ResCoM D5.1 Initial Standard Data Model

Eurostep

29-9-2014

This project has received funding from the European Union's Seventh Programme for research, technological development and demonstration under grant agreement No 603843

Contents

1 Introduction 3

1.1 Overview of PLCS 3
1.2 ResCoM initial standard data model 4
1.2.1 Objective and scope 4
1.2.2 Method 4
1.2.3 Roles of ResCoM initial standard data model 5

2 Initial standard data model description 7

2.1 The bases of the initial standard data model 7
2.2 SysML block definition diagram 8
2.3 ResCoM_ProductRCBreakdown_and_Design 9
2.4 ResCoM_Requirements 14
2.5 ResCoM_InLife 17
2.6 ResCoM_Activities_and_Methods model 20

3 Conclusion 26

References 28

Annex - SysML block definition diagram introduction

1 Introduction

This deliverable is a documented **domain information model** based on ISO 10303-239 and OASIS PLCS ed. 2 standard supporting the envisioned ResCoM processes and information. It also includes both defined terminology and UML class diagrams defining the used information structures [1].

1.1 Overview of PLCS

ISO 10303-239 (PLCS) [2] represents the information needed to operate and maintain a product during its entire life, from initial concept to disposal, as it evolved in an environment containing heterogeneous organizations, processes, and IT systems set required to support a product [3].

The development of PLCS was driven by the exploitation of aftermarket opportunities. A coherent information environment in which the necessary data are available to all participating actors is a prerequisite for efficient product support. PLCS can be thought of as a new "world map" for product support. PLCS has three main components [4]:

- A business vision (Figure 1-1), outlining the vision is to a single source of truth for assured product and support information for use across the enterprise.

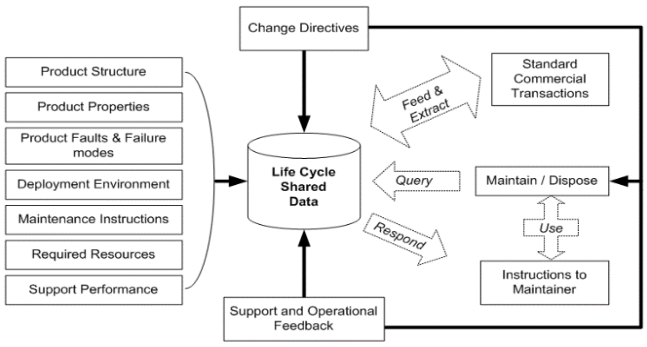


Figure 1-1. The PLCS vision

- An activity model [5], comprising a coherent set of processes which support continuous optimization over the full product life cycle - from product requirements to product disposal.
- An information model [6], specifying an information environment for the integration and exchange of the data that are required by the processes described in the activity model.

This project has received funding from the European Union's Seventh Programme for research, technological development and demonstration under grant agreement No 603843

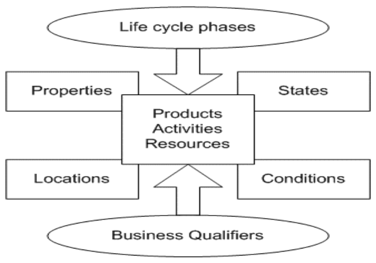


Figure 1-2. The key concepts in the PLCS information model

The central concepts in PLCS (Figure 1-2) are: product, activity, and resource. Each of them may be associated with properties, states, or locations. Conditions may be applied to relationships between these concepts.

1.2 ResCoM initial standard data model

1.2.1 Objective and scope

Eurostep is developing the Share-A-space server solution for secure PLM collaboration. The ResCoM project will develop it further to support closed loop multi life cycle applications. Share-A-space is based on the ISO 10303 (i.e. ISO 10303-214 Automotive Design, ISO 10303- 239 Product Life Cycle Support and ISO 10303-233 System Engineering). Among those standards Product Life Cycle Support (PLCS) and System Engineering [7] are considered as the foundations of the ResCoM collaborative software platform.

ResCoM prospects a new business opportunity which could benefit both the enviroment and economy. One the other hand, it puts increasing requests on managing information throughout the complete product multiple lifecycles. The objective of the work described in this document is to provide a simplified AP233/239 based data model extended with semantics from the closed loop and multiple lifecycles area to support closed loop multiple lifecycle products. The model will be validated using Share-A-space and industrial test cases.

1.2.2 Method

The initial data model is modelled by utilizing methodologies and tools developed and used in the support of products with a linear life cycle. In the context of ResCoM firstly we extracted the main ResCoM concepts and product requirements from the anterior ResCoM work e.g. Asif [8], Rashid et al. [9], DOW [1], D2.1 [10], D2.2 [11], D2.3 [12] and D3.2 [13]. A generic activity model was then defined to identify what and how the closed-loop production activities should be carried out and integrated throughout the product's multiple lifecycles. The activity model will be continuously updated according to the up-to-date research results from WP2, WP3, WP4, and WP6. This ResCoM generic activity model is designed to be extended and tailored for use in different cases (in ResCoM project, different cases means the four cases from OEMs). The generic activity model is modelled in [14], though it is not an element of this deliverable, it provides a good basis for defining processes (which provide the context of data) and responsibilities within organizations and in the ResCoM collaborative closed-loop production hub.

Accordingly the ResCoM initial data model i.e. the domain information model is created, which is specifying an information environment for the integration and exchange of the data that are required by the processes described in the ResCoM generic activity model. The generic concepts presented in the data model are influenced by the ISO/STEP PLCS and Systems Engineering standards. The initial standard data model is also a generic model which is applicable to the four industrial cases of ResCoM project, and even applicable to future products which are designed based on the ResCoM methodologies. The data model is divided into four distinct domains domains in order to cover all ResCoM relevant processes and information, each focusing on different aspect of the model. Each of the four domains is represented by one SysML block definition diagram:

- ResCoM_ProductRCBreakdown_and _design
  Concepts and properties for representing the product lifecycles including their relationships.

- ResCoM_Requirements
  Various types of ResCoM product specific requirements including their traceability and structure.

- ResCoM_InLife
  Multiple lifecycles design, lifecycles definition and predefined product structure of each lifecycle for specific manufactured products.

- ResCoM_Activities_and_methods
  Activities for closing the loop of the product system and supporting multiple including predefined activity methods for supporting first manufacturing activities (such as design) as well as operational value recovery methods in each of its end of lifecycle taking into account plan and execute manufacturing and operational value recovery specific product individuals.

Roles of ResCoM initial standard data model
The development processes of ResCoM collaborative software platform are shown in Figure 1-3.

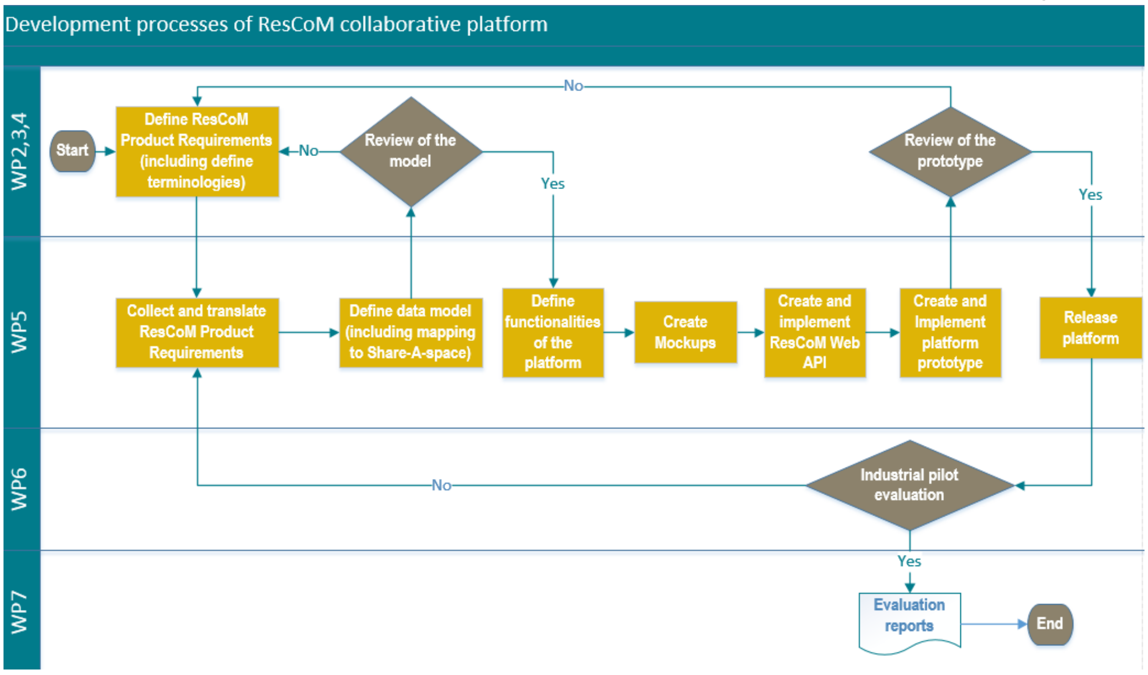

Development processes of ResCoM collaborative platform
WP2,3,4
WP5
WP6
WP7
Start
Define ResCoM Product Requirements (including define terminologies)
No
Review of the model
Yes
No
Review of the prototype
Yes
Collect and translate ResCoM Product Requirements
Define data model (including mapping to Share-A-space)
Define functionalities of the platform
Create Mockups
Create and implement ResCoM Web API
Create and Implement platform prototype
Release platform
No
Industrial pilot evaluation
Yes
Evaluation reports
End

Figure 1-3. Development processes of ResCoM collaborative software platform

ResCoM standard data model aims to specify an information environment for the integration and exchange of the data that are required by the processes described in ResCoM requirement framework. The intended use of the initial standard data model is first to provide a basic framework as a support for collecting further ResCoM requirements on multiple lifecycles product management from other three pillars (i.e. product design, business model, and supply chain). Along with the research of the ResCoM project, the information on integrating and exchanging data required by the processes of close-loop production will be progressively identified by the three pillars, which will be used for extending and refining this initial standard data model.

Secondly, the initial standard data model is the basis of the ResCoM collaborative software platform development, including ResCoM Web API, and Share-A-space solution for managing product having multiple lifecycles. ResCoM initial standard data model is modelled to represent all the entities/objects and their relationships which are needed to be managed in Share-A-space to support the complete lifecycles of the RCP (Resource Conservative Product). It will be validated using Share-A-space and industrial test cases.

This will be an iterative process and when final closed loop information management requirements are set, the profile will be updated and ready (D5.2 Final standard data model) for communicate with the OASIS and ISO standards. The information ready for standardization will include activity models giving context to the standard proposal, vocabulary (taxonomy) establishing a semantic base for the information management in this area and information models including utilized reference data.

2 Initial standard data model description

2.1 The basis of the initial standard data model

The bases of the initial standard data model includes two parts: concepts of the ISO/STEP PLCS and Systems Engineering standards, and concepts of ResCoM.

