

Description of the Logical Platform Information Model v. 3.3

ITEM CONCEPTS

The first page of the Logical Platform model describes the major relationships for the main object *Item*. An *Item* is used to create a common identifier of its specialisations. *Item* is an abstract entity, so it must be instantiated as a *Part*, a *Document*, an *Operation* or a *Production_resource*, but one *Item* can not be a *Part* and a *Document* at the same time.

Item_relationship is also an abstract entity that describes the relationship between two *Items*. The allowed relationships between two *Items* are:

- *Part* to *Part* (Part_relationship)
- *Document* to *Document* (Document_relationship)
- *Operation* to *Operation* (Operation_relationship)
- *Production_resource* to *Production_resource* (Production_resource_control_relationship)
- *Production_resource* to *Part* (Production_resource_to_part_relationship)

Each *Item* has *Item_versions*. You can keep track of the order of the versions by using the relationship object *Item_version_precedence*. Every *Item* and/or *Item_version* can be identified with the objects *Item_identifier* and *Item_version_identifier*, which also identifies an *Organisation_unit* as the owner of the *Item*.

To learn more about the Volvo standardised use of *Items*, *Parts*, *Item Identifier* et c, please read the Volvo Standard 9123,13, which can be found on the Volvo Corporate Standards homepage:
<http://violin.vtd.volvo.se/depts/06850/eng/index.htm>.

PERSONS AND ORGANISATION + PERS./ORG. CONTACT INFORMATION

These two models describe the organisation of users of a system. A *Person* is identified by its *person_ID*, and can have different names such as a family name and a first name. The relationship between a *Person* and an *Organisation_unit* is either an *Assignment* or an *Employment*. Also, a *Person* can play different *Roles_in_organisation*. For instance, *Person* Mr X can be employed in *Organisation_unit* 12345, but have two roles: Engineer and Project Manager.

An *Organisation_unit* can be specialised into a *Market* or a *Factory*, which both may be defined by a *Variant*.

There is a possibility to categorise the *Organisation_units* so that one *Organisation_unit* belongs to many categories and one category can refer to many *Organisation_units*. Also, the object *Organisation_relationship* can show the relationship between two *Organisation_units*. This relationship may also be categorised. For instance, you can define one *Organisation_unit* to be the supplier of another *Organisation_unit*, which is categorised as customer.

The *Person_and_organisation_with_date* entity is used for sign-off. The *Person_organisation_select* entity can also be used to connect *Contact_information* with a *Person* with a relationship to an *Organisation_unit*. This *Contact_information* can be categorised as personal information, office information et c.

DATE AND TIME CONCEPTS

This model describes the usage of setting a *Time_period* that is then connected to the *UTC_timestamp*, which holds the year count from the beginning of the year according to ISO 8601.

PRODUCT STRUCTURE – PRODUCTS

A *Product* is a *Professional_Service* or a *Commodity*, or both at the same time. A *Commodity* is a *Product_commodity* or a *Part_commodity*, or both at the same time. A *Commodity* is a physically realisable *Product* that a Volvo company can sell. It is a *Product_commodity* if it is defined by a string of *Variants*, and a *Part_commodity* if it is defined as a *Part* (with the *Item_identifier*). Note that a *Commodity* can be both a *Product_commodity* and a *Part_commodity* at the same time.

The *Part_commodity* is both a specialisation of *Part* and *Commodity*, so it inherits its behaviour from both those entities.

The *Product_commodity* is related to a *Product_class*. The *Product_commodity* is defined by a *Product_commodity_definition*. This definition can be a *Product_string*, which is the complete set of validated *Variants* that are needed to build the product. This can be related to a *Pre_defined_commodity*, which specifies a fully defined and buildable *Product_commodity*. An example of this is Volvo Cars' use of Product Number 34. This low-specified product can then be further specified by *Options*. So a *Product_commodity* can also be specified by a *Pre_defined_commodity* in combination with *Options*.

