Verband der
Automobilindustrie

ECM Recommendation Part 0 (ECM)

Version 2.0, issued Aug. 2009

Replacements: Version 1.0

VDA 4965 – Part 0

Version 3.0, issued Jan. 2010

Replacements: Version 2.0

SASIG

strategic automotive product data standards industry group

A Joint Publication

SASIG

Automotive Industry Action Group (U.S.), Groupement pour l'Amelioration des Liaisons dans l'Industrie Automobile (France), Japan Automobile Manufacturers Association (Japan) / Japan Auto Parts Industries Association (Japan), Odette Sweden AB (Sweden), and Verband der Automobilindustrie (Germany) are members of the Strategic Automotive product data Standards Industry Group- SASIG. These organisations are cooperating in the creation, distribution and use of joint documents, including PDQ (Product Data Quality), PDM Assembly Data Exchange, XMTD (Exchange and Management of Technical Data), DEV (Digital Engineering Visualization).

SASIG Publications

A SASIG publication reflects a consensus of those substantially concerned with its scope and provisions. A SASIG publication is intended as a guide to aid the manufacturer, the consumer and the general public. The existence of a SASIG publication does not in any respect preclude anyone from manufacturing, marketing, purchasing, or using products, processes, or procedures not conforming to the publication.

Cautionary Notice

SASIG publications are subject to periodic review and users are cautioned to obtain the latest editions.

Maintenance Procedure

Recognizing that this joint publication may not cover all circumstances, SASIG has established a maintenance procedure. Please refer to the Maintenance Request Form at the back of this document to submit a request.

Contact information

Contacts information to SASIG is available on http://www.sasig.com.

Copyright and Trademark Notice

Table of Contents

Figures

Tables

1 Abstract

1.1 Preface

Today, considerable effort is expended in the automobile development value chain to constantly reduce throughput times in the development processes and to enhance the quality of digital product and process descriptions. Simultaneous engineering in product development and manufacturing planning and intensive cooperation between the various divisions within a company and with external partners are key factors for success in this context. One of the key challenges to be surmounted when shortening development times and ramp-up times in series manufacturing is to successfully and rapidly implement the changes which occur during these phases in a joint effort with partners from the fields of development, planning and manufacturing. Some factors which regularly motivate the wish to initiate changes include:

- Legislative changes
- Changes to market conditions or the competitive situation
- Internal inadequacies in development, planning or manufacturing
- Quality or safety problems
- Exploitation of additional optimization potential

The desire to make changes also necessitates corresponding changes to the cross-company engineering change management processes between the partners, suppliers and service providers to achieve greater dynamism and flexibility. Thus, mastery of engineering change management processes is increasingly becoming a crucial factor in the success of cooperative ventures in the automotive industry. Suppliers are today integrated into the development and planning process at an early stage. Differences which result from the nature of the partners and the projects in which they are involved mean that it is a challenging task to integrate partners into engineering change management. Small changes can, however, often have a significant impact on costs, processes, and development and manufacturing times. This means that standards and recommendations for structuring and implementing cross-company engineering change management processes and engineering change management data right down to the level of defining company-specific interfaces are indispensable if integration is to be efficient. According to estimates of the ECM project group, cost savings of up to around 40% can be expected as a result of reducing the times involved in retrieval, coordination and processing during the course of cross-partner engineering change management processes.

Harmonization of private (i.e. partner-specific) ECM processes is one key contribution towards exploiting the potential for improvement during development and when starting up series manufacturing. This current recommendation is based on the requirements and experiences of the partners in the joint SASIG ECM workgroup. This recommendation has been validated by means of pilot projects in the companies involved.

1.2 Objectives of the recommendation

The objective of the recommendation is to provide support for ECM processes with respect to the communication of change data between customers and suppliers. Use of the recommendation allows more efficient integration of all the partners involved in a proposed change as a result of:

- a reduction in the variety of communications methods used in the partner network
- a reduction in the throughput time for a proposed change
- improvements in the quality and reliability of the process
- improvement with respect to the transparency of changes for all those involved
- the reusability of process agreements, interfaces and tools

1.3 Structure of the recommendation

The recommendation is subdivided into individual parts reflecting the ECM Reference Sub-Process (3.3) involved during the course of a change; this subdivision is subject to change as a result of further developments.

Overview and general recommendations for ECM

This document gives an overview of and offers general recommendations for the overall ECM Reference Process, its modelling and a partner model. It describes fundamentals and basic recommendations valid for each ECM Reference Sub-Process. This document goes on to define

- ECM Reference Sub-Processes
- Interaction Scenarios
- Messages for the communication
- ECM Data Model objects and

which are shared by and relate to the specific parts of the recommendation as follows:

Part 1: ECR – Specification and Decision on Change

Part 1 offers recommendations with respect to communicating Engineering Change Request (ECR) data to support the specification of proposed changes and the associated decisions. It describes the ECR Reference Process which permits synchronization of the different internal procedures of the partners involved using ECR Messages. It offers recommendations for permitted sequences of ECR Messages to provide support for business-typical ECR Interaction Scenarios (IS) within the scope of bilateral communication between the cooperation partners and recommendations with respect to the minimum requirements applicable to communication and message content. The change data contained in the individual communication steps is described in the ECR Data Model.

Part 2: ECO – Engineering Implementation of Change

Part 2 offers recommendations for communicating Engineering Change Order (ECO) data compliant with Part 1 in order to allow implementation in engineering, such as the ECO Reference Process and the necessary ECO Messages and the ECO Data Model.

Further part is expected to cover the Sub-Process Alternative Solutions (AS). To provide solutions for the identified and described potential for improvement, one or more Alternative Solutions are generally developed which describe different approaches to implementing an idea for a change.

Part 2 is being planned and will be published when it is completed. Changes to structure and contents may however result from new insights and other information obtained during the course of the work.

1.4 Notational conventions

The following notational conventions are used in the context of this recommendation and are intended to highlight the specific names used in this document.

The abbreviations and terms defined in sections 1.5 and are valid within the scope of the recommendation.

Text in *italics as* used in the documents is intended to identify the following objects, as well as the objects which are defined within each of these domains:

- ECM Reference Process and ECM Reference Sub-Processes (e. g., phases, activities, process roles)
- ECM Interaction Scenarios (*request*, *respond* and *notify* messages)
- ECM Messages

The objects defined in the ECM Data Model (entities) are written in SMALL CAPITALS to distinguish them from the terms used for process-related objects.

1.5 Abbreviations

The following abbreviations apply within this document:

AS	**A**lternative **S**olution
API	**A**pplication **P**rogramming **I**nterface
ECM	**E**ngineering **C**hange **M**anagement
ECO	**E**ngineering **C**hange **O**rder
ECR	**E**ngineering **C**hange **R**equest
IS	**I**nteraction **S**cenario
ISO	**I**nternational **O**rganization for **S**tandardization
IT	**I**nformation **T**echnology

MCO	**M**anufacturing **C**hange **O**rder
OEM	**O**riginal **E**quipment **M**anufacturer
OMG	**O**bject **M**anagement **G**roup
PDM	**P**roduct **D**ata **M**anagement
PLM	**P**roduct **L**ifecycle **M**anagement
STEP	**ST**andard for the **E**xchange of **P**roduct Model Data
UML	**U**nified **M**odeling **L**anguage
XML	e**X**tensible **M**arkup **L**anguage

1.6 Definitions

The following definitions apply within this document:

activity:

Processing step during the course of a process which is used in activity diagrams for detailed description of a reference process (see phase), i.e. with conditional branches etc.

coordinator:

The organization, responsible for coordinating the ECM activities to solve an accepted issue against a product.

Note: *The coordinator will be in most cases the owner of the product.*

cross-phase:

Refers to relationships between two or more (process) phases or elements of these phases

dialog:

Synonym for an interaction instance

ECM objects:

Key data model objects which provide the basis for communication between partners within the ECM Reference Sub-Processes such as ECR, ECO or MCO

ECM Interaction Scenario (ECM IS):

Interaction scenarios defined within the scope of a specific ECM Reference Sub-Process. There are ECR Interaction Scenarios, ECO Interaction Scenarios, etc.

ECM Message:

Messages defined within the scope of the ECM Reference Process or within the scope of a specific ECM Reference Sub-Process. There are common ECM Messages, ECR Messages, ECO Messages, etc.

ECM Reference Process:

The overall reference process as it relates to ECM. The ECM Process comprises several ECM Reference Sub-Processes

ECM Reference Sub-Process

ECR Reference Process, ECO Reference Process, etc. are ECM Reference Sub-Processes. Note that, for example, the ECR Reference Process and ECO Reference Process may overlap

ECM Reference Sub-Process-independent:

Indicates elements which can be used in each ECM Reference Sub-Process in the same way

ECO Reference Process (synonym: Engineering Implementation of Change Process):

The specific ECM Reference Sub-Process which relates to an Engineering Change Order

ECR Reference Process (synonym: Specification and Decision on Change Process):

The specific ECM Reference Sub-Process which relates to an Engineering Change Request

Engineering change management (ECM):

The coordinated management and uniform tracking of changes, i.e. collecting ideas or need for product and product related changes, elaborating one or more possible solutions, evaluating them with respect to technical and cost aspects and implementing them with respect to engineering and manufacturing

Engineering Change Management (ECM):

Engineering change management according to this recommendation

Engineering Change Order (ECO):

A request to implement an engineering change, consisting of its planning, design, acceptance, testing, etc, documentation, review and release. An ECO is processed in a number of processing steps defined by the ECO Reference Process.

Engineering Change Request (ECR):

A request to evaluate a change to one or more products, involving changing parts, assemblies, documentation, or other items belonging to these products or by changing activities necessary to engineer or build these products.

An ECR is processed in a number of processing steps defined by the ECR Reference Process, thus enriching the ECR with the information required to enable a final decision to be made about its implementation by one or more ECO later on. (short: Change Request)

instantiation:

A Data Model groups objects from the real world into information classes (or simply "classes"). A class describes the general properties in the form of attributes and at the same time defines a data type. An instance of a class is a uniquely identified unit of data and represents the object from the real world. Instantiation refers to the generation of instances of a class. For example, an object in the real world is a person, and their name is a property. The class or data type 'person' represents all people. An instance of the data type 'person' assigns a value to each of the attributes defined in the class (name, age etc.) and can be identified using a unique ID.

interaction:

Results in an exchange of messages between partners[1]. An interaction can be a unidirectional interaction where a single message of the type *notify* is passed or a bidirectional interaction where one message each of the types *request* and *respond* are exchanged. Each unidirectional interaction comprises a send action on the part of the sender and a receive action on the part of the recipient. A bidirectional interaction comprises a second send- and receive-action with sender and receiver swapped.

interaction instance:

Instantiation of an interaction scenario between a specific coordinator organization and a specific participant organization (i.e. an "actual" interaction).

interaction scenario (IS):

Simplified description of the sequences used and the conditions under which data is exchanged between partners for a particular purpose using interactions2 in order to implement process integration for a specific type of cooperation. This recommendation defines interaction scenarios between the two interaction roles coordinator and participant by 1) defining the interactions which can occur, must occur or must not occur; 2) defining the permitted sequences in which interactions or messages may occur using syntax diagrams; 3) assigning send and receive actions of coordinator interactions to specific

[1] Based on N. Kavantzas, D. Burdett, G. Ritzinger, Y. Lafon (Eds.). “Web Services Choreography Description Language Version 1.0”. W3C, Working Draft 12 October 2004.

[2] Ibid.

points (synchronization points) in a reference process, thus defining additional permitted interactions and conditions and sequences for interactions (see protocol specification).

mandatory:

Indicates all activities or data objects which are obligatory, i.e., which must be carried out under all circumstances, or data objects which are absolutely necessary

Manufacturing Change Order (MCO):

A request to implement a change by manufacturing up to its release for production. An MCO is processed in a number of processing steps.

MCO Reference Process (synonym: Manufacturing Implementation of Change Process):

Denotes the specific ECM Reference Sub-Process relating to an MCO

message:

Sequence of symbols which contain information[3]. This recommendation distinguishes between three types of messages: 1) *request* 2) *respond* 3) *notify*.

milestone:

Describes a state in a process at which a particular process objective is achieved. Milestones are used in phase diagrams in particular.

optional:

Indicates all activities or data objects which may be used as the need arises or which may be required in certain cases

participant:

A partner who is involved in an ECM Process but who is not responsible for coordinating it

partner:

All people and organizations who participate in an ECM Reference Process, i.e. both the *coordinator* and the *participants*

phase:

Subdivision in a process used in phase diagrams for a rough description of an ECM Reference Sub-Process. Phase diagrams are used as an easily read representation of an ECM Reference Sub-Process and its component parts. For this reason, a number of activities are typically grouped together to form a phase in this form of representation

[3] H. Engesser, V. Claus. "Duden Informatik". 2nd Edition; Dudenverlag, 1993.

private process:

Describes a partner-specific process that is used internally, in contrast to a reference process, which is used as a common representation of a process.

protocol specification:

A protocol specification is used to precisely define the permitted message sequence per interaction scenario from a coordinator's and from a participant's point of view

reference process:

A workflow[4] for harmonized communication for data and process integration described using a schema. A reference process specifies the primary activities, possible interactions with partners and the sequence and conditions for activities and interactions. For the purposes of communication as per this recommendation, each partner maps his own internal process (i.e. terminology, process, data) onto a reference process.

role:

Describes a set of responsibilities in a process which can be embodied by a person or an organizational unit. Two types of roles are used in this recommendation: 1) process roles, in which the persons/organizations fulfilling the role carry out activities in an ECM Reference Sub-Process, such as the *Requestor* etc. 2) interaction roles where organizational units fulfil the role and interact as *coordinator* or *participant* in an ECM Interaction Scenario by carrying out a number of process roles. Further modelling elements of the organizational structure, such as group, (organizational) entity and organizational unit are described in detail in the respective parts.

synchronization point:

Defined point in an ECM Reference Sub-Process at which a specific interaction with a partner is possible to synchronize the private engineering change management processes of both the *coordinator* and *participant*

1.7 References to standards

CMMI:

Capability Maturity Model® Integration (CMMISM); Version 1.1
CMMISM for Systems Engineering, Software Engineering, Integrated, Product and Process Development, and Supplier Sourcing (CMMI-SE/SW/IPPD/SS, V1.1); March 2002

ISO 10303-11:

Industrial automation systems and integration – Product data representation and exchange – Part 11: Description methods: The EXPRESS language reference manual, November 1994

[4] Based on: S. Jablonski, M. Böhm, W. Schulze (Eds.). "Workflow Management". Heidelberg: dpunkt-Verlag, 1997.

ISO 10303-21:

Industrial automation systems and integration – Product data representation and exchange – Part 21: Implementation methods: Clear text encoding of the exchange structure

ODETTE PID:

ODETTE: Partner Identification and Database. Version 1.0. 2006

QDXReport8D-1.1

Quality Data eXchange; 31.03. 2006; www.vda-qmc.de

RFC 1034:

P. Mockapetris. Domain Names – Concepts and Facilities. RFC 1034. November 1987, http://www.ietf.org/.

VDA 5006:

Einheitliche Identifizierung von Partnerunternehmen – UPIK (Unique Partner Identification Key). VDA-Empfehlung 5006, 1st edition, August 2003.

2 Scope and Field of Application

This recommendation is intended for manufacturing companies and suppliers in the automotive industry and development service providers who are integrated in the engineering change management processes of their clients. It is also intended for management, those responsible for product and process documentation, heads of development departments, those responsible for data exchange and those responsible for IT systems.

The topics dealt with in the recommendation concentrate on the engineering change management processes and procedures, including the associated documentation, for making changes to the product and the documentation during the development of the product, and start-up and manufacture of the series[5]. The guidelines can, however, also be applied throughout the entire product development process, so that the recommendation can, for example, be used during the early phase of development in individual cases.

This recommendation can be used for two types of integration in the context of engineering change management: data-oriented integration and process-oriented integration, which also can be combined.

In the case of data-oriented integration – e.g. when suitable clients are used for direct access to the partner systems – change data is communicated between the partners using the same methods as with process-oriented integration, for example using file-based exchange of product data (see 2.2).

In the case of process-oriented integration, the partners are incorporated into the engineering change management process directly. This is typically achieved by the partners receiving at predetermined points in engineering change management process (synchronization points) specific information like problem descriptions, proposed changes, decisions which are relevant to them or achievements of a particular status or milestone. Furthermore, it may also be used to archive requests for specific change data or assessments, or requests to carry out particular activities to which the partner reacts or responds in an appropriate manner. According to the recommendation, process-based integration then proceeds on the basis of agreed ECM Interaction Scenarios between the partners (see 2.2.2).

The recommendations are largely independent of the implementation method and can be realized using either asynchronous data exchange or synchronous communication (see Appendix F for conformance requirements). They are thus applicable to both online and offline system integration, and support the use of Web clients, in-house ECM systems of the partners and also the use of smart-tool solutions for small and medium-sized enterprises. In addition, this permits the solution to be used by a number of partners without the need to create individual solutions for every partner-specific ECM communication method.

2.1 Engineering change management tasks

The task of engineering change management is uniform management and tracking of changes, i.e. collecting ideas for changes, putting these into concrete form, evaluating them

[5] Software and signals, for example, have at present not been taken into account.

and implementing them with respect to development and manufacturing. The use of organizational measures with clearly defined scopes of authority and responsibilities on the one hand allow the component tasks of engineering change management to be addressed more clearly, and on the other hand the establishment of an end-to-end engineering change management process allows the progress of the change to be better monitored, controlled and retraced by those involved.

The need to make changes to a product arises as a result of any problems which occur or any potential for optimization detected during the life cycle of the product, i.e. from its initial conception through to its disposal. The initial idea for a change can therefore in principle come from any area of the company (e.g. procurement, development, manufacturing, sales, after-sales) but also from partners who participate in customer/supplier relationships such as development partners, manufacturing partners or even the end customers.

The result of the ECM process is a change which has been implemented in the manufacturing environments after it has been evaluated and implemented and released in the development process. It is of course possible that the idea for a change may be rejected, for example if it appears unpromising, if no tangible benefit can be identified or if the change is not approved for other reasons.

2.2 Application scenarios

2.2.1 Data oriented integration

In the "data-oriented integration" application scenario, one partner provides the other partner with an external view of his change data. The second partner can use this data to obtain or reconstruct information on the current status and the (external) view of the partner with respect to the proposed change during the course of the ECM process or after the ECM process has been completed (Figure 1).

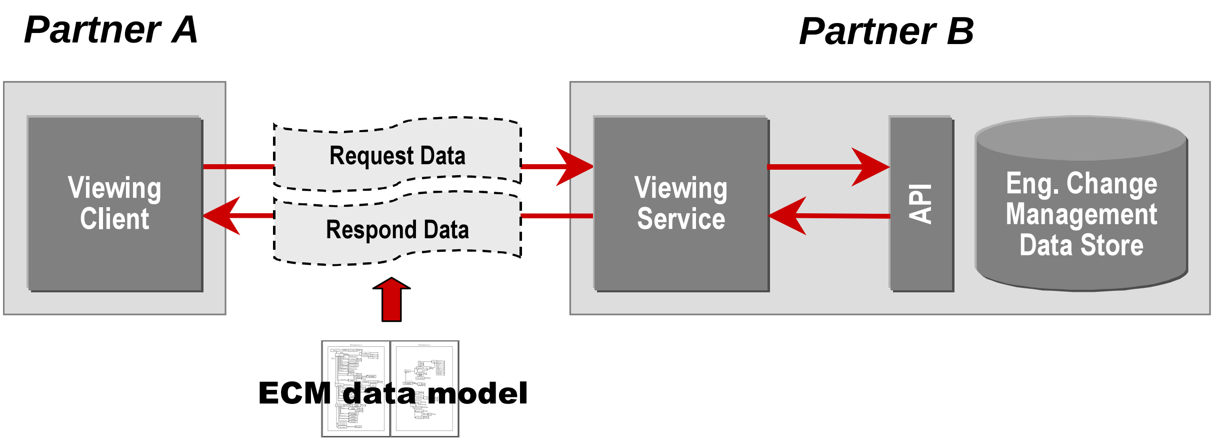


Figure 1: Schematic representation of the "data-oriented integration" application scenario

This can be done independently of the ECM process to a large extent since the partner can at any time request information on the changes to the product data relevant to him as the need arises. Requests for data can, however, also be used for simple incorporation of partners without the need to implement systems for exchanging change data or fully integrate the partners into the ECM process. Data exchange mechanisms between the partners can also operate on a mutual basis. In this case, each partner can use the external, harmonized view of the change data of the other partner in order to compare his own change data with that of his partner.

When requesting data, partners can use a client, for example, to query specific change data available from the data provider. To achieve this, the data provider must implement a data viewing service which is able to access the provider's internal data storage services and carries out conversion of the data in accordance with the harmonized Data Model laid down in this recommendation. The core functions which should be provided by the data viewing service include querying specific change data as well as retrieving individual items of change data.

2.2.2 Process oriented integration

When there is a customer/supplier relationship between two partners, there are typically dependencies between the private ECM processes of the individual partners. At different times in the engineering change management process, one of the partners proposes a change or one partner must be integrated in the other partner’s own process in order to permit activities such as technical elaboration, evaluation or decision-making. This is generally achieved by sending suitable messages to the partner (Figure 2).

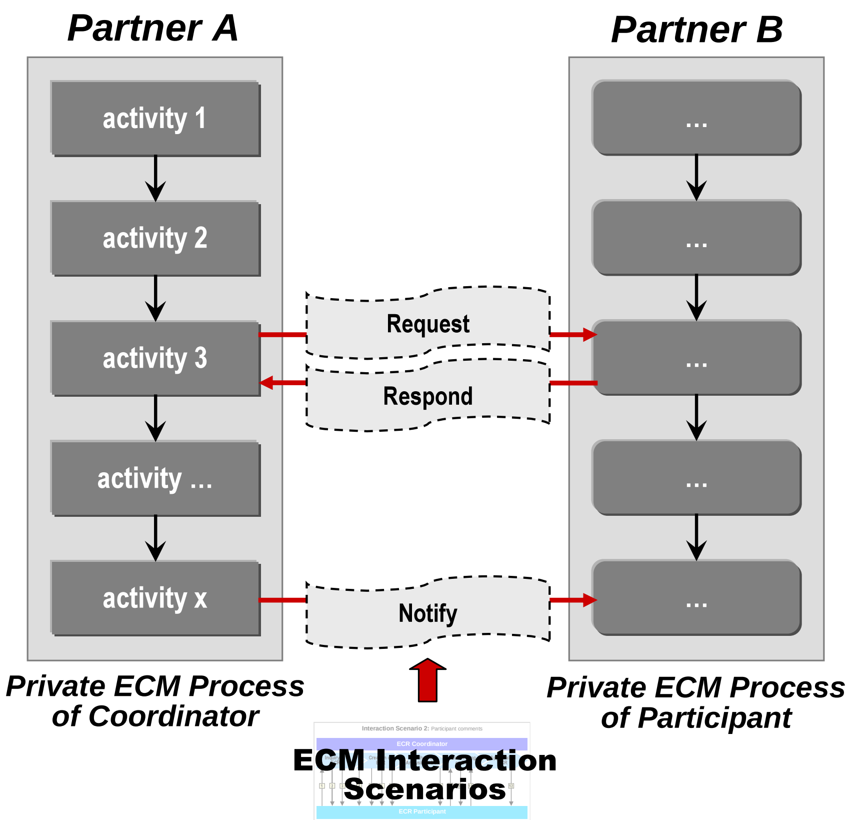


Figure 2: Schematic representation of the “process-oriented integration” application scenario

The underlying communication model used in this recommendation is based on the bilateral exchange of messages. This means that the private ECM process of one partner interact with the private ECM process of the other partner on the basis of *request*, *response* or *notify* messages. Together, these interactions make up the communication between the two partner processes.

The sequence of messages in an interaction must correspond to that laid down in the harmonized ECM Reference Process. In addition the information contained in the body of each messages must correspond to the information, which is specifed in the process context of the message

The quantity and scope of the required interactions can vary depending on the way in which tasks and process roles are distributed between the partners. The use of different ECM Interaction Scenarios can also be influenced by the content of the change. For example, the interaction will be different if the *participant* is responsible for the costs of the parts involved and if the *participant* only has responsibility for development-related aspects of the parts. The use of the standardized ECM Reference Sub-Processes, ECM Interaction Scenarios and messages as well as the standardized data model permit a significant reduction of the effort necessary for agreeing on and implementing ECM Interaction Scenarios in the context of cross-partner engineering change management. This is because a uniform approach can be adapted for integrating several pairs of partners as opposed to constantly agreeing on and implementing bilateral specifications and custom adaptations.

The communication between the partners is based on a standardized data model which is the result of the mapping of the ECM Data Model onto a standardized data representation and format.

2.2.3 Combinations of data and process oriented integration

Data-oriented integration and process-oriented integration can be used independently of each other as described above. However, both kinds of integration may also be used in combination. In this case, process-oriented integration may be used for interactions depending on the state of processes. Data-oriented integration may be used to communicate current data at arbitrary points during the course of the process. For example, if an ECM implementation of two partners A and B uses both kinds of integration, the combination version may be used as shown in Figure 3.

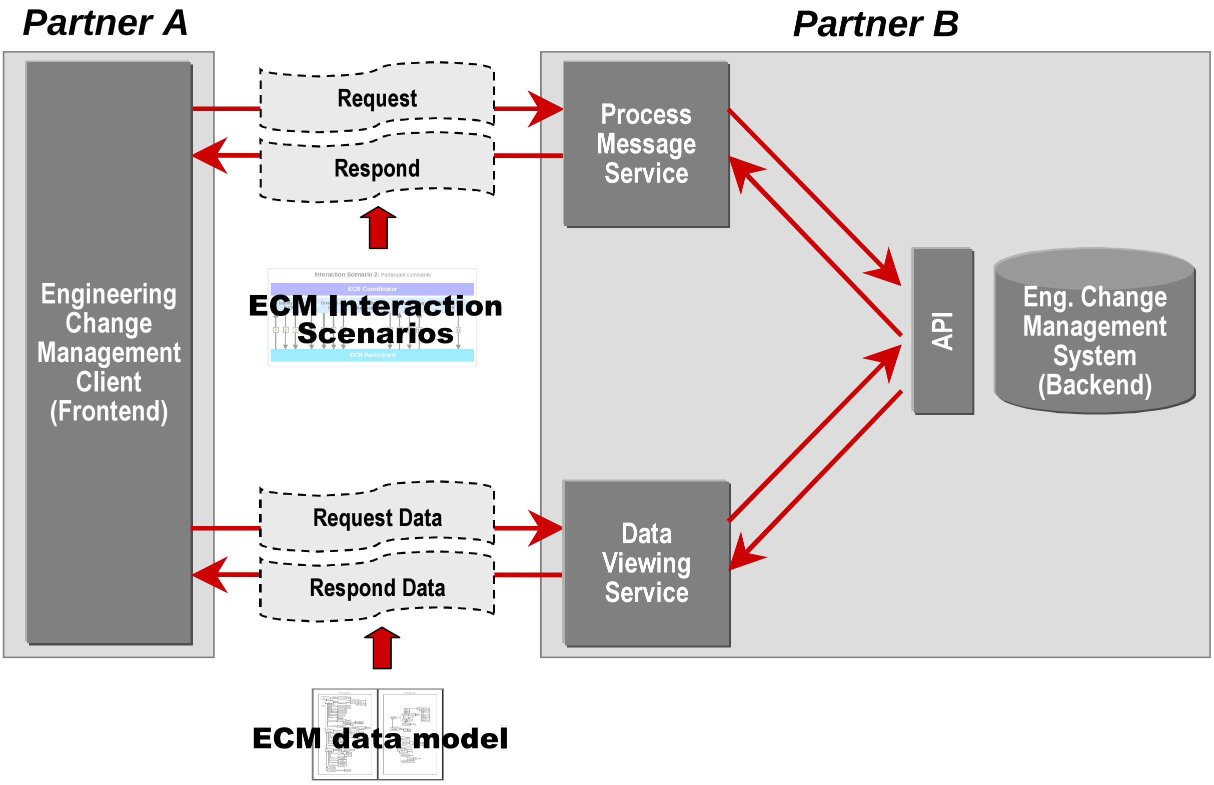


Figure 3: Schematic representation of the combination of "process-" and "data-oriented integration"

Partner A has been involved in an ECM Interaction Scenario where he sends a engineering change proposal to partner B. At an arbitrary point during (or after) the ECM Interaction Scenario, Partner A can request an update on the current state of that engineering change proposal.

2.3 Recommendations for use

The set of application guidelines is not only a recommendation with respect to implementation in the field of engineering change management but can also be used as a general guideline for cooperation relationships. The

- generally applicable ECM Reference Sub-Processes and their milestones (necessary synchronization points), the
- information units to be exchanged (messages), the
- ECM Reference Sub-Process-specific ECM Data Model and the
- ECM Interaction Scenarios

represent elements which can be used for any form of communication between partners. They support analysis and harmonization of any existing partner processes as well as any necessary adaptation of these processes. Once a common understanding has been established, it facilitates the specification of further tasks. The quantity of relevant data to be exchanged can be verified on the basis of the data models. Together, the ECM Reference Process and the ECM Data Model thus serve as a checklist for the following issues:

- Have the individual private ECM process steps been harmonized?
- When must the data be synchronized between the partners involved?
- What data must be exchanged and when?
- Has all the necessary data been specified?

This recommendation is therefore also suitable for optimizing current ECM processes.

3 ECM Reference Process Fundamentals

This section provides an overview of the fundamental principles and key Sub-Processes of the ECM Reference Process. It describes the solution approaches on which the ECM Recommendation is based. Section 3.3 provides information on how the communication relationships between *coordinators* and *participants* can be supported in a cooperative network using the ECM Reference Process. A global set of regulations is derived from this, laying down the foundations for the ECM Reference Sub-Processes, the ECM Interaction Scenarios, ECM Messages and the data which is communicated between the partners.

The ECM Reference Process fundamentals may be described in more detail in other parts of this recommendation depending on the specific requirements of the ECM Sub-Processes.

3.1 ECM process modeling

Irrespective of individual requirements and project-specific variants of private ECM processes at the partner companies, it is still by and large possible to classify the ECM activities of the partners in identical ECM Reference Sub-Processes. The first Sub-Process involves identification and description of the problem which triggers wish to initiate a change. One or more alternative solutions are initially developed and the changes to the current state which become necessary as a result of these are defined. During the implementation Sub-Process, the ECM activities which have been planned and approved are implemented in the design. These activities can in turn lead to new or modified manufacturing processes and, for example, to new or modified parts, procedures or organizational structures.

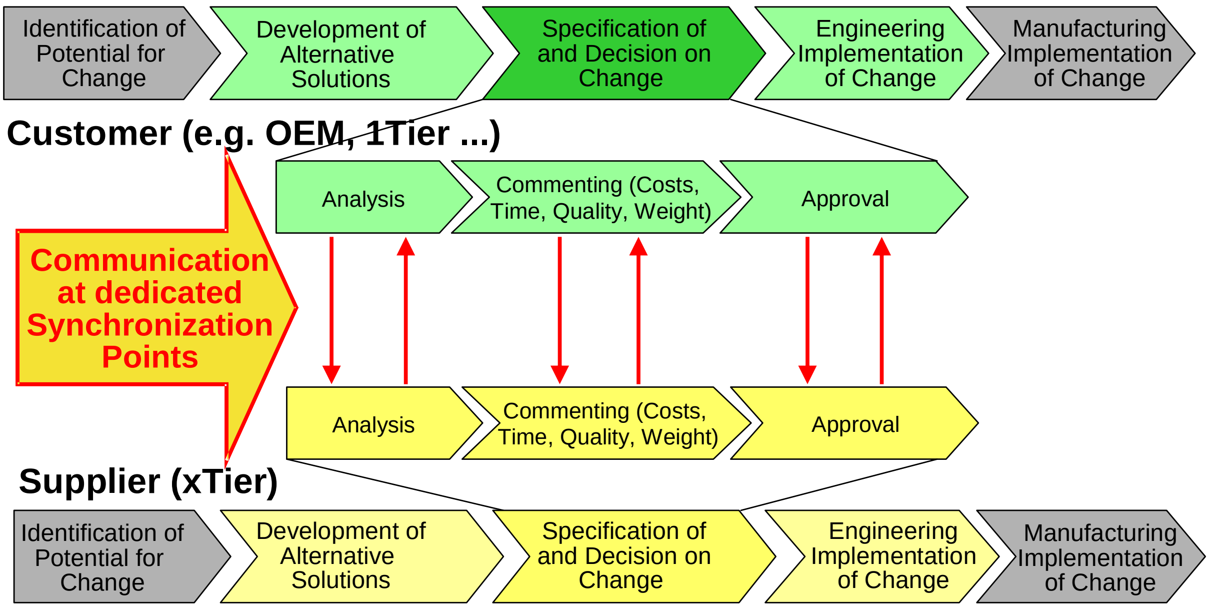


Figure 4: Strategic process synchronization as a fundamental principle in ECM communication

Since the private ECM processes are based on the same fundamental principles but can otherwise vary considerably from case to case, it is recommended that cross-company communication is carried out using individual data exchange processes (interactions) at specific synchronization points during the course of the process. The varying private ECM processes at the different partners can be synchronized by exchanging defined messages. This approach reduces the effort involved in communication at the interface between the partners.

In order to define the meaning and context of the possible interactions and to arrange them in a logical sequence, the recommendations for ECM are based on ECM Reference Sub-Processes. The agreed ECM Reference Sub-Processes form the basis for common agreement with respect to communication between the partners involved and provides the basis for standardized interactions. ECM Interaction Scenarios are derived from the ECM Reference Sub-Processes and define in detail specific roles and activities for the partners and interactions among each other for specific partnership constellations.

3.2 ECM Reference Process

Figure 5 shows the ECM Reference Process in a phase diagram involving the ECM Reference Sub-Processes[6]. In most cases, ECM activities are triggered by identifying ideas for improvements (M1) which are then analyzed with respect to their potential benefits in the first phase (*Identification of Potential for Change*). It is, however, also possible for such activities to be triggered by current issues or risks. The need to make a change can be identified by the customer, the development department or a supplier; it can result from tests, trial construction or prototyping or during series start-up, manufacturing or service activities. The need for change can also arise as a result of changed conditions such as new legislation, organizational influences or cost optimization measures. *Identification of Potential for Change* can result in reaching milestone *M2: Change Potential Identified*.

Irrespective of the type or source of an idea for a change, processing and management of the change request always in principle follow the ECM Reference Process.

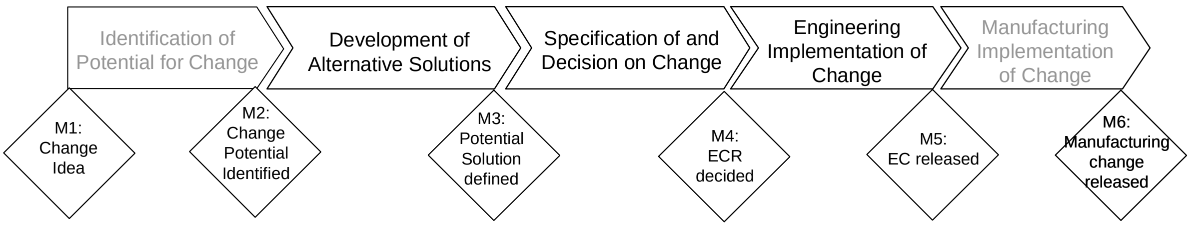


Figure 5: ECM Reference Process: Phase overview

The second phase of the ECM Reference Process, *Development of Alternative Solutions*, represents the description of different approaches to implement an idea for a change, since the idea is initially analyzed with respect to its potential benefit irrespective of the way in which it might be implemented. This phase also includes the evaluation and selection of favoured solution variants which will then be studied more closely and given preferential treatment during the course of an ECR, and/or it may involve a preliminary decision.

Once milestone M3: *Potential Solution Defined,* has been reached, a proposed solution is analyzed with respect to technical and commercial considerations in the phase *Specification*

[6] Note: The ECM Sub-Processes here are introduced as consecutive phases for simplicity reasons but in general may be executed partly concurrently. In addition, not all ECM Reference Sub-Processes need to be performed for every change. The possibilities and restrictions have to be defined in a later version of this document.

and Decision on Change (representing the ECM Sub-Process ECR) and the resulting change requirements are specified in detail. Appropriate evaluations and comments from the specialist departments and partners who are involved respectively affected lead to decision as to whether the proposed change is approved or rejected. The ECR object forms the basis for these activities and provides the necessary authorization.

If the ECR is approved (M4 *ECR Decided*), the change is implemented in development in phase four *Engineering Implementation of Change* (representing the ECM Sub-Process ECO), for example by making changes to the relevant documents such as CAD models, drawings, Bill of Materials (BoM) or product structures. The ECO object forms the basis for these activities and provides the necessary authorization. Depending on the scope and type of change, different units may be involved in implementation at the partners, for example, planning, logistics, operating resources and procurement (series).

When the change is implemented by engineering and released *(M5: Engineering Change released) Manufacturing Implementation of Change* (representing the ECM Sub-Process MCO) begins. For example, production processes and tools producing the parts affected by the change have to be adapted, new and changed parts need to pass quality assurance and changes concerning logistics need to be performed. The MCO object forms the basis for these activities and provides the necessary authorization. This phase ends with M6: Manufacturing Change Released.

During the course of the ECM Reference Process, ideas for changes, implementation alternatives and change requests are evaluated in succession and decisions are taken, e. g. to discard solutions and to evaluate different solutions, or not to pursue some change potential idea any further. As a result, it is also possible for the ECM Reference Process as a whole or for specific ECM Reference Sub-Processes to be completed in the middle of the process shown.

The core ECM processes focus on the development of alternative solutions, the specification of and decision on change and the engineering implementation of change. These processes are:

- “Development of Alternative Solutions”
- “Specification of and Decision on Change”, and
- “Engineering Implementation of Change”

The processes

- “Identification of Potential for Change” and
- “Manufacturing Implementation of Change”

are not considered as ECM core processes. The activity „Identification of Potential for Change“ is considered to be part of the “Issue Management Process“. If there is further interest in this specific activity, the following documents provide ideas on how to implement this type of process and potential data exchange between partners:

- CMMI: Capability Maturity Model® Integration (CMMISM); Version 1.1
  CMMISM for Systems Engineering, Software Engineering, Integrated, Product and Proc-

ess Development, and Supplier Sourcing (CMMI-SE/SW/IPPD/SS, V1.1)
CMMI describes how to manage requirements in general and how to identify inconsistencies between those requirements and project / product specific conditions. It therefore gives ideas on a potential process for issue management. (see CMMI, V. 1.1, Chapter 7, Engineering, Requirements Management and Requirements Development).

- QDX: QDXReport8D-1.1; Quality Data eXchange
  QDX describes a format that enables partners to perform the exchange of quality data. The xml-Schema provides ideas on exchange problems and possible solutions before the detailed description of alternative solutions begins.

The activity “Manufacturing Implementation of Change” is considered to be part of "Production Processes".

3.3 ECM partner model

Cooperative activities within the network of manufacturers and suppliers are characterized by a high degree of variability. This variability and the large number of systems involved, coupled with interfaces which are often incompatible with each other, make it particularly difficult to integrate process and product data across the boundaries of different locations and different companies. For this reason, it is recommended that the private ECM processes, which naturally differ from each other, should be synchronized by means of targeted communication steps (interactions) carried out at defined synchronization points, thus enabling cross-partner process synchronization with IT support.

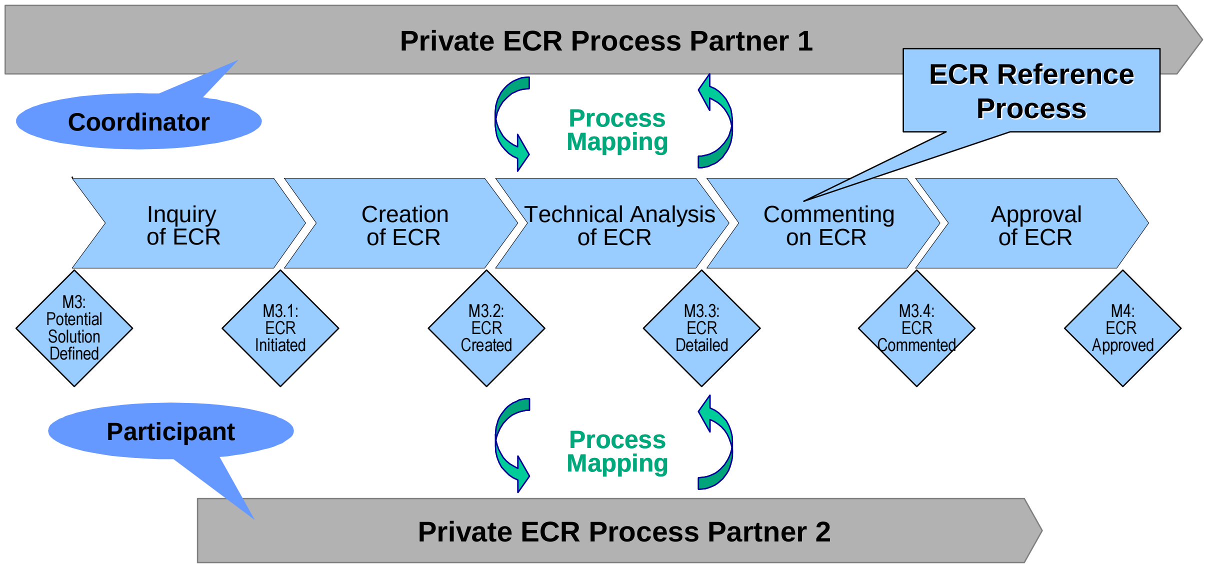


Figure 6: Harmonized synchronization, e.g., via the ECR Reference Process

In the context of collaborative engineering change management as described by this recommendation, synchronization of the various private ECM processes of the partners involved is done using ECM Reference Sub-Processes. As illustrated in Figure 6 using the example of an ECR, this means that for any concrete change request, the *coordinator* takes responsibility for managing the process and interacts with the outside world according to the ECR Reference Process. The partner (*participant*) is incorporated by means of the interactions and messages which have been defined within the ECR Reference Process. The partner in turn then implements these in terms of his private ECM process and data.

By reducing the complexity to a simple set of rules governing the relationship between the *coordinator* and the *participant* in the context of the ECM Reference Process, it becomes possible to use IT solutions to support even deeply nested and variable cooperative networks between manufacturers and suppliers without the need to provide separate solutions for each cooperation relationship. Transparency of the harmonization and decision processes is considerably enhanced because the content of each message during communication with the partner is embedded in the context of the relevant ECM Reference Sub-Process. For example, the list of relevant parts is consolidated over all the affected partners during *Technical Analysis and Detailing of ECR*, and forms the basis for comments during *Commenting on ECR*.

Since an ECM Reference Process generally involves several partners, it is possible to cascade process synchronization in a cooperative ECM network (Figure 7). This means that an ECM change network is broken down into individual bilateral relationships between one *coordinator* and one *participant*.

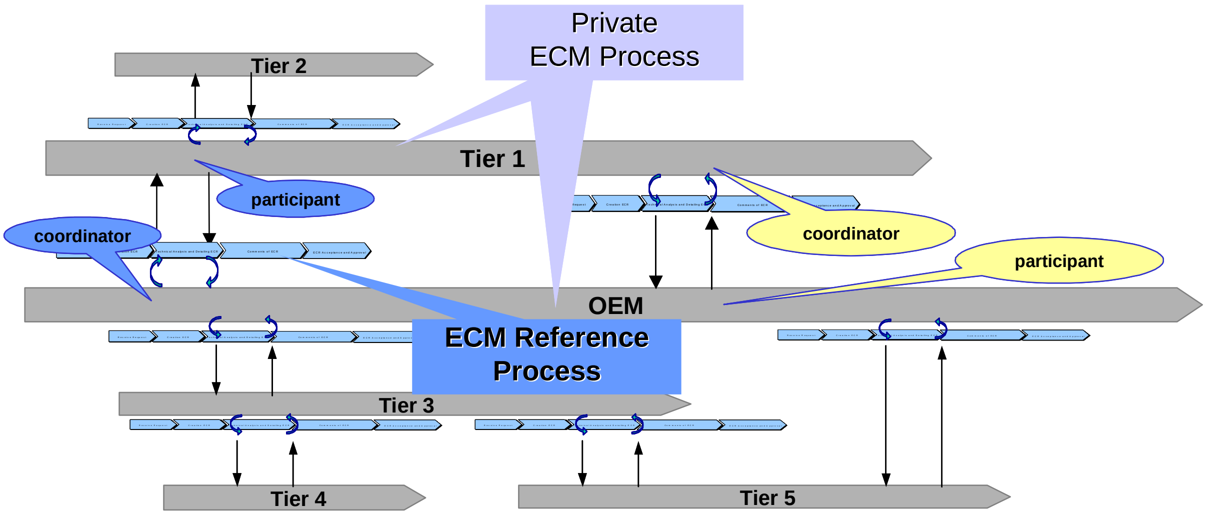


Figure 7: Cascading ECM communication in a cooperative ECM network

Each partner within an ECM network may have the role of the *coordinator* or *participant* depending on the specific cooperation.

The partners involved in an ECM Reference Process work within their own private ECM processes which are individual company-specific and/or project-specific. Generally, they will use different ECM systems and components for managing the relevant data in a proposed change. Each of these relationships should comply with the ECM Reference Process and with the global rules laid down in the following sections. Figure 8 shows an example of a communication breakdown in a partner network in an ECR Reference Process.

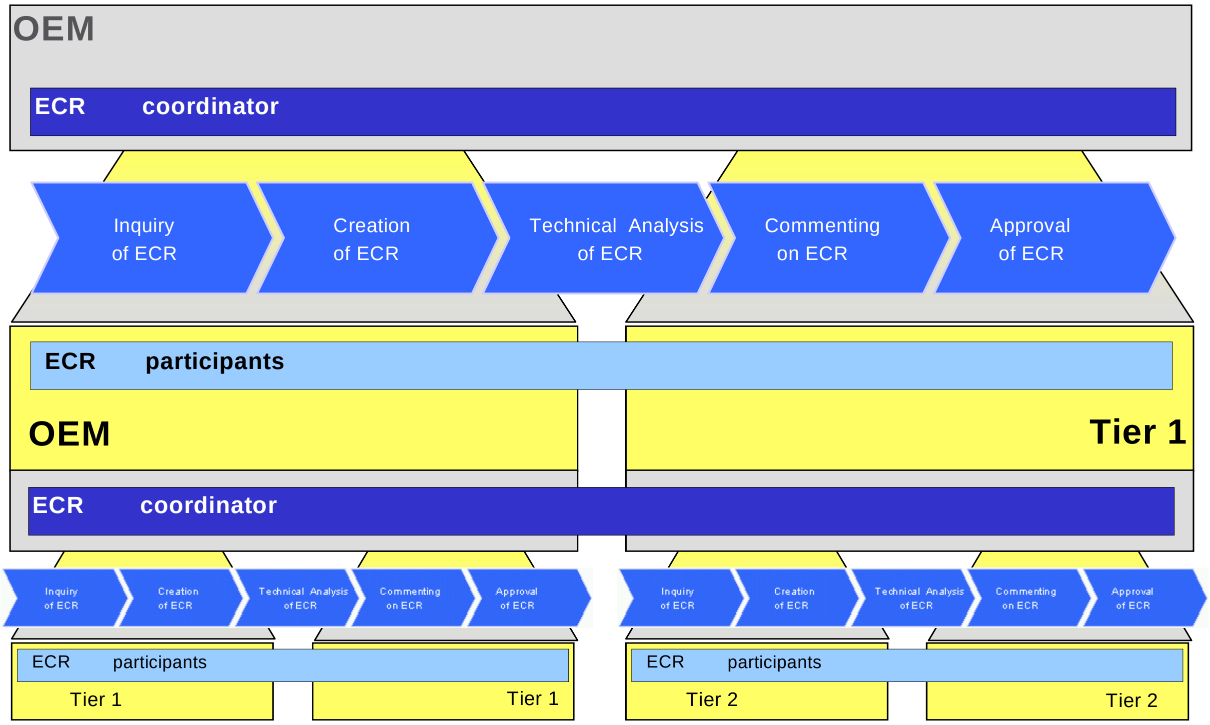


Figure 8: Communication breakdown in a partner network using an ECR as an example

3.4 Global rules

This section defines rules used as the basis for the specification of the recommendation. The rules specify assumptions and points of view and, in that context, support a better understanding of the recommendation.

3.4.1 ECM Reference Process rules

- In the context of an ECM Interaction Scenario, the organization responsible for coordinating the specific ECM Reference Sub-Process and for ensuring that a decision is made takes on the role of coordinator and a partner involved in this ECM Interaction Scenario takes on the role of participant.
- *Coordinator* and *participant* communicate within the ECM Reference Sub-Processes via ECM objects like ID (including ID.process_phase) to identify the object of change and the related ECM Sub-Process.
- The ECM Reference Sub-Process of the *coordinator* is presented externally as laid down in the ECM Reference Process by mapping the private ECM Sub-Process onto the ECM Reference Process. This defines the context for messages during bilateral communication with the *participant* in ECM Interaction Scenarios.
- Each ECM Reference Process defines the common context and the terms and semantics used to achieve a common understanding between the two partners.
- The *participant* maps its private ECM process to the ECM Reference Process.
- By chaining ECM Interaction Scenarios between two specific partners, an ECM network comprising different OEMs and tier-1 through tier-n suppliers can be created in which

each bilateral ECM Interaction Scenario complies with the ECM Reference Process agreed with the *coordinator* (Figure 8).

- It is recommended that the *coordinator* and *participant* roles should not be changed within an ECM Interaction Scenario.

3.4.2 Interaction rules for the partners involved

- Communication between two ECM partners takes place on the basis of interactions in which individual messages compliant with one ECM Interaction Scenario are exchanged.
- An ECM Interaction Scenario represents the permitted sequence of interactions for a typical cooperation situation. Each interaction comprises one or two[7] messages. The sequence of interactions or messages is also specified in the specific ECM Reference Sub-Processes and defines communication of the data at specified points in time in the process.
- The permitted use of a specific message is defined within an ECM Reference Sub-Process on the basis of the synchronization points specified in the ECM Reference Sub-Process. Further detail on the permitted use is provided by additional specifications in the ECM Interaction Scenario used.
- If a *participant* is to supply data following a request, the *coordinator* must generally wait for the respond message from the *participant* before he is able to carry out further interactions in accordance with the ECM Reference Sub-Processes, even if his private ECM process may already be in progress.
- Both partners must act in compliance with one ECM Interaction Scenario.
- Version management of ECM objects is not supported by this recommendation. If different versions of an ECM object are needed, it is recommended that a new ECM object be created and that this new ECM object be linked to the previous one by a relationship.

3.4.3 Data content rules

- As a minimum, all the mandatory data objects of a message must be communicated at the relevant synchronization point.
- References between different ECM object identifiers are directional. (Example: The ECM object identifier of a *participant* in an ECM network is derived from the ECM object identifier of the *coordinator*.)
- Data objects in a message can be mandatory or optional. If a data object is declared as mandatory, it should be agreed as mandatory in every cooperation and in every project.
- If a data object is declared as mandatory, it is expected that the sender communicates at least one instance of this data object.
- If no user data is available for a mandatory data object at a given time, this must be communicated explicitly.

[7] In special circumstances, an interaction may comprise several messages.

- If a data object is declared as optional, the sender is permitted to communicate this data object. Whether or not it is necessary to communicate optional data objects depends on the context and must be decided on a case-by-case basis.
- Individual data objects which have been declared as optional can, if required, be set to mandatory during a cooperation activity. The reverse is not true.

4 Overview of the Modelling Concept and the Notation

The ECM Reference Sub-Processes, ECM Interaction Scenarios, ECM IS Protocol Specifications, ECM Messages and ECM Data Model objects involved in engineering change management are described on the basis of diagrams. The modelling methods and notations are common to all parts of this recommendation.

4.1 ECM process notation

With regard to ECM Reference Sub-Process definition, separate diagrams are used to provide a general overview of a Sub-Process and detailed views of the Sub-Processes. The ECM Sub-Processes are described from the point of view of the primary process owner (*co-ordinator*). Partners (*participants*) can fulfill certain roles in an ECM Reference Sub-Process. In this case, interactions with the partner will take place. Phase diagrams are used to provide a general overview of the ECM Reference Sub-Process. These show a linear representation of consecutive phases. Typically, phases are initiated when milestones are reached and, in the same way, milestones can be reached at the end of phases. If one of the phases can be carried out by a partner, this phase is shown in a line entitled *participant*, and the primary process is shown in the line entitled *coordinator*. (If the partner does not have an active role in a phase, but only introduces or receives notifications, this is not indicated separately in the phase diagram.)

Activity diagrams are used to provide a more detailed description of the processes in an ECM Reference Sub-Process. These are described in UML 2 (Unified Modelling Language 2). In the activity diagrams, the activities are modelling processing steps which are carried out by roles. The sequence of the activities is described in more detail. Composite activities are described in further detail in separate diagrams or are derived from patterns which are described in Appendix A. In the case of interaction between partners, individual interactions between roles, during which message objects are exchanged, are modelled. The relationships between the roles used in the ECM Reference Sub-Processes are drawn in the diagram for its organization model. In particular, this shows which roles in the activity diagram are assigned to the *coordinator* and which roles can be embodied by *participants*. Specific statuses which describe the positive or negative result of an activity, for example, are modelled using specific objects.

To achieve greater clarity, a distinction is made between the normal flow of an activity and special variants of this flow and exceptions for the activity. The normal flow which is represented need not always be the normal case during actual execution; instead it describes the primary activities in a way which is easy to understand. Appendix A contains notes on the notation used.

The following should be noted with respect to the terms "phase" and "activity": In the description of the phase diagrams, the term "phase" is used throughout to refer to the processing steps; in the descriptions of the activity diagrams, the term "activity" is used throughout. For each of the phases described in detail, there is either one activity which has the same name, or the phase comprises a number of activities.

4.2 ECM Interaction Scenario notation

An ECM Interaction Scenario describes how data is exchanged between partners for a particular purpose using messages in order to implement ECM Sub-Process integration for a specific type of cooperation. ECM Interaction Scenarios make use of certain possibilities for interaction provided by the ECM Reference Process in order to exchange the associated messages in a specific application context. A distinction is made between two different types of interaction: In unidirectional interactions, a single message of type *notify* is sent; in bidirectional interactions, one message each of the type *request* and *respond* are exchanged.

ECM Interaction Scenarios are briefly described by illustrating the application context and showing the distribution of the tasks between the *coordinator* and the *participant* in the development process. This is supplemented by a rough description (if necessary for each variant of the ECM Interaction Scenario) of the message flow involved in exchange of messages. An overview diagram similar to an interaction diagram shows the flow of messages specific to the ECM Interaction Scenario from the *coordinator* to the *participant* and vice versa. The messages are assigned to the phases of the ECM Reference Process of the *coordinator* in order to show the process context in which a message is exchanged. This process context is precisely defined in the ECM Reference Sub-Processes. This involves assigning the sending and receiving actions of the *coordinator* to specific points in the ECM Reference Sub-Process. This is complemented by message flow diagrams. These diagrams describe the permitted sequences of messages specific to the ECM Interaction Scenario. A distinction is made between mandatory messages and optional messages in the ECM Interaction Scenario.

The meanings of the messages which can occur in different ECM Interaction Scenarios is described in separate sections.

Other messages and other conditions which apply to the message sequences, in particular for the flow variants and exceptions, are described by the ECM Reference Process only and are not explicitly illustrated in message flow diagrams.

4.3 ECM Interaction Scenario Protocol Specification notation

An ECM IS Protocol Specification precisely describes in detail the message flow specific to each mentioned ECM Interaction Scenario. For every ECM Interaction Scenario, one diagram is provided for the *coordinator* specification and one for the *participant* specification.

The protocols are described using UML activity diagrams just like for the ECM Reference Sub-Processes (see section 4.1), although the content is more detailed and specific for a certain ECM Interaction Scenario. References to data model attributes which affect the message flow are given. In addition, references to cooperation-specific settings are modelled explicitly as annotations in order to provide transparency regarding needed agreements and the impact on the expected behaviour of the message flow. If necessary, specifications are divided into several diagrams to improve readability.

4.4 ECM Data Model notation

ISO 10303-11 defines EXPRESS as a formal language for specifying data models used to describe product information. Use of a formal language provides a clear and consistent representation and facilitates implementation. EXPRESS allows data and the associated constraints to be described.

EXPRESS-G is the graphical representation of EXPRESS and is defined in Appendix D of ISO 10303-11. A brief overview (of the elements used) is provided in Appendix B.

The common ECM Data Model is specified using EXPRESS-G diagrams (see Appendix E). Implementations of the EXPRESS code can be derived from these diagrams. The detailed specifications are given in the data dictionaries in Appendix E. They define all data types (entities) and attributes including attribute types, cardinalities, (application) rules and predefined attribute values. The ECM Sub-Process dependent specifications are defined in the related parts of this recommendation.

The basis for the ECM Data Model is formed by the units of information in the ECM Reference Sub-Processes as well as by the analysis of the data models of ECM application systems used by the companies participating in the working group, taking into consideration what data needs to be exchanged with partners in the context of change requests. Generalization of the application data resulted in the ECM Data Model (described in chapter 7.2) as well as in ECM Sup-Process specific data models (e.g. the ECR Data Model) described in other parts of this recommendation.

Notation:

- Data types are written in small capitals (ORGANIZATION)
- Attributes of the data type are added after a dot ('.') (ORGANIZATION.name)

5 Cross-Sub-Process Dependencies and Relationships between Different Phases

This chapter is work in progress at SASIG

5.1 Dependencies between different ECM Reference Sub-Processes

This section is work in progress at SASIG

5.2 Dependencies between ECM Interaction Scenarios of different ECM Reference Sub-Processes

This section is work in progress at SASIG

5.3 Dependencies between the partial ECM Data Models of different ECM Reference Sub-Processes

This section is work in progress at SASIG

6 Procedural Model for the Application of this Recommendation

An all-inclusive approach is recommended in order to exploit the potential of collaborative engineering change management. Simply developing or purchasing the appropriate interfaces and tools will not deliver the desired results. From the moment the partnership is set up, it is recommended that a common understanding be established with respect to the organization, the methodology, and agreements be reached on ensuring geometrical, functional and manufacturing-related viability and on the depth of integration. Process harmonization within the partnership is supported by an initial selection desired ECM Interaction Scenarios (Figure 9).

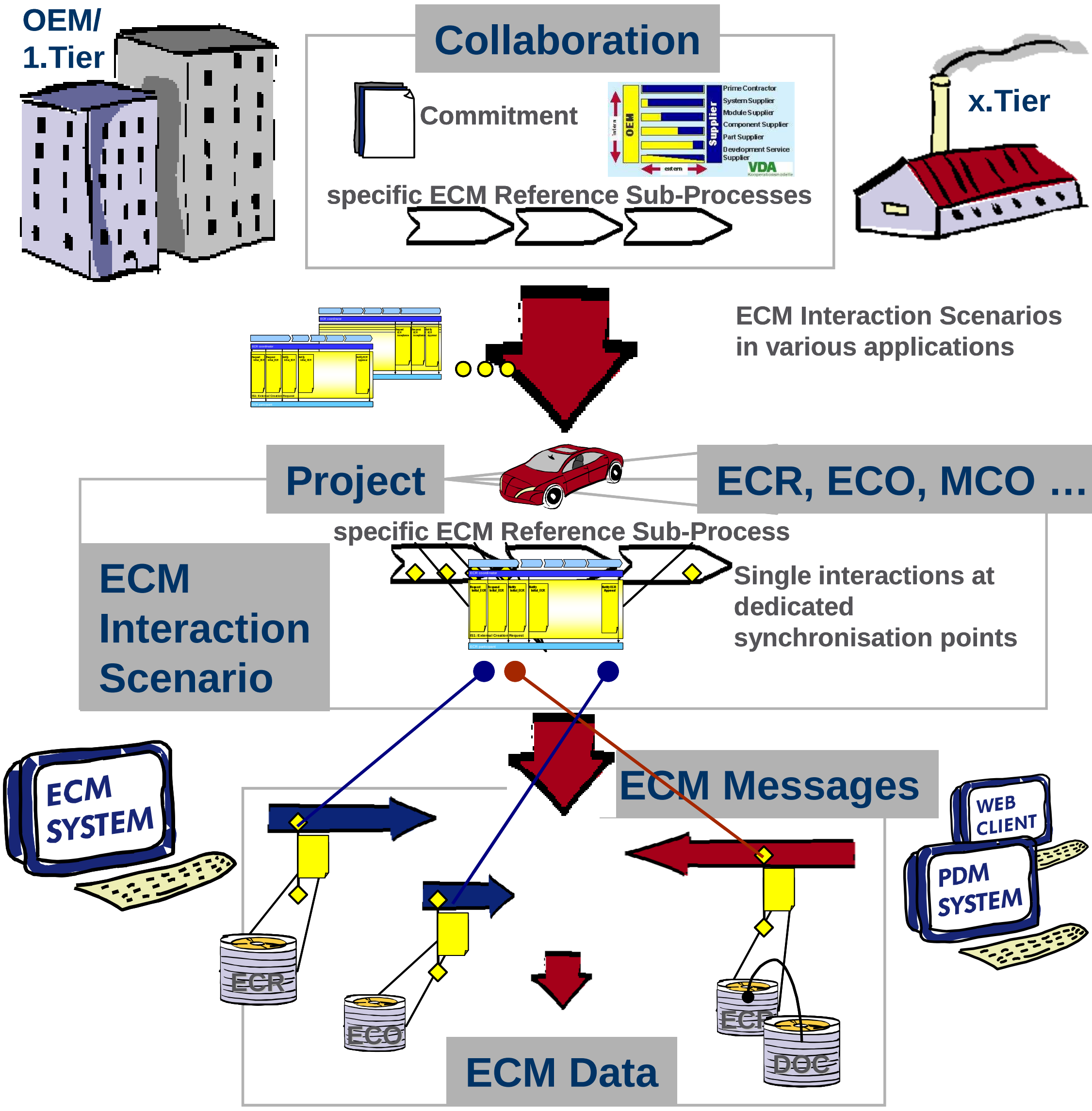


Figure 9: Using the ECM Reference Sub-Processes and ECM Interaction Scenarios

The ECM Reference Sub-Processes and the ECM Interaction Scenarios are used to reach an agreement regarding responsibilities, process steps, expected messages, and the scope of change data to be exchanged between the partners. These can be configured in the systems involved and in the associated interfaces (Figure 9)

This allows misunderstandings to be avoided successfully, the total number of errors to be reduced, and any gaps in coordination to be detected at an early stage. The modularity of the system and the use of reference models mean that at least 60% of the coordination effort can be saved in new projects and cooperative partnerships and that preconfigured components (e.g. the ECM Reference Sub-Process descriptions, extensive sections of supplier agreements, interfaces, communication tools) can be reused or adapted dynamically.

The contacts on the *coordinator* and *participant* sides assume special responsibility if the ECM Reference Sub-Processes are used systematically. Before cooperation starts, the contact roles for each partner and the communication relationships between them should be specified by

- definition/listing of the contact roles,
- definition/listing of the possible communication relationships.

During implementation, steps must be taken at system level to ensure that the agreed contact roles are informed about the ECR, ECO etc. or that they can obtain information as required. The communication relationships must be observed during processing of a concrete change request. Depending on the nature of the cooperation, the contacts who are involved and the roles assigned to them can be specified at any time, but should always be kept up-to-date.

Since the recommendation is standards-based and is largely independent of any architecture, there is a high degree of flexibility with regard to implementation, particularly for suppliers. This means that ECM Reference Sub-Processes, ECM Interaction Scenarios and messages can for example be implemented synchronously on the supplier side by 'resident engineers' when using customer systems, by Web client technology or by 'fat clients'. Online or offline communication between an in-house ECM system and the partner system is also possible, as is a connection via neutral Web clients.

The following represents a simple checklist to serve as an introduction to using the ECM Reference Sub-Processes:

- Familiarization with the ECM Reference Sub-Processes
- Comparison between the private ECM Sub-Processes and the ECM Reference Sub-Processes
- Familiarization with the ECM Interaction Scenarios
- Decision on whether to use online or offline communication (see Appendix C)
- Decision on whether to exchange data or messages (see Appendix C)
- Initial selection and definition of the desired ECM Interaction Scenarios
- More precise definition (or expansion) of the selected ECM Interaction Scenario(s)
- Mapping of the attribute lists of the in-house processes and the ECM Reference Sub-Processes

- Definition/listing of the contact roles
- Definition/listing of the possible communication relationships
- Configuration of the systems and interfaces involved

7 Common Resources for ECM

7.1 Common ECM Messages

This section provides information on the common resources messages and data objects, which can be used in each ECM Reference Sub-Process in the same way, i.e., which are independent of the specific ECM Sub-Processes. Usage of an element relating to the common resources may be described in more detail in AppendixD.

7.1.1 Administrative messages

To improve data quality and ease of use, administrative messages can be defined to communicate the values permitted for selected ECM data objects and their attributes. These messages can be used in each of the specific ECM Reference Sub-Processes and can be applied to a huge range of data objects defined in the specific parts of this recommendation.

The administrative ECM Messages support communication of the values permitted for one or more attributes of one ore more selected ECM Reference Sub-Process-specific Data Model objects.

Request_permitted_values

This message from the *participant* asks the *coordinator* for the values permitted for one or more attributes of one or more objects of an ECM Reference Sub-Process-specific Data Model.

Parameters:

- $Class_n$: the name of the ECM Reference Sub-Process-specific Data Model object n [optional]
- $Class\_attribute_{n,m}$: the name of the attribute m of the ECM Reference Sub-Process-specific Data Model object n [optional]

If no parameter is specified in this message, the semantics are defined as follows: The *coordinator* must send back a list of all permitted values (see note) of all data objects according to the ECM Reference Sub-Process-specific Data Model.

Note: *The decision as to what "a list of all permitted values" and "all data objects according to the Data Model" mean is implementation specific.*

Respond_permitted_values

This message from the *coordinator* is the response to the *Request_permitted_values* message. It contains a list of all the values permitted for the attributes specified in the *Request_permitted_values* message.

Parameters:

- $Class_n$: the name of the ECM Reference Sub-Process-specific Data Model object n [optional]

- $Class\_attribute_{n,m}$: the name of the attribute m of the ECM Reference Sub-Process-specific Data Model object n [optional]
- $Property\_value\_representation_{n,m,k}$: the permitted value k of the attribute m of the ECM Reference Sub-Process-specific Data Model object n [optional]

Notes: *If the permitted values for an attribute is a non-restrictive list (i.e. the communicated values are permitted, but other values are also permitted), exactly one Property_-value,m,k object exists for the corresponding Property_value_representationn,m,k object that contains a value of '…' in its attribute .value_specification. Otherwise (i.e., if no other values except the permitted ones are allowed), no such Property_valuen,m,k object exists for the corresponding Property_value_representationn,m,k object.*

If the coordinator does not provide information on the values permitted for an attribute, no Property_value_representationn,m,k objects (and therefore also no Property_valuen,m,k objects) are referenced by the corresponding Class_attributen,m object.

7.1.2 Acknowledgement, Warning, and Error Messages

This section describes the ECM messages for signalling errors, warnings and acknowledgement messages that may occur during cross-partner ECM communication.

Notify_Acknowledgement

This optional message from the recipient signals the recipt of any ECM Message. See Appendix G for details.

Parameters:

- identifier of the previously received ECM Message [mandatory]

Notify_Warning

This message from the recipient signals a waring during the processing of the received ECM message at the recipients side. See Appendix G for details.

Parameters:

- identifier of the previously received ECM Message that causes the warining [mandatory]

Notify_Error

This message from the recipient signals an error during the processing of the received ECM message at the recipients side. See Appendix G for details.

Parameters:

- identifier of the previously received ECM Message that causes the error [mandatory]

7.1.3 ECM Message header and control information

To enable communication between partners, any implementation must fulfil the following requirements by providing control information using the message header format specified below:

1. The organization of the sender and of the designated receiver needs to be uniquely identified globally or at least between coordinator and participant.
2. Every message exchanged between coordinator and participant must be uniquely identifiable on a technical level, i.e. on the level of ECM context information such as the message name, ECM Interaction Scenario etc. This is needed in order to be able to determine the significance of the message in the context of the ECM Reference Sub-Process and in the application context determined by the ECM Interaction Scenario.
3. It must be possible to uniquely assign every respond message to the corresponding request message in order to identify the request data to which the response data refers.
4. It must be possible to assign every message uniquely to an ECM Interaction Scenario flow. This ensures, for example, that it is possible to simultaneously exchange information about different change requests with the same participant even using the same ECM Interaction Scenario. At the same time, the assignment of the message to the context of preceding messages relating to a change is retained. In particular, support must be provided to allow for a change of the responsibility for defining an ID called InteractionId, which facilitates this assignment to be changed during an ECM Interaction Scenario flow.
5. It must be possible to assign messages to subdialogs of an ECM Interaction Scenario flow. During an ECM Interaction Scenario flow, there may be subdialogs that may be needed to be processed separately. For example, in the ECR Reference Process in Commenting on ECR, different kinds of comments need to be exchanged in concurrent sub-flows between one coordinator and one participant. Therefore, it is necessary to be able to identify which subdialog a message belongs to.
6. It must be possible to reconstruct the sequence of messages for each sender in an ECM Interaction Scenario flow. This is needed with respect both to the order in which the messages were sent and to their completeness. This is intended to ensure that communication errors such as incorrect sequences or lost messages can be detected.

To promote clarity, the data fields used to implement these requirements should be separate from the user data in each message. Thus, the data fields listed below are expected as part of the message header.

Table 1: Message header data for messages of the types request or notify:

Data object	Description (types/format, sample value, meaning)
ORGANIZATIONIDTYPE	Field whose value specifies the method used to identify the sending organization in the field *ORGANIZATIONID.* ***Type:*** String; possible values: “DUNS_NUMBER”, “ORGANIZATION_NAME”, “SUPPLIER_IDENTIFIER”, “DNS_DOMAIN”, “ODETTE_PID”; with the following meaning: ▪ “DUNS_NUMBER” – Use of a DUNS number, which is a globally unique

	number code recognized as a global standard for company identification (DUNS: Data Universal Numbering System) (see VDA 5006). ▪"ORGANIZATION_NAME" - Use of an organization name, i.e. the name of the organization. ▪"SUPPLIER_IDENTIFIER" – Use of an identifier/number of the supplier, usually only unique to the customer. If SUPPLIER_IDENTIFIER is used to identify a supplier, it is recommended that the customer be identified using an organization name. ▪"DNS_DOMAIN" - Use of a domain name managed by the global internet domain naming service (DNS) (see RFC 1034). ▪"ODETTE_PID" – Use of an Universal Party Identifier (UPI), which is a globally unique number code defined by Odette (see ODETTE PID). The use of one type of identification only is recommended. The use of the DUNS number in particular is recommended. It is not recommended that the type of identification be changed during an ECM IS dialog.
ORGANIZATIONID	Field whose value uniquely identifies the organization sending this message in the context of communication between *coordinator* and *participant*. ***Type:*** String; type and format of this field depends on the method used to identify the sending organization (see field OrganizationId-Type): ▪"DUNS_NUMBER": ▪Type: Integer, format: 9 digits; example: "315744388". ▪"ORGANIZATION_NAME": ▪Type: String; format: alphanumeric; example: "Verband der Automobilindustrie". ▪"SUPPLIER_IDENTIFIER": ▪Type: String; format: arbitrary; example: "12345678". ▪"DNS_DOMAIN" : ▪Format: string (for details, see [RFC 1034]); example: "vda.de" ▪"ODETTE_PID": ▪Type: String; format: 19 characters; example: "006012345123400000". ***Note:*** The value of *ORGANIZATIONID* should not contain a "/" because it is used in the *INTERACTIONID*.
INTERACTIONID	***Type:*** String; For process-oriented messages: format*: ORGANIZATIONID "/" IDSTRING1 "/" SUBDIALOGNO;* example: "717743322/AEA553421/1" *IDSTRING1:* ***Type:*** String; format/value: generally defined by the ECM Reference Sub-Process-specific recommendation (e.g. an ECR_id (see Part 1 - ECR)) *SUBDIALOGNO:* Type: integer; values: 1, 2, … **Note:** *The* IDSTRING *may contain the "/" character.* For data-oriented messages: format: *ORGANIZATIONID "/" IDSTRING2* *IDSTRING2:* Type: String; value: not further specified. Unique identifier defined by the Organization issuing the *request* message. Together with the *ORGANIZATIONID* of the *coordinator* and the *participant*, the *INTERACTIONID* (and, in the same way, the NewInteractionId) uniquely and globally identifies the assignment of the message to a current interaction dialog, i.e. a sequence of messages belonging to an ECM Interaction Scenario that is currently being processed, and a possible subdialog. As a result of this requirement that assignment should be unique on the side of both the sender and the recipient, and because of the pos-

	sibility of changing this *INTERACTIONID* during the course of the interaction, it should be possible to uniquely assign the *INTERACTIONID* to the party issuing the *INTERACTIONID*; this party must also ensure that the *INTERACTIONID* is unique within his company. 1. If this is the first message in the interaction dialog, the following applies: ➔ The sender assigns a new *INTERACTIONID.* 2. If this is not the first message in the interaction dialog, the following applies: a) If *NEWINTERACTIONID* = "" in the previous message, the following applies: *INTERACTIONID* in the current message must have the same value as *INTERACTIONID* from the most recent message of this interaction dialog. b) If *NEWINTERACTIONID* <> "" in the previous message, the following applies: ➔ *INTERACTIONID* in the current message must have the value of the *NEWINTERACTIONID* from the previous message ➔ The *SUBDIALOGNO* should have the value “1”, if no subdialog is active. After completion of a subdialog, *SUBDIALOGNO* should have again the value “1” in every message. ➔ For data-oriented messages, no value should be provided here.
NEWINTERACTIONID	***Type/format/example:*** as for *INTERACTIONID* Contains the value to be used as the new *INTERACTIONID* as of the following message. ➔ If the value is "" (empty string), this indicates that the existing *INTERACTIONID* is retained or that this is the first message in the interaction. ➔ If the value <> "", *INTERACTIONID* contains the currently valid *INTERACTIONID* and *NEWINTERACTIONID* contains a new valid *INTERACTIONID* which uniquely identifies an interaction with a partner within the organization of the sender of the *request* or *notify* message. This new value must be used as the *INTERACTIONID* in the subsequent messages of this interaction (until this is again changed, if applicable). The possibility of changing the *INTERACTIONID* is specified for each ECM Interaction Scenario. For data-oriented messages, no value should be provided here.
INTERACTIONSCENARIONAME	***Type:*** String; values: defined by the ECM Reference Sub-Process-specific recommendation; example: “IS1” (see Part 1 - ECR) As of the first message, the *INTERACTIONSCENARIONAME* specifies which ECM Interaction Scenario forms the basis for the subsequent dialog. For data-oriented messages, no value should be provided here.
MESSAGENAME	***Type:*** String; values: defined by the ECM Reference Sub-Process-specific recommendation; example: "Request_ECR_details" (see Part 1 - ECR) Contains the name of this message.
SEQUENCENO	***Type:*** Integer; values: 1, 2, … Sequential number specifying what number message from the sender this is in the current ECM Interaction Scenario dialog. This means that *SEQUENCENO* starts at "1" with reference to the dialog identified by

	INTERACTIONID / NEWINTERACTIONID for the sender defined by *ORGANIZATIONID*. This holds also if another subdialog is started. *SEQUENCENO* must be incremented (even in the event of rollbacks) as soon as a sender sends a further message during the course of an ECM Interaction Scenario dialog. For data-oriented messages, no value should be provided here.
ISCOMPLETE	***Type:*** Boolean; values: TRUE, FALSE The value FALSE identifies the message as a preliminary status, i.e. an interim status which may not be complete, which has been sent from the sender to the recipient for information purposes and which will be followed (possibly after further preliminary statuses) by a further message containing the final status (see the flow variant for exchanging preliminary statuses described in Part 1 - ECR, section 2.4.3). If the object is part of a *request* message, it is not generally necessary to respond to this message with a *respond* message. If the recipient wishes to respond, he can (only) do this with one or more *respond* messages, each of which are flagged as IsComplete = FALSE. It is only necessary to send a *respond* message in response to a *request* message where IsComplete = TRUE. In this case, *respond* messages which are flagged as preliminary statuses may follow, but at the end, it is mandatory that a *respond* message where IsComplete = TRUE is sent. For data-oriented messages, no value should be provided here.
KINDOFSUBDIALOG	***Type:*** String; values: “” (empty string) if no subdialogs are used, additional values may be defined by Part 1 - ECR, etc.; example: “COMMERCIAL_COMMENTS” (see Part 1 - ECR). Information field indicating the type of interaction in the subdialog. For example, according to Part 1 - ECR, two subdialogs may be executed and indicated by the values “1” and “2” respectively for *SUBDIALOGNO* in the *INTERACTIONID*. *KINDOFSUBDIALOG* may contain the values “TECHNICAL_COMMENTS” and “COMMERCIAL_COMMENTS” respectively in the subdialogs (see Part 1 - ECR). For data-oriented messages, no value should be provided here.

Table 2: Message header data for messages of the type respond:

Data object	Description
ORGANIZATIONIDTYPE	As for messages of the type *request* or *notify*
ORGANIZATIONID	As for messages of the type *request* or *notify*
INTERACTIONID	As for messages of the type *request* or *notify*
NEWINTERACTIONID	As for messages of the type *request* or *notify*
INTERACTIONSCENARIONAME	As for messages of the type *request* or *notify*
MESSAGENAME	As for messages of the type *request* or *notify*
SEQUENCENO	As for messages of the type *request* or *notify*
ISCOMPLETE	As for messages of the type *request* or *notify*
REQUESTOR_ORGANIZATIONIDTYPE	Copy of the value of *ORGANIZATIONIDTYPE* in the corresponding *request* message
REQUESTOR_ORGANIZATIONID	Copy of the value of *ORGANIZATIONID* in the corresponding *request* message
REQUESTOR_INTERACTIONSCENARIONAME	Copy of the value of *INTERACTIONSCENARIONAME* in the corresponding *request* message
REQUESTOR_MESSAGENAME	Copy of the value of *MESSAGENAME* in the corresponding *request* message

REQUESTOR_SEQUENCENO	Copy of the value of *SEQUENCENO* in the corresponding *request* message
REQUESTOR_ISCOMPLETE	Copy of the value of *ISCOMPLETE* in the corresponding *request* message
REQUESTOR_KINDOFSUBDIALOG	Copy of the value of *KINDOFSUBDIALOG* in the corresponding *request* message

Respond messages must also include a reference to the corresponding *request* message to which the *respond* message refers. This is done by repeating the values *ORGANIZATIONIDTYPE*, *ORGANIZATIONID*, *INTERACTIONSCENARIONAME*, *MESSAGENAME*, *SEQUENCENO*, *ISCOMPLETE* from the *request* message in the corresponding data objects *REQUESTOR_ORGANIZATIONIDTYPE*, …, *REQUESTOR_ISCOMPLETE*.

Figure 10 shows an example illustrating the use of the message header data in a permitted sequence of messages in a dialog as per ECR Interaction Scenario 4.

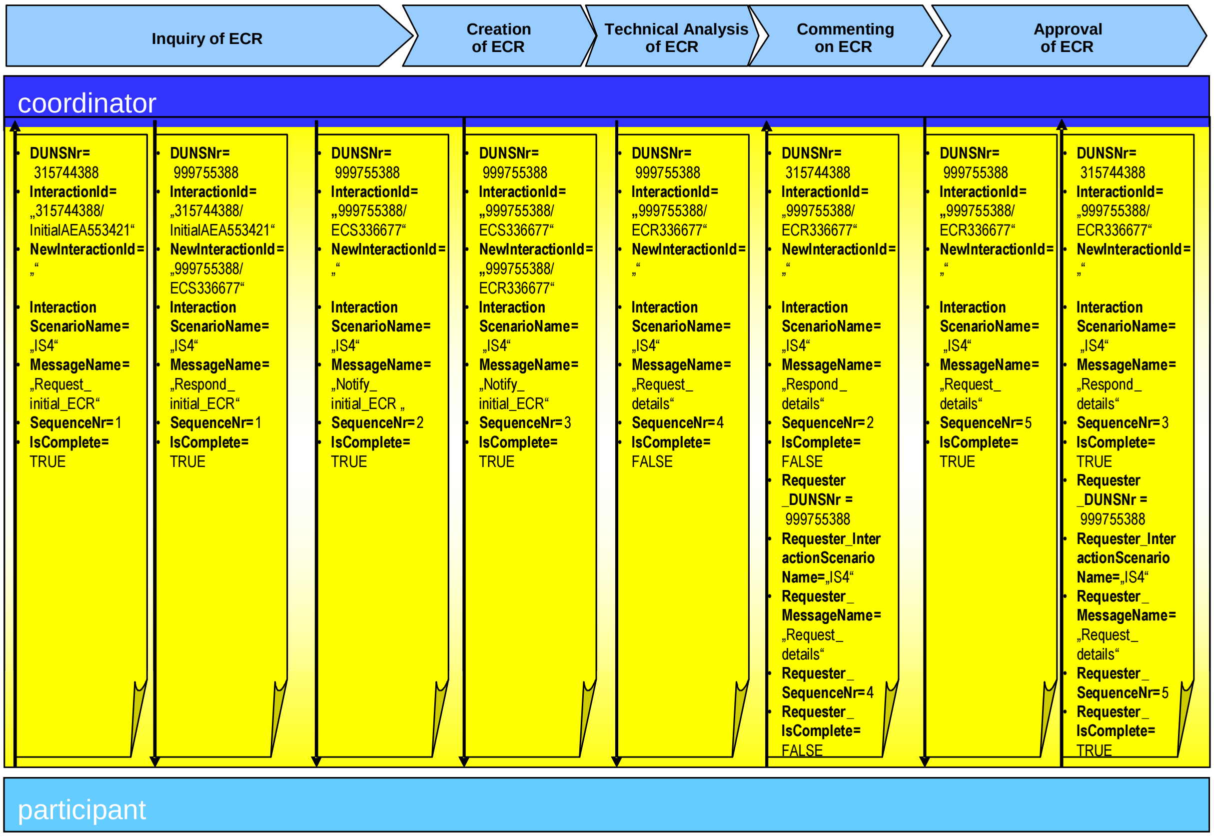


Figure 10: Example illustrating the use of the message header data.

7.2 ECM Data Model

This section provides an introduction to the data model,in the context of ECM which reflects the requirements of the ECM Reference Sub-Processes.

The ECM Data Model supports the description of resource constructs for objects and their properties. The following fall into the scope of this section:

- the definition of header information and details of an change object

- the identification of organizational data (people, organizations, dates, times and scheduled tasks)
- the definition of scopes as the context of engineering changes
- the definition of objects affected by engineering changes
- the description of comments if available
- the provision of a mechanism for generic class definitions
- the description of acceptance and status information of Sub-Processes
- the definition of specific (costs and mass) and generic properties including a mechanism for referring to information objects
- the references to attachmens

Appendix E specifies the ECM Data Model by EXPRESS-G representations (overview of the defined data types, their properties and relationships) as well as by a detailed specification of all data types (data dictionary).

Appendix A: ECM Process and Protocol Notation

Overview of the symbols used in the phase diagrams.

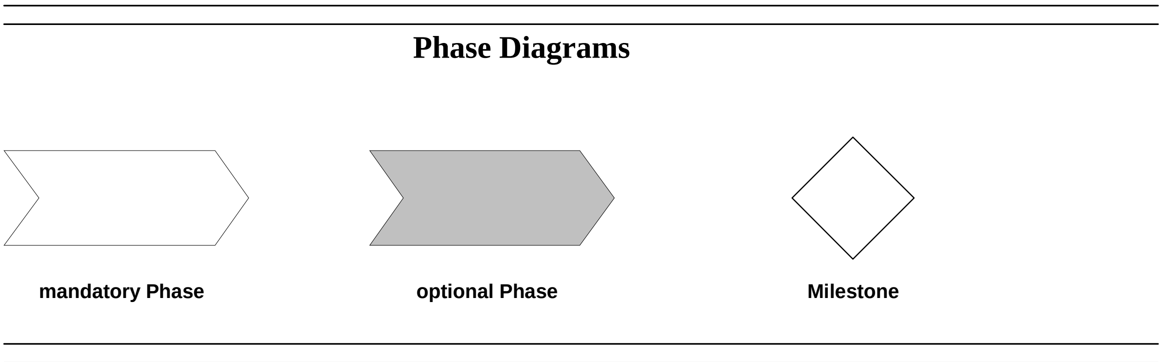


Figure 11: Symbols used in the phase diagrams

The following provides an overview of the notation used in UML 2 activity diagrams and state machine diagrams[8] .

[8] Object Management Group. UML Superstructure Specification, v2.0. Document -- formal/05-07-04, http://www.omg.org/.

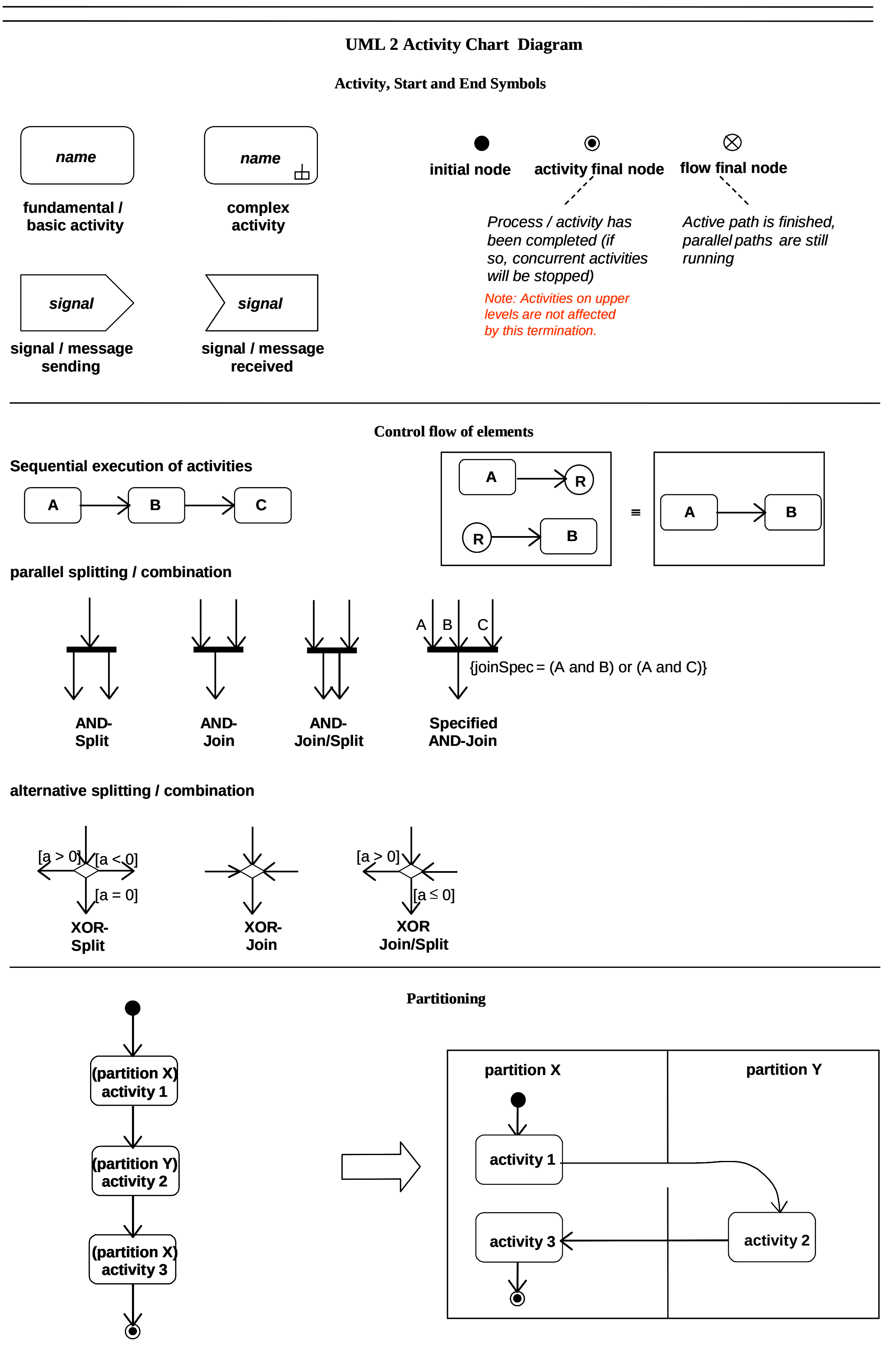


Figure 12: Notation used in UML 2 activity diagrams and state machine diagrams (1/4)

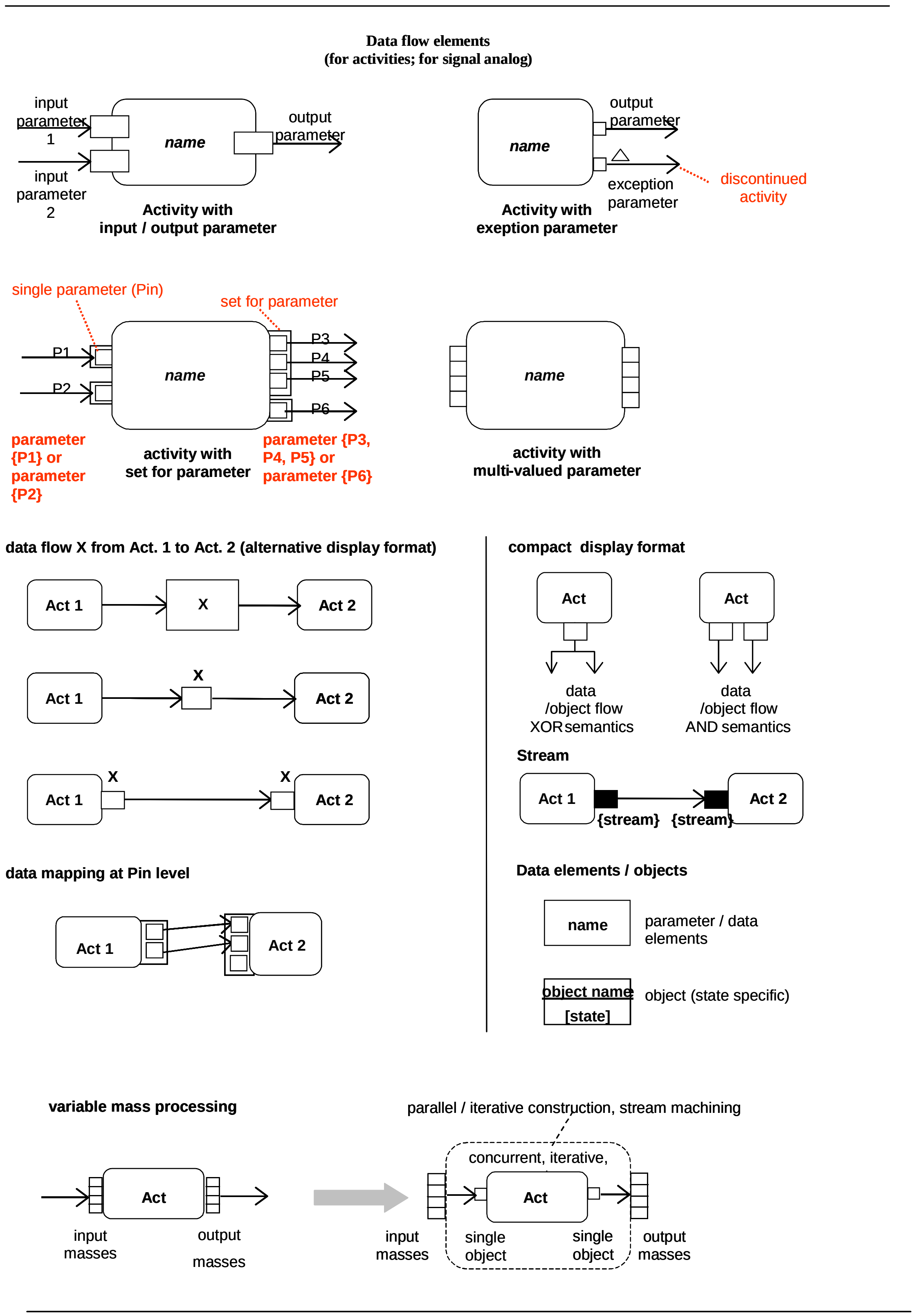


Figure 13: Notation used in UML 2 activity diagrams and state machine diagrams (2/4)

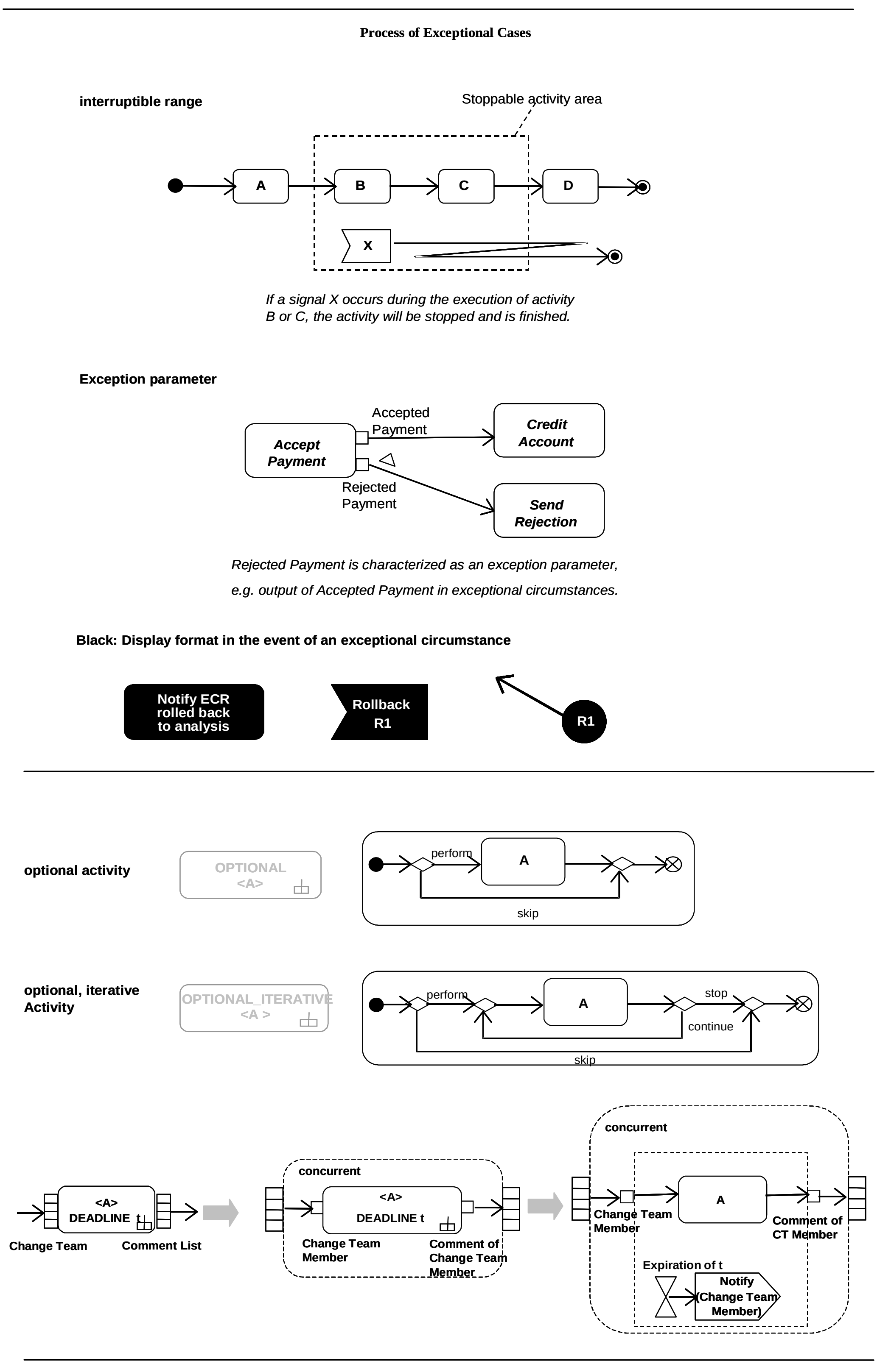
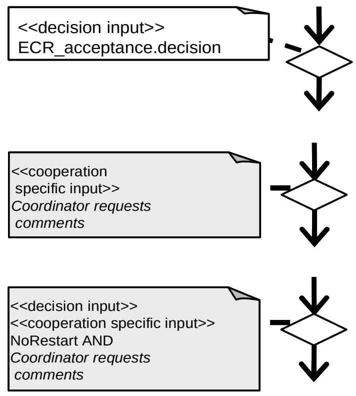


Figure 14: Notation used in UML 2 activity diagrams and state machine diagrams (3/4)

Annotations



Branch decision depending on parameters of the process

Branch decision depending on cooperation-specific regulations (coop.-specific input parameters in *italics*)

Branch decision depending on parameters of the process and on cooperation-specific regulations (coop.-specific input parameters in *italics*)

UML 2 State machine Diagram

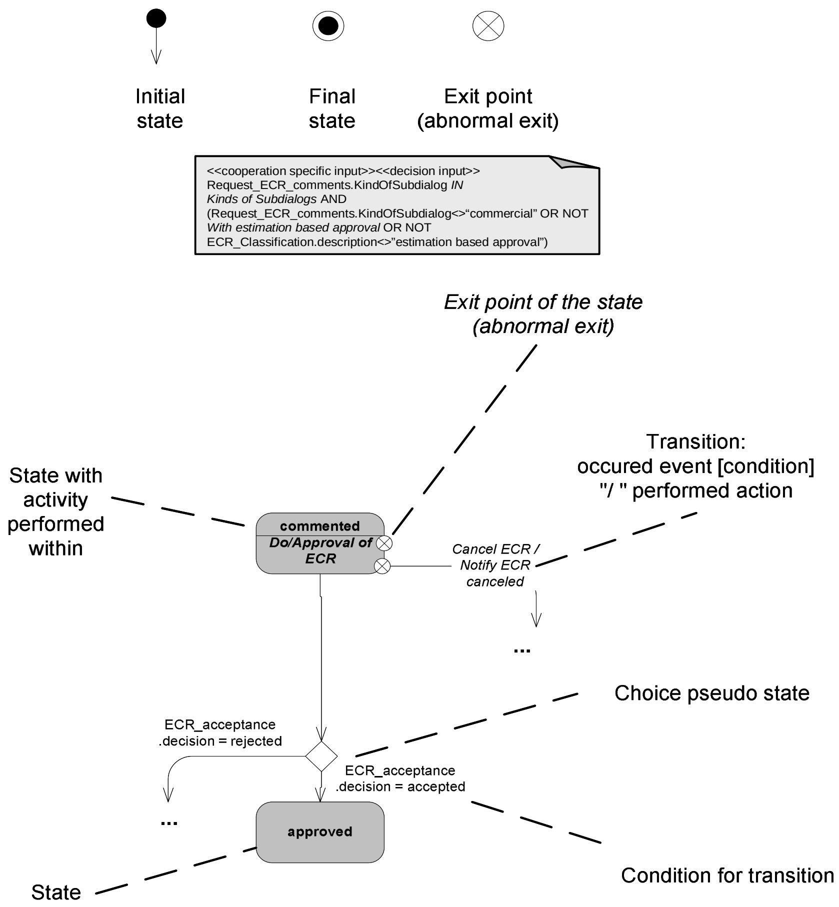


Figure 15: Notation used in UML 2 activity diagrams and state machine diagrams (4/4)

Appendix B: Data Model Notation

B.1 ECM Data Model notation

The data models are specified in EXPRESS-G. The international standard ISO 10303-11 defines EXPRESS as a formal language for specifying data models used to describe product information. Use of a formal language provides a clear and consistent representation and facilitates implementation. EXPRESS allows data and the associated constraints to be described. EXPRESS-G is the graphical representation of EXPRESS and is defined in Appendix D of ISO 10303-11.

This Appendix provides a brief introduction to EXPRESS-G. It is reduced to those graphical elements that are used in the specifications of the common and specific ECM data models of this recommendation series. The complete specification of EXPRESS can be found in ISO 10303-11.

B.2 Introduction to EXPRESS-G

The following tables give a brief description of the elements used in the diagrams of the ECM Data Model. The complete description of the notation of the information modelling language EXPRESS and its graphical representation EXPRESS-G is included in ISO 10303-11.

Table 3: Description of the elements used in the Data Model diagrams (1/3)

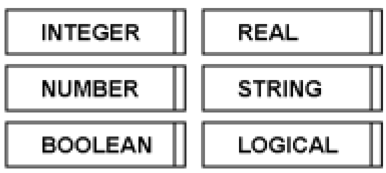
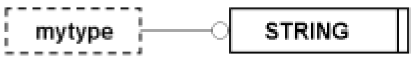
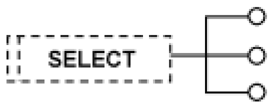
Graphical elements	Description
	***Simple Data Type:*** Simple Data Types are INTEGER, REAL, NUMBER (for INTEGER or REAL), STRING, BOOLEAN (true or false) or LOGICAL (true, false or unknown) ***Note***: Simple Data Types are not used in the diagrams for reasons of clearness. They are only used in paragraph 3.
	***Defined Type:*** Defined Types are types defined by the user.
	***Select Data Type:*** The Data Type SELECT is an agreement about a sample of Data Types
	***Set Data Type:*** The Aggregation Data Type SET is an unsorted amount of elements of the same Data Type with specifications about the cardinality. The specifications mean S[0:?] – 0 .. n ; S[1:?] – 1 .. n ; S[2:3] – 2 or 3.

Table 4: Description of the elements used in the Data Model diagrams (2/3)

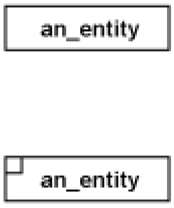
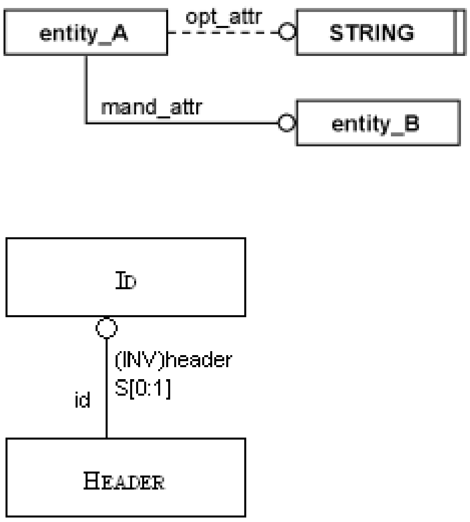
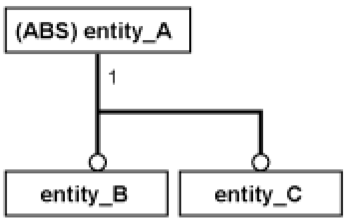
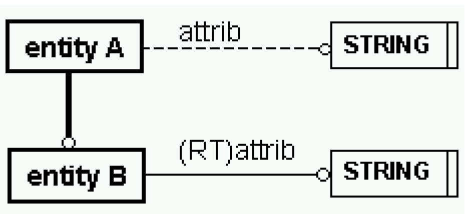
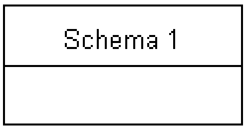
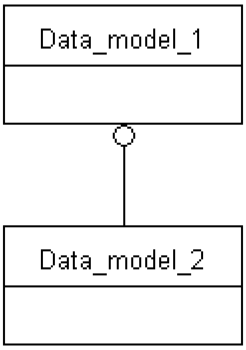
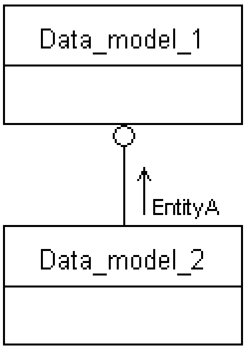
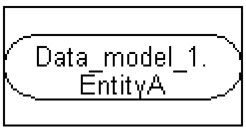
Graphical elements	Description
	***Entity Data Type:*** Describes objects of the “real” world by attributes and relations. EXPRESS uses the term Entity to specify Data Types. Different values of a Data Type are called instances. A small square at the upper left corner indicates an entity which is specified at another location in the diagram.
	***Entity data type - attributes:*** The attributes of a Data Type represent characteristics of the described object. ***Explicit Attributes*** can be mandatory or optional. They reference simple or aggregate Data Types. ***Inverse Attributes*** are suitable for specifying an attribute in an entity that refers to an attribute of another entity, which in turn defines a reference to itself. This is normally used to regiment relations between both entities. Example: Every instance of HEADER references exactly one instance of ID. But every instance of ID is referenced by none or exactly one HEADER.
	***Subtype Supertype:*** An Entity (B) can be specified as a Subtype of another Entity (A). This creates a relation so that the specified Subtype inherits all attributes from its Supertype. ***Supertype Specifications:*** A Supertype which has been declared an ABSTRACT Supertype (symbolized by ’(ABS)’) is, on its own, insufficiently specified. It needs to be specified by at least one Subtype. If there are more entities specified as Subtypes of a Supertype, the specification ONEOF (symbolized by ’1’) determines that only one instance of a Subtype and no combinations of Subtypes are valid.
	***Redefined attributes:*** Inherited attributes of an entity may be specialized, this is indicated by the prefix '(RT)'. ***Example:*** The data type B is a specialization of data type A and inherits the optional attribute 'attrib'. But this attribute is mandatory in the context of data type B. This kind of specialization is indicated by the line types used (optional and mandatory) and by the prefix '(RT)' for the redefined attribute 'attrib'.

Table 5: Description of the elements used in the Data Model diagrams (3/3)

Graphical elements	Description
	***Schema*** A schema declaration defines a common scope for a collection of related entity and other data type declarations. A schema defines a name space for all data types defined within it.
	***Interface specification*** All data type specifications declared in schema. Data_model_1 are made usable within the schema Data_model_2 by way of a use specification. If there are no named_types specified, all of the named data types declared within the schema Data_model_1 are treated as if declared locally in the schema Data_model_2.
	***Explicit interface specification*** The entity data type Entity A declared in schema Data_model_1 is made usable within the schema Data_model_2 by way of a use specification.
	***Inter-Schema reference*** The entity data type Entity A declared in schema Data_model_1 can be used as attribute type in a data type declaration.

Appendix C: IT-Architecture

C.1 Overview

This Appendix is intended to provide an overview of possible use cases and implementations with regard to this recommendation. Pilot implementations of this recommendation involving different types of integration have been performed by BMW and MAGNA STEYR Fahrzeugtechnik, as well DaimlerChrysler and Delphi. These pilots have shown that different implementation alternatives can be used within a single collaboration framework without conflict.

The following section provides an introduction to the reference architecture for ECM applications. This is followed by an explanation of two general design decisions relating to integration which are based on this architecture. Other implementation alternatives are then discussed, followed by four methods of integration including examples.

C.2 Reference architecture for ECM integration

This section provides a rough outline of the architecture used in ECM applications (see Figure 16). This reference architecture provides the basis for the methods of integration used and is also used to illustrate various options for integrating ECM applications.

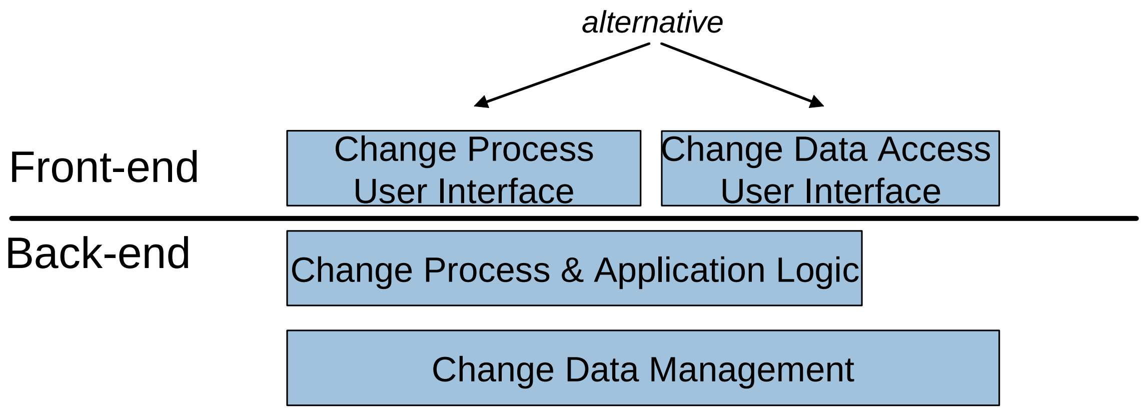


Figure 16: Reference architecture for integrated Engineering Change Management

In an implementation of an ECM application, a distinction is made between front-end and back-end components. Front-end components are user-oriented, client-side components that offer users access to the ECM application, including the user interface, etc. Back-end components include the server-side components for the application, such as data and state support for ECRs, etc.

Another distinction can be made with regard to the functional scope. Data management components are responsible for access to and administration of change data. Process-related components and application logic components include ECM process-specific functionality for the administration and control of engineering change management processes, including the corresponding applications necessary for processing the activities (e.g. evaluation of an amendment of an ECR etc.).

Therefore, a distinction is made between the following components:

- A Change Process User Interface facilitates access to the ECM process, i.e. to activity processing, process status query, etc. This component usually also provides the functionality of a Change Data Access User Interface and can therefore be considered to be an alternative to or enhancement of a Change Data Access User Interface.
- A Change Data Access User Interface only allows access to the change data (typically read access since write access is predominantly process dependant).
- Change Process & Application Logic facilitates the administration and control of the ECM process (including the corresponding applications).
- Change Data Management is responsible storing and managing the change data.

A clear line cannot always be drawn between the different components although making this distinction facilitates the following descriptions with regard to the types of integration that are possible in an ECM environment.

C.3 General decisions regarding integration

In principle, a distinction can be made between different types of integration involving ECM applications:

- Process-oriented or data-oriented integration: In the case of data-oriented integration, change data (such as, for example, an ECR) is exchanged or made accessible. This allows a partner to obtain information about current or completed ECRs and, for example, view the comments on a change request made by the partner in good time or understand the reasons behind changed assembly versions by calling up the underlying change requests. In the case of a standardized solution, the ECM Data Model in particular is relevant as a source.
  In the case of process-oriented integration, the integration solution means that partner A can, depending on the status, receive activities for processing (such as, for example, Commenting on ECR) from the change process of partner B and that the results from B can be transferred automatically to the ECM process of A. Functionality for data-oriented integration is typically included in process-oriented integration. If a standardized solution is used, not only the data model but also the ECM Interaction Scenarios and messages are relevant.
  Please note that process-oriented integration and data-oriented integration can be used simultaneously or can be combined. For example, data-oriented integration can be used to provide an overview of ECRs and their status while process-oriented integration can be used to respond to an open request for one of these ECRs.
- Front-end or back-end integration: In the case of front-end integration, a front-end is made available to an external partner to enable this partner to access the ECM system. In the case of back-end integration, back-end components of partner A are integrated with the back-end components of partner B.
- This allows the types of integration described in section 0 to be derived. In order to explain these using more concrete examples, basic implementation alternatives can be realized for a number of different types of integration. These implementation alternatives are described in the following section.

C.4 Implementation alternatives for integration solutions

In an integration context, a distinction can be made between synchronous and asynchronous communication. In the case of synchronous communication, the integrated components (front-end/back-end or back-end/back-end) are online concurrently and communicate with blocking (i.e., they wait until a message/data has reached the partner). This communication is typically stream-based; an example of this type of communication is synchronous Web service communication.

In the case of asynchronous communication, the integrated components (back-end/back-end or front-end/back-end) can continue operating after transmission, i.e. the messages must be buffered within the communication channel (e.g. in the middleware or at the receiver's end). This means that the integrated components do not have to be online or connected with each other at the same time (i.e. they can also communicate offline). File-based data exchange is an example of offline communication.

Other design-related decisions to be made when implementing integration solutions are the protocols to be used to exchange files or messages. The following list provides a few important examples of standards and references:

- ENGDAT[9] (ENGineering DATa Message): package file format for forwarding product data as package files including information about sender, receiver and a description of each physical file within a package.
- OFTP (Odette file transfer protocol): standardized protocol for data exchange[10].
- STEP Part 21 (ISO 10303-21): plain text presentation of product data as sequential file[11]
- STEP Part 28: (ISO 10303-28): XML representation of product data
- Web Services

In addition, the OMG standard PLM Services V2.0 (http://www.omg.org/spec/PLM) should be mentioned in connection with STEP application integration environments. This standard defines a platform-independent format and protocol for data and application logic for synchronous and asynchronous access to PLM-relevant data based on the information model defined in the 3rd edition of ISO 10303 STEP AP214 "Core Data for Automotive Mechanical Design Processes" (http://www.iso.org). A platform-specific implementation for Web Services is also specified in the OMG PLM Services standard. It represents an important and sustainable implementation basis for ECM.

C.5 Types of integration

The following sections provide a detailed description of the types of integration that result from the previously-mentioned design-related decisions regarding integration.

[9] VDA recommendation 4951, based on Odette ENGDAT Version 2.

[10] see VDA4914/2

[11] VDA Recommendation "Product Data Exchange", VDA 4956, November 2002. http://www.vda.de/.

C.5.1 Back-end data integration

In the case of back-end data integration, data is exchanged between the back-end systems (see Figure 17).

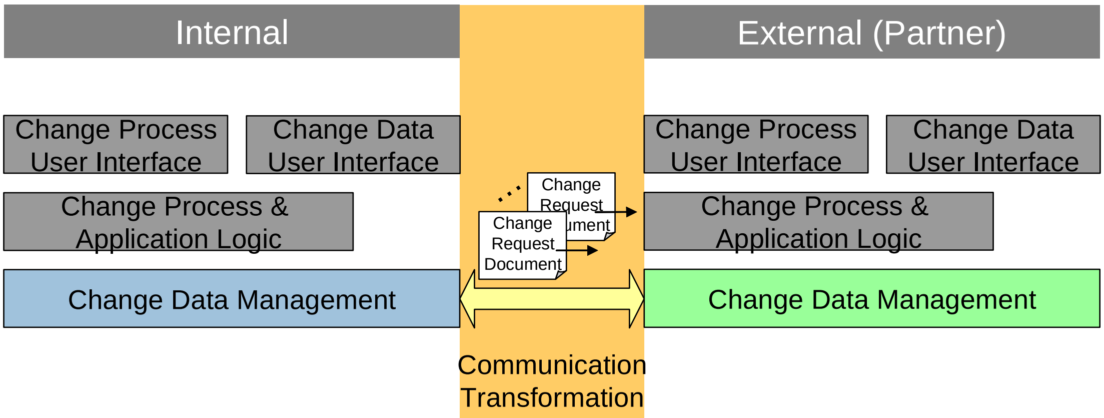


Figure 17: Back-end data integration

C.5.2 Back-end process integration

In the case of back-end process integration, private ECM processes exchange messages with each other based on a standardized process and data model (see Figure 18). This type of integration has been deployed in the pilot implementation realized by BMW and MAGNA STEYR Fahrzeugtechnik.

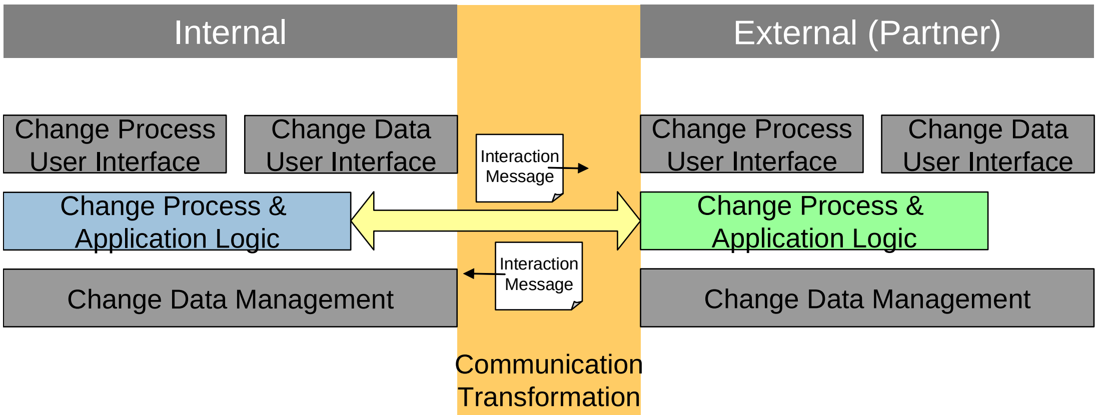


Figure 18: Back-end process integration

C.5.3 Front-end data integration

In the case of front-end data integration, one partner is authorized to access change data (this is typically read access) (see Figure 19). This type of integration provides the basis for the pilot implementation at DaimlerChrysler and Delphi, where one partner can access the change data of another partner online via a neutral client based on a standardized interface such as PLM Services. One variant of this would be a Smart Tool for viewing change data such as, for example, an ECR (see next section).

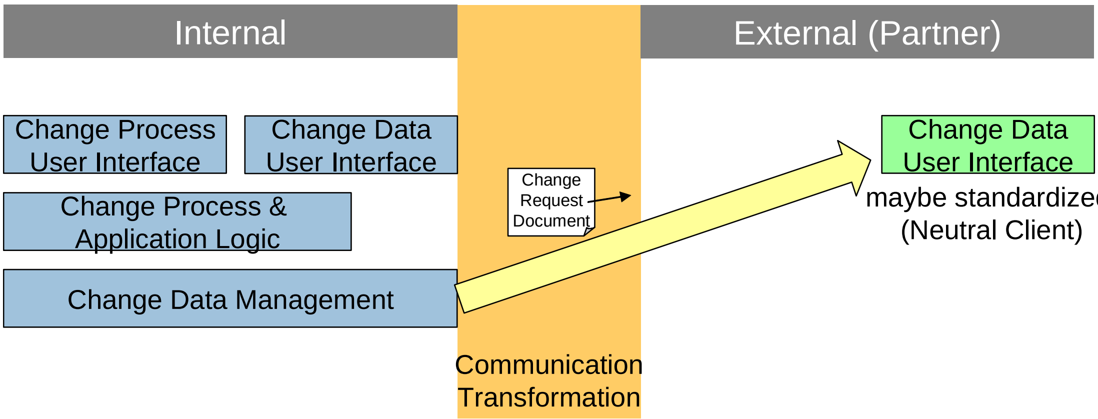


Figure 19: Front-end data integration

C.5.4 Front-end process integration

In the case of front-end process integration, one partner has access to process and application data (see Figure 20) without any need for an ECM back-end.

One example of this would be a neutral process client that allows a partner to access the back-end online and thus, for example, analyze and comment on ECRs for which he has been requested to do so.

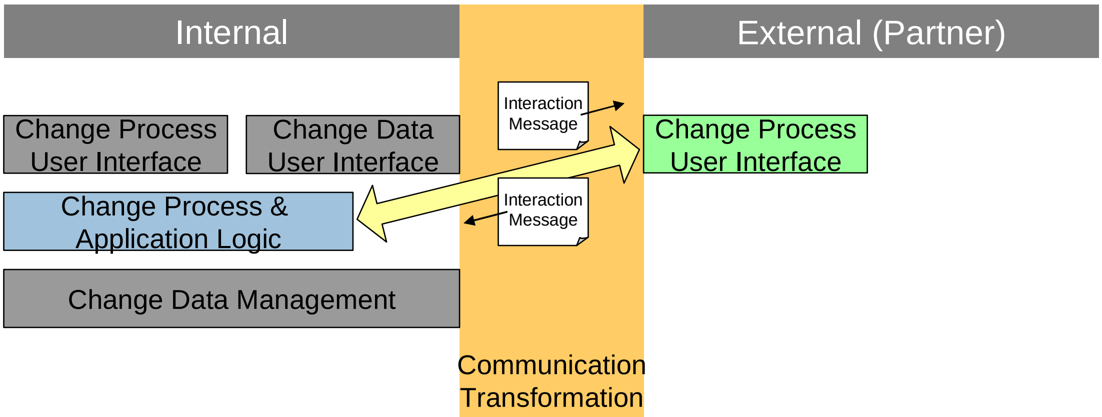


Figure 20: Front-end process integration

Appendix D: Description of Common Messages

As mentioned in Section 6.6.1, this appendix provides a more detailed description of the administrative ECM Messages.

D.1 Request_permissive_values

The general format of the parameters for this ECM Message is as follows:

[$\mathtt{Class}_1$, $\mathtt{Class\_attribute}_{1,1}$, … $\mathtt{Class\_attribute}_{1,k}$, … $\mathtt{Class}_2$, $\mathtt{Class\_attribute}_{2,1}$, … $\mathtt{Class\_attribute}_{2,l}$, , … $\mathtt{Class}_n$, $\mathtt{Class\_attribute}_{n,1}$, … $\mathtt{Class\_attribute}_{n,m}$]

Each *Class*$_x$ object must include the name of the object type of the ECM Data Model in its attribute *.id.* Each *Class_attribute*$_{x,y}$ object must reference its *Class*$_x$ object and must include in its attribute *.id* the name of the attribute of the object type of the ECM Data Model for which the *participant* wants the permitted values.

In addition, the *.definition* attribute *for* each *Class_attribute*$_{x,y}$ object must refer to a separate *Property*$_{x,y}$ object with a value of '/NULL' in its *.property_type* and *.id* attribute.

Example: Assume the *participant* issues a request for the values permitted for the attributes *.id* and *.name* for all the *Series* currently defined on the coordinator side. According to this example, the signature of this ECM Message is as follows:

```
Request_permissive_values("Series", ".id", ".name")
```

The parameters of this ECM Message are instantiated as follows:

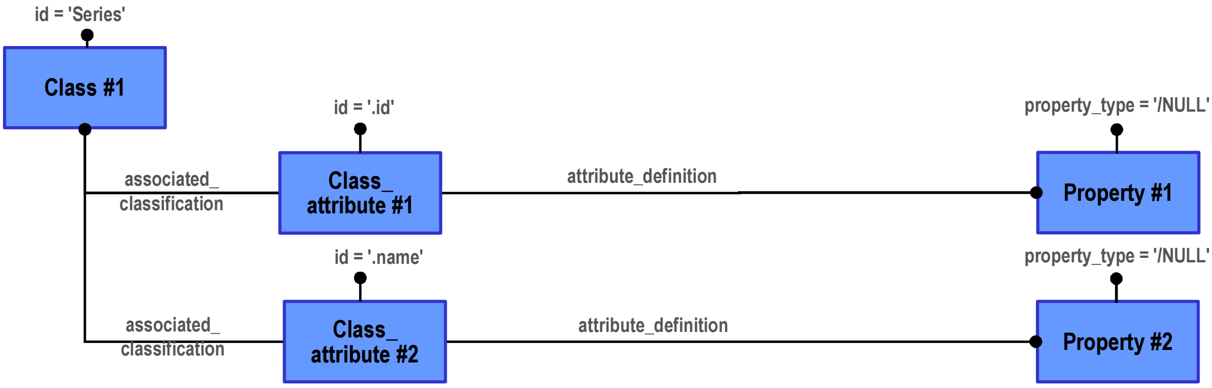


Figure 21: Sample instantiation of the Request_permissive_values ECM Message

If no parameter is specified in this ECM Message, the semantics are defined as following: The *coordinator* must return a list of all the values permitted for all the data objects in accordance with the ECM Data Model.

D.2 Respond_permissive_values

The general format of the parameters for this ECM Message is as follows:

[$\mathtt{Class}_1$,

$\mathtt{Class\_attribute}_{1,1}$,

```
Property_value_representation1,1,1, … Property_value_representation1,1,p, …
Class_attribute1,k, …
Class2,
Class_attribute2,1, … Class_attribute2,l, , …
Classn,
Class_attributen,1, … Class_attributen,m, …
Property_value_representationn,m,q]
```

Each *Class*$_x$ object must include the name of the object type of the ECM Data Model in its attribute *.id.* Each *Class_attribute*$_{x,y}$ object must reference its *Class*$_x$ object and must include in its attribute *.id* the name of the attribute of the object type of the ECM Data Model for which the *participant* wants the permitted values.

The *.definition* attribute for each *Class_attribute*$_{x,y}$ object must refer to a separate *Property*$_{x,y}$ object with a value of '/NULL' in its *.id* attribute. The *.property_type* attribute in this ECM Message may contain the sender-specific nomenclature of the field instead of the value '/NULL'.

The*.allowed_values[i]* attribute for each *Class_attribute*$_{x,y}$ object must refer to a separate *Property_value_representation*$_{x,y,z}$ object. Each *Property_value_representation*$_{x,y,z}$ object must refer to the *Property*$_{x,y}$ object that is also referenced by the *Class_attribute*$_{x,y}$ object in its *.attribute_definition* attribute. In addition, each *Property_value_representation*$_{x,y,z}$ object must refer, in its attribute *.specified_value,* to a *Property_value*$_{x,y,z}$ object that represents one of the values permitted for the attribute denoted in the attribute *.id* of the *Class_attribute*$_{x,y}$ object. The *.value_name* attribute of the *Property_value*$_{x,y,z}$ object must always be 'permissive value'.

Example: Assume the *coordinator* sends a response to the *Request_permissive_values* ECM Message from the example above. Since the *coordinator* does not provide any permitted values for the *.id* attribute for *Series* objects, no permitted value is returned. For the *.name* attributes of the *Series* object however, there are two permitted list values defined: “ABC” and “DDE”. These two values are returned by this ECM Message. According to this example, the signature of the ECM Message *Respond_permissive_values* is as follows:

```
Respond_permissive_values(“Series”, “.id”, “.name”, “ABC”, “DDE”)
```

The parameters of this ECM Message are instantiated as follows:

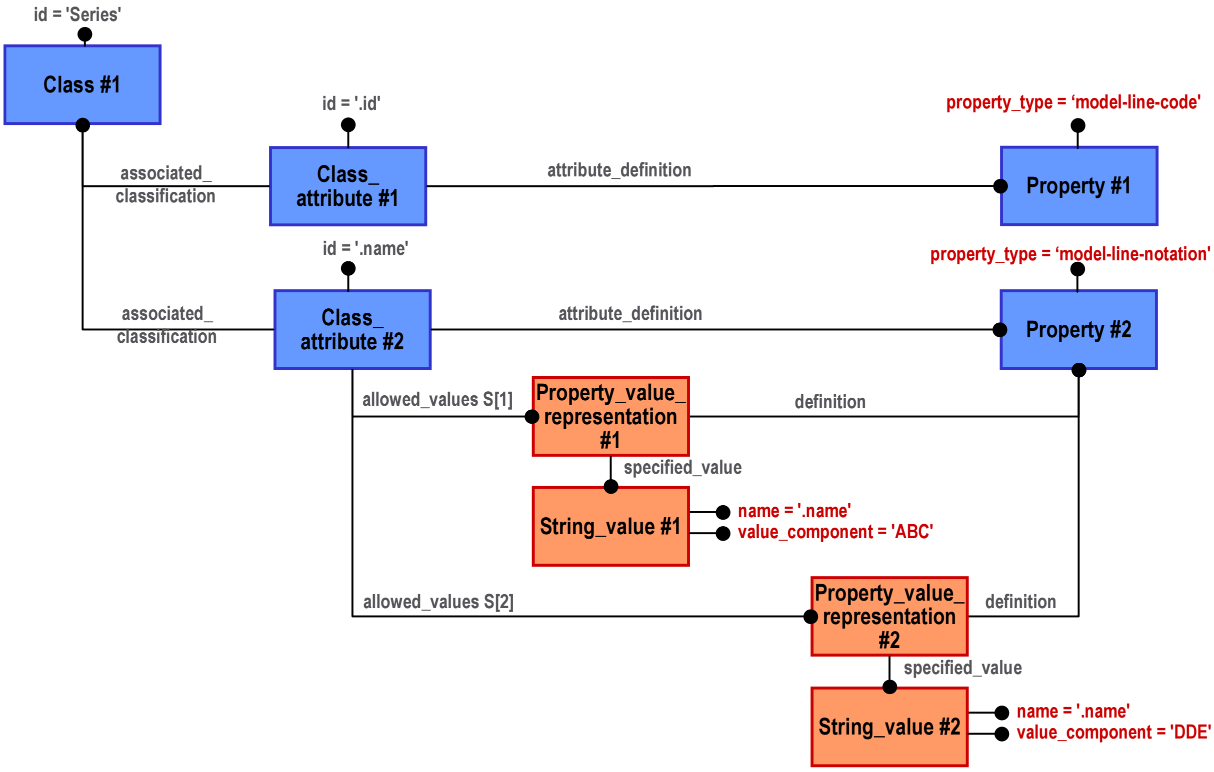


Figure 22: Sample instantiation of the Respond_permissive_values ECM Message

Appendix E: ECM Data Model and Data Dictionary

E.1 ECM Data Model – EXPRESS-G

The following figures are EXPRESS-G representations of the ECM Data Model. A brief introduction to EXPRESS-G notation can be found in Appendix B. The figures provide an overview of the defined data types and their properties.

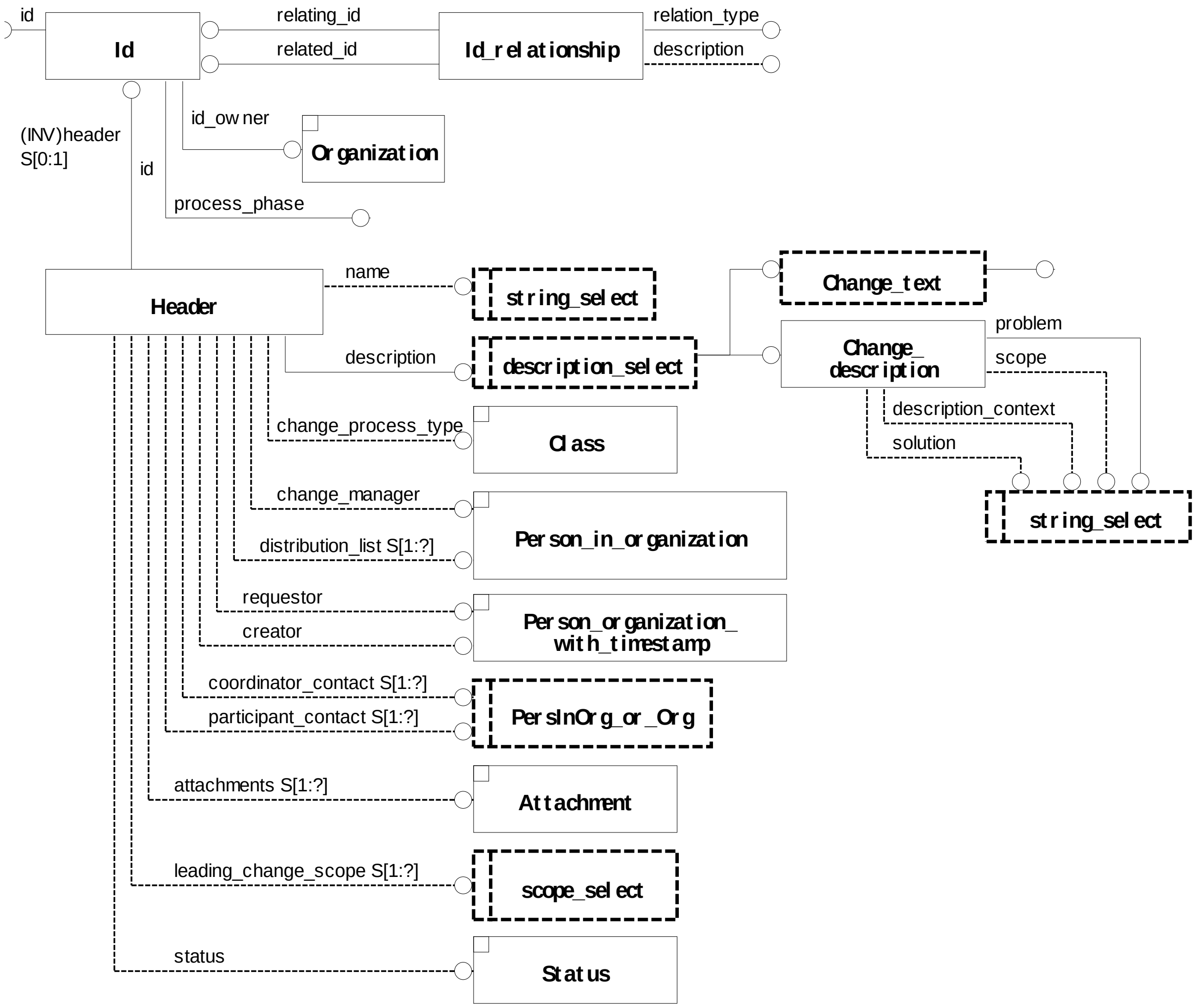


Figure 23: Header definition

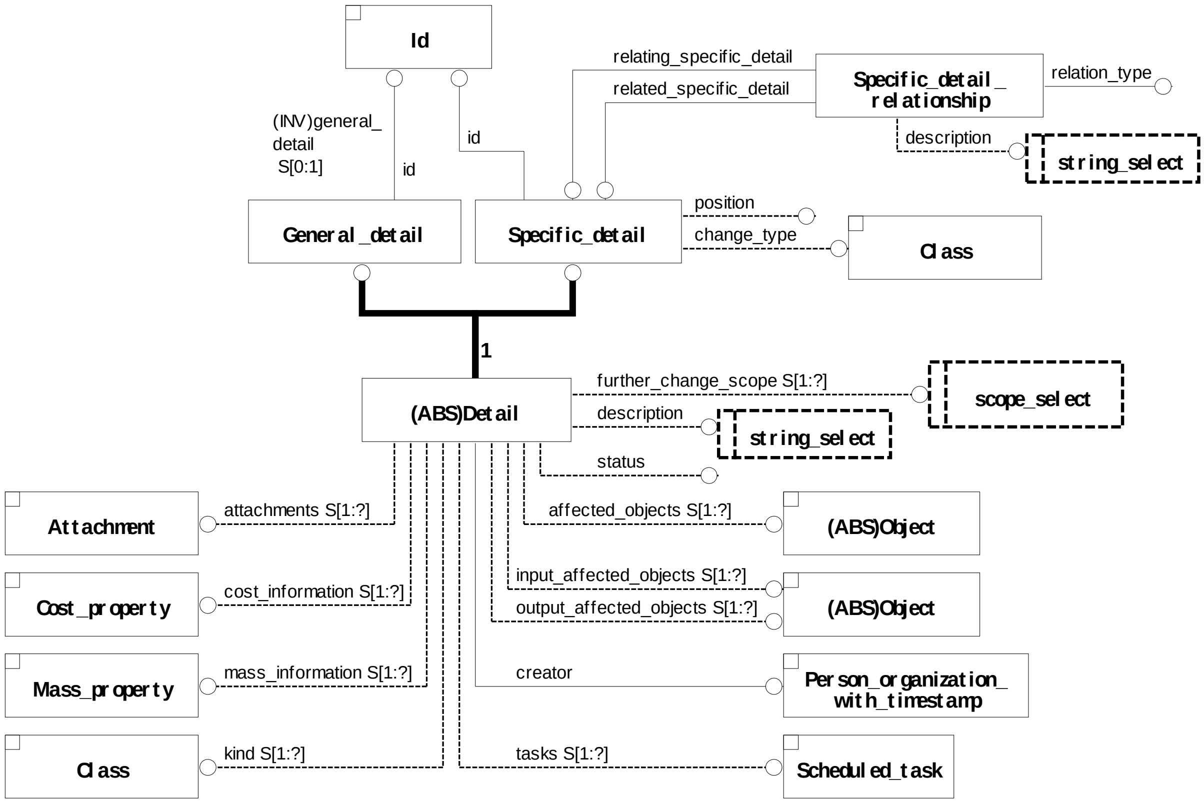


Figure 24: Detail definition

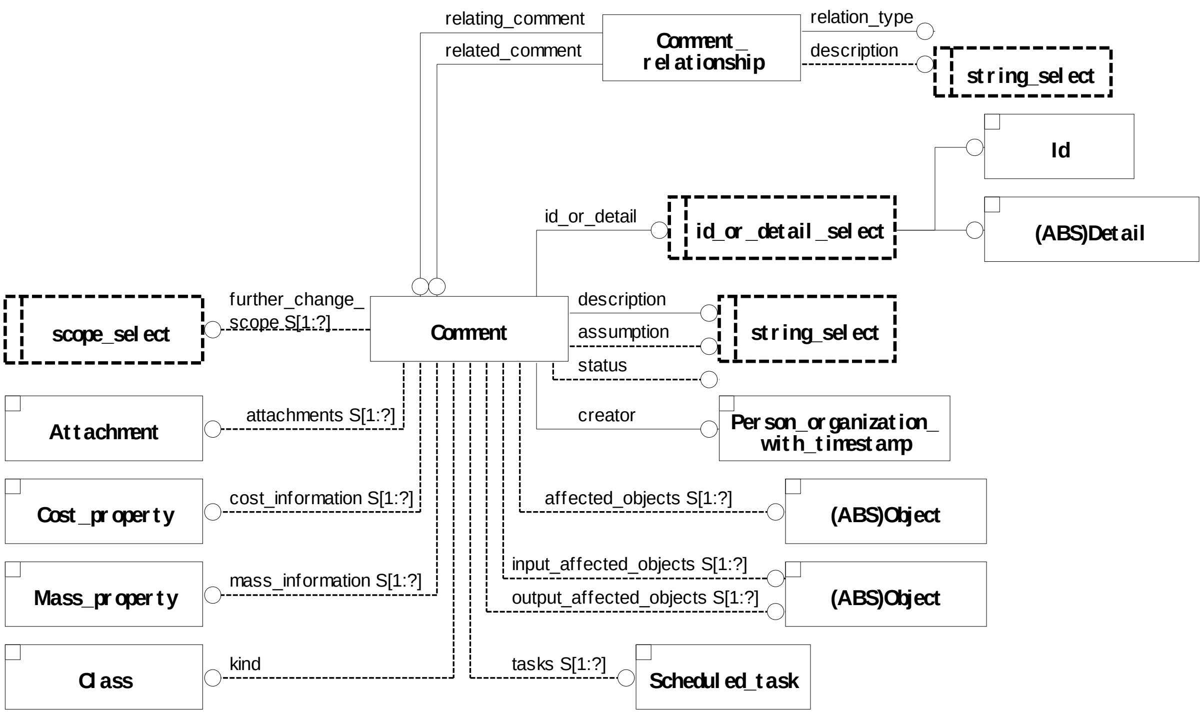


Figure 25: Comment definition

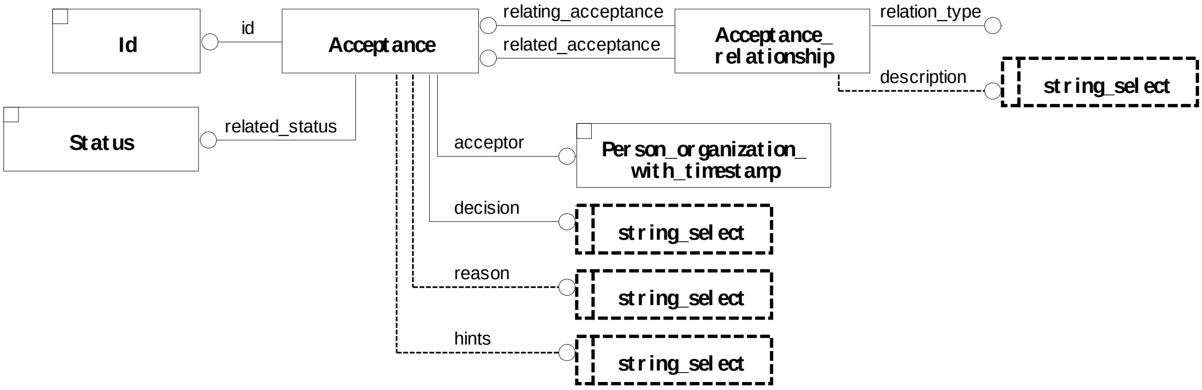


Figure 26: Acceptance definition

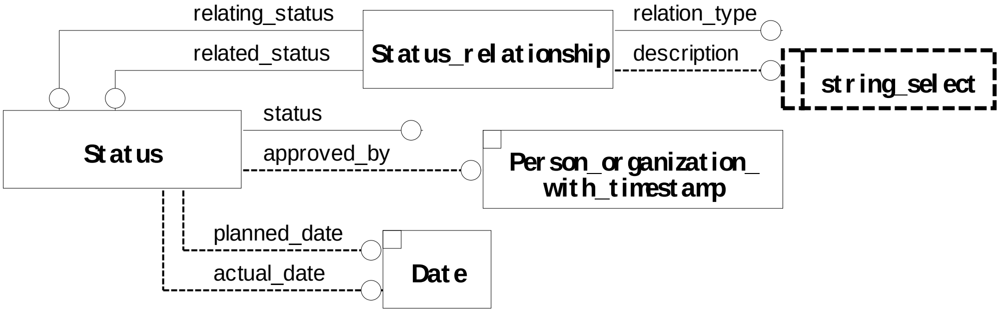


Figure 27: Status definition

A status is the ranking that specifies the state of the Reference Sub-Processes. Permitted values are defined in the specific data models (e.g. ECR, ECO, etc.) in the corresponding parts of this recommendation.

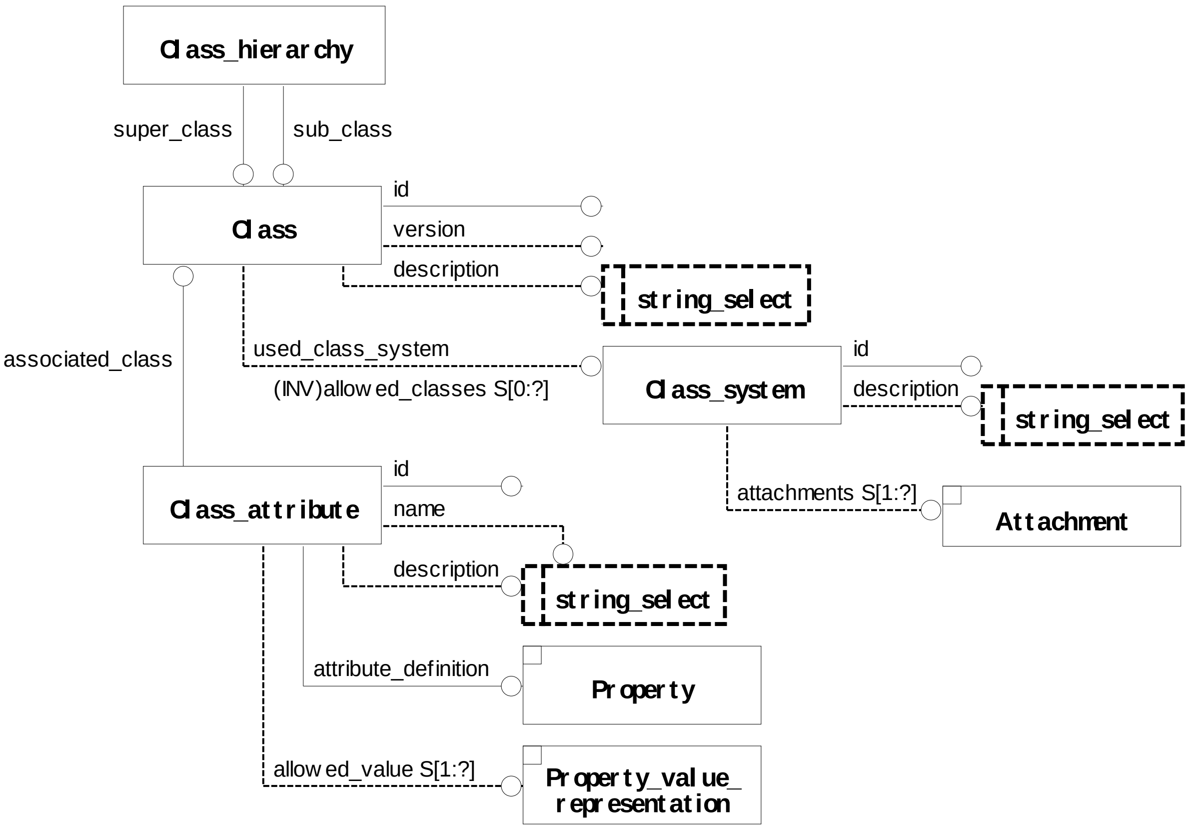


Figure 28: Class definition

The class concept supports the definition of classified property sets for data type definitions. In general, these data type definitions include the specification of allowed attribute values. The specification contains the context and the values of the property. The following figure provides an instantiation example for the usage of CLASS to classify a change process type as a 'fast run' process. Detailed classification information is specified in the data dictionary appendixes of the Reference Sub-Processes.

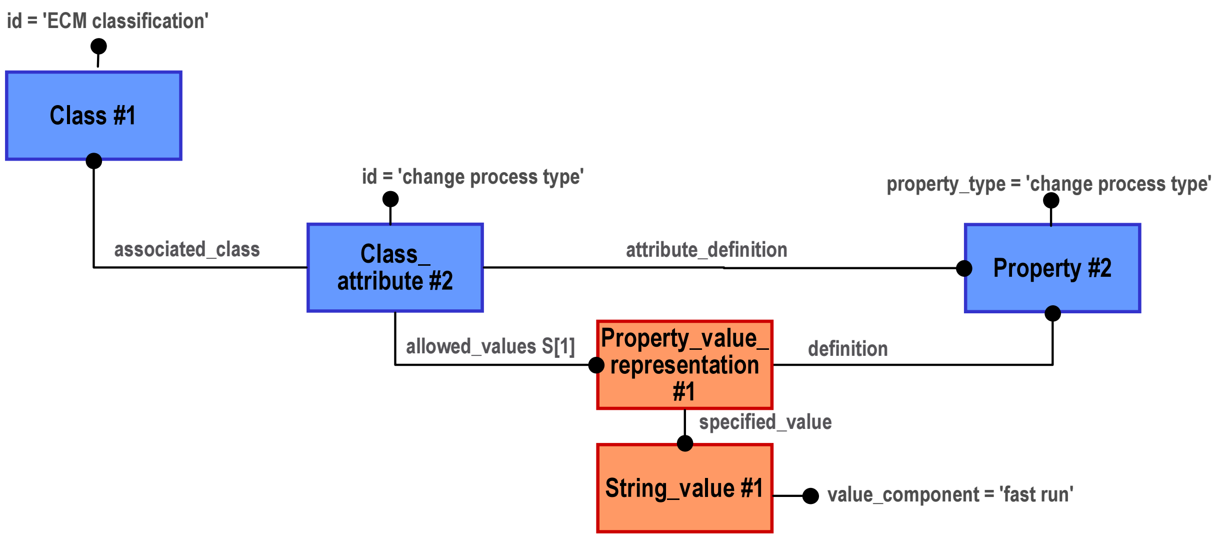


Figure 29: Instantiation example for using Class data type

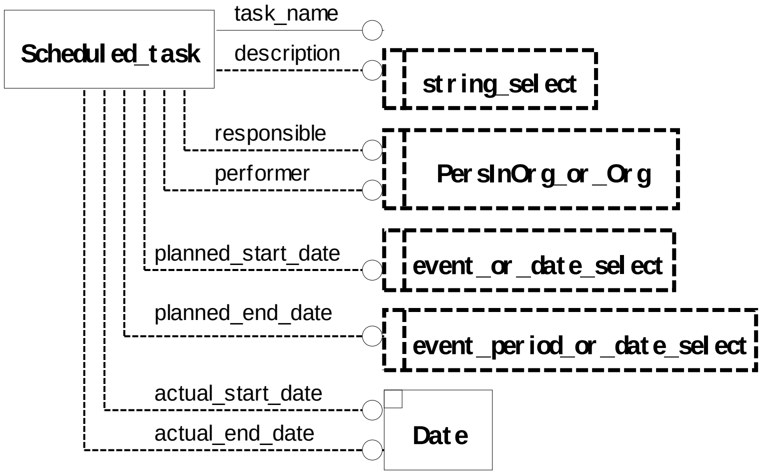


Figure 30: Scheduled task definition

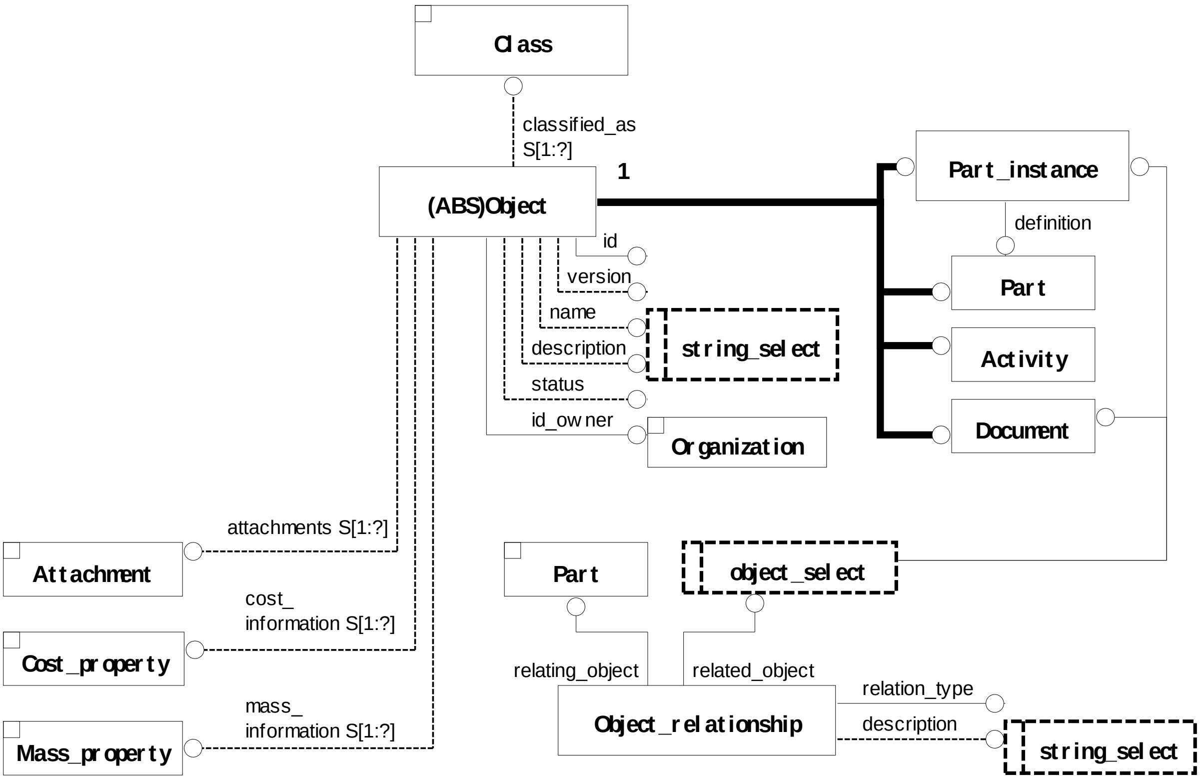


Figure 31: Affected object definition

Elements that are affected by engineering changes are:

- parts representing single parts, assemblies, vehicles as well as tools
- use of a part as a component in an assembly
- documents
- activities describing a measure that is neither related to a part nor to a document but affected (added, changed, omitted) by an engineering change

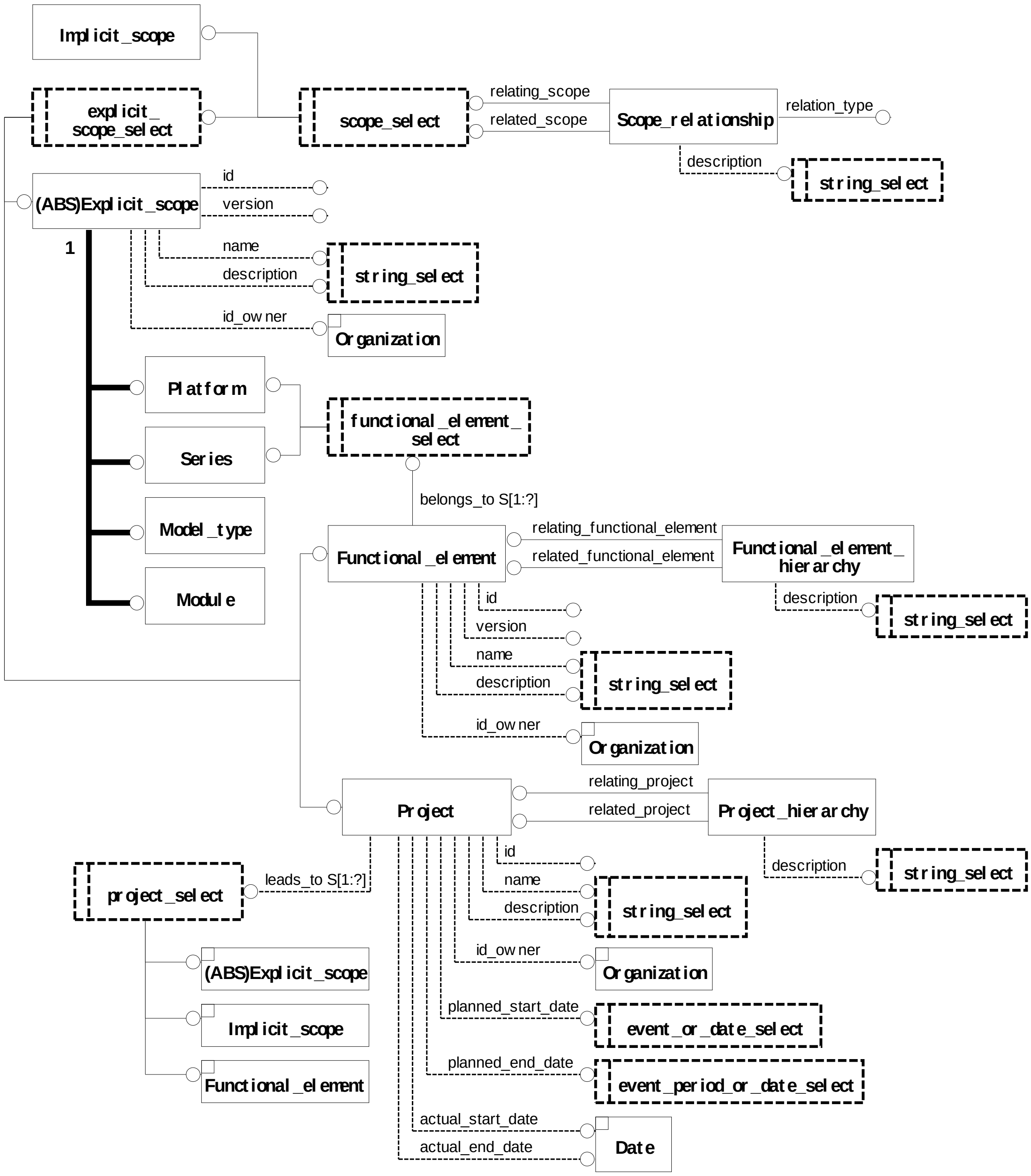


Figure 32: Data type Scope definition

SCOPE data types define the scope affected by an engineering change. Elements of different structuring categories represent the scope. Structuring Categories are

- Platforms, series, model types and modules (as a hierarchy)
- Functional sets of platforms or series
- Vehicle projects

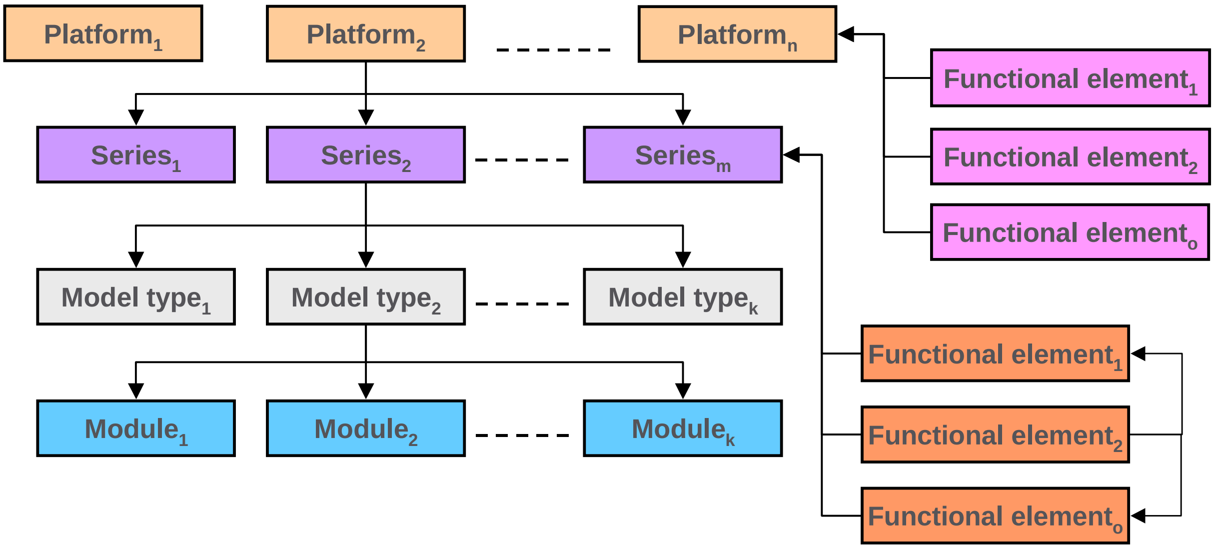


Figure 33: Hierarchy of structuring categories

The scope as a property of an object references the top level element of the structuring category hierarchy for which the property is valid. This means (implicitly) that all lower-level elements in the hierarchy are also elements of the scope - unlike the higher-level elements in the hierarchy.

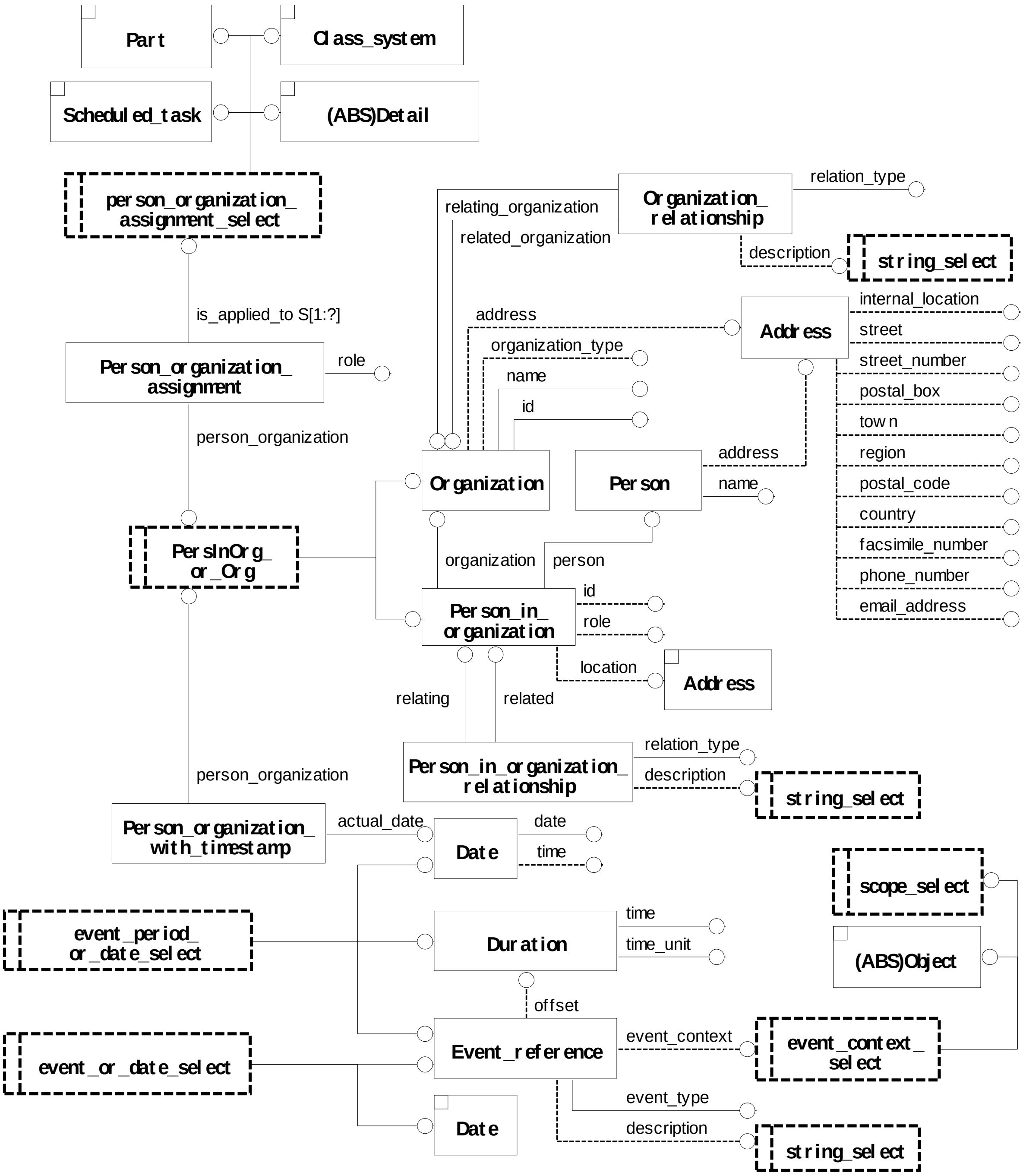


Figure 34: Organizational data types

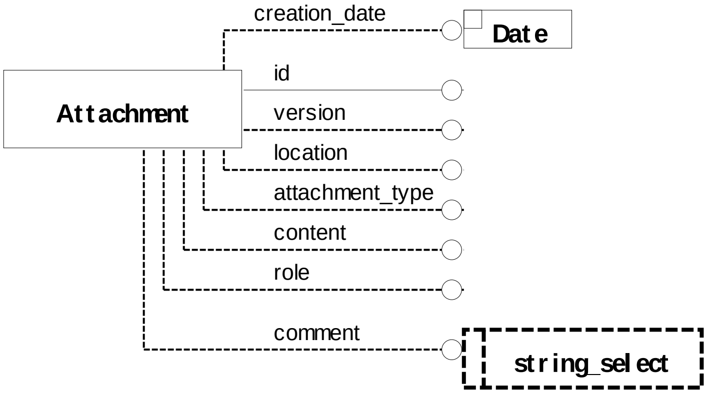


Figure 35: Attachment definition

The data type ATTACHMENT specifies parts of document systems, single files in a file system of a computer or links to documents which are intended to be additional documents for describing circumstances that are not specified explicitly within the data types of the data model.

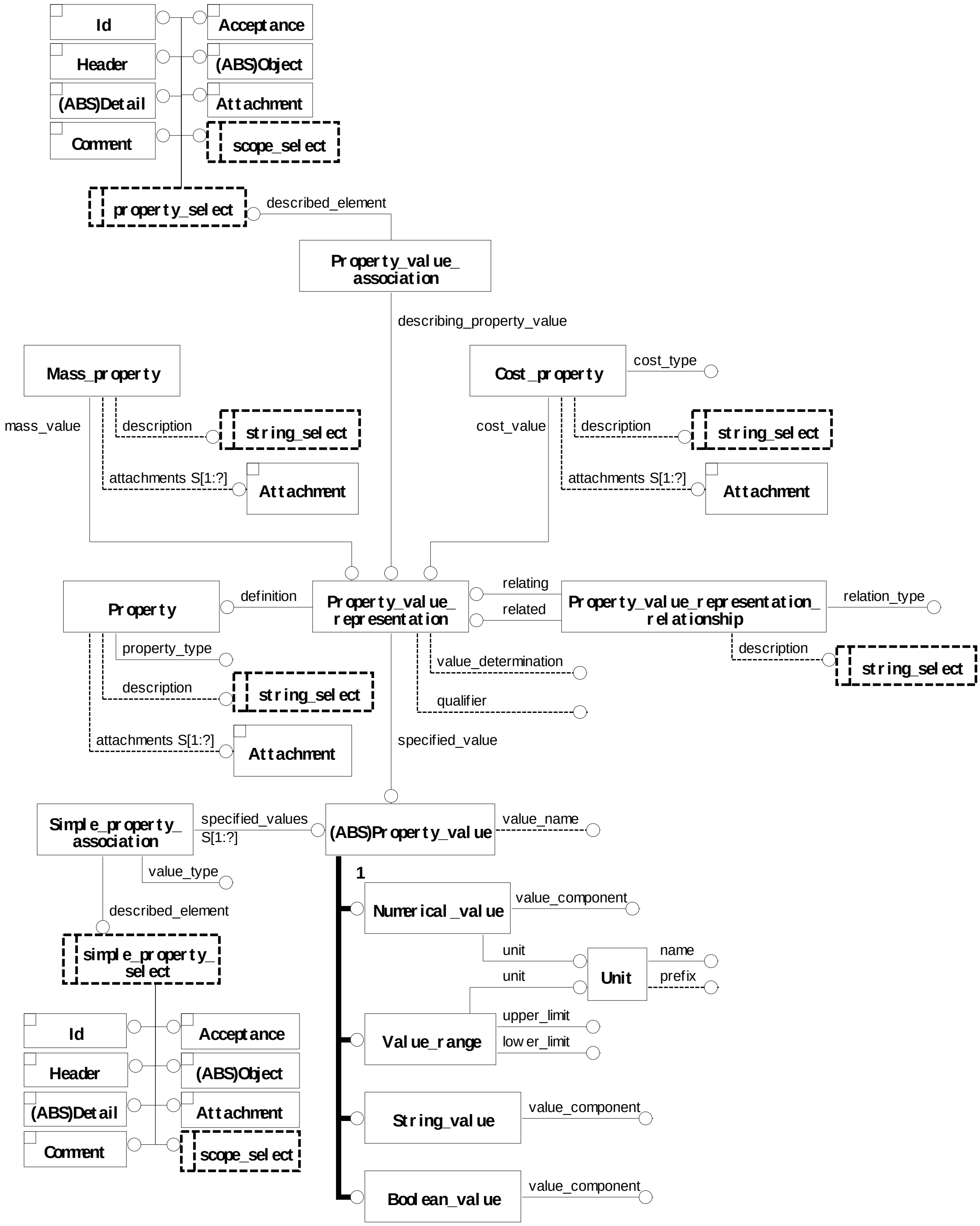


Figure 36: Property data types

The generic property definition concept is added by the typical properties cost and mass.

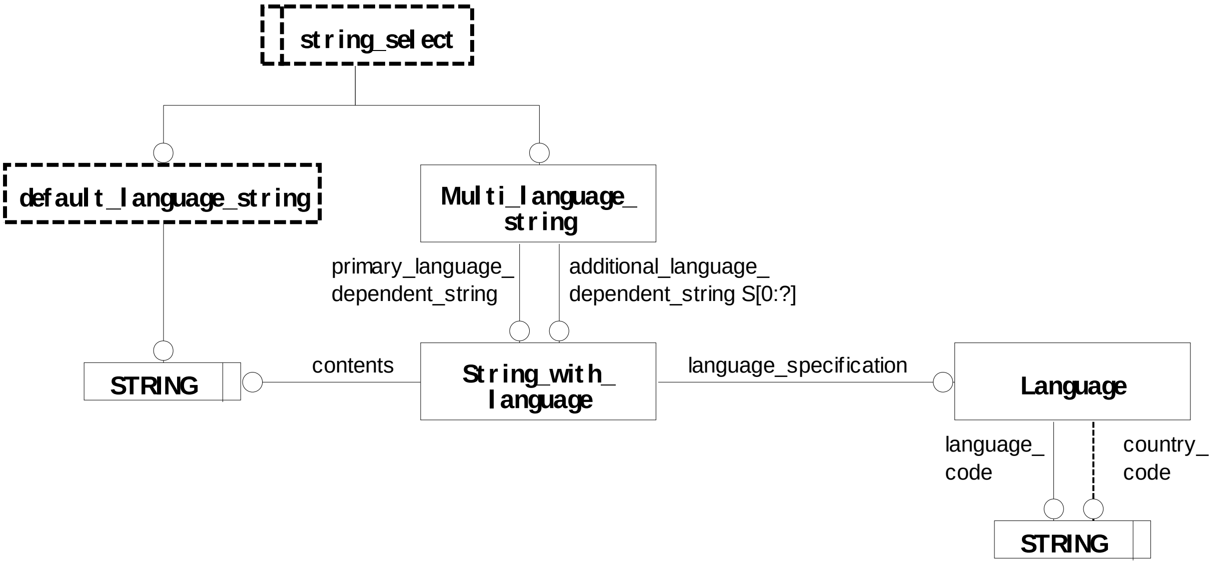


Figure 37: Multi-language data types

The multi-language data types support the representation of text information in one or more languages. Therefore the multi-language concept of ISO 10303-214 is used.

E.2 ECM Data Model - Data Dictionary

This chapter specifies all the elements of the ECM Data Model together with their attributes, attribute types and their predefined attribute values, as well as rules for their usage. Data types are arranged in alphabetical order. The order of the descriptions of the attributes is context dependent.

The following sections describe data objects that belong to the ECM Data Model:

- the names of the attribute including a short description of the meaning of the attribute type
- the type of attribute (as a basic type (string, integer) or as a reference to another data object)
- the cardinality of the attribute (1: mandatory; 0,1: optional; 1 .. n: one or more instances of the attribute type; 0 .. n: none, one or more instances of the attribute type
- predefined attribute values with description (the values should be used if possible, because their meaning is specified)
- rules for the usage of data object in the context of this guideline

E.2.1 Acceptance

Description of an acceptance declaration by a partner relating to an engineering change.

- attribute **acceptor**: represents the signature of the person or organization unit, which associates the acceptance declaration to the engineering change.
  - ***attribute type:*** PERSON_ORGANIZATION_WITH_TIMESTAMP
  - ***cardinality:*** 1

- attribute **decision**: defines the result of a decision activity in the ECM.
  - ***attribute type:*** string_select
  - ***cardinality:*** 1

  Use one of the following values:
  - **'accepted':** The ECM was accepted to proceed through the next phases in the ECM process (depending on its current status) or for implementation.
  - **'rejected':** The ECM was rejected with regard to further consideration in the ECM process or implementation.
  - **'incomplete':** The ECM was considered to be incomplete during external approval and should be resubmitted after completion at the *coordinator.*
- attribute **reason**: reasons for the decision
  - ***attribute type:*** string_select
  - ***cardinality:*** 0,1
- attribute **hints**: any information concerning the acceptance
  - ***attribute type*:** string_select
  - ***cardinality:*** 0,1
- attribute **related_status:** specifies the ECM Reference Process status
  - ***attribute type:*** STATUS
  - ***cardinality:*** 1
- attribute **id:** identifier of the engineering change referenced by the acceptance declaration
  - ***attribute type:*** ID
  - ***cardinality:*** 1

E.2.2 Acceptance_relationship

ACCEPTANCE_RELATIONSHIP describes a relationship between two ACCEPTANCE objects.

- attribute **relating_acceptance:** specifies the first acceptance in the relationship.
  - ***attribute type:*** ACCEPTANCE
  - ***cardinality:*** 1
- attribute **related_acceptance:** specifies the second acceptance in the relationship.
  - ***attribute type:*** ACCEPTANCE
  - ***cardinality:*** 1
- attribute **relation_type:** specifies the meaning of the relationship.
  - ***attribute type:*** string
  - ***cardinality:*** 1

Use the following value:

- **'sequence':** defines a sequence in which 'relating acceptance' is the preceding acceptance and 'related_acceptance' is the succeeding acceptance.

- attribute **description:** any annotations
  - ***attribute type:*** string_select
  - ***cardinality:*** 0,1

E.2.3 Activity

ACTIVITY is a specialization of OBJECT and is identified as an affected object. ACTIVITY describes a measure that is neither related to a part nor to a document. Such a measure regarding change data can be omitted, changed, added, or can be valid unaltered.

- attribute **id**: see OBJECT
- attribute **id_owner**: see OBJECT
- attribute **version**: see OBJECT

  Rule: *A versioning of ACTIVITY makes no sense. Therefore the attribute is not attached with a value in this use case.*

- attribute **name**: see OBJECT
- attribute **description**: see OBJECT
- attribute **attachments**: see OBJECT
- attribute **cost_information**: see OBJECT
- attribute **mass_information**: see OBJECT
- attribute **status**: see OBJECT
- attribute **classified_as**: see OBJECT

E.2.4 Address

ADDRESS describes an address including phone and fax number as well as email-address. All attributes are optional, but at least one attribute has to be attached with a value (***Rule***).

- attribute **internal_location**: company internal address
  - ***attribute type:*** string
  - ***cardinality:*** 0,1
- attribute **street**: street
  - ***attribute type:*** string
  - ***cardinality:*** 0,1
- attribute **street_number**: street number
  - ***attribute type:*** string
  - ***cardinality:*** 0,1
- attribute **postal_box**: postal box
  - ***attribute type:*** string
  - ***cardinality:*** 0,1

- attribute **town**: town
  - ***attribute type:*** string
  - ***cardinality:*** 0,1
- attribute **region**: region
  - ***attribute type:*** string
  - ***cardinality:*** 0,1
- attribute **postal_code**: postal code
  - ***attribute type:*** string
  - ***cardinality:*** 0,1
- attribute **country**: country
  - ***attribute type:*** string
  - ***cardinality:*** 0,1
- attribute **facsimile_number**: facsimile number
  - ***attribute type:*** string
  - ***cardinality:*** 0,1
- attribute **phone_number**: phone number
  - ***attribute type:*** string
  - ***cardinality:*** 0,1
- attribute **email_address**: e-mail address
  - ***attribute type:*** string
  - ***cardinality:*** 0,1

E.2.5 Attachment

ATTACHMENT specifies parts of document systems ('managed documents', e.g. versioned CAD models), single files in the file system of a computer (e.g. Word, Excel or STEP data) or links to documents (e.g. internet addresses, identifier of drawing archives) which are intended to be additional documents for describing circumstances that are not specified explicitly within the data types of the data model.

- attribute **id**: identification of the additional document
  - ***attribute type:*** string
  - ***cardinality:*** 1
- attribute **version**: version number of the document (e.g. number of a standard)
  - ***attribute type:*** string
  - ***cardinality:*** 0,1

Rule: *This attribute is mandatory for documents in a document system ('managed documents').*

- attribute **location**: location at which the document can be found. (e.g. internet address)
  - ***attribute type:*** string
  - ***cardinality:*** 0,1

Rule: *This attribute is mandatory for referenced documents that are not generally available elsewhere (e.g. standards, VDA recommendations).*

- attribute **attachment_type**: format of the documents
  - ***attribute type:*** string
  - ***cardinality:*** 0,1

If possible, use one of the following values:

- all MIME types,
  see www.iana.org/assignments/media-types/, e.g. 'http', 'jpeg', 'pdf', 'text', 'xml'
- **'doc'**: MS Word format
- **'ppt'**: MS PowerPoint format
- **'xls'**: MS Excel format
- **'stp'**: ISO 10303 (STEP), Part 21 format

- attribute **content**: content description of the appendix
  - ***attribute type:*** string
  - ***cardinality:*** 0,1
- attribute **comment**: any additional information to the attachment (content or usage)
  - ***attribute type:*** string_select
  - ***cardinality:*** 0,1
- attribute **creation_date**: creation date of the appendix
  - ***attribute type:*** DATE
  - ***cardinality:*** 0,1
- attribute **role**: specifies the role of the attachment as additional information
  - ***attribute type:*** string
  - ***cardinality:*** 0,1

If possible, use one of the following values:

- **'mandatory'**: the attachment shall conform to the content of the assigned document
- **'informative'**: the attachment may or may not be considered

- o **'additional Information'**: the attachment provides information that is relevant for the associated object, but is not a description of the associated object itself

E.2.6 Boolean_value

BOOLEAN_VALUE is a specialization of a PROPERTY_VALUE and defines a Boolean value (true or false).

- attribute **value_name**: see PROPERTY_VALUE
- attribute **value_component**: Boolean value
  - ***attribute type:*** boolean
  - ***cardinality:*** 1

Rule: *Valid values are: 'true' and 'false'*

E.2.7 Change_description

CHANGE_DESCRIPTION provides a method for the structured description of a engineering change. The description of the ECM can be stated as a simple text (Change_text) as well as structured (CHANGE_DESCRIPTION).

- attribute **problem*:*** describes the original idea of the engineering change as well as the expectations concerning the result
  - ***attribute type:*** string_select
  - ***cardinality:*** 1
- attribute **scope:** describes the engineering change from different point of views (OEM, Supplier) and defines excluded technical and organizational
  - ***attribute type*:** string_select
  - ***cardinality:*** 0,1
- attribute **solution:** describes in as much detail as possible the expected solution including applied methods and tools.
  - ***attribute type:*** string_select
  - ***cardinality:*** 0,1
- attribute **description_context:** specifies the context of an ECM regarding the expected benefit.
  - ***attribute type:*** string_select
  - ***cardinality:*** 0,1

E.2.8 Class

CLASS is a classification of an object which characterizes all objects of the same kind; such a classification is independent from the application of the classified object.

- attribute **id**: specifies the identifier of the CLASS
  - ***attribute type:*** string

- ***cardinality:*** 1

- attribute **description**: additional information about the CLASS
  - ***attribute type:*** string_select
  - ***cardinality:*** 0,1
- attribute **version**: specifies the identification of a particular version of the CLASS
  - ***attribute type:*** string
  - ***cardinality:*** 0,1
- attribute **used_class_system**: specifies the CLASS_SYSTEM that contains the information about the definition of the classification and how to interpret the name of the Class
  - ***attribute type:*** CLASS_SYSTEM
  - ***cardinality:*** 0,1

E.2.9 Class_attribute

CLASS_ATTRIBUTE is a characteristic used to classify an object associated with the corresponding CLASS. The definition attribute of each 'allowed_value' refers to the property identified within 'attribute_definition'.

- attribute **id**: specifies the identifier of the CLASS_ATTRIBUTE that must be unique within the scope of the associated CLASS.
  - ***attribute type:*** string
  - ***cardinality:*** 1
- attribute **name**: specifies the word or group of words used to refer to the CLASS_ATTRIBUTE
  - ***attribute type:*** string_select
  - ***cardinality:*** 0,1
- attribute **description**: specifies additional information about the CLASS_ATTRIBUTE
  - ***attribute type:*** string_select
  - ***cardinality:*** 0,1
- attribute **associated_class**: specifies the CLASS of which the CLASS_ATTRIBUTE is a characteristic
  - ***attribute type:*** CLASS
  - ***cardinality:*** 1
- attribute **attribute_definition**: specifies the PROPERTY that characterizes the allowed values
  - ***attribute type:*** PROPERTY
  - ***cardinality:*** 1

- attribute **allowed_value**: specifies the set of PROPERTY_ VALUE_REPRESENTATION objects that represent characteristic values of the CLASS_ATTRIBUTE
  - ***attribute type:*** PROPERTY_VALUE_REPRESENTATION
  - ***cardinality:*** 0 .. n

E.2.10 Class_hierarchy

CLASS_HIERARCHY defines a hierarchical relationship between two instances of CLASS. The CLASS_HIERARCHY is used to create hierarchical structures involving CLASS objects.

- attribute **super_class**: specifies the higher level of CLASS in a CLASS_HIERARCHY that includes the sub class
  - ***attribute type:*** CLASS
  - ***cardinality:*** 1
- attribute **sub_class**: specifies the lower level of CLASS in a CLASS_HIERARCHY that is included in the super class
  - ***attribute type:*** CLASS
  - ***cardinality:*** 1

E.2.11 Class_system

CLASS_SYSTEM is the scheme used to define the categorization of data types.

- attribute **id**: specifies the identifier of the CLASS_SYSTEM
  - ***attribute type:*** string
  - ***cardinality:*** 1
- attribute **description**: additional information about the CLASS_SYSTEM
  - ***attribute type:*** string_select
  - ***cardinality:*** 0,1
- attribute **attachments**: additional, externally available information as appendix to the CLASS SYSTEM
  - ***attribute type:*** ATTACHMENT
  - ***cardinality:*** 0 .. n
- attribute **allowed_classes (INV)**: inverse attribute as a set of CLASSES that reference the CLASS_SYSTEM as 'used_class_ system'
  - ***attribute type:*** CLASS
  - ***cardinality:*** 0..n

E.2.12 Comment

A COMMENT specifies a comment on an engineering change (ID) or more detailed information (DETAIL). It includes statements regarding the affected objects, calculation, changes in weight and/or statements concerning feasibility and consequences of the change. A comment on an ECM comprises comment text and optional statements concerning the assumptions that provided the basis for the comment.

If the comment refers to all aspects of the details, it references ('id_or_detail') GENERAL_DETAIL.

If the comment only refers to several aspects of the ECM (e.g. a subset of the affected objects, certain statements regarding prices, costs or weight or certain appointments), it references ('id_or_detail') SPECIFIC_DETAIL.

- attribute **id_or_detail:** specifies the relation for which the comment is given.
  - ***attribute type:*** id_or_detail_select
  - ***cardinality:*** 1
- attribute **description:** text representing the comment.
  - ***attribute type:*** string_select
  - ***cardinality:*** 1
- attribute **status:** status of the given comment.
  - ***attribute type:*** string
  - ***cardinality:*** 0,1

  Use one of the following values:

  - **'preliminary':** is an preliminary comment
  - **'final':** is the final comment
- attribute **Creator:** originator of the comment with date of creation.
  - ***attribute type:*** PERSON_ORGANIZATION_WITH_TIMESTAMP
  - ***cardinality:*** 1
- attribute **input_affected_objects:** affected objects before the change (parts to change, assembly structure, etc.) for which the comment is given.
  - ***attribute type:*** OBJECT
  - ***cardinality:*** 0 .. n
- attribute **output_affected_objects:** affected objects after the change (e.g. newly developed parts, new versions, changed assembly structures, etc.) for which the comment is given.
  - ***attribute type:*** OBJECT
  - ***cardinality:*** 0 .. n
- attribute **affected_objects:** affected objects without distinction of input and output for which the comment is given.
  - ***attribute type:*** OBJECT

- ***cardinality:*** 0 .. n

- attribute **cost_information:** information about the costs of the ECM that refer to the scope of the specified (input/output) affected objects.
  - ***attribute type:*** COST_PROPERTY
  - ***cardinality:*** 0 .. n
- attribute **mass_information:** information about the weight of the ECM that refer to the scope of the specified (input/output) affected objects.
  - ***attribute type:*** MASS_PROPERTY
  - ***cardinality:*** 0 .. n
- attribute **attachments:** any additional documents and/or files that relate to the ECM comment.
  - ***attribute type:*** ATTACHMENT
  - ***cardinality:*** 0 .. n
- attribute **assumption:** description of assumptions or constraints that provided the basis for the comment.
  - ***attribute type:*** string_select
  - ***cardinality:*** 0,1
- attribute **tasks:** specification of tasks concerned with the given comment.
  - ***attribute type:*** SCHEDULED_TASK
  - ***cardinality:*** 0 .. n
- attribute **kind:** allows a distinction to be made between different kinds of comment that are exchanged simultaneously in one message.
  - ***attribute type:*** CLASS
  - ***cardinality:*** 1 .. n

The following can be used to define the kind of a comment (Instantiation example see Part 0 of this recommendation, Figure 17):

- CLASS.id = 'ECM comment classification'
- CLASS_ATTRIBUTE.id = 'ECM kind of comment'
- CLASS_ATTRIBUTE.allowed_values S[1] = PROPERTY_VALUE_REPRESENTATION
- PROPERTY_VALUE_REPRESENTATION.specified_value = STRING_VALUE
- STRING_VALUE.value_component = one of the predefined values below
- PROPERTY_VALUE_REPRESENTATION.definition = PROPERTY
- PROPERTY.property_type = 'ECM kind of comment'

If possible, use one of the following values for String_value.value_component:

- **'general':** indicates a common, universal kind of comment. This value is to be used if there is only one kind of comment exchanged.
- **'commercial':** Comments on the ECM regarding cost effects of the change

  - **'development':** comments on the engineering change with a technical focus, i.e. regarding effects from the view of engineering (design, test, …)
  - **'logistics':** comments on the engineering change regarding logistical effects of the change (delivery time changes, container changes, …)
  - **'manufacturing':** comments from the view of manufacturing
- attribute **further_change_scope:** more scope elements that are within the scope of the ECM but which are not specified as leading in HEADER. leading_change_ scope
  - ***attribute type:*** scope_select
  - ***cardinality:*** 0 .. n

E.2.13 Comment_relationship

COMMENT_RELATIONSHIP describes the dependency between two COMMENT instances.

- attribute **relating_comment:** specifies the first comment in the relationship.
  - ***attribute type:*** COMMENT
  - ***cardinality:*** 1
- attribute **related_comment:** specifies the second comment in the relationship.
  - ***attribute type:*** COMMENT
  - ***cardinality:*** 1
- attribute **relation_type:** describes the dependency between the two comments.
  - ***attribute type:*** string
  - ***cardinality:*** 1

Use the following value:

- **'sequence':** defines a sequence where 'relating_comment' is the preceding comment and 'related_comment' is the succeeding comment
- attribute **description:** any annotations
  - ***attribute type:*** string_select
  - ***cardinality:*** 0,1

E.2.14 Cost_property

COST_PROPERTY specifies all costs that are relevant to engineering changes as well as objects affected by the changes.

- attribute **cost_type**: specifies the type of costs
  - ***attribute type:*** string
  - ***cardinality:*** 1

  If possible, use one of the following values:

  - **'one time costs'**: costs that arise only once

- **'part costs'**: proportional costs, e.g. total costs per vehicle, assembly or part

- attribute **cost_value**: specifies the cost value
  - ***attribute type:*** PROPERTY_VALUE_REPRESENTATION
  - ***cardinality:*** 1
- attribute **description**: any annotations
  - ***attribute type:*** string_select
  - ***cardinality:*** 0,1
- attribute **attachments**: additional, externally available information as appendix to the cost data
  - ***attribute type:*** ATTACHMENT
  - ***cardinality:*** 0 .. n

Note: *The attribute 'qualifier' of the data type PROPERTY_VALUE_REPRESENTATION includes an additional possibility for classifying the costs as annual or delta costs:*

If possible, use one of the following values for PROPERTY_VALUE_REPRESENTATION.QUALIFIER:

- **'annual'**: the specified costs are costs per year
- **'delta'**: the specified costs describe the change (delta) of a cost type

Note: *The kind of cost is specified in the attribute 'name' of PROPERTY referenced by the attribute 'definition' of PROPERTY_VALUE_REPRESENTATION.*

If possible, use one of the following values for PROPERTY.PROPERTY_TYPE:

- **'prototype development costs'**: prototype development costs
- **'prototype material costs'**: prototype material costs
- **'development costs'**: development costs
- **'material costs '**: material costs
- **'tool costs'**: tool costs
- **'production costs'**: production costs
- **'scrapping costs'**: scrapping costs
- **'transport costs'**: transport costs
- **'handling costs'**: handling costs

Note: *A distinction between calculated and estimated costs can be specified in PROPERTY_ VALUE_REPRESENTATION.value_determination.*

If possible, use one of the following values for PROPERTY_VALUE_REPRESENTATION.value_determination:

- **'calculated'**: the specified cost value has been calculated

- **'estimated'**: the specified cost value has been estimated

E.2.15 Date

Date specifies a calendar date.

- attribute **date**: specifies the date
  - ***attribute type:*** string
  - ***cardinality:*** 1

  notation according to ISO 8101: yyyy-mm-dd

  e.g. '2004-12-06' for the 6th of December 2004
- attribute **time**: specifies the time
  - ***attribute type:*** string
  - ***cardinality:*** 0,1

  notation according to ISO 8101: hh:mm:ss-hh:mm', e.g. '13:20:00-05:00' for 1:20 pm Eastern Standard Time, which has a 5-hour time shift backwards with regard to Coordinated Universal Time (UTC).

E.2.16 default_language_string_select

DEFAULT_LANGUAGE_STRING_SELECT is of data type SELECT and represents the data types STRING_WITH_LANGUAGE or STRING.

E.2.17 description_select

DESCRIPTION_SELECT is of data type SELECT and represents the data types CHANGE_TEXT or CHANGE_DESCRIPTION.

E.2.18 Detail

DETAIL specifies detailed information about the engineering change from a technical point of view. The analysis of a change results in a successive refinement of the description of an ECM. All objects affected by the change as well as consequences (e.g. costs, appointments) have to be identified. (Figure 24)

The detailed description of the change is summarized in all instances of DETAIL.

Rule: *DETAIL is an abstract data type, that can only be instantiated in one of its values (GENERAL_DETAIL, SPECIFIC_DETAIL).*

- attribute **status:** describes the state of detailing within the ECM Reference Process
  - ***attribute type:*** string
  - ***cardinality:*** 0,1

  Use one of the following values:

  - **'informative':** DETAIL specifies technical detail information that is given before the phase Technical Analysis of ECM of the Reference Process, e.g. it represents informative detail information that must be evaluated during the technical analysis.
  - **'in work':** DETAIL specifies technical detail information that is given after the start and before the end of the phase Technical Analysis of ECM of the Reference Process, e.g. it represents a preliminary result of the technical analysis.
  - **'completed':** DETAIL specifies technical detail information that is given after the completion of the phase Technical Analysis of ECM of the Reference Process, i.e. it represents the result of the technical analysis.

- attribute **creator:** originator of the detailed ECM information with the date of creation
  - ***attribute type:*** PERSON_ORGANIZATION_WITH_TIMESTAMP
  - ***cardinality:*** 1
- attribute **kind:** classifies the detailed information for the ECM
  - ***attribute type:*** CLASS
  - ***cardinality:*** 0 .. n

The following structure and fixed values should be used to define the kind of detailed data:

- CLASS.id = 'ECM detail'
- CLASS_ATTRIBUTE.id = 'ECM kind of detail'
- CLASS_ATTRIBUTE.allowed_values S[1] = PROPERTY_VALUE_REPRESENTATION
- PROPERTY_VALUE_REPRESENTATION.specified_value = STRING_VALUE
- STRING_VALUE.value_component = one of the predefined values below
- PROPERTY_VALUE_REPRESENTATION.definition = PROPERTY
- PROPERTY.property_type = 'ECM kind of detail'

- attribute **input_affected_objects:** affected objects before the change (parts to change, assembly structure, etc.)
  - ***attribute type:*** OBJECT
  - ***cardinality:*** 0 .. n

Note: *If .input_affected_object references an object of the type Part, no dependency is specified with regard to use. Otherwise the type* PART_INSTANCE *has to be used.*

- attribute **output_affected_objects:** affected objects after the change (e.g. newly developed parts, new versions, changed assembly structures, etc.)
  - ***attribute type:*** OBJECT
  - ***cardinality:*** 0 .. n

Note: *If .output_affected_object references an object of the type Part, no dependency is specified with regard to use. Otherwise the type* PART_INSTANCE *has to be used.*

- attribute **affected_objects:** affected objects without distinction of input and output for which the comment is given
  - ***attribute type:*** OBJECT
  - ***cardinality:*** 0 .. n

Note: *If .affected_object references an object of the type Part, no dependency is specified with regard to use. Otherwise the type* PART_INSTANCE *has to be used.*

- attribute **description:** any annotations
  - ***attribute type:*** string_select
  - ***cardinality:*** 0,1
- attribute **cost_information:** information about the costs of the ECM that refer to the scope of the specified 'input/output_affected_ objects'
  - ***attribute type:*** COST_PROPERTY
  - ***cardinality:*** 0 .. n
- attribute **mass_information:** information about the weight of the ECM that refer to the scope of the specified 'input/output_affected_ objects'
  - ***attribute type:*** MASS_PROPERTY
  - ***cardinality:*** 0 .. n
- attribute **attachments:** any additional documents and/or files that relate to the product change
  - ***attribute type:*** ATTACHMENT
  - ***cardinality:*** 0 .. n
- attribute **further_change_scope:** more scope elements that fall within the scope of the ECM, but which are not specified as leading in HEADER. leading_change_scope
  - ***attribute type:*** scope_select
  - ***cardinality:*** 0 .. n
- attribute **tasks:** specifies tasks that are concerned with the engineering change
  - ***attribute type:*** SCHEDULED_TASK
  - ***cardinality:*** 0 .. n

E.2.19 Document

DOCUMENT is a specialization of OBJECT and is identified as an affected object in a change process. This document can be omitted, changed, added, or can be valid unaltered in the context of the Change.

- attribute **id**: see OBJECT
- attribute **id_owner**: see OBJECT
- attribute **version**: see OBJECT

Rule: *The attribute is mandatory for document systems ('managed documents') and, in this case, must be unique together with the Identifier.id.*

- attribute **name**: see OBJECT
- attribute **description**: see OBJECT
- attribute **classified_as**: see OBJECT
- attribute **attachments**: see OBJECT

- attribute **cost_information**: see OBJECT
- attribute **mass_information**: see OBJECT
- attribute **status**: see OBJECT

E.2.20 Duration

DURATION describes a period of time in terms of length.

- attribute **time**: length of time (as integer or real value)
  - ***attribute type:*** number
  - ***cardinality:*** 1
- attribute **time_unit**: unit in which the duration is measured
  - ***attribute type:*** string
  - ***cardinality:*** 1

  If possible, use one of the following values:
  - **'hour'**: length of time in hours
  - **'day'**: length of time in days
  - **'week'**: length of time in weeks
  - **'month'**: length of time in months

E.2.21 event_or_date_select

- EVENT_OR_DATE_SELECT is of data type SELECT and represents the data types EVENT_REFERENCE or DATE.

E.2.22 event_periode_or_date_select

- EVENT_PERIODE_OR_DATE_SELECT is of data type SELECT and represents the data types EVENT_REFERENCE, DURATION or DATE.

E.2.23 event_context_select

EVENT_CONTEXT_SELECT is of data type SELECT and represents the data types OBJECT or SCOPE_SELECT.

E.2.24 Event_reference

EVENT_REFERENCE specifies a point in time relative to an event.

Rule: EVENT_REFERENCE can be used by SCHEDULED_TASK.start_date only if SCHEDULED_TASK.task_name is 'operation date'.

- attribute **event_type**: specifies the type of event, e.g. ‘start of production’
  - ***attribute type:*** string

- ***cardinality:*** 1
- attribute **description**: any annotations
  - ***attribute type:*** string_select
  - ***cardinality:*** 0,1
- attribute **offset**: offset (earlier, later) with regard to the specified event
  - ***attribute type:*** DURATION
  - ***cardinality:*** 0,1
- attribute **event_context**: specifies the context of the event
  - ***attribute type:*** DURATION
  - ***cardinality:*** 0,1

E.2.25 Explicit_scope

EXPLICIT_SCOPE describes elements explicitly affected by a change process and supports different structuring methods of a vehicle.

EXPLICIT_SCOPE is an abstract data type and can be instantiated only as one of it subtypes (PLATFORM, SERIES, MODEL_TYPE, MODULE).

Rule: At least one of the optional attributes .id and .name has to be attached with a value.

- attribute **id:** identifies the scope of an change
  - ***attribute type:*** string
  - ***cardinality:*** 0,1
- attribute **id_owner:** reference to an organization in which the attribute 'id' is uniquely defined, e.g. the pair (id, id_owner) must be unique for the interpretation of the attribute ‘id’
  - ***attribute type:*** ORGANIZATION
  - ***cardinality:*** 0,1
- attribute **version:** version of a the scope
  - ***attribute type:*** string
  - ***cardinality:*** 0,1
- attribute **name:** name of the scope
  - ***attribute type:*** string_select
  - ***cardinality:*** 0,1
- attribute **description:** any annotations
  - ***attribute type:*** string_select
  - ***cardinality:*** 0,1

E.2.26 explicit_scope_select

EXPLICIT_SCOPE_SELECT is of data type SELECT and represents the data types EXPLICIT_SCOPE, FUNCTIONAL_ELEMENT, or PROJECT.

E.2.27 Functional_element

A FUNCTIONAL_ELEMENT is an element of a functional breakdown of platform or series oriented views to a vehicle.

For more information see Scope.

- attribute **id**: identifies the FUNCTIONAL_ ELEMENT
  - ***attribute type:*** string
  - ***cardinality:*** 0,1
- attribute **id_owner**: reference to an organization in which the attribute 'id' is uniquely defined, e.g. the pair (id, id_owner) must be unique for the interpretation of the attribute '.id'
  - ***attribute type:*** ORGANIZATION
  - ***cardinality:*** 0,1
- attribute **version**: specifies a version of this FUNCTIONAL_ELEMENT
  - ***attribute type:*** string
  - ***cardinality:*** 0,1
- attribute **name**: specifies a name for the FUNCTIONAL_ELEMENT
  - ***attribute type:*** string_select
  - ***cardinality:*** 0,1
- attribute **description**: any additional information
  - ***attribute type:*** string_select
  - ***cardinality:*** 0,1
- attribute **belongs_to**: references platforms and/or series that are elements of the FUNCTIONAL_ELEMENT
  - ***attribute type:*** functional_element_select
  - ***cardinality:*** 1.. n

E.2.28 Functional_element_hierarchy

FUNCTIONAL_ELEMENT_HIERARCHY defines a hierarchical relationship between two instances of FUNCTIONAL_ELEMENT. The FUNCTIONAL_ELEMENT_HIERARCHY is used to create hierarchical structures involving FUNCTIONAL_ELEMENT objects.

- attribute **relating_functional_element**: specifies the higher level of the hierarchy
  - ***attribute type:*** FUNCTIONAL_ELEMENT

- ***cardinality:*** 1

- attribute **related_functional_element**: specifies the lower level of the hierarchy
  - ***attribute type:*** FUNCTIONAL_ELEMENT
  - ***cardinality:*** 1
- attribute **description**: any annotations
  - ***attribute type:*** string_select
  - ***cardinality:*** 0,1

E.2.29 functional_element_select

FUNCTIONAL_ELEMENT_SELECT is of data type SELECT and represents the data types PLATFORM, or SERIES.

E.2.30 General_detail

General_DETAIL is a specialization of DETAIL that summarizes all the common detailed information (Figure 24).

Rule: *Every change has a maximum of one instance of GENERAL_DETAIL.*

- attribute **status:** see DETAIL
  - ***attribute type:*** string
  - ***cardinality:*** 0,1
- attribute **creator:** see DETAIL
  - ***attribute type:*** PERSON_ORGANIZATION_WITH_TIMESTAMP
  - ***cardinality:*** 1
- attribute **input_affected_objects:** see DETAIL
  - ***attribute type:*** OBJECT
  - ***cardinality:*** 0 .. n
- attribute **output_affected_objects:** see Detail
  - ***attribute type:*** OBJECT
  - ***cardinality:*** 0 .. n

Note: *Both lists of affected objects in GENERAL_DETAIL only have to be complete if the involved systems in a concrete cooperation cannot interpret the detailed information (SPECIFIC_DETAIL). This has to be defined in the partner agreement.*

- attribute **description:** see DETAIL
- attribute **cost_information:** costs that refer to the ECM as a whole
  - ***attribute type:*** COST_PROPERTY
  - ***cardinality:*** 0 .. n
- attribute **mass_information:** information about weight that refer to the ECM as a whole
  - ***attribute type:*** MASS_PROPERTY

- ***cardinality:*** 0 .. n

- attribute **attachments:** see DETAIL
  - ***attribute type:*** ATTACHMENT
  - ***cardinality:*** 0 .. n
- attribute **further_change_scope:** see DETAIL
  - ***attribute type:*** SCOPE_SELECT
  - ***cardinality:*** 0 .. n
- attribute **tasks:** see DETAIL
  - ***attribute type:*** SCHEDULED_TASK
  - ***cardinality:*** 0 .. n
- attribute **id:** reference to the change object to which the detailed information refers
  - ***attribute type:*** ID
  - ***cardinality:*** 0,1

E.2.31 Header

HEADER contains the general information regarding a product-related issue from the Engineering change (Figure 23).

- attribute **id:** identifier of the engineering change
  - ***attribute type:*** ECM_ID
  - ***cardinality:*** 1

Rule: *id and id_owner as tuple have to be unique, i.e. the identifier has to be unique within a company.*

- attribute **name:** name of the ECM as a short description of the problem
  - ***attribute type:*** string_select
  - ***cardinality:*** 0,1
- attribute **description:** description of the main subject of the engineering change – either as unstructured text or as reference to an instance of CHANGE_DESCRIPTION
  - ***attribute type:*** description_select
  - ***cardinality:*** 1
- attribute **change_process_type:** classification of the change object. Due to different classification criteria, more statements are possible
  - ***attribute type:*** CLASS
  - ***cardinality:*** 0 .. 1

  The following attributes within the class concept have been predefined for classifying an ECM_header:

  - CLASS.id = change classification cases regarding the Reference Process phases (see Part 1 and following)
  - CLASS_ATTRIBUTE.id = classification case (see below)

- CLASS_ATTRIBUTE..attribute_definition = PROPERTY
  CLASS_ATTRIBUTE.allowed_values S[1] = PROPERTY_VALUE_REPRESENTATION
- PROPERTY_VALUE_REPRESENTATION.specified_value = STRING_VALUE
- STRING_VALUE.value_component = one of the predefined values for the classification case (see below)
  PROPERTY_VALUE_REPRESENTATION.definition = PROPERTY
- PROPERTY.property_type = classification case (see below)

Predefined classification cases are:

- 'change type'
- 'reason for change'
- 'effectivity'
- 'change class'
- 'cost absorption'
- 'change process type'

If possible, use one of the following values for STRING_VALUE.value_component:

- classification case **'change type'**
  - **'deviation':** temporary solutions/immediate solutions, which deviates from the released status of the PDM documentation
  - **'process changes':** changes to the manufacturing process (no changes to the PDM documentation)
- .classification case **'reason for change'**
  - **'customer wish':** customer wish (market need, change of requirement specification etc.)
  - **'corrections':** correction of mistakes in technical product documentation, adaptation to reality
  - **'functional improvement':** functional improvement, model upgrading
  - **'legal requirement':** legal requirements, requirements regarding security
  - **'rationalization':** rationalization regarding the fabrication, cost reduction, standardization
  - **'adoption':** adaptations (tolerances, purchase articles, materials..)
  - **'carry over parts':** change of carry-over-parts (COP)
  - **'recycling requirement':** requirements regarding recycling
  - **'after sale requirement':** requirements regarding customer service (improvement of service)
  - **'change of project targets':** change of project targets in the project requirements
  - **'quality':** Improvement of quality
- .classification case **'effectivity'**

  - **'no build':** implement immediately to avoid 'no build' situation at the customer. All products that have not been delivered to the customer must be reworked or scraped.
  - **'urgent change':** implement immediately. It is not necessary to rework finished products
  - **'coordinated change':** implement the change in accordance with the customer's need to coordinate to a specific date or vehicle number.
  - **'running change':** implement the change as soon as possible (but no later than a specific date or at a predefined time after change approval) so that no excess or obsolete material is generated, existing parts are depleted.
- .classification case **'change class'**
  - **'obligatory change request':** changes that need to be applied by the customer
  - **'reportable change request':** changes that need to be reported by the customer
  - **'not reportable change request':** changes that do no have to be reported
- .classification case **'cost absorption'**
  - **'absorption of cost by customer':** customer accepts the costs
  - **'absorption of costs by supplier':** supplier accepts the costs
  - **'renegotiation of costs':** costs are renegotiated
  - **'estimation based approval'**
- classification case **'change process type'**
  - **'fast run':** an early decision is made because of the importance of the change; if necessary, the comment phase can be omitted.
  - **'estimation based approval':** decision about the ECM is based on estimations

- attribute **change_manager:** person or organizational unit at the *coordinator* of a ECM who is responsible for pushing the ECM in the phases of the Reference Process.
  - ***attribute type:*** PERSON_IN_ORGANIZATION
  - ***cardinality:*** 0,1

Rule: *The change_manager is optional only in the early phases of the Reference Process (phase 1 and 2). Subsequently, a manager responsible for controlling and monitoring the process has to be named.*

- attribute **distribution_list:** persons or organizational units who are to be informed about the ECM.
  - ***attribute type:*** PERSON_IN_ORGANIZATION
  - ***cardinality:*** 0 .. n
- attribute **requestor:** initiator of the ECM (person or organizational unit), including date of the suggestion for the change. The ECM can be initiated by either the *coordinator* or any *participant*.
  - ***attribute type:*** PERSON_ORGANIZATION_WITH_TIMESTAMP
  - ***cardinality:*** 0,1

- attribute **creator:** person or organizational unit who created the ECM in phase 2 of the Reference Process in the application system at the *coordinator*, including the creation date
  - ***attribute type:*** PERSON_ORGANIZATION_WITH_TIMESTAMP
  - ***cardinality:*** 0,1
- attribute **coordinator_contact:** persons or organizational units of the *coordinator* with responsibility for the ECM or components of the ECM (construction, purchase, etc.). Their task within the Reference Process is specified in PERSON_IN_ORGANIZATION. role or in ORGANIZATION. organization_type
  - ***attribute type:*** PERSINORG_OR_ORG
  - ***cardinality:*** 0 .. n
- attribute **participant_contact:** persons or organizational units of the *participant* with responsibility for the ECM or components of the ECM (construction, purchase, etc.). Their task within the Reference Process is specified in PERSON_IN_ORGANIZATION. role or in ORGANIZATION. organization_type
  - ***attribute type:*** PERSINORG_OR_ORG
  - ***cardinality:*** 0 .. n
- attribute **attachments:** any additional documents or files specifying the ECM
  - ***attribute type:*** ATTACHMENT
  - ***cardinality:*** 0 .. n
- attribute **status:** current status of the ECM regarding the Reference Process
  - ***attribute type:*** STATUS
  - ***cardinality:*** 1
- attribute **leading_change_scope:** current status of the ECM regarding the Reference Process
  - ***attribute type:*** scope_select
  - ***cardinality:*** 1 .. n

E.2.32 Id

ID identifies a change object in the context of a company. A unique identifier as well as its range of validity (company, in which it was defined) specify this identifier.

- attribute **id:** identifier of the change object
  - ***attribute type:*** string
  - ***cardinality:*** 1
- attribute **id_owner:** organization in which the attribute 'id' is uniquely defined, i.e. the tuple (id, id_owner) must be unique for the interpretation of the attribute 'id'
  - ***attribute type:*** ORGANIZATION
  - ***cardinality:*** 1

Rule: *ID_id and id_owner as tuple have to be unique, i.e. the identifier is unique within a company.*

- attribute **process_phase:** specifies the related subprocess within the ECM Reference Process
  - ***attribute type:*** string
  - ***cardinality:*** 1
- attribute **header (INV):** inverse attribute as a set of HEADER that reference the ID as 'id'
  - ***attribute type:*** HEADER
  - ***cardinality:*** 0,1

  Rule: *Each ECM must uniquely reference one ID by its header (HEADER). On the other hand, each instance of ID may be referenced by a maximum of one instance of HEADER.*
- attribute **general_detail (INV):** inverse attribute as a set of GENERAL_DETAIL that reference the ID as 'id'
  - ***attribute type:*** GENERAL_DETAIL
  - ***cardinality:*** 0,1

  Rule: *Each ECM must uniquely reference one ID by its general detail (General_detail). On the other hand, each instance of ID may be referenced by a maximum of one instance of GENERAL_DETAIL.*

E.2.33 Id_or_detail_select

ID_OR_DETAIL_SELECT is of data type SELECT and represents the data types ID or DETAIL.

E.2.34 Id_relationship

ID_RELATIONSHIP specifies the relationship between two instances of ID. Typical applications are e.g., the allocation of HEADER, the dependency between two engineering changes or the subdivision of one ECM into a number of ECMs.

- attribute **relating_id:** specifies the first ID in the relation. The role is defined by .relation_type
  - ***attribute type:*** ID
  - ***cardinality:*** 1
- attribute **related_id:** specifies the second ID in the relation. The role is defined by .relation_type
  - ***attribute type***: ID
  - ***cardinality:*** 1
- attribute **relation_type:** describes the type of relationship
  - ***attribute type:*** string
  - ***cardinality:*** 1

  If possible, use one of the following values for the attribute .relation_type:

- o **'decomposition':** 'related_id' is a component of 'relating_id' without any statements regarding the order or dependence
- o **'dependency':** 'related_id' depends on 'relating_id'
- o **'precedence':** 'related_id' has a higher priority than 'relating_id'
- o **'sequence':** 'relating_id' is always in front of the 'related_id' concerning the order
- o **'simultaneity':** 'relating_id' and 'related_id' identify two ECM identifiers (.id, .id_owner) for two change projects which are considered as occurring during the same time period or shall be performed together in order to ensure consistency and enhance efficiency
- o **'supplied id':** defines a relationship between two id objects representing the same object in different organizational contexts. The 'relating_id' is in the organizational context of the coordinator while the 'related_id' is same in the organizational context of the participant
- o **'initial ECM to ECM':** 'relating_id' identifies an initial ECM and the 'related_id' identifies an ECM representing the accepted initial ECM at the *coordinator*

- attribute **description:** any annotations
  - ***attribute type:*** string
  - ***cardinality:*** 0,1

E.2.35 Implicit_scope

IMPLICIT_SCOPE defines a scope for the change without using the predefined scopes of EXPLICIT_SCOPE but using the SIMPLE_PROPERTY_ASSOCIATION. IMPLICIT_SCOPE and EXPLICIT_SCOPE may be used alternatively.

E.2.36 Language

LANGUAGE specifies a language in which an information is given.

- attribute **language_code**: the language of the text information in the Alpha-3 bibliographic code specified in ISO 639-2
  - ***attribute type:*** string
  - ***cardinality:*** 1

Example: *Possible values for language_code are, e.g., 'eng' for English, 'fre' for French, 'rus' for Russian, or 'ger' for German.*

- attribute **country_code**: the country, as addition to the language, according to the alpha-2 code specified in ISO 3166-1.
  - ***attribute type:*** string
  - ***cardinality:*** 0,1

Example: *Possible values for country_code are, e.g., 'GB' for the United Kingdom or 'US' for the United States of America.*

E.2.37 Mass_property

MASS_PROPERTY specifies a statement regarding the weight. The value can be classified using calculated, estimated, and absolute or delta.

- attribute **mass_value**: value of the mass
  - ***attribute type:*** string
  - ***cardinality:*** 1
- attribute **description**: any annotations
  - ***attribute type:*** string_select
  - ***cardinality:*** 0,1
- attribute **attachments**: any additional documents or files specifying the mass property
  - ***attribute type:*** ATTACHMENT
  - ***cardinality:*** 0 .. n

  If possible, use one of the following values for PROPERTY_VALUE_REPRESENTATION.qualifier:

  - **'absolute'**: The specified weight value represents a weight
  - **'delta'**: The specified weight value represents a change in weight

  Note: *The kind of mass is specified in the attribute 'name' of PROPERTY referenced by the attribute 'definition' of PROPERTY_VALUE_REPRESENTATION.*

  Note: *A distinction between calculated and estimated mass information can be specified in PROPERTY_VALUE_REPRESENTATION.value_determination.*

  If possible, use one of the following values for PROPERTY_VALUE_REPRESENTATION. value_ determination:

  - **'calculated'**: the specified mass value has been calculated
  - **'estimated'**: the specified mass value has been estimated

E.2.38 Model_type

MODEL_TYPE is a specialization of EXPLICIT_SCOPE and specifies a model type of a vehicle.

For more information see EXPLICIT_SCOPE.

- attribute **id**: see EXPLICIT_SCOPE
- attribute **version**: see EXPLICIT_SCOPE
- attribute **name**: see EXPLICIT_SCOPE
- attribute **description**: see EXPLICIT_SCOPE
- attribute **id_owner**: see EXPLICIT_SCOPE

E.2.39 Module

MODULE is a specialization of EXPLICIT_SCOPE and specifies a functional unit in a vehicle. A module in an engineering change context serves to allocate the change, i.e. describe the scope of the validity of the change.

The classification into "modules" corresponds to a functional grid of the vehicle - **not** a geometrical allocation, i.e. a module does not have to be a coherent component.

Since the parts list (list in the sense of an amount (SET)) is unstructured in the Change Request, the list with affected modules will not appear in the parts list.

For more information see EXPLICIT_SCOPE.

- attribute **id**: see EXPLICIT_SCOPE
- attribute **version**: see EXPLICIT_SCOPE
- attribute **name**: not allocated
- attribute **description**: see EXPLICIT_SCOPE
- attribute **id_owner**: see EXPLICIT_SCOPE

E.2.40 Multi_language_string

MULTI_LANGUAGE_STRING specifies text information, expressed in one or more languages.

- attribute **primary_language_dependent_string**: text information in the original language
  - ***attribute type:*** STRING_WITH_LANGUAGE
  - ***cardinality:*** 1
- attribute **additional_language_dependent_string**: text information in a particular language
  - ***attribute type:*** STRING_WITH_LANGUAGE
  - ***cardinality:*** 0 .. n

E.2.41 Numerical_value

NUMERICAL_VALUE is a specialization of a PROPERTY_VALUE and defines a numerical value.

- attribute **value_name**: see PROPERTY_VALUE
- attribute **value_component**: floating point number or integer
  - ***attribute type:*** number
  - ***cardinality:*** 1
- attribute **unit:** unit of the numerical value
  - ***attribute type:*** UNIT
  - ***cardinality:*** 1

E.2.42 Object

OBJECT is a placeholder for identifying product data that is affected by the ECM subprocess. It includes parts, assemblies, vehicles, tools, their use, as well as documents. An Instantiation can only take place in one of its specializations.

- attribute **id:** identifies the object
  - ***attribute type:*** string
  - ***cardinality:*** 1
- attribute **id_owner:** organization in which the attribute 'id' is uniquely defined, i.e. the pair (id, id_owner) must be unique for the interpretation of the attribute '.id'
  - ***attribute type:*** ORGANIZATION
  - ***cardinality:*** 1
- attribute **version:** version of the object
  - ***attribute type:*** string
  - ***cardinality:*** 0,1
- attribute **name:** name of the object
  - ***attribute type:*** string_select
  - ***cardinality:*** 0,1
- attribute **description:** any annotations
  - ***attribute type:*** string_select
  - ***cardinality:*** 0,1
- attribute **cost_information:** costs referring to an object
  - ***attribute type:*** COST_PROPERTY
  - ***cardinality:*** 0 .. n
- attribute **mass_information:** weight information referring to an object
  - ***attribute type:*** MASS_PROPERTY
  - ***cardinality:*** 0 .. n
- attribute **classified_as:** classification of objects
  - ***attribute type:*** CLASS
  - ***cardinality:*** 0 .. n

To classify OBJECTS the following structure and fixed values must be used:

CLASS.id = **'object classification'**

CLASS_ATTRIBUTE.id = **''** (nothing predefined)

CLASS_ATTRIBUTE.allowed_values S[1] = PROPERTY_VALUE_REPRESENTATION

PROPERTY_VALUE_REPRESENTATION.specified_value = STRING_VALUE

STRING_VALUE.value_component = (nothing predefined)

PROPERTY_VALUE_REPRESENTATION.definition = PROPERTY

PROPERTY.property_type = '' (nothing predefined)

- attribute **attachments:** documents and/or files completing the statements regarding the weight
  - ***attribute type:*** ATTