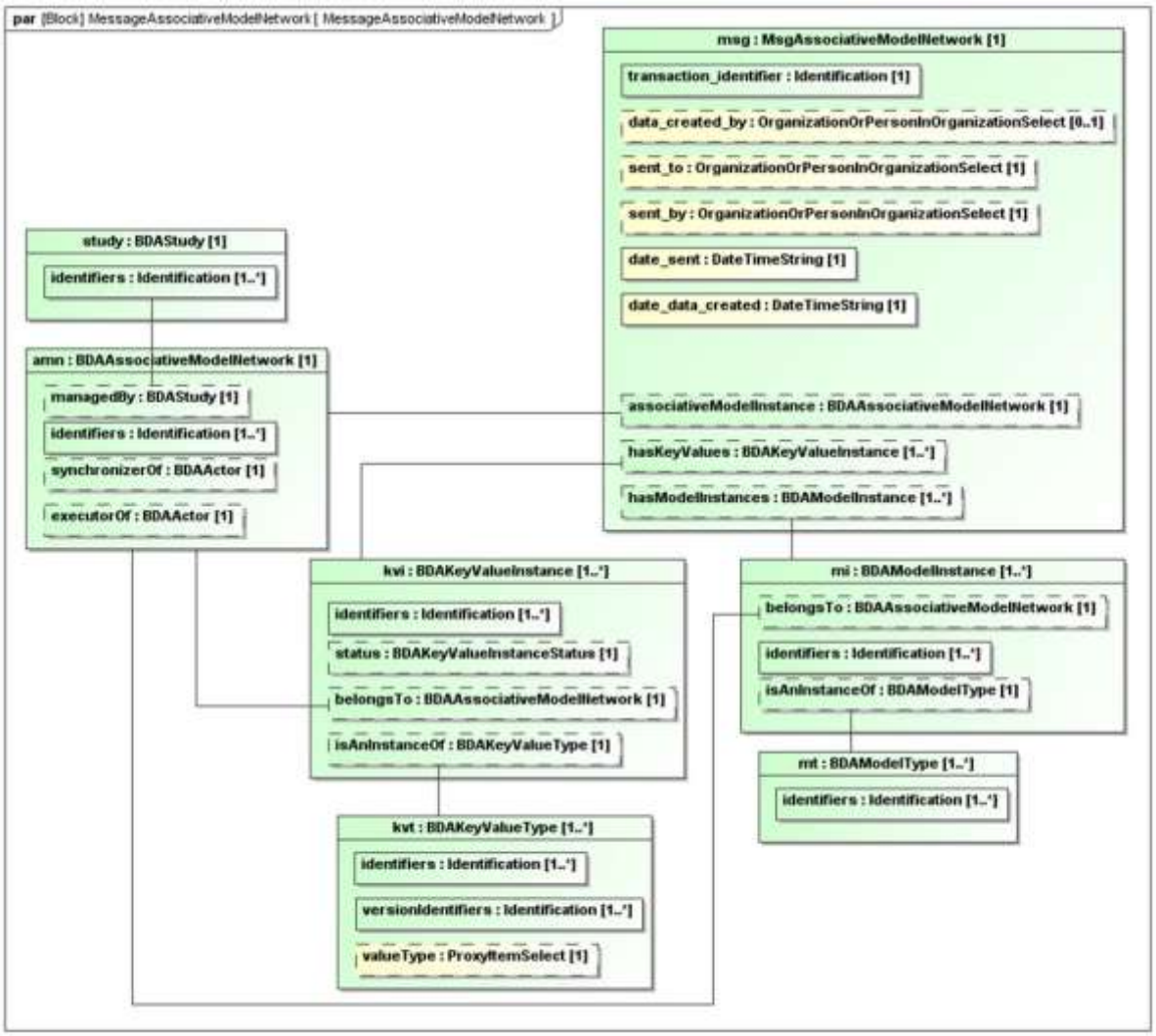


Figure 18Associative Model Network PLCS PSM representation

3.3 Model Instance DEX

The Model Instance DEX specifies how to use the PLCS PSM to exchange a Model Instance. The DEX represents information such as:

- Model Instance
- Documents associated to the Model Instance
- Digital files that comprise a Document
- Reference to the Associative Model Network to which the Model Instance belongs
- Reference to the Study to which the Associative Model Network belongs

It is intended to support the exchange between the systems used by an architect to manage a study and the systems used by M&S engineers to evaluate the model instance.

The scope of the exchange is limited to the exchange of a single model instance and the information necessary for M&S engineers to evaluate the model instance and return the results to the architect. If a complete network is to be exchanged then the Associative Model Network DEX should be used.

3.3.1 Scope

The following are within the scope of this Data Exchange Specification (DEX):

- Model Instances

The following are outside the scope of this Data Exchange Specification (DEX):

- Collaborative Model Template
- Associative Model Network
- Requirement
- Study

3.3.2 PLCS PSM Representation

The necessary information to be exchanged by the DEX is shown in the SysML Parametric Diagram Figure 19 below.

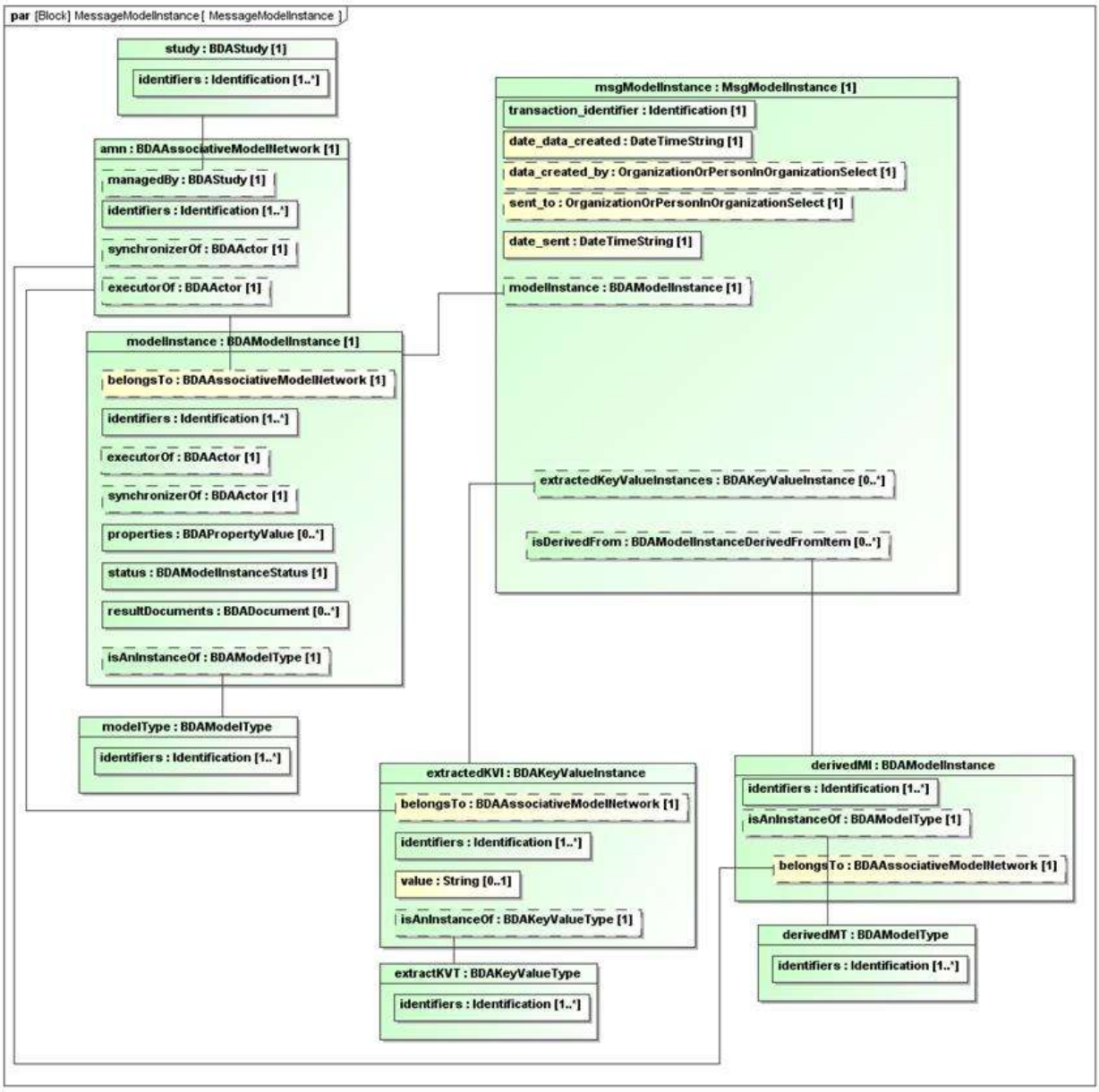


Figure 19 Model Instance PLCS PSM representation

3.4 Reference Data

The Reference Data Library used by the DEX is ref [4]. If the DEX is to be used in a data exchange scenario for which additional reference data is required, then a new OWL ontology should be created that imports this library. The new OWL ontology then becomes Reference Data Library used by the DEX.

4 MoSSEC Services

4.1 Clients and Servers for the MoSSEC Web Services

A client is something that consumes and publishes data, and a server is hosting and serving data for those clients. This is illustrated in Figure 20 below.

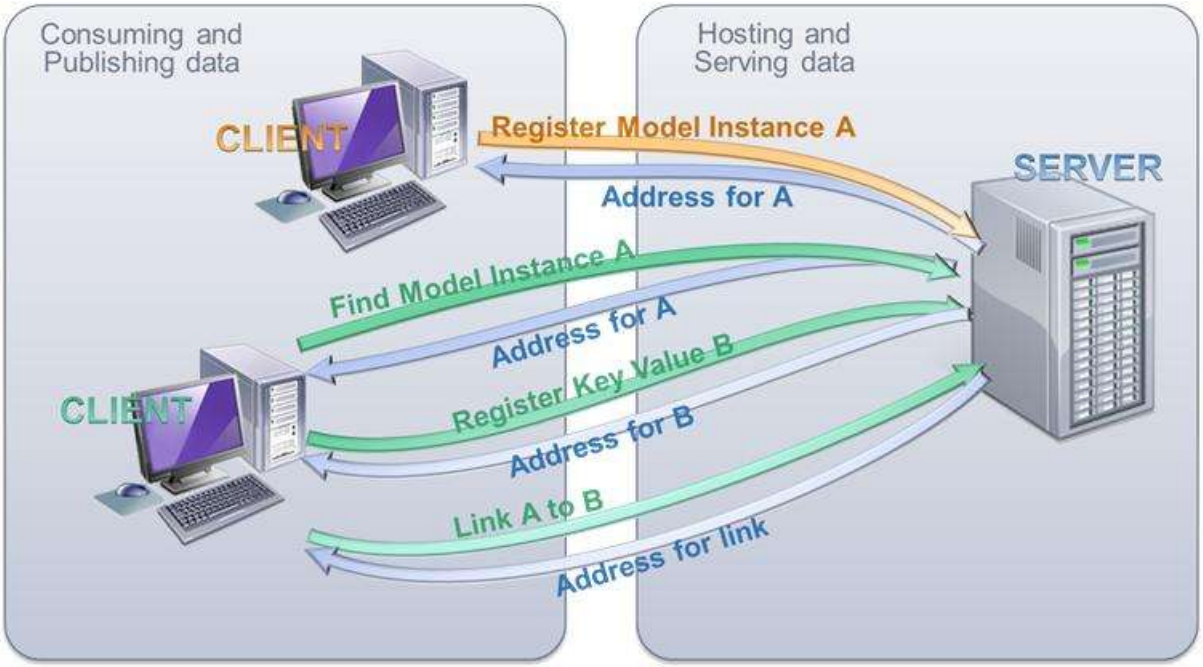


Figure 20 Client server interaction example

Figure 20 above illustrates a typical scenario of multiple clients accessing a server and manipulating the same piece of data. Firstly, the orange client adds some information (Model Instance A) to the server and receives feedback stating whether it has been added successfully or not. At this point the green client is able to query the server for the item that has been added by the orange client and receives the server's identifier for the item (this is assuming the information was successfully added). Lastly, the green client then adds more information (key Value B) to the server and links it to the that was uploaded by orange client.

4.2 MoSSEC services categories

The MoSSEC Services provide services for communication of elements from the MoSSEC Business Object Model. They serve as the primary reference for all services required in a collaborative context, requiring that meta-data is passed among heterogeneous and/or proprietary IS/IT environments.

The services are architected into three categories:

1. System (implementation) services e.g. Login, Logout,

2. Late Bound Services e.g. Register, Read, Update, Delete, Search

3. Early Bound Services e.g. CollaborativeModelTemplate.getAllItems()

4.2.1 System or Implementation Services

The System or Implementation Services define additional characteristics necessary for using the Business Objects in an implementation environment. For example, attributes for data provenance, internal system identification tokens, last updated, last modified, created by, created on, etc.

4.2.2 Late Bound Services

Rather than having individual “get” and “set” services for every class in the BDA Business Object Model it was decided to use late bound services that would be common for all classes, so that fewer services are needed. There are additional benefits in that updates to the BDA Business Object Model do not necessitate updates to existing services or additional services to be defined.

The naming convention for the common services is CRUD, or Create, Read, Update, Delete. However the “create” operation for an object is performed inside a BDA enabled capability and the object is then registered with other platforms. Therefore it was decided to use the name “Register” in place of “Create”.

4.2.3 Early Bound Services

In addition to the common services, services specific to individual classes were identified as part of the BDA Business Object model. These were included as operations in the UML definition of the BDA Business Object Model.

The majority of these services are for when it was identified that a particular “target” object regularly needs to return a “source” of a relationship. A constraint was enforced for the BDA Business Object Model to only use one-directional relationships from a source to a target. This was to avoid the potential inconsistency in the distributed dataset where the reference on one object could be deleted without the removing the reference on the other. This means that the common “Read” service for a source would return the target references, but not vice versa. The common “Search” service could be used to find the source, but this would be inefficient with frequent use. In these cases operations were added to the target to return the source. For example in the relationship source “belongsTo” target, the target has the operation getItems().

4.3 Services Implementation

The MoSSEC Services can be implemented using various technologies, for example by using Web Services, or any other XML Schema based technology, since the MoSSEC Business Object model and the Services are defined independent from any communication protocol

The CRESCENDO used a web service implementation of these services to successfully demonstrate several scenarios as illustrated in Figure 21 below.

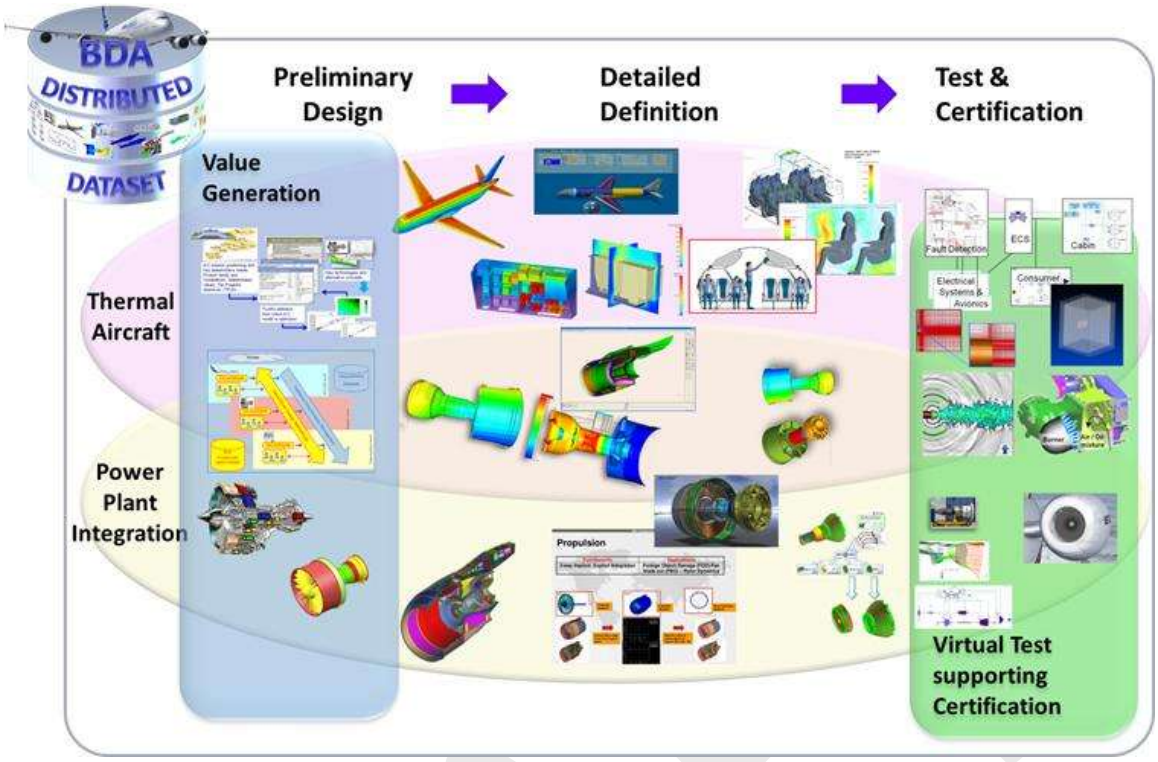


Figure 21 CRESCENDO scenarios

The implementations were across different organisations in different countries and using many different clients, though only one server, Share-A-space™ provided by Eurostep. The installations are shown in Figure 22 below.

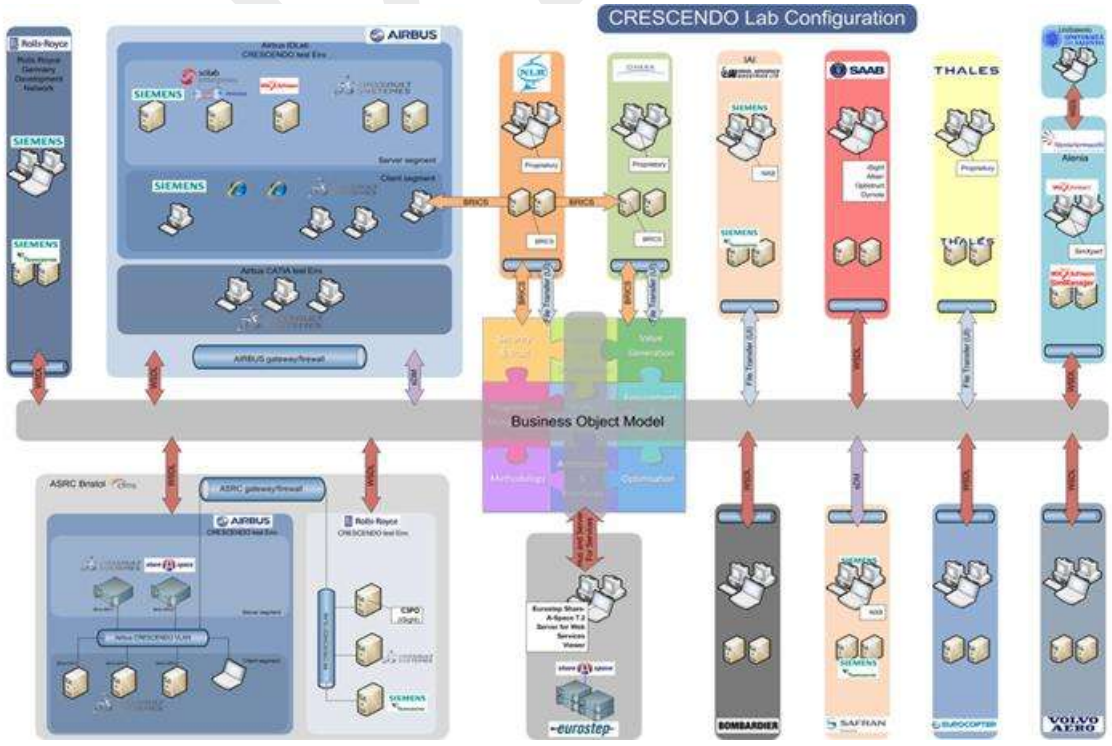


Figure 22 Installations used during CRESCENDO project

5 Conclusions & Next steps

The scope of the proposed MoSSEC standard has been introduced in terms of the primary information objects and services, which build on the foundations of AP233 & AP239.

A MoSSEC Project is proposed to review and update this scope and to drive the international standardisation activity and related implementor forums.

Potential participants and contributors to the MoSSEC Project are invited to contact the project leader adrian.murton@airbus.com.

A number of web-meetings will be held to allow potential participants to confirm their level of interest and to share related work, before face-to-face meetings are held to formalise the organisation principle of the MoSSEC Project.

References

1. ISO standards 10303-233 Systems Engineering and 10303-239 Product Life Cycle Support

   http://www.iso.org

2. PLCSlib can be found at http://www.plcs.org/plcslib/plcslib/ Note: the MoSSEC context and DEXs will not be visible here until approved for public release

   http://www.plcs.org/plcslib/plcslib/

3. CRESCENDO context and DEXs which form the basis for MoSSEC. Including the DEXs and Templates.

   http://crescendo-new.share-a-space.com/DEX/plcslib/

4. CRESCENDO Reference Data Reference Data Library used as the bases for MoSSEC

   http://crescendo-new.share-a-space.com/DEX/plcslib/data/contexts/Crescendo/refdata/Crescendo-rdl-en.owl.htm

5. CRESCENDO project: Collaborative and Robust Engineering using Simulation Capability Enabling Next Design Optimisation. EU FP7 co-funded project was coordinated by Airbus with a consortium of 59 partners from 13 countries, including aircraft and aero-engine manufacturers & suppliers, PLM and simulation software solution providers, research centres and academic institutions

   http://www.crescendo-fp7.eu/

6. "Trade-off management technology for architects within designing the robust virtual aircraft", by O. Tabaste (MSC), at PDT Europe, The Hague, 2012

   http://www.pdteurope.com/media/4896/3_next_generation_simulation___life_cycle_management_environment_for_virtual_aircraft_collaborative_trade-off.pdf

7. "**Enabling the Comprehensive Behavioral Digital Aircraft**", by V. Tuloup et al of Dassault Systemes @ GPDIS 2012

   http://www.gpdisonline.com/presentations/44MaciasEtAlDassaultEnablBehDigAir.pdf

Annex 1: MoSSEC Services

Services for MoSSEC are listed below grouped by their type.

System Services

Login():sessionId

login in to a server. It returns a string for the session ID for the server which is the authentication. This has to be kept by the client application to be used when invoking further operations. All other operations require this authentication.

Parameters

Name	Type	Multiplicity	Notes
userName	string	0..1	The name of the user that is logging in to the server
password	string	0..1	The password for the user that is logging in to the server
project	string	0..1	optional argument for a project, when the server has multiple projects.

Logout():boolean

Log out of a server. After a successful logout, the session ID is no longer active on the server so cannot be used for further operation calls.

Parameters

Name	Type	Multiplicity	Notes
sessionId	string	0..1	The session ID for which the logout is requested

GetMyNotifications():Notification[]

The fetches the notifications that the user has received on the BDA server. It returns an array of Notifications

Parameters

Name	Type	Multiplicity	Notes
sessionId	string	0..1	The ID of the session on the server for authentication

UploadDigitalFileContent():boolean

Allows the registration of a Digital File for a Document and the uploading of its binary content. Note, the Document must be already created on the server, and provided as an Object Reference to the operation.

Parameters

Name	Type	Multiplicity	Notes
document Reference	Object Reference	1	The object reference for the Document for which the digital file will be associated
digitalFile	Digital File	1	an instance of a DigitalFile of which the content is provided.
content	Binary Content	1	The binary content to be uploaded for the digital file
sessionId	string	0..1	The ID of the session on the server for authentication

DownloadDigitalFileContent():BinaryContent

Enables the download of the binary content of a Digital File. It returns the binary content of the given digital file object reference.

Parameters

Name	Type	Multiplicity	Notes
digitalFile Reference	Object Reference	1	The object reference for the digital file content that is to be downloaded
sessionId	string	0..1	The ID of the session on the server for authentication

Table 2 System Data Services

Late Bound Services

Register():ObjectReference

Enables the registration of a new object on the BDA server. It returns the object reference for the newly created object. This is also used to update an existing object.

Parameters

Name	Type	Multiplicity	Notes
managed Object	Managed Object	1	An object which is a sub type of ManagedObject
sessionId	sting	0..1	The ID of the session on the server for authentication

Read():ManagedObject

Enables the reading of an object from the BDA Server based on an Object Reference (e.g. something returned by a Search operation). It returns the appropriate object which is a sub type of ManagedObject

Parameters

Name	Type	Multiplicity	Notes
object Reference	Object Reference	1	an instance of an ObjectReference, e.g. returned by a search operation
sessionId	string	0..1	The ID of the session on the server for authentication

Delete():boolean

Enables the deletion of an object from the BDA server given an Object Reference.

Parameters

Name	Type	Multiplicity	Notes
object Reference	Object Reference	1	an instance of an ObjectReference, e.g. returned by a search operation
sessionId	string	0..1	The ID of the session on the server for authentication

Search():ObjectReference[]

Enables search for existing objects on a BDA Server with filters for ID, name, description and class type. it returns an array of object references.

Parameters

Name	Type	Multiplicity	Notes
id	string	0..1	The ID of the item being searched for.
name	string	0..1	The name of the item being searched for.
description	string	0..1	The description of the item being searched for.
type	string	0..1	The class of the item being searched for.
sessionId	string	0..1	The ID of the session on the server for authentication

Table 3 Late Bound Data Services

Note: All these Early Bound Services also need and input of “sessionId” string as shown for the late bound services above.

Early Bound Services (definitions included in Additional Material listed in Appendix A)
Document Management
Class: **Document** getAssignedTo():IdentifiableObject[]

Methodology Management
Class: **ModelType** getDerivedFrom(context:TemplateContext[0..1]):ModelTypeDerivedFromItem[] getIsExtractedFrom(context:TemplateContext[0..1]):KeyValueType[] getQualityReports(context:TemplateContext[0..1]):QualityReportType[] getAllIsExtractedFrom(context:TemplateContext[0..1]):KeyValueType[] getPrefferedMethod(context:TemplateContext[0..1]):ApplicableMethod[] getRankWeight(context:TemplateContext[0..1]):RankWeightType[]
Class: **CollaborativeModelTemplate** getImmediateItems():CollaborativeModelTemplateItem[] getAllItems():CollaborativeModelTemplateItem[]
Class: **AccessibleModelTypeConstituent** getModel():ModelType
Class: **MethodInputAssignment** getAllSensitivities():Sensitivity[] getSensitivityForOutput(output:ApplicableMethod):Sensitivity
Class: **ApplicableMethod** getAllSensitivities():Sensitivity[] getSensitivityForInput(input:MethodInputAssignment):Sensitivity
StudyManagement
Class: **Baseline** getCreatingStudy():Study
Class: **Study** getManagingStudy():Study getConsolidatedStatus():string

ProgrammeManagement
Class: **AccessibleModelInstanceConstituent** getModel():ModelInstance
Class: **AssociativeModelNetwork** getItems():AssociativeModelNetworkItem[] getConsolidatedStatus():string
Class: **KeyValueInstance** getConsolidatedStatus():string
Class: **Programme** getManagedBaselines():Baseline[]
Class: **ModelInstance** getIsExtractedFrom():KeyValueInstance[] getAllIsExtractedFrom():KeyValueInstance[] getDerivedFrom():ModelInstanceDerivedFromItem[] getPlannedMethods():Method[] getConsolidatedStatus():string getQualityReports():QualityReportInstance[] getRankWeightExtractedFrom():RankWeightInstance[] getAllRankWeightExtractedFrom():RankWeightInstance[]
RequirementManagement
Class: « abstract » **RequirementObject** getParentRequirements(relationshipType:string[0..1]):RequirementObject[] getChildRequirements(relationshipType:string[0..1]):RequirementObject[]

Class: **Requirement** referencedBy():IdentifiableObject[] shouldBeSatisfiedBy():SatisfiedByObject[] supportingVerificationEvidence(instance:SatisfiedByObject):VerificationEvidenceObject[] shouldBeSatisfiedByRelationships(SatisfiedByObject:SatisfiedByObject):RequirementShouldBeSatisfied[] verifications(SatisfiedByObject:SatisfiedByObject):Verification[]
Class: **Verification** getRequriement():Requirement getSatisfiedBy():SatisfiedByObject
Interface: **SatisfiedByObject** satisfyingRequirements():Requirement[]
Interface: **VerificationEvidenceObject** evidenceForRequirements():Requriement[]
ActorsManagement
Class: **Organization** getActors(actorType:string[0..1]):Actor[]

Table 4 Early Bound Data Services

Annex 2: Web Services

The services specification described in *Annex 1: MoSSEC Services* were used in the CRESCENDO project to define web services. An html representation of the WSDL for these services in embedded below.

CRESCENDOServices_as_html.htm

The WSDL and XSD for these services are in the embedded zip file below.

CRESCENDO_Services.zip

Note, the embedded files above also include some services that are intended for a later MoSSEC release.