PRODUCT CLASSES AND VARIANTS

A *Product_class* defines a set of similar products. It is the highest level in the Product Structure. The relationship between two *Product_classes* can either be with control or without control, e.g. Vehicle- and Component Product Class. With control means that the lower *Product_class* only can use the *Variants* and *Variant_families* that are used by the higher *Product_class*. Without control means that there are no such restrictions.

Product_type is a specialisation of *Product_class*, and describes a portion of the products in a *Product_class*. It is defined by a *Variant_expression*, which is a Boolean set of *Variants*. Each possible product in a *Product_class* should belong to one and only one *Product_type* of that *Product_class*.

There are three kinds of relationships between the *Product_class* and the *Variants*. The first points out the ownership connection. The *Product_class_owner* relates a *Product_class* to its owner, and a *Variant_family_owner* sets the owner of a *Variant_family*.

The second relationship defines which *Variant_families* a *Product_class* may use. The specialisation *Product_type_defining_variant_family* is used to define which *Variant_family* is used to define a specific *Product_type*.

The third relationship describes which *Variants* are used in the *Product_class*. The *Product_class_to_Variant_relationship* can be one of its two specialisations. *Identifying_variant_for_class* is used to specify the *Variant* that identifies a *Product_class* (or a *Product_type*). The *Selected_variant_for_class* is used to authorise a *Variant* to be used within a *Product_class* (or a *Product_type*).

A *Variant* can only belong to one *Variant_family*, but a *Variant_family* can include many *Variants*. The *Variant* can be identified with one *Variant_symbols* at a time, but a *Variant_symbol* related to a *Variant* can be changed with *Effectivity*.

VARIANT CONTROL

The *Variant_control* entity describes which control that is allowed of the *Variants* in a *Product_class*. The explanation of the reason to choosing a certain control of *Variants* have been chosen is held by the entity *Reason_for_constraint*, which is related to the *Organisation_unit* that sets the reason.

The *Variant_control* is either a *Variant_restriction* or a *Variant_inclusion*. The *Variant_restriction* describes a *Variant_expression* that is either forbidden or allowed. Example of forbidden restriction:

Variant 11 cannot be used together with *Variant 12*. The forbidden restrictions state all the forbidden combinations of variants.

The allowed *Variant_restriction* state all the allowed variant combinations. You need to state all the allowed combinations by using *Variants* from all appointed *Variant_families* within a certain “package”.

A *Variant_inclusion* has two parts, a condition (one or many *Variants*) and the consequential part (one or many *Variants*). E g: *Variant 11* and *Variant 21* (condition) require *Variant 31* and *Variant 42* (consequence).

PDM PROPERTY

The entity *Property* describes a measurable or non-measurable characteristic for the entity to which the *Property* is assigned. This could for example be the weight of a *Part*, the fuel volume of a *Variant* et c. The entity *Feature_dependency* describes the relationship between two *Properties*, so that you can create a structure of *Properties*. The *Property* can be categorised, and the categories can be related to a *Variant_family*.

PDM PROPERTY- MEASURES

The main entity here is the entity *Measure*. It is either an *Actual_measure* or an *Estimated_measure*, or both at the same time. The *Estimated_measure* can be the planned measure for a planned object, and the *Actual_measure* is the result of a measuring action. The values of the *Measure* can be a decided value (*its_value*) or within two values, decided by the plus and minus attributes. The *Unit_of_measure* keeps information on the selected unit of measure. It is either a *SI_unit* or a *Customary_unit*.

The entity *Unit_conversion* keeps track of the conversion factors needed to convert one unit to another. It can either be a *Customary_SI_conversion* or a *Customary_conversion*. The entity *Customary_SI_conversion* allows you to specify that the parameters of a *Customary_unit* should be converted to those of a *SI_unit*. The *Customary_conversion* allows you to convert one *Customary_unit* to another *Customary_unit*.

PART CONCEPT

A *Part* is a specialisation of an *Item* and must have a *Part_version*. The relationship between two *Parts* (*Part_relationship*) can be one of three types: *Part_made_from* (used to relate a *Part* to the *Material* from which it can be made), *Part_precedence* (used to describe that one *Part* is the preceding design solution to another *Part*) and *Part_alternate_factory_relationship* (describes that one *Part* is the common replacement for another *Part* when one particular factory is involved).

A *Part* can be specialised as a *Part_commodity* or as either *Material*, *Single_part* or *Assembly*. The description of a *Part* is made by the relationship to the entity *Part_title_structure*, which can be related to a dictionary such as the Techla system.

PART LOCATION

Describes the possibility to position any *Part* inside the product by using either absolute or relative co-ordinate systems. Those co-ordinate systems can in turn have variable positions depending on *Variants*. This model needs to be further developed.

SOLUTION

A *Solution* is used to describe a product or one of its constituents in any stage of its lifecycle or any stage within its manufacturing process. It can be generic (“cooling system”) or defined by a *Part* or a *Product_commodity*. A *Solution* can be specialised as a *Design_solution* or a *Production_solution*, or be both at the same time. The *Design_solution* is used to represent a distinct level of the designers’ view of the breakdown of a product, and the *Production_solution* is used to represent a distinct level of the production view of the breakdown of a product.

The *Quantified_solution_usage* represents the quantified usage of a *Solution*. It is either a *Single_solution_usage* (one piece), a *Specified_quantity_solution_usage* (a specified amount is used, e.g. 4 wheels) or a *Selected_quantity_solution_usage* (the amount to be used cannot be defined until it’s produced, but the maximum and minimum quantities can be defined).

Also, the entity *Single_to_specified_quantity_link* can be used to describe a relationship between a number of *Single_solution_usages* of a *Solution* and a *Specified_quantity_solution_usage* of the same *Solution*. The quantity attribute of the *Specified_quantity_solution_usage* has to be equal with the number of *Single_solution_usages*.

Example: A *Single_to_specified_quantity_link* can be defined between 4 *Single_solution_usages* of the *Solution* “Wheel” and a *Specified_quantity_solution_usage* of the *Solution* “Wheel” with quantity information “4”. This relationship defines a relation between the single instance BOM and the quantity BOM.

OPERATION STRUCTURE

The entity *Operation* is used to describe the generic operation in a manufacturing process. The description from a Volvo dictionary can be used via the entity *Part_title_structure*. The entity *Operation_relationship* gives the possibility to create structures of *Operations*, and each *Operation* must have one or many *Operation_versions*.

The generic *Operation* can have a connection to an *Operation_instance*, which is a description of the usage of the *Operation* in its context. It has a reference to *Process_area* via the *Operation_instance_to_process_area_relationship* entity. Example: The usage of the *Operation* “Laser welding ABC” can be the *Operation_instance* “Laser welding ABC in Process Area X for Product Y with the Effectivity Z”.

The entity *Performed_operation* describes an executed realisation of an *Operation_instance*. It allows you to keep track of specific information of an operation that was actually performed on a specific *Physical_instance*, with a specified *Production_resource_instance*.

MANUFACTURING RESOURCE STRUCTURE

The *Production_resource*, which is a specialisation of *Item*, is a description of a resource (= tool) that is planned to be used in a generic *Operation*. Each *Production_resource* can have one or many *Production_resource_versions*. You can create a relationship between two *Production_resources* with the entity *Production_resource_control_relationship*.

You can also use the entity *Production_resource_to_part_relationship* to specify the relationship between a generic *Production_resource* and a *Part*. *Production_resource_to_part_relationship* is either a *Production_resource_defined_by_part*, which relates a *Production_resource* to the *Part* that defines it, or *Part_produced_with_production_resource* that defines which *Production_resource* was used to create the *Part*.

A *Production_resource* can also be related to one or many *Resource_functions*, which describe the function of a *Production_resource*. E.g: the *Production_resource* “Welding Robot” performs the *Resource_function* “Spot Welding”.

When it has been decided which tool to be used for the *Operation_instance*, this is described in the *Production_resource_instance* (for example Robot nr 25). A *Production_resource_instance* can be located in a *Process_area* via the entity *Resource_location*. The *Process_areas* can be interrelated in three different ways with the *Process_area_relationship* entity:

- *Process_area_hierarchy* (the spatial relationship between a *Process_area* and one of its constituent *Process_areas*) or
- *Process_area_interface* (describes the interface between two *Process_areas*, e.g. adjacent, below) or
- *Process_area_specialisation* (the relationship between for example “a generic welding factory” and “The Torslanda welding factory”).

PRODUCT STRUCTURE – GENERIC

The abstract entity *Product_structure_relationship* describes the product structure relationships within a Product Class that is Variant controlled. The *Product_structure_relationship* structure can be seen as a network, which consists of breakdowns that point at the parent entity *Complex_product* (the whole) and the child entity *Product_constituent* (a part of the whole).

To read this model page: start with *Product_class* and follow the breakdowns clockwise.

The allowed breakdowns of the structure (that is the possible specialisations of the *Product_structure_relationship*) are described in the following pages that are called “PRODUCT STRUCTURE – xxx”. The allowed kinds of breakdowns are:

- Decomposition (= consists of)
- Specialisation (=is a)
- Realisation (=is realised with)
- Functionality (=fulfils the function)

One page is reserved for the Operation relationships, and one for the Resource relationships.

The entity *Alternative_solution* is directly imported from AP214. Whether the entity *Solution* also covers this entity or not is the subject for further discussions. Until that has been decided, there is a page concerning the relationship between *Alternative_solution*, *Solution* and *Product_function*.

The entity *Product_structure_viewpoint* is used to be able to filter out data from the *Product_structure_relationship* depending on what you are interested in.

PRODUCT STRUCTURE – DECOMPOSITION

The entity *Solution_decomposition* is a specialisation of the *Product_structure_relationship*, and it describes the decomposition relationship of a *Single_solution_usage* to one of its constituent *Quantified_solution_usages*: the *Single_solution_usage* is a kind of *Quantified_solution_usage*, which in turn is a kind of *Complex_product* (see models SOLUTION and PRODUCT STRUCTURE-GENERIC). The *Single_solution_usage* is therefore connected to the *Quantified_solution_usage* that is a kind of *Product_constituent* (see PRODUCT STRUCTURE-GENERIC) with the entity *Solution_decomposition* that is a kind of *Product_structure_relationship*. Example: a Car (a *Single_solution_usage*) has a relationship to the *Quantified_solution_usage* 4 Wheels (a *Specified_quantity_solution_usage*).

The entity *Assembly_decomposition* is a specialisation of the *Solution_decomposition*. It gives the possibility to decompose a *Solution*. The *Assembly_decomposition* is used to describe the relationship between a *Single_solution_usage* (referring to an assembly) and one of its constituent, a *Quantified_solution_usage* (that refers either to a single part or an assembly). To be read clockwise. Example: the *Single_solution_usage* “wheel” consists of another *Quantified_solution_usage*, in this case the special case *Single_solution_usage* “tire”. Another *Assembly_decomposition* describes the relationship between the *Single_solution_usage* “wheel” and the special case *Single_solution_usage* “rim” et c.

The other specialisations of the *Product_structure_relationship* (e.g. *Product_function_decomposition*, *Operation_decomposition*) describe the possibilities to build a consists-of structure of the specialisations of the *Product_structure_relationship*. You can for example relate a *Product_function* to each one of the *Product_functions* that it consists of.

PRODUCT STRUCTURE- SPECIALISATION

This model describes the allowed specialisation relationships between the specialisations of the *Complex_product* and the *Product_constituent*, as they are described in PRODUCT STRUCTURE – GENERIC. The entities that can be broken-down into specialisation structures are *Product_function*, *Solution*, *Operation* and *Resource_function*.

The entity *Solution_specialisation* holds information about the relationship between a *Solution* and its specialisations, which also are *Solutions*. Example: The *Solution* “Brake” has one specialisation relationship to the *Solution* “Disc brake” and one specialisation relationship to the *Solution* “Drum brake”.

PRODUCT STR.- OPERATION REL.

This model page describes the possible relationships between operations. This gives the possibility to identify the item before and after a certain operation (input/output), or to simply define that “what comes out from that operation should go into this”, when you don’t want to identify the item in-between two operations. It also gives you the possibility to relate operations to another in a sequence.

The *Input_and_output_relationship* is used in the manufacturing BOM to present that the output of a given operation (parent) is used as the input to another operation (child). The *Input_and_output_relationship* is used when there is no reason to identify the result in-between two consecutive operations.

The entity *Operation_input_relationship* describes the relationship between a generic *Operation* and one of its generic input materials (which is a *Production_solution*).

Operation_instance_input_relationship describes the relationship between an *Operation_instance* and one of its input materials. The input material is here called “*Quantified_production_solution_usage*”, which is an entity that doesn’t really exist in the Logical Platform version 3.3. It is only written so here to make clear that the input material to an *Operation_instance* must be a *Quantified_solution_usage* that refers to a *Production_solution*.

Operation_instance_output_relationship describes the relationship between an *Operation_instance* and one of its output materials. As described above, the output material must be a *Quantified_solution_usage* which refers to a *Production_solution* (the entity “*Quantified_production_solution_usage*” doesn’t really exist in the model).

Operation_sequence_restriction describes a partial sequence restriction between two *Operations*. This can be used in the early planning of an operation line, when you just know that you will need to perform certain generic operations in a specific order. The entity *Operation_instance_sequence_restriction* describes this partial sequence restriction between two *Operation_instances*.

PRODUCT STR. – RESOURCE REL.

This page describes the allowed relationships between the operations and the resources that are used to perform the operations.

The *Utilised_resource_function* relationship describes the relation between an *Operation* and one of its *Resource_functions*. The *Utilised_resource_for_operation* entity describes the relationship between a generic *Operation* and one of its *Production_resources*. The *Utilised_resource_for_operation_instance* relationship describes the relation between an *Operation_instance* and one of its *Production_resource_instances*.

PRODUCT STR. – REALISATION

This model page generally describes the realisation of an entity by relating it to another. For example, a *Product_class* can be realised with the usage of a structure of *Solutions*.

Generic_realisation_of_product_class describes the relationship between a *Product_class* and one of the *Solutions* which realises it. *Product_class_operational_realisation* describes the relationship between a *Product_class* and one of the final *Operations* on the production line that can realise it. *Product_function_realisation* describes the relationship between a *Product_function* and one of the *Design_solutions* that can realise it. *Specific_realisation_of_product_class* describes the relationship between a *Product_class* and one of the *Quantified_solution_usages* that can realise it.

PRODUCT STR. – FUNCTIONALITY

This model page describes the relationship between a *Product_class* or a *Design_solution* and its functionality, which is described in *Product_function*.

PRODUCT STR. – ALT. SOL. REL.

The concept of Alternative Solution is used in AP214. It is still a part of the Logical Platform (cf PRODUCT STRUCTURE – GENERIC), but it remains an issue if the *Alternative_solution* can be replaced or not by the usage of *Solutions* and the *Product_structure_relationships*. Until this issue is solved, there exists a possibility to let an *Alternative_solution* have a relationship to the *Solutions* which realises it, and the *Product_functions* whose functionality the *Alternative_solution* solves.

As all other specialisations of the *Product_structure_relationship*, there is one *Alternative_solution_realisation* for every relationship between an *Alternative_solution* and each one of its realising *Solutions*. This also applies for *Alternative_solution_has_functionality*.

PRODUCT STR. – CATEGORISATION

The entity *Viewpoint_category* is used to handle viewpoints of the *Product_structure_relationships* (cf PRODUCT STRUCTURE - GENERIC). A *Viewpoint_category* can be used to filter out a certain viewpoint of product data. A *Viewpoint_category* could for example be the manufacturing viewpoint, or the electrical system viewpoint.

The *Viewpoint_category_hierarchy* describes the relationship between *Viewpoint_categories*. They can either be in a hierarchy or be equivalent. The *Viewpoint_categorisation* describes the relationship between a *Viewpoint_category* and an element that can be categorised.

DOCUMENTS – CATEGORISATION

This model page describes the relationship between documents and the possibility to categorise documents. A *Document* is a specialisation of *Item* and it is anything that carries information/describes objects in the PDM-system. E.g: drawing, CAD-models, paper documents, CD's, videos, photography. A document could either be electronically accessible through a computer system or available as a

physical paper/sample. Note: the software programs loaded in the product are *Parts* and not *Documents*.

Documents can be categorised by using the entity *Document_category*, which is an identified category for a document (e.g. text document, program code, and geometry) according to the Volvo Standard 9123. *Categorisation_of_document_category* is a relationship between *Document_categories* that establishes whether the categories are in a hierarchy or that they are equivalent.

Two documents can be related to each other with the relationship entity *Document_relationship*, which is a specialisation of *Item_relationship*. The *Document_relationship* can be specialised as:

- *Document_references* (identifies the related document as being referenced by the relating document)
- *Document_precedence* (identifies the related document as succeeding the relating document independent of their respective issue) or
- *Document_hierarchy* (identifies the related document as being a subordinate to the relating document).

DOCUMENTS – STRUCTURE

This model page describes the structure of documents and their storage.

The *Document_issue* is a specialisation of *Item_version* and is a version of a document. It has a release status and an associated content (*Document_body*). The *Document_body* describes the partial content of a document and is an abstract entity that can be specialised as either *Computer_accessible_document* or *Non_computer_accessible_document*.

The *Document_body_relationship* can be used to describe the relationship between different elements of the same document (e.g. a CAD file embedded in a product structure tree breakdown document). Further, the *Document_body* can be located in an *Archive*, which in turn is located in a *Domain*. *Document_body* and *Document_body_relationship* can also be used to manage multi-sheet drawings.

The *Computer_accessible_document* can be created and edited in a software application, which can be identified with the entity *Editing_tool*. The different digital document format standards that the *Editing_tool* supports are represented with the entity *Digital_format* (e.g. MS Word 6.0 format).

DOCUMENTS - ASSIGNMENT

The entity *Document_assignment* identifies the relationship of a document to the PDM related information that it documents. *Document_assignment* can relate to both a *Document* or a *Document_issue* (the version of the *Document*, related to the *Document_body*) to the object that it documents.

EFFECTIVITY CONCEPTS

The entity *Effectivity* describes the validity of an object or a relationship between objects. *Effectivity* can be defined as date and time, an *Event_reference* or by another *Effectivity*. The start can be a *Date_and_time* or triggered by an event. The stop is either described by the same entities as for start, a period of time or described as “until further notice”.

The *Effectivity_relationship* between two *Effectivities* establishes the type of relationship.

The *Effectivity_assignment* relates the *Effectivity* to the element it concerns.

Note: *Effectivity_change* has to be included as an *Effectivity_element* in the future.

APPROVAL CONCEPTS

An *Approval* represents a statement made by technical personnel or management personnel on whether certain requirements are met. *Approval* also includes the judgement of the quality for the statement.

The *Approvals* can be related to each other and categorised with the entity *Approval_relationship*. An *Approval* can also have a Status, and a *Date_and_time*. The select-entity *Approval_elements* relates the *Approval* to the element that is to be approved.

The description of the *Approval* is made with the relationship to the right *Document*, which also is related to the real signature of the *Approval*, which is the entity *Personal_approval*. This entity is related to a *Person_and_organisation_with_date*.

CHANGE MANAGEMENT: WORK REQUEST

The entity *Work_request* holds the information of the formal development requirement, which often results in activities to develop one or many parts or other technical solutions. The *Work_request* comes from a requestor (*Person_and_organisation_with_date*) and it is also related to one or many notified persons (*Person_and_organisation_with_date*). The objects that are affected by the *Work_request* are related with the entity *Affected_elements*.

The entity *Activity_resolves_work_request* relates the *Work_request* to one or many *Activities*, which are to be solved. These activities can be identified with the help of an *Activity_chain_template*. The *Activity_chain_templates* can be related with each other with the *Activity_template_relationship* entity. This relationship can also be specialised as a precedence (one *Activity_chain_template* should be performed before another *Activity_chain_template*), hierarchy (one *Activity_chain_template* can be specialised into another *Activity_chain_template*) or co-ordination (two *Activity_chain_templates* should be performed at the same time).

CHANGE MANAGEMENT – ACTIVITY

This page describes the *Activity* more detailed. An *Activity* can have an id and an *activity_type*. An *Activity* can be requested by a *Person_and_organisation_with_date*, and it is related to the supplying and the concerned *Organisation_units*. The responsibility of the *Activity* is made by a *Role_in_organisation*. The planned start date for the *Activity* is decided by the *Introduction*, which is an effectivity date. This is also applicable for planned_end_date (*Termination*). The *Activity* will later get an actual_start_date (*Date_and_time*) and may also have an actual_end_date (*Date_and_time*). The chosen working method can be related to the *Activity* by using the relation *chosen_method* to the entity *Activity_chain_template*.

Project is a specialisation of *Activity*, since a *Project* can have gates but not an *Activity*. If you build a hierarchy of *Activities*, the top-level *Activity* could be a *Project* with *Gates*. The entity *Gate* is a specialisation of an *Event_reference*, and can be related to a *Gate_template* that is a generic definition of a specific *Gate*.

The relationships *Gate_status_depends_on_activity* and *Activity_depends_on_Gate_status* describe the possible relationships between a *Gate* and an *Activity*. A *Gate* can have dependencies on many *Activities* before it can be opened (the entity *Gate_status_depends_on_activity* identifies each one of these relationships). Also, an *Activity* can have dependencies on many *Gates* whether it can start or not (the entity *Activity_depends_on_Gate_status* identifies each one of these relationships).

CHANGE MANAGEMENT – EC

The *Engineering_change* is a result of an *Activity*, which describes the change of one object. An *Activity* can be related to another *Activity* with the use of the entity *Activity_relationship*. This relationship between two entities can be specialised as an *Activity_precedence* (one *Activity* comes before another), *Activity_hierarchy* (one *Activity* is the specialisation of another) or *Activity_coordination* (two *Activities* have to be co-ordinated).

The *Engineering_change* object may have an id, a description and a status. The element affected by the change is related via the select-entity *Affected_elements*. The attribute *added_or_deleted* can be used to define whether an affected element is added or deleted. The start time of the *Engineering_change* to be valid is made with the entity *Introduction*, which sets the correct effectivity control. The entity *Change_category* can be used to categorise the type of change that is to be performed.

The *Engineering_change* can be related to other *Engineering_changes* with the relationship entity *Engineering_change_relationship*. It can be specialised as either an *Engineering_change_sequence*, an *Engineering_change_hierarchy* or an *Engineering_change_coordination*.

The entity *ERN_general_info* is used to describe common attributes for a group of *Engineering_changes*. This is often called Front page at Volvo today. Information about decided change sequences and change co-ordinations (information which derives from the *Engineering_change_relationship*-entity) is also described here.

AUTHORISATION AND SECURITY

This model page describes the authorisation of users in a PDM-system. The *Actor* is used to represent an end user of the system or an application, that may access or change the content of the information in the system. The *Actor* can be either a *Role_in_organisation*, a *Person* or an *Application*.

The access rights of a user on a specific object are set with the entity *Access_rights*. This entity defines the relation between the user and the specific object, with the level of access rights defined in the attribute named “attributes”. A user can have *Access_rights* to one or many objects. The additional relationships in the model make it possible for one *Actor* to give another *Actor* *Access_rights*.

PHYSICAL INSTANCES

The entity *Physical_instance* describes the physical realisation of a *Production_resource_version*, a *Part_version*, a *Production_instance* or maybe even a *Person*. The *Physical_assembly_relationship* describes the relationship between a *Physical_instance* and one of its constituent *Physical_instances*. The *Production_instance* represents the physical instance of a *Production_solution* when it has been produced.