The generic concepts of the model are influenced by the ISO/STEP PLCS and Systems Engineering standards. The following main concepts, which are outlined in the PLCS concept model (Figure 2-1), influence the ResCoM initial standard data model [15]:

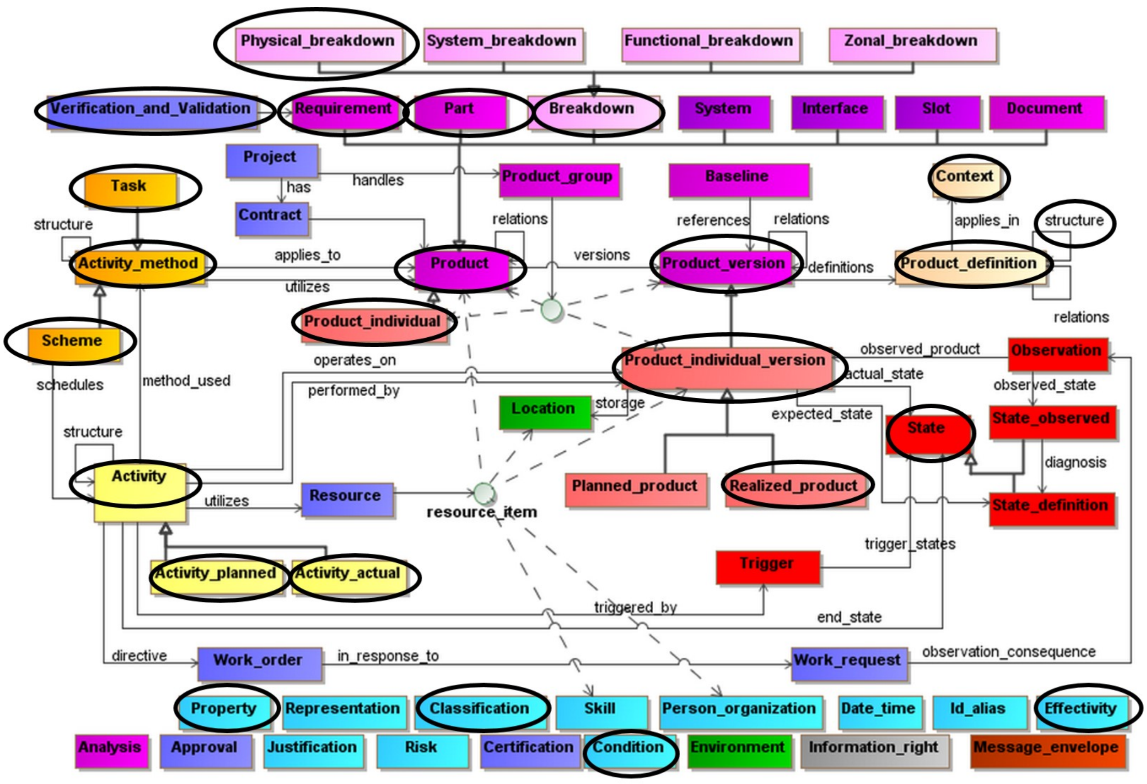


Figure 2-1. PLCS concept model

However, ResCoM initial standard data model must support the envisioned ResCoM processes and information. Thus, it also adds semantics from the closed-loop production and multiple lifecycles area. For the initial standard data model, the task is how to represent the following ResCoM concepts/terminologies (from [1] [8] [9] [12]) in the data model:

- ResCoM specific product requirement [1]
    - Sustainability baseline: LCA value decides the optimum number of product lifecycles
    - Closing the loop: business requirements, product design requirements, supply chain requirements, information technology requirements
    - Interaction among the factors of the requirements
- Multiple lifecycles design

This project has received funding from the European Union’s Seventh Programme for research, technological development and demonstration under grant agreement No 603843

  - RCP: Resource Conservative Product (labelling the product as a RCP brand) [1] [8] [9]
  - RCL (Resource Conservation Level) [1] [8] [9]
    - Optimum number of lifecycles (product-level, component-level)
    - Predetermined length (time or performance) of each lifecycle (predefined lifecycle plan for supporting each lifecycle)
    - RCL_i: Resource Conservation Level, where i = 0,1,2…RCL_0 represents RCP in its 1st, 2nd, 3rd…designed lifecycles
  - Material indexes [1]
    - MCI: material criticality index
    - MRI: material recyclability index
  - Product/component indexes [1]
    - CRI: component reusability index
    - PMI: product modularity index
    - PUI: product upgradability index
    - PRI: product reusability index
    - EPI: EoL potential index
- Processes to closing the loop [12]
  - Manufacturing
  - Distribution
  - Core collection
  - Operational value recovery
  - Redistribution
  - In-service
- Individual product management [12]
  - Identification of an individual product during its multiple lifecycles
  - Traceability of the product changes during multiple lifecycles
  - How to represent the RCL_i relationships of an individual product

2.2 SysML block definition diagram

The initial standard data model is modelled by using SysML (Systems Modelling Language) block definition diagram. SysML is a general purpose graphical modelling language for specifying, analysing, designing and verifying complex systems that may include hardware, software, information, personnel, procedures and facilities. It is a subset of the UML (Unified Modelling Language) with extensions (e.g. requirements, parametrics, item flows etc.). The SysML block definition diagram is based on UML class diagram, and describes the structure of a system by showing the systems classes, their attributes (part property, reference property, value property), and represents the relationships among the classes, e.g. specialization and generalization, association, aggregation, and composition. Before you start reading the initial standard data model in this document, the "SysML block definition diagram introduction" (extracted from the book 'SysML for Systems Engineering' of Jon Holt and Simon Perry) in the Annex is recommended to read.

A system typically has a number of SysML block definition diagrams – not all classes (i.e. blocks) are inserted into a single SysML block definition diagram. A class (i.e. block) can have multiple levels of meaning and participate in several SysML block definition diagrams. The initial standard data model is divided into four distinct domains, and each of them focuses on different aspect of the model and is represented by one SysML block definition diagram:

- ResCoM_ProductRCBreakdown_and _Design

    - The identification and composition of a design from a multiple lifecycles viewpoint.
    - The definition of product configuration.
    - The identification of a breakdown from a multiple lifecycles configuration management viewpoint.
    - The definition and composition of RCLs and their realizations to designs.
- ResCoM_Requirements
    - The identification and composition of product requirements of the ResCoM product specific requirements.
    - The definition of verifications and their applicability to design and verify requirements of a design.
    - The identification of evidences that support the verifications.
- ResCoM_InLife
    - The identification and composition of serialized assets throughout their multiple lifecycles.
    - The capture of feedback on the usage and condition of a product.
    - The realization of serialized assets to product designs and RCLs.
- ResCoM_Activities_and_Methods
    - The identification and composition of activities which are required for closing the product system.
    - The methods, planning and scheduling of such activities.
    - The objects that the methods perform on.

2.3 ResCoM_ProductRCBreakdown_and_Design

The ResCoM_ProductRCBreakdown_and_Design model in Figure 2-2 captures the essential concepts and properties for representing the design of multiple lifecycles products, the product lifecycles definitions i.e. RCL definitions, as well as defines the relationships between the concepts.

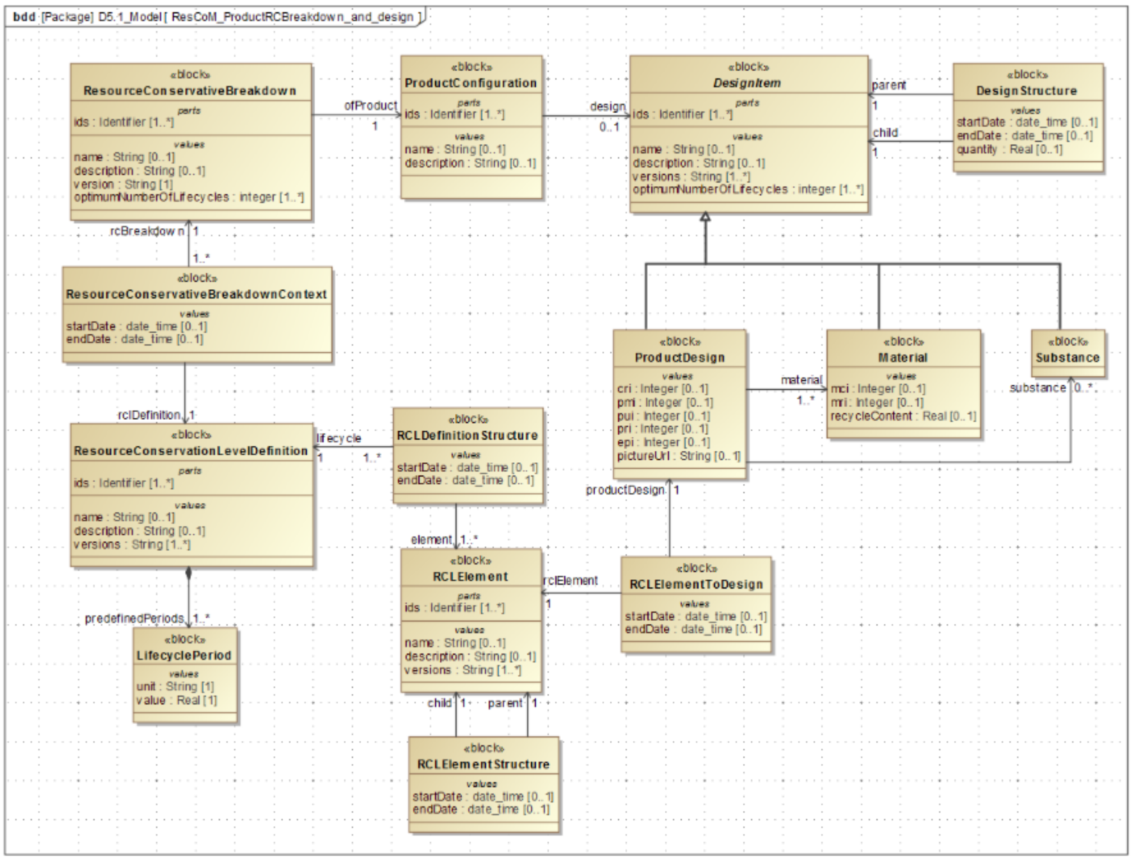

bdd [Package] D5.1_Model [ResCoM_ProductRCBreakdown_and_design]
«block» ResourceConservativeBreakdown
«block» ProductConfiguration
«block» DesignItem
«block» DesignStructure
«block» ResourceConservativeBreakdownContext
«block» ProductDesign
«block» Material
«block» Substance
«block» ResourceConservationLevelDefinition
«block» RCLDefinitionStructure
«block» RCLElement
«block» RCLElementToDesign
«block» LifecyclePeriod
«block» RCLElementStructure
ofProduct
design
parent
child
rcBreakdown
rclDefinition
lifecycle
material
substance
productDesign
element
rclElement
predefinedPeriods

Figure 2-2. ResCoM_ProductRCBreakdown_and_Design as a SysML block diagram

The ResCoM_ProductRCBreakdown_and_Design model concepts, relationships and properties are described in Table 1.

Table 1. The concepts, relationships and properties of the ResCoM_ProductRCBreakdown_and_Design model

Block / Relationship	Description	Concept reference
Block: ResourceConservativeBreakdown	A ResourceConservativeBreakdown provides a mean to subdivide a RCP (Resource Conservative Product) into a set of related RCLs (Resource Conservative Levels) to which additional information can be attached. This usually takes the form of a tree.	• ResCoM: RCP [1] • PLCS: Breakdown [2]
Block: ResourceCoversativeBreakdownContext	Membership relationship between ResourceConservativeLevelDefinition and ResourceConservativeBreakdown of which the definition is a member.	• PLCS: BreakdownContext, Effectivity and EffectivityAssignment [2]
Relationship: rcBreakdown	An association represents that at least one ResourceConservativeBreakdownContext is associated to one ResourceConservativeBreakdown.	
Block: ResourceConservationLevelDefinition	Identification of RCL definitions i.e. lifecycles definitions of a RCP.	• ResCoM: RCL [1] • PLCS: PhysicalElement [2]

Relationship: rclDefinition	An association represents in each ResourceConservativeBreakdownContext one ResourceConservativeLevelDefinition (i.e.RCL) will be defined.	
Block: LifecyclePeriod	Predefined length (time or performance) of a RCL, which is a Property of RCL.	• ResCoM: Predefined length [9]
Relationship: predefinedPeriods	A composition represents one ResourceConservationLevelDefinition defines at least one LifePeriod.	
Block: RCLDefinitionStructure	Relationship between a ResourceConservativeLevelDefinition and a RCLElement where RCLElement is regarded as a child.	• PLCS: PhysicalElementUsage, Effectivity and EffectivityAssignment [2]
Relationship: lifecycle	An association represents that in one RCL there will be at least one RCLDefinitionStructure.	
Block: RCLElement	A basic element of a ResourcConservationLevelDefinition that represents a physical sub part (module or component) of RCP.	• ResCoM: Module [12] • PLCS: PhysicalElement [2]
Relationship: element	An association represents that one RCLDefinitionStructure consists of at least one RCLElement.	
Block: RCLElementStructure	Relationship between a parent and child RCLElement.	• PLCS: PhysicalElementUsage, Effectivity and EffectivityAssignment [2]
Relationship: child, parent	Associations between RCLElement and its constituent lower level RCLElement.	
Block: *DesignItem*	An abstract class. The identification of a design, a generalization of the design objects. It is a collector of data common to all revisions of the *DesignItem*.	• PLCS: Product [2]
Block: DesignStructure	Relationship between a parent and child *DesignItem*	• PLCS: NextAssemblyViewUsage , Effectivity and EffectivityAssignment [2]
Relationship: child, parent	Associations between *DesignItem* and its constituent lower level *DesignItem*.	
Block: ProductConfiguration	The identification of a product concept (multiple lifecycles design) as a configuration.	• ResCoM: Standardization and Compatibility, upgradability and adaptability [12] • PLCS: ProductConfiguration [2]
Relationship: design	An association represents that one ProductConfiguration is associated to zero or one *DesignItem*.	
Relationship: ofProduct	An association represents that one ResourceConservativeBreakdown is a breakdown of one ProductConfiguration (product concept).	

Block: ProductDesign	Specialization of *DesignItem* that collects the definitional information of the versions of a product.	• ResCoM: Product/component indexes [1] • PLCS: Part [2]
Block: Material	Specialization of *DesignItem* that collects the definitional information of the versions of a non-countable material.	• ResCoM: Material indexes [1] • PLCS: Part [2]
Block: Substance	Specialization of *DesignItem*.	• ResCoM: Conformity to legislation [12] • PLCS: Part [2]
Relationship: *Specialization* Note: a child block will inherit any properties that its parent block has, but may have additional properties that make the child block special.	A specialization where *DesignItem* is parent block and ProductDesign, Material, and Substance are child blocks. It represents that one *DesignItem* could have three types which are ProductDesign, Material, and Substance.	
Relationship: material	An association represents one ProductDesign is associated with at least one Material.	
Relationship: substance	An association represents one ProductDesign is associated with zero or more Substance.	
Block: RCLElementToDesign	Relationship between a RCLElement and a ProductDesign that is realized by the element.	• PLCS: BreakdownElementReali-zation, Effectivity and EffectivityAssignment [2]
Relationship: rclElement	An association represents one RCLElementToDesign is associated with one RCLElement.	
Relationship: productDesign	An association represents one RCLElementToDesign is associated with one ProductDesign.	
Property	**Description**	**Comment**
Parts		
ids : Identifier [1..*]	A set of Identifiers for the ResourceConservativeBreakdown/ ResourceConsercationLevelDefinition/ RCLElement/ProductConfiguration/ *DesignItem.*	An identifier provides the identifying code for the product data.
Values		
name : String [0..1]	A name provides the identifying name in terms of String for the ResourceConservativeBreakdown/ ResourceConsercationLevelDefinition/ RCLElement/ProductConfiguration/ *DesignItem.*	
description : String [0..1]	A set of text based descriptions of the ResourceConservativeBreakdown/ ResourceConsercationLevelDefinition/ RCLElement/ProductConfiguration/ *DesignItem*.	

version : String [1]	A version identifies a version of a ResourceConservativeBreakdown i.e. a RCP.	A version serves as the collector of the data characterizing a realizable object in various application contexts.
version : String [1..*]	A version identifies a version of ResourceConsercationLevelDefinition/ RCLElement/*DesignItem*.	
startDate : date_time [0..1]	A date is assigned to a relationship/structure between two objects showing when this relationship/structure begins.	Relationship/structure between: • ResourceConservativeBreak-down and ResourceConservationLevel-Definition • ResourceConservationLevel-Definition and RCLElement • RCLElement and its sub parts • RCLElement and ProductDesign • *DesignItem* and its sub parts
endDate : date_time [0..1]	A date is assigned to a relationship/structure between two objects showing when this relationship/structure ends up.	
optimumNumberOfLifecycles : integer [1..*]	The optimum number of lifecycles of defined for a ResourceConservativeBreakdown (RCP) /*DesignItem*.	ResCoM concept [1]
unit : String [1]	A LifecyclePeriod is specified by a value and a unit.	
value : Real [1]		
quantity : Real [1]	The quantity of a child *DesignItem* constitutes its parent *DesignItem*.	
cri : Integer [0..1]	Component reusability index for a ProductDesign.	ResCoM concept [1]
pmi : Integer [0..1]	Product modularity index for a ProductDesign.	
pui : Integer [0..1]	Product upgradability index for a ProductDesign.	ResCoM concept [1]
pri : Integer [0..1]	Product reusability index for a ProductDesign.	
epi : Integer [0..1]	EoL potential index for a ProductDesign.	
pictureUrl : String [0..1]	The URL of an image of a ProductDesign. This URL must be accessible from the reporting services server.	Integration with Granta BOM Analyzer
mci : Integer [0..1]	Material criticality index for a Material.	ResCoM concept [1]
mri : Integer [0..1]	Material reusability index for a Material.	
recycleContent : Real [0..1]	The percentage of recycled Material in the current supply. If omitted then 0% is assumed.	Integration with Granta BOM Analyzer

2.4 ResCoM_Requirements

The ResCoM_Requirements model in Figure 2-3 captures mainly the different types of ResCoM product specific requirements, and the traceability and structure of the requirements.

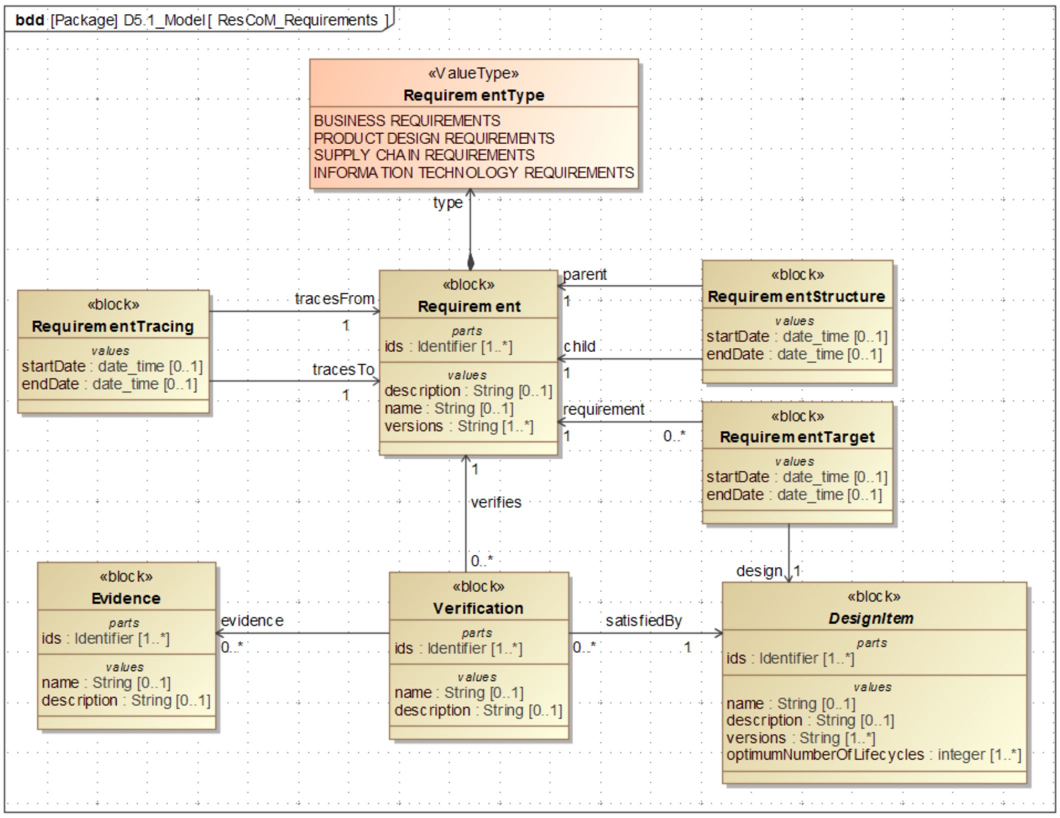


Figure 2-3. ResCoM_Requirements model as a SysML block diagram

Note that a block can participate in several SysML block definition diagrams for different purposes, e.g.”*DesignItem”* block which has been shown in the previous diagram also appears in this diagram to represent relationships with other blocks, and it will also appear in other diagrams in the latter part of this document.

The ResCoM_Requirements model concepts, relationships and properties are described in Table 2.

Table 2. The concepts, relationships and properties of the ResCoM_Requirements model

Block / Relationship	Description	Concept reference
Block: Requirement	A statement that identifies a necessary capability, function, characteristic, or quality of a target item, e.g. a product or service. A Requirement is used to uniquely identify a requirement.	• ResCoM: ResCoM product specific requirements [12] • PLCS: Requirement [2] • System Engineer : Requirement [7]
Relationship: type	A composition represents that one Requirement defines one Value Type.	
Block: RequirementTracing	A RequirementTracing shows tracing from one Requirement to another.	• ResCoM: essential and critical factors [D4.1 and D4.2] • PLCS: TracingRelationship, Effectivity and EffectivityAssignment [2]
Relationship: tracesFrom, tracesTo	An association represents tracing relationship from one Requirement (TracesFrom) to another Requirement (TracesTo).	
Block: RequirementStructure	Relationship between a parent and child Requirement.	• ResCoM: ResCoM product specific requirements [12] • PLCS: RequirementCollection-Relationship, Effectivity and EffectivityAssignment [2]
Relationship: child, parent	Associations between one Requirement and its constituent lower level Requirement.	
Block: RequirementTarget	A RequirementTarget is used to relate a Requirement to *DesignItem* which are affected by the Requirement.	• ResCoM: ResCoM product specific requirements [12] • PLCS: RequirementAssignment, Effectivity and EffectivityAssignment [2]
Relationship: requirement	An association represents zero or more RequirementTarget is associated with one Requirement.	
Block: *DesignItem*	An abstract class. The identification of a design, a generalization of the design objects. It is a collector of data common to all revisions of the *DesignItem*.	• PLCS: Product [2]
Relationship: design	An association represents one RequirementTarget targets to one *DesignItem*.	
Block: Verification	A Verification is an objective assertion that a claim that a requirement is satisfied by a particular item is. Verification ensures that the specified requirements have been met. Verification uses the methods of Test, Analysis, Inspection, Demonstration, and Similarity.	• PLCS: Verification [2]

Relationship: satisfiedBy	An association represents zero or more Verification claims that a Requirement is satisfied by a particular *DesignItem*.	
Relationship: verifies	An association represents zero or more Verification claims that a Requirement is satisfied by a particular *Designitem* has been verified.	
Block: Evidence	An Evidence is a collector of items used together to provide a single piece of evidence within a Verification.	• PLCS: Evidence [2]
Relationship: evidence	An association represents zero or more Evidence supports one Verification.	
Property	**Description**	**Comment**
Value Type		
Business requirements	Requirements on the service models that supports product returns to OEMs	ResCoM concept [1]
Product design requirements	Requirement on the product design for multiple lifecycles	
Supply chain requirements	Requirements on integrating forward and reverse supply chains solutions that can handle the dynamics of multiple lifecycles.	
Information technology requirements	Requirement on the collaborative product lifecycle management software platform supporting multiple lifecycles.	
Parts		
ids : Identifier [1..*]	A set of Identifiers for the Requirement /*DesignItem*/Verification/Evidence.	An identifier provides the identifying code for the product data.
Values		
name : String [0..1]	A name provides the identifying name in terms of String for the Requirement /*DesignItem*/Verification/Evidence.	
description : String [0..1]	A set of text based descriptions of the Requirement /*DesignItem*/Verification/Evidence.	
version : String [1..*]	A version identifies a version of a Requirement/*DesignItem*.	A version serves as the collector of the data characterizing a realizable object in various application contexts.
startDate : date_time [0..1]	A date is assigned to a relationship/structure between two objects showing when this relationship/structure begins.	Relationship/structure between: • Requirement (TracesFrom) and Requirement (TracesTo) • Requirement and its sub requirement • Requirement and its target *DesignItem*
endDate : date_time [0..1]	A date is assigned to a relationship/structure between two objects showing when this relationship/structure end up.	

optimumNumberOfLifecycles : integer [1..*]	The predefined optimum number of lifecycles for a *DesignItem*.	• ResCoM concept [1]

2.5 ResCoM_InLife

The ResCoM_InLife model in Figure 2-4 covers the multiple lifecycles design, lifecycles definition, predefined product structure of each lifecycle for a specific manufactured product i.e. SerializedAsset.

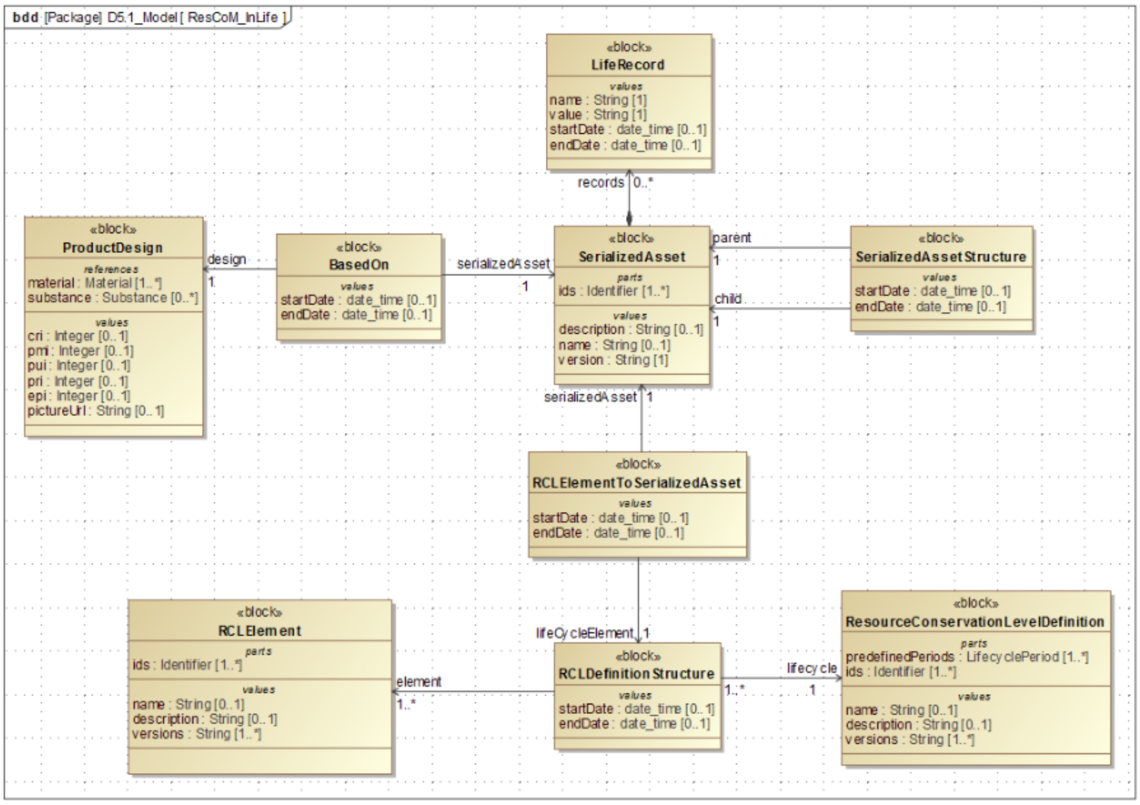

bdd [Package] D5.1_Model [ResCoM_InLife]
«block» LifeRecord
values
name : String [1]
value : String [1]
startDate : date_time [0..1]
endDate : date_time [0..1]
records
0..*
«block» ProductDesign
references
material : Material [1..*]
substance : Substance [0..*]
values
cri : Integer [0..1]
pmi : Integer [0..1]
pui : Integer [0..1]
pri : Integer [0..1]
epi : Integer [0..1]
pictureUrl : String [0..1]
design
1
«block» BasedOn
values
startDate : date_time [0..1]
endDate : date_time [0..1]
serializedAsset
1
«block» SerializedAsset
parts
ids : Identifier [1..*]
values
description : String [0..1]
name : String [0..1]
version : String [1]
parent
1
child
1
«block» SerializedAssetStructure
values
startDate : date_time [0..1]
endDate : date_time [0..1]
serializedAsset
1
«block» RCLElementToSerializedAsset
values
startDate : date_time [0..1]
endDate : date_time [0..1]
lifeCycleElement
1
«block» RCLElement
parts
ids : Identifier [1..*]
values
name : String [0..1]
description : String [0..1]
versions : String [1..*]
element
1..*
«block» RCLDefinitionStructure
values
startDate : date_time [0..1]
endDate : date_time [0..1]
1..*
lifecycle
1
«block» ResourceConservationLevelDefinition
parts
predefinedPeriods : LifecyclePeriod [1..*]
ids : Identifier [1..*]
values
name : String [0..1]
description : String [0..1]
versions : String [1..*]

Figure 2-4. ResCoM_InLife model as a SysML block diagram

The ResCoM_InLife model concepts, relationships and properties are presented in Table 3**Table 3**.

Table 3. The concepts, relationships and properties of the ResCoM_InLife model.

Block / Relationship	Description	Concept reference
Block: SerializedAsset	A SerializedAsset identifies an individual product that has been manufactured.	• PLCS: ProductAsRealized [2]
Block: SerializedAssetStructure	Relationship between a parent and child SerializedAsset.	• PLCS: RealizedAssemblyRelation-ship, Effectivity and EffectivityAssignment [2]
Relationship: child, parent	Associations between SerializedAsset and its constituent lower level SerializedAsset.	

Block: ProductDesign	Specialization of *DesignItem* that collects the definitional information of the versions of a product.	• ResCoM: Product/component indexes [1] • PLCS: Part [2]
Block: BasedOn	A BasedOn is a relationship between a ProductDesign, and the product that has been made from the design, represented by SerializedAsset.	• PLCS: ProductDesignVersionTo-Individual, Effectivity and EffectivityAssignment [2]
Relationship: design	An association represents that one BasedOn basese on one ProductDesign.	
Relationship: serializedAsset	An association represents that one BasedOn associating with one SerializedAsset.	
Block: LifeRecord	The identification of a life record that records each RCL a SerializedAsset has gone through and the usage state representing by value within this RCL.	• ResCoM: traceability of each RCL [1] • PLCS: Effectivity and EffectivityAssignment [2]
Relationship: records	A composition represents one SerializedAsset defines zero or more LifeRecord.	
Block: RCLElementToSerializedAsset	Relationship between a SerializedAsset and its predefined structure in each RCL.	• PLCS: BreakdownElementReali-zation, Effectivity and EffectivityAssignment [2]
Relationship: serializedAsset	An association represents that one RCLElementToSerializedAsset associating with one SerializedAsset.	
Block: RCLDefinitionStructure	Relationship between a ResourceConservativeLevelDefinition and a RCLElement where RCLElement is regarded as a child.	• PLCS: PhysicalElementUsage, Effectivity and EffectivityAssignment [2]
Relationship: lifecycleElement	An association represents that one RCLElementToSerializedAsset associating with one RCLDefinitionStructure.	
Block: RCLElement	A basic element of a ResourcConservationLevelDefinition that represents a physical sub part (module or component) of RCP.	• ResCoM: Module [12] • PLCS: PhysicalElement [2]
Relationship: element	An association represents that one RCLDefinitionStructure consists of at least one RCLElement.	
Block: ResourceConservationLevelDefinition	Identification of RCL definitions i.e. lifecycles definitions of a RCP.	• ResCoM: RCL [1] • PLCS: PhysicalElement [2]
Relationship: lifecycle	An association represents that in one RCL there will be at least one RCLDefinitionStructure.	

Property	Description	Comment
Parts		

ids : Identifier [1..*]	A set of Identifiers for the SerializedAsset/ ResourceConservationLevelDefinition/RCL Element.	An identifier provides the identifying code for the product data.
predefiendPeriods: LifecyclePeriod [1..*]	Predefined length (time or performance) of a RCL, which is a property of RCL.	The hidden block "LifecyclePeriod" which is shown in Figure 2-2 ResCoM_ProductRCBreak down_and_Design model.
references		
material :Material[1..*]	Specialization of *DesignItem* that collects the definitional information of the versions of a non-countable material.	The hidden block "Material" which is shown in Figure 2-2 ResCoM_ProductRCBreak down_and_Design model.
substance : Substance [0..*]	Specialization of *DesignItem*.	The hidden block "Substance" which is shown in Figure 2-2 ResCoM_ProductRCBreak down_and_Design model.
Values		
name : String [0..1]	A name provides the identifying name in terms of String for the SerializedAsset/ ResourceConservationLevelDefinition/RCL Element/LifeRecord.	
description : String [0..1]	A set of text based descriptions of the SerializedAsset/ ResourceConservationLevelDefinition/RCL Element.	
version : String [1]	A version identifies a version of a SerializedAsset i.e. a manufactured RCP.	A version serves as the collector of the data characterizing a realizable object in various application contexts.
version : String [1..*]	A version identifies a version of ResourceConsercationLevelDefinition/RCL Element.	
startDate : date_time [0..1]	A date is assigned to a relationship/structure between two objects showing when this relationship/structure begins.	Relationship/structure between: • ProductDesign and SerializedAsset • SerializedAsset and its sub parts • SerializedAsset and its property LifeRecord • SerializedAsset and RCLDefinitionStructure • RCLElement and ResourceConservation-LevelDefinition
endDate : date_time [0..1]	A date is assigned to a relationship/structure between two objects showing when this relationship/structure end up.	
value : Real [1]	A LifeRecord is specified by a value.	
cri : Integer [0..1]	Component reusability index for a ProductDesign.	ResCoM concept [1]
pmi : Integer [0..1]	Product modularity index for a ProductDesign.	

pui : Integer [0..1]	Product upgradability index for a ProductDesign.	
pri : Integer [0..1]	Product reusability index for a ProductDesign.	
epi : Integer [0..1]	EoL potential index for a ProductDesign.	
pictureUrl : String [0..1]	The URL of an image of a ProductDesign. This URL must be accessible from the reporting services server.	Integration with Granta BOM Analyzer

2.6 ResCoM_Activities_and_Methods model

The ResCoM_Activity_and_Methods model in Figure 2-5 covers activity for closing the loop of the product system and supporting multiple lifecycles of the RCP, including predefined activity methods for supporting first manufacturing for a RCP (design) and operational value recovery methods in each of its end of lifecycle, plan and execute the manufacturing and operational value recovery for a specific product individual (SerializedAsset) throughout its lifecycles.

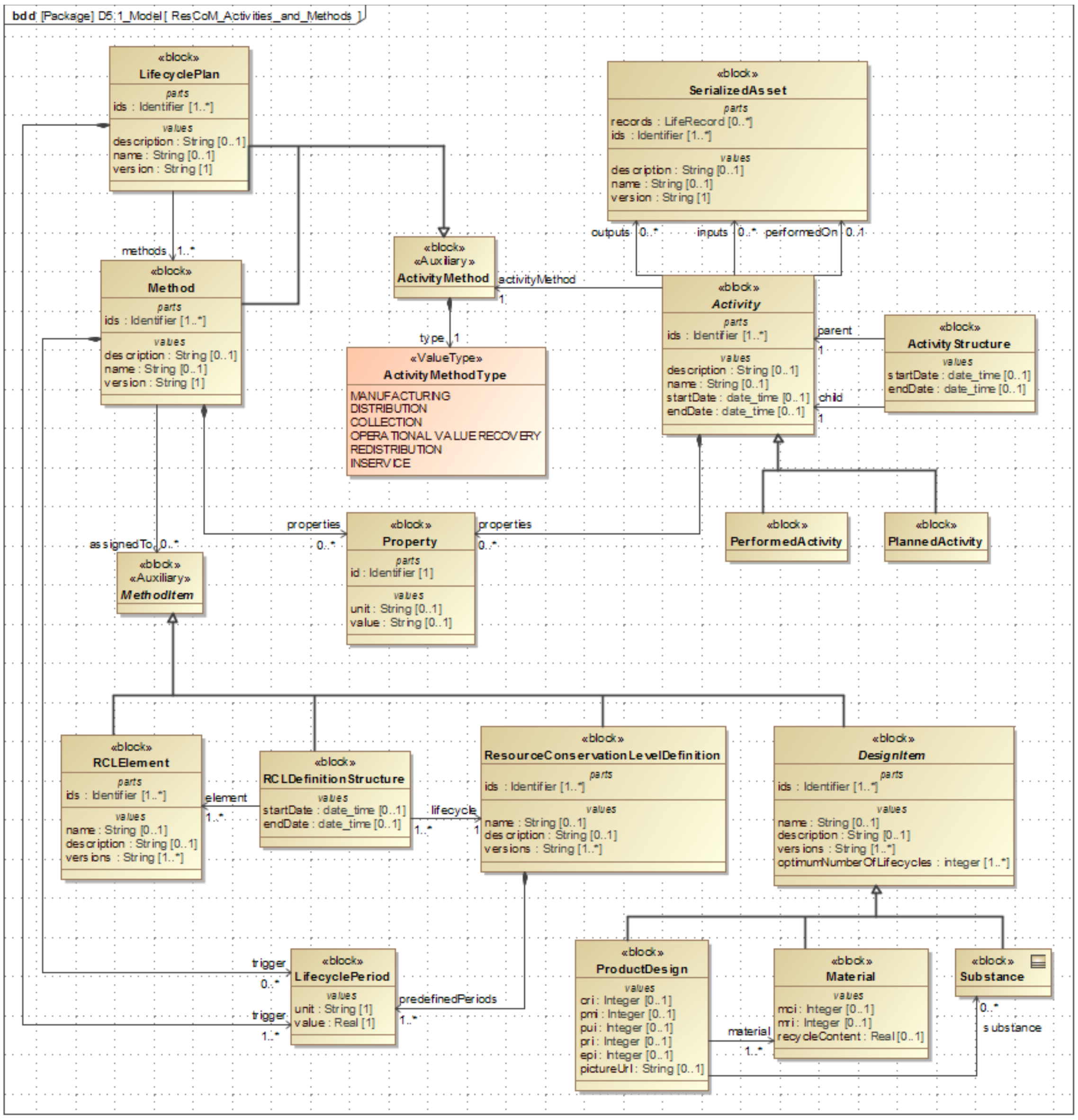


Figure 2-5. ResCoM_Activities_and_Methods model a SysML block diagram

The ResCoM_Activities_and_Methods model concepts, relationships and properties are presented in Table 4.

Table 4. The concepts, relationships and properties of the ResCoM_Activities_and_Methods model

Block / Relationship	Description	Concept reference
Block: SerializedAsset	A SerializedAsset identifies an individual product that has been manufactured.	• PLCS: ProductAsRealized [2]
Block: *Activity*	An abstract class. An *Activity* is an action or a set of actions that consume time and resources and whose performance is necessary to achieve, or contribute to, the realization of one or more outcomes.	• ResCoM: Production activities, supply chain activities [12] • PLCS: Activity [2]
Relationship: outputs	An association represents that zero or more SerializedAsset is the input to one *Activity*.	
Relationship: inputs	An association represents that zero or more SerializedAsset is the output to one *Activity*.	
Relationship: performedOn	An association represents that one *Activity* is performed on zero or one SerializedAsset.	
Block: ActivityStructure	Relationship between a parent and child *Activity*.	• PLCS: ActivityRelationship [2]
Relationship: child, parent	Associations between *Activity* and its constituent lower level *Activity*.	
Block: PerformedActivity	A PerformedActivity is an *Activity* that has started but not necessarily finished.	• PLCS: ActivityActual [2]
Block: PlannedActivity	A PlannedActivity is an *Activity* which, when first defined, has yet to start and so is being planned. It is a record of the intent to perform an *Activity*.	• PLCS: ActivityPlanned [2]
Relationship: *Specialization* Note: a child block will inherit any properties that its parent block has, but may have additional properties that make the child block special.	A specialization where *Activity* is parent block and PerformedActivity and PlannedActivity are child blocks. It represents that one *Activity* could have two types which are PerformedActivity and PlannedActivity.	
Block: ActivityMethod	An ActivityMethod is a specific way to carry out an *Activity*.	• PLCS: ActivityMethod [2]
Relationship: type	A composition represents that one ActivityMethod defines one Value Type.	

Relationship: activityMethod	Associations between *Activity* and ActivityMethod i.e. the specific way to carry out the *Activity*.	
Block: LifecyclePlan	A LifecyclePlan is a specialization of ActivityMethod predefines a set of activity methods for supporting a RCP (design) from first manufacturing to operational value recovery in each end of its lifecycle.	• ResCoM: Right time, Right quality, Right Quantity [12] • PLCS: Scheme [2]
Block: Method	A TaskMethod is a specialization of ActivityMethod. It is a specification of work.	• PLCS: TaskMethod [2]
Relationship: *Specialization* Note: a child block will inherit any properties that its parent block has, but may have additional properties that make the child block special.	A specialization where ActivityMethod is parent block and LifecyclePlan and Method are child blocks. It represents that one ActivityMethod could have two types which are LifecyclePlan and Method.	
Relationship: methods	An association represents that one LifecyclePlan is associated with at least one Method.	
Block: Property	A Property is the record of an attribute or characteristic that is applicable to something.	• PLCS: Property [2]
Relationship: properties	An association between Method and Property represents that one Method has zero or more Property.	
Relationship: properties	An association between *Activity* and Property represents that one *Activity* has zero or more Property.	
Auxiliary Block: *MethodItem*	A *MethodItem* is an abstract generalization of instances.	• PLCS: *TaksAssignmentSelect* [2]
Relationship: assignedTo	An association represents that one Method is assigned to zero to more *MethodItem*.	
Relationship: *Specialization* Note: a child block will inherit any properties that its parent block has, but may have additional properties that make the child block special.	A specialization where *MethodItem* is parent block and RCLElement, ResourceConservationLevelDefinition, *DesignItem* and RCLDefinitionStructure are child blocks. It represents that one MethodItem could have four types which are RCLElement, ResourceConservationLevelDefinition, *DesignItem* and RCLDefinitionStructure.	
Block: RCLElement	A basic element of a ResourcConservationLevelDefinition that represents a physical sub part (module or component) of RCP.	• ResCoM: Module [12] • PLCS: PhysicalElement [2]
Block: RCLDefinitionStructure	Relationship between a ResourceConservativeLevelDefinition and a RCLElement where RCLElement is regarded as a child.	• PLCS: PhysicalElementUsage, Effectivity and EffectivityAssignment [2]
Relationship: element	An association represents that one RCLDefinitionStructure consists of at least one RCLElement.	

Block: ResourceConservationLevelDefinition	Identification of RCL definitions i.e. lifecycles definitions of a RCP.	• ResCoM: RCL [1] • PLCS: PhysicalElement [2]
Relationship: lifecyle	An association represents that in one RCL there will be at least one RCLDefinitionStructure.	
Block: *DesignItem*	The identification of a design, a generalization of the design objects. It is a collector of data common to all revisions of the *DesignItem*.	• PLCS: Product [2]
Block: ProductDesign	Specialization of *DesignItem* that collects the definitional information of the versions of a product.	• ResCoM: Product/component indexes [1] • PLCS: Part [2]
Block: Material	Specialization of *DesignItem* that collects the definitional information of the versions of a non-countable material.	• ResCoM: Material indexes [1] • PLCS: Part [2]
Block: Substance	Specialization of *DesignItem*.	• ResCoM: Conformity to legislation [12] • PLCS: Part [2]
Relationship: *Specialization* Note: a child block will inherit any properties that its parent block has, but may have additional properties that make the child block special.	A specialization where *DesignItem* is parent block and ProductDesign, Material, and Substance are child blocks. It represents that one *DesignItem* could have three types which are ProductDesign, Material, and Substance.	
Relationship: material	An association represents one ProductDesign is associated with at least one Material.	
Relationship: substance	An association represents one ProductDesign is associated with zero or more Substance.	
Block: LifecyclePeriod	Predefined length (time or performance) of a RCL, which is a property of RCL.	ResCoM: Predefined length [9]
Relationship: predefinedPeriods	A composition represents one ResourceConservationLevelDefinition defines at least one LifecyclePeriod.	
Relationship: trigger	A composition (LifecyclePeriod is the composition of a LifecyclePlan) represents one LifecyclePlan includes at least one LifecyclePeriod which will trigger the LifecyclePlan.	
Relationship: trigger	A composition (LifecyclePeriod is the composition of a Method) represents one Method includes zero or more LifecyclePeriod which will trigger the Method.	

Property	Description	Comment
Value Type		
Manufacturing	RCL_0 production	ResCoM concept [12]
Distribution	Distribution of RCP in its first lifecycle	

Collection	Collection of RCP in each of its end of lifecycle	
Operational value recovery	RCL$_i$ Production	
Redistribution	Distribution of RCP in its 2nd, 3rd… lifecycle	
In service	Maintenance during each lifecycle	
Parts		
ids : Identifier [1..*]	A set of Identifiers for the SerializedAsset/ *Activity*/LifecyclePlan/Method/Property/ RCLElement /ResourceConservationLevelDefinition/ *DesignItem*.	An identifier provides the identifying code for the product data.
records : LifeRecord [0..*]	The identification of a life record that records each RCL a SerializedAsset has gone through and the usage state representing by value within this this RCL.	The hidden block "LifeRecord" which is shown in Figure 2-4 ResCoM in life model.
Values		
name : String [0..1]	A name provides the identifying name in terms of String for the SerializedAsset/ *Activity*/LifecyclePlan/Method/ RCLElement /ResourceConservationLevelDefinition/ *DesignItem*.	
description : String [0..1]	A set of text based descriptions of the SerializedAsset/ *Activity*/LifecyclePlan/Method/ RCLElement /ResourceConservationLevelDefinition/ *DesignItem*.	
version : String [1]	A version identifies a version of a SerializedAsset/LifecyclePlan/Method. A version serves as the collector of the data characterizing a realizable object in various application contexts.	A version serves as the collector of the data characterizing a realizable object in various application contexts.
version : String [1..*]	A version identifies a version of RCLElement /ResourceConservationLevelDefinition/ *DesignItem*.	
startDate : date_time [0..1]	A date is assigned to a relationship/structure between two objects showing when this relationship/structure begins or a date is assigned to start an activity.	Relationship/structure between: • *Activity* and its sub parts • Start and end dates of an *Activity* • RCLElement and ResourceConservation-LevelDefinition
endDate : date_time [0..1]	A date is assigned to a relationship/structure between two objects showing when this relationship/structure end up, or a date is assigned to end up an activity.	
optimumNumberOfLifecycles : integer [1..*]	The optimum number of lifecycles of defined for a *DesignItem*.	ResCoM concept [1]
unit : String [1]	A Property/LifecyclePeriod is specified by a value and a unit.	
value : Real [1]		

cri : Integer [0..1]	Component reusability index for a ProductDesign.	ResCoM concept [1]
pmi : Integer [0..1]	Product modularity index for a ProductDesign.	
pui : Integer [0..1]	Product upgradability index for a ProductDesign.	
pri : Integer [0..1]	Product reusability index for a ProductDesign.	
epi :Integer [0..1]	EoL potential index for a ProductDesign.	
pictureUrl : String [0..1]	The URL of an image of a ProductDesign. This URL must be accessible from the reporting services server.	Integration with Granta BOM Analyzer
mci : Integer [0..1]	Material criticality index for a Material.	ResCoM concept [1]
mri : Integer [0..1]	Material reusability index for a Material.	
recycleContent : Real [0..1]	The percentage of recycled material in the current supply. If omitted then 0% is assumed.	Integration with Granta BOM Analyzer

3 Conclusion

This document describes the ResCoM initial standard data model which was modelled by starting from industrial proven life cycle standards for product life cycle information management (ISO 10303 233 and 239), and then semantics from the closed loop, multi life cycle area were added based on the ResCoM product specific requirements collected from anterior ResCoM work.

The initial standard data model is the basis of the ResCoM collaborative software platform development, including development of ResCoM Web API, and Share-A-space solution for managing product having multiple lifecycles. The data model will be validated using Share-A-space and industrial test cases.

Another intended uses of the initial standard data model is to provide a basic framework as a support for collecting further information on multiple lifecycles product management or requirements on the collaborative software platform from other three pillars (i.e. product design, business model and supply chain), including but not limit on collecting the following information for extending and refining the data model on the following work:

- Collaborative Software Platform
    - What IT tools will be used for analysis and decision making in product design, business model, and supply chain.
    - How will the tools be used?
    - What inputs are needed for doing the analysis and what outputs can be gotten?
- Product requirement management
    - Definition of the generic ResCoM product specific requirements (from D2.3)
    - Definition of the ResCoM product specific requirements for each of the OEMs' case study (from D2.3)
    - Hierarchy of the requirements
    - Interactions among the requirements (from D4.1 and D4.2)
    - Requirements verification
- Product configuration (from WP3 and WP4)
    - Product BOM (engineering BOM, manufacturing BOM, maintenance BOM etc.)
    - Definition of configuration management activities over the complete multiple lifecycles
    - Compatibility of the modular/components
- Product traceability (from WP3 and WP4)
    - Definition of the multiple lifecycles of a product design including Product level, modular level, component level, and material level.
    - How to identify an individual product, and on what the level that the product should be kept tracing?
    - How to capture of feedback on the usage and condition of a product, and who owns the data and who has right of access?
- Value creation and recovery activities, processes, resources, distribution channels, partners, supplier and customers

Modelling, refining, validating and extending the data model will be an iterative process and when final closed loop information management requirements are set, the profile will be updated and be documented as D5.2 Final standard data model to be ready for communicate with the OASIS and ISO standards.

References

[1] ResCoM DOW (2013) *603843_DOW*

[2] ISO International Organization for Standardization (2012) *ISO 10303-233:2012. Industrial automation systems and integration – Product data representation and exchange – Part 233: Application protocol: Systems engineering.* Geneva.

[3] Dunford, J., Bergtröm, P. (2007). *Standards-based PLM: Re-engineering the Aftermarket with PLCS Part 2 Technologies and Deployments*. Eurostep and John Stark Associates Technology White Paper

[4] PLCS Introduction http://www.plcs.org/plcslib/plcslib/. Accessed 30 Sep 2014

[5] PLCS activity model http://www.plcs.org/plcslib/plcslib/sys/activity_model_index_base.html. Accessed 30 Sep 2014.

[6] PLCS information model http://www.plcs.org/plcslib/plcslib/data/PLCS/psm_model/model_base.html. Accessed 30 Sep 2014.

[7] ISO International Organization for Standardization (2012) *ISO 10303-239:2012 Ed. 2. Industrial automation systems and integration – Product data representation and exchange – Part 239: Application protocol: Product life cycle support.* Geneva.

[8] Asif FMA (2011) *Resource conservative manufacturing: a new generation of manufacturing. Licentiate thesis*, KTH Royal Institute of Technology.

[9] Rashid, A., et al., (2013). *Resource Conservative Manufacturing: an essential change in business and technology paradigm for sustainable manufacturing.* Journal of Cleaner Production

[10] Fraunhofer Project Group Process Innovation (IPA-BT) (2014) *ResCoM deliverable 2.1: Product specific requirements-current practices.*

[11] Technische Universiteit Delft (2014) *ResCoM deliverable 2.2: Product specific requirements-state-of-the-art.*

[12] INSEAD (2014) *ResCoM deliverable 2.3: Product specific requirements-ResCoM.*

[13] IDEAL&CO Explore, DUT (2014) *ResCoM deliverable 3.2: Best design practices.*

[14] Ye, X., Zhang, X. (2013), *PLM for Multiple Lifecycle Product: Concepts, terminologies, processes for collaborative information management,* Master Thesis, KTH Royal Institute of Technology, http://www.diva- portal.org/smash/get/diva2:693889/FULLTEXT01.pdf

[15] PLCS concept model http://www.plcs.org/plcslib/plcslib/data/PLCS/concept_model/model_base.html. Accessed 30 Sep 2014.

This project has received funding from the European Union's Seventh Programme for research, technological development and demonstration under grant agreement No 603843

ResCoM

Annex - SysML block definition diagram introduction

Contents

1 Basic modelling ... 3

2 Adding more detail to blocks ... 5

2.1 Block property types ... 5

3 Adding more detail to relationships ... 6

3.1 Specialization and generalization ... 6

3.2 Aggregation and composition ... 8

3.3 Dependencies ... 9

4 Summary ... 9

References ... 11

This project has received funding from the European Union's Seventh Programme for research, technological development and demonstration under grant agreement No 603843

1 Basic modelling

There are two basic elements that make up a block definition diagram, which are the '**block**' and the '**relationship**'. A 'block' represents a type of 'thing' that exists in the real world and, hence, should have a very close connection to reality. Figure 1 shows two very simple blocks. Blocks are represented graphically by rectangles in the SysML and each must have a name, which is written inside the rectangle. In order to understand the diagram, it is important to read the symbols. The diagram here shows that two blocks exist: 'Block 1' and 'Block 2'. This is one of the

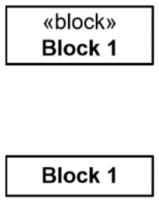


Figure 1. Representing blocks

The upper block has the word 'block' in chevrons ('«block»') in it, which signifies that this block is stereotyped. As every block in a block definition diagram contains this same adornment, it is usually left off the diagram, as shown in the lower block. For the purposes of this introduction, the «block» **stereotype** will be omitted from most diagrams.

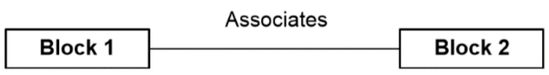


Figure 2. Representing a relationship

Figure 2 shows how to represent a relationship between two blocks. This particular relationship is known as an **association** and is simply ageneral type of relationship that relates one or more blocks. The association is represented by a line that joins two blocks, with the association name written somewhere on the line. This diagram reads: two blocks exist: 'Block 1' and 'Block 2' and 'Block 1' associates 'Block 2'.

Figure 3 shows two more examples that are based on real life. The top part of the diagram reads: there are two blocks: 'Dog' and 'Cat' where 'Dog' chases 'Cat'. Likewise, the lower part of the diagram reads: there are two blocks: 'Cat' and 'Mouse' where 'Cat' eats 'Mouse'. Figure 3 is actually ambiguous, as it could be read in one of two ways, depending on the direction in which the association is read. Take, for example, the top part of the diagram: who is to say that the diagram is to be read 'Dog' chases 'Cat' rather than 'Cat' chases 'Dog', as it is possible for both cases to be true. Therefore, there is some ambiguity as the diagram must be read in only one direction for it to be true; thus, a mechanism is required to indicate direction.

This project has received funding from the European Union's Seventh Programme for research, technological development and demonstration under grant agreement No 603843

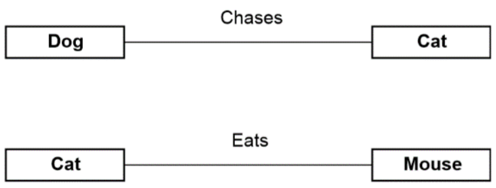


Figure 3. Examples of blocks and associations

The simplest way to show direction is to **placea direction marker on the association** that will dictate which way the line should be read, as shown in the top part of Figure 4. The diagram now reads 'Dog' chases 'Cat' and definitely not 'Cat' chases 'Dog' and is thus less ambiguous than Figure 3.

The second way to show direction is to **define a 'role' on each end of the association**, as shown in the middle part of Figure 4. In this case, the two roles that have been defined are 'chaser' and 'chasee', which again eliminates the ambiguity that existed in Figure 3.

The lower part of Figure 4 introduces a new association called 'dislikes'. This time, however, the lack of direction is intentional, as both statements of 'Dog' dislikes 'Cat' and 'Cat' dislikes 'Dog' are equally true. Therefore, when no direction is indicated, the association is said to be **bidirectional**.

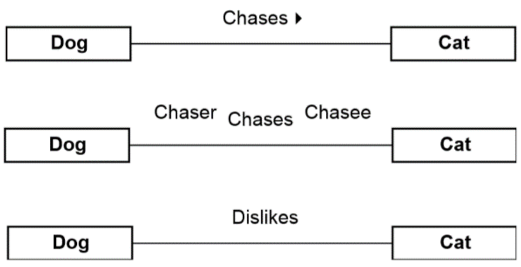
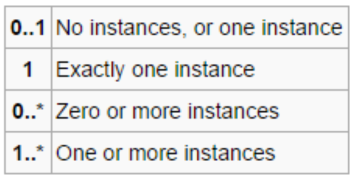


Figure 4. Showing direction

There is no concept of the number of cats and dogs involved in the chasing of the previous diagrams. Expressing this numbering is known as **multiplicity** (Figure 5), which is illustrated in Figure 6. The top part of Figure 6 shows that each 'Dog' chases one or more 'Cat'.



0..1	No instances, or one instance
1	Exactly one instance
0..*	Zero or more instances
1..*	One or more instances

Figure 5. Multiplicity

If no number is indicated, as in the case of the 'Dog' end of the association, it is assumed that the number is 'one'. Although the number is one, it does not necessarily indicate that there is only one dog, but rather that the association applies for each dog. The multiplicity at the other end of the 'chases' association states '1..*', which means'one or more' or some where between one and many. The lower part of the diagram is read as: one or more 'Dog' chases one or more

'Cat'. This could mean that a single dog chases a single cat, a single dog chases any number or a herd of cats, or that an entire pack of dogs is chasing a herd of cats.

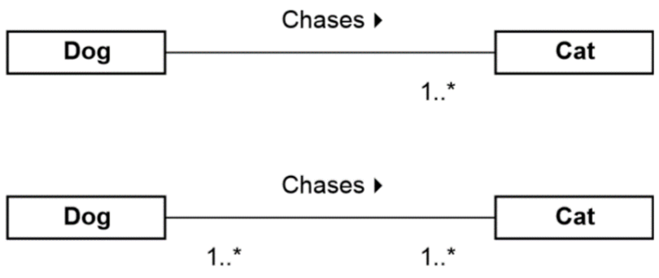


Figure 6. Showing numbers using multiplicity

2 Adding more detail to blocks

The block 'Cat' represented all cats that looked and behaved in the same way, but it is not defined anywhere how a cat looks or behaves. This section examines how to add this information to a block by using '**properties**'. For this example, suppose that we wish to represent the features 'age', 'weight', 'colour' and 'favourite food' on the block 'Cat'. These features are represented on the block as 'properties' – one for each feature (Figure 7).

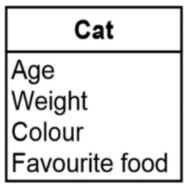


Figure 7. Properties of the block 'Cat'

Properties are written in a box below the block name box. When modelling, it is possible to add more detail at this point, such as the **visibility** of the property, **type**, **default values** and so forth.

The block 'Cat' is now fully defined for our purposes and the same exercise may be carried out on any other blocks in the diagram in order to populate the model fully. It should also be pointed out that the blocks may be left at a high level with no properties or operations. As with everything in the SysML, use only as much detail as is necessary, rather than as much as is possible.

2.1 Block property types

Property is a structural feature of a block, there are three property types (see Figure 14):

- Part property: typed by a block
  - Usage of a block in the context of the enclosing block
  - E.g. right-front: wheel
- Reference property: typed by a block
  - A part that is not owned by the enclosing block (not composition)
  - E.g. logical interface between 2 parts

- Value property: typed by value type
    - Defines a value with units, dimensions, and probability distribution
    - E.g. tirePressure:psi=30

3 Adding more detail to relationships

There are four types of relationship that will be discussed here: **‘association’, ‘specialization and generalization’ ‘aggregation and composition’, and ‘dependency’**. Many types of relationship exist, but these three represent the majority of the most common uses of relationships. Associations have already been introduced and shown to be a very general type of relationship that relate together one or more block.

3.1 Specialization and generalization

‘Specialization’ refers to the case when a block is being made more special or is being refined in some way. Specialization may be read as ‘has types’ whenever its symbol, a small triangle, is encountered on a model. If the relationship is read the other way around, then the triangle symbol is read as ‘is a type of’, which is a generalization.

Specialization is used to show ‘child’ blocks, sometimes referred to as ‘sub- blocks’, of a ‘parent’ block.

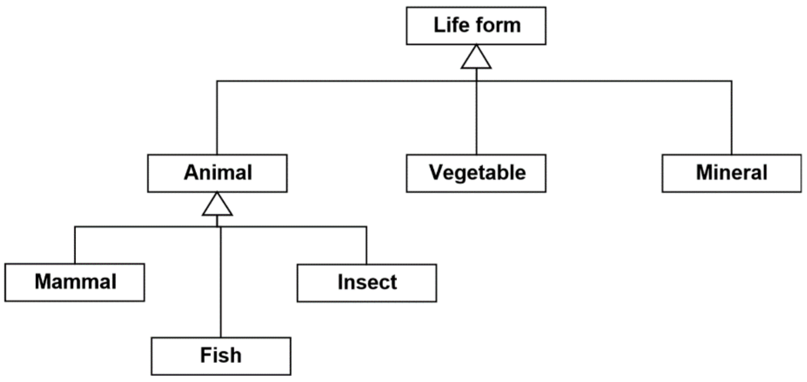


Figure 8. Life-form hierarchy

Figure 8 shows different types of life known to man. The top block is called ‘Life form’ and has three child blocks: ‘Animal’, ‘Vegetable’ and ‘Mineral’, which makes ‘Life form’ the parent block. Going down one level, it can be seen that ‘Animal’ has three child blocks: ‘Mammal’, ‘Fish’ and ‘Insect’. Notice now how ‘Animal’ is the parent block to its three child blocks, while still being a child block of ‘Life form’.

The diagram may be read in two ways:

- from the bottom up: ‘Mammal’, ‘Fish’ and ‘Insect’ are types of ‘Animal’ – ‘Animal’, ‘Vegetable’ and ‘Mineral’ are all types of ‘Life form’;
- from the top down: ‘Life form’ has three types: ‘Animal’, ‘Vegetable’ and ‘Mineral’ – the block ‘Animal’ has three types: ‘Mammal’, ‘Fish’ and ‘Insect’.

In SysML terms, a child block will inherit any properties and operations that its parent block has, but may have additional properties or operations that make the child block special. Figure 9 shows an expanded version of Figure 8 by adding some properties and operations to the blocks. It can be seen that the block ‘Animal’ has three identifiable properties: ‘age’, ‘gender’ and ‘number of legs’. These properties will apply to all types of animal and will therefore be inherited by their child blocks. That is to say that any child blocks will automatically have the same three properties.

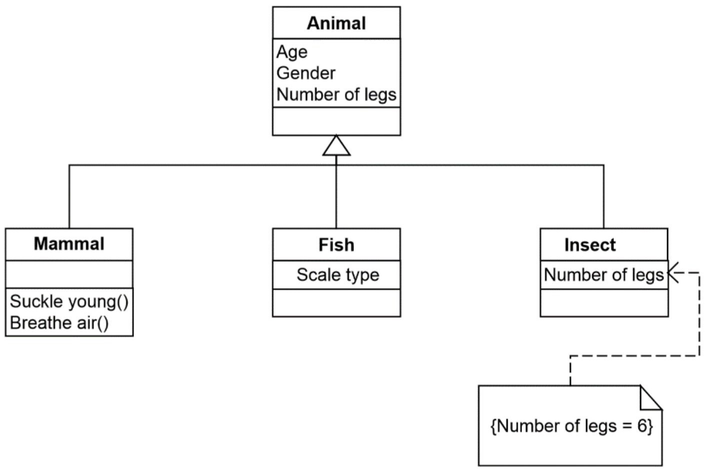


Figure 9. Example of inheritance

What makes the child block different or special and therefore an independent block in its own right is the addition of extra properties or constraints on existing properties. The block ‘Fish’ has inherited the three properties from its parent block (which are not shown), but has had an extra property added that its parent block will not possess: ‘scale type’. This makes the child block more specialized than the parent block. The block ‘Insect’ has no extra properties or operations but has a constraint on one of its property values. The property ‘number of legs’ is always equal to six, as this is in the nature of insects. Such a limitation is known in the SysML as a ‘**constraint**’. From a modelling point of view, it may be argued that the property ‘number of legs’ should not be present in the block ‘Animal’, since it is not applicable to fish. This is fine and there is nothing inherently wrong with either model except to say that it is important to pick the most suitable model for the application at hand. **Remember that there are many correct solutions to any problem, and thus people’s interpretation of information may differ.** By the same token, it would also be possible to define two child blocks of ‘animal’ called ‘male’ and ‘female’, which would do away with the need for the property ‘gender’ as shown in Figure 10.

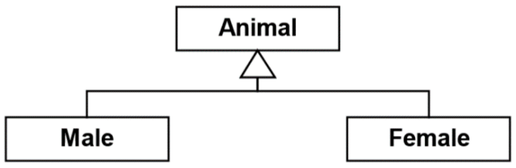


Figure 10. Another way to model gender

Which approach is the better of the two, the one shown in Figure 9 or the one shown in Figure 10? Again,it is necessary to **pick the most appropriate visual representation of the information and one that you, as the modeller, are comfortable with.**

3.2 Aggregation and composition

The second type of relationship is a special type of association that allows assemblies and structures to be modelled and is known as aggregation.

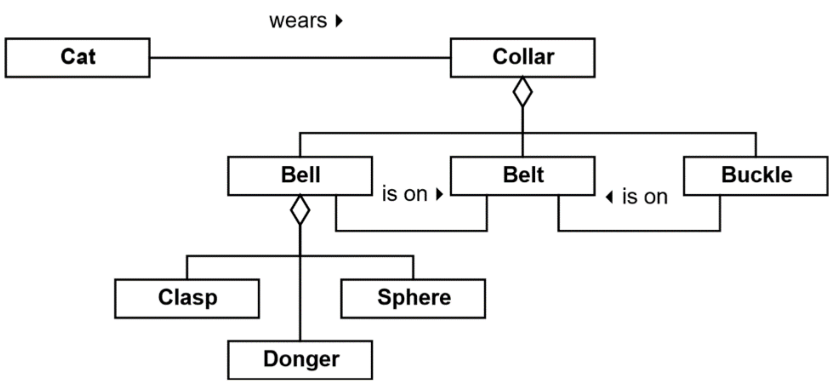


Figure 11. Example of aggregation

Figure 11 provides an example of **aggregation**. Aggregation is shown graphically in the SysML by a diamond or rhombus shape and, when reading the diagram, is read by saying ‘is made up of’. Starting from the top of the diagram, the model is read as: ‘Cat’ wears ‘Collar’. The direction is indicated with the small arrow and there is a one-on-one relationship between the two blocks. The multiplicity here is implied to be one to one as there is no indication. The ‘Collar’ is made up of (the aggregation symbol) a ‘Bell’, a ‘Belt’ and a ‘Buckle’. The ‘Bell’ is on the ‘Belt’ and the ‘Buckle’ is on the ‘Belt’. The ‘Bell’ is made up of (the aggregation symbol) a ‘Clasp’, a ‘Donger’ and a ‘Sphere’.

There is also a second special type of association that shows an aggregation style relationship, known as **composition**. The difference between composition and aggregation is subtle but very important and can convey much meaning. The simplest way to show this difference is to consider an example, as shown in Figure 12.

From the model, there are three types of ‘Weapon’: ‘Foil’, ‘Epée’ and ‘Sabre’. Each weapon is made up of a ‘Handle’, a ‘Pommel’, a ‘Blade’ and a ‘Guard’. An **aggregation** is made up of **component parts** that **may exist in their own right**. It is possible to buy or make any of the components under the block ‘Weapon’, as they are assembled into the completed block ‘Weapon’. The block ‘Blade’, **an composition**, however, has three components that **cannot exist independently of the block ‘Blade’**. This is because a fencing blade is a single piece of steel that is composed of three distinct sections. For example, there is no such thing as a ‘Foible’, since it is an inherent part of the ‘Blade’ rather than being an independent part in its own right. Of course, any of these blocks may have their own properties and/or operations, even when used as part of the composition relationship.

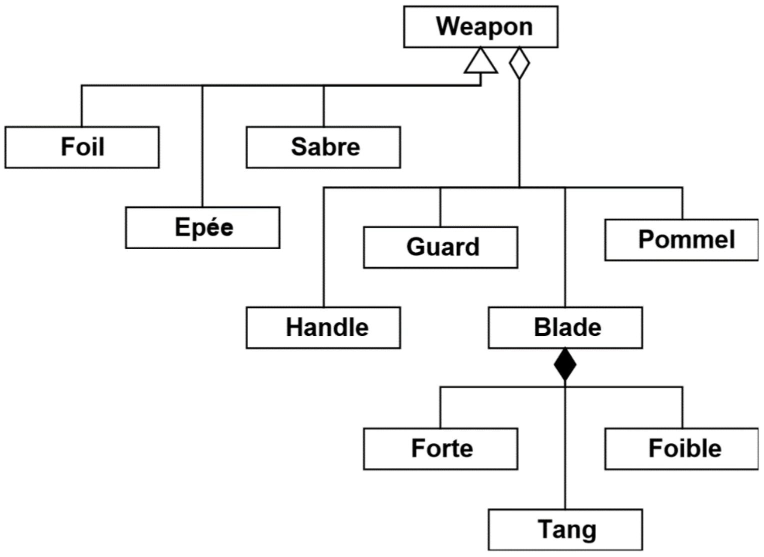


Figure 12. Example of the difference between composition and aggregation

3.3 Dependencies

A dependency is used to show that one block is dependent on another. This means that a change in one block may result in a change in its dependent block. A dependency is represented graphically by a dashed line with an arrow on the line end. For example, consider the simple diagram of Figure 13.

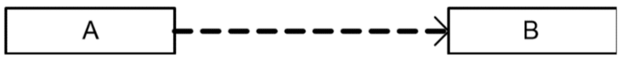


Figure 13. A simple dependency

In this example, any change to block 'B' will result in a change to block 'A'. By their very nature, dependencies are quite weak relationships and really need to be further adorned using stereotypes to add any real meaning. Many diagrams use the dependency in conjunction with stereotypes, such as the use case diagram and the requirements diagram.

4 Summary

An overview of the elements of the SysML block definition diagram is summarized as Figure 14. A system typically has a number of SysML block definitioan diagrams – not all classes (i.e. blocks) are inserted into a single class diagram. A class (i.e. block) can have multiple levels of meaning and participate in several class diagrams.

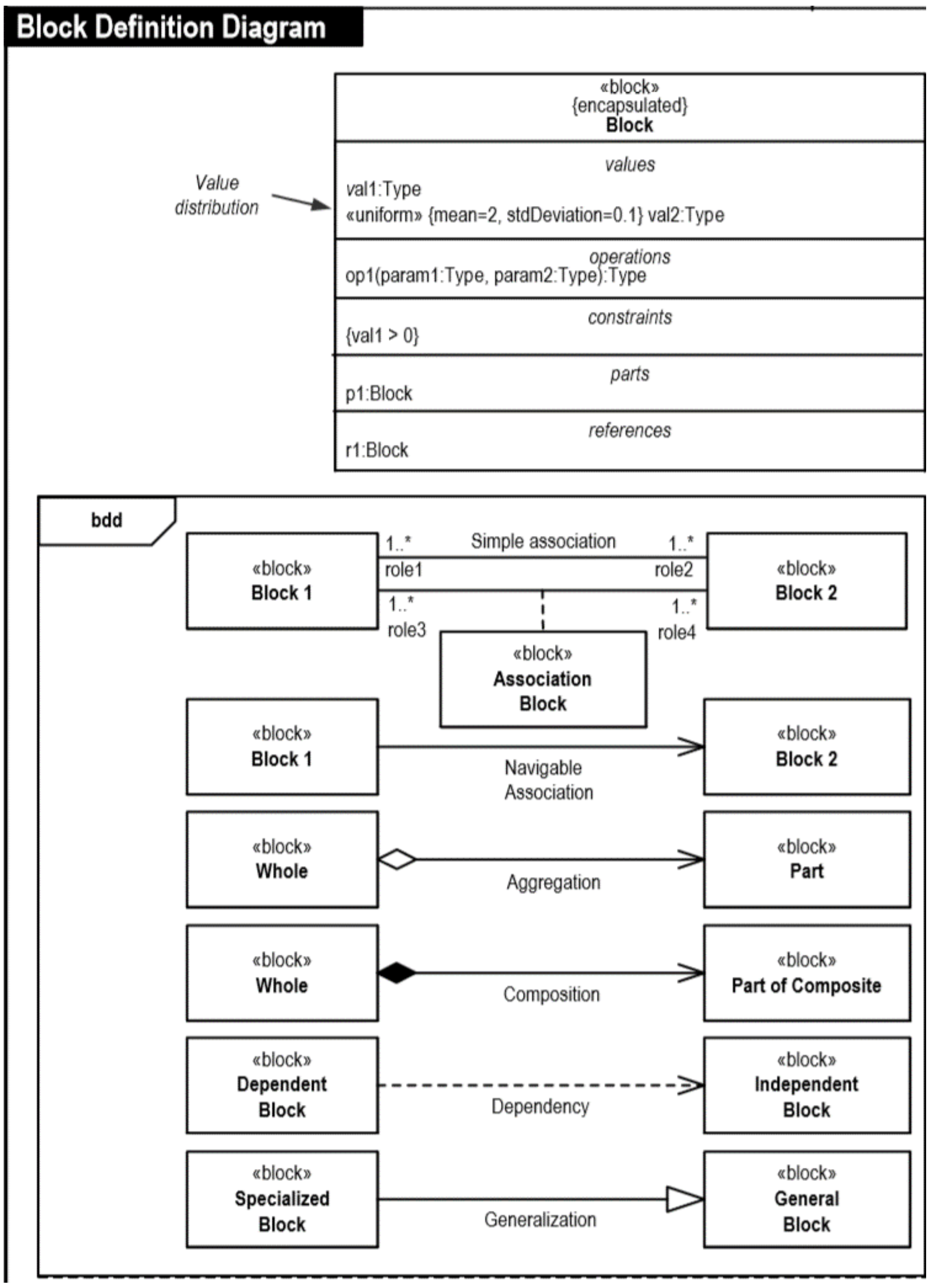


Figure 14. An overview of the elements of the SysML block definition diagram

References

[1] Jon Holt and Simon Perry. (2008) *SysML for Systems Engineering*. The Institution of Engineering and Technology, London, United Kingdom.

This project has received funding from the European Union’s Seventh Programme for research, technological development and demonstration under grant agreement No 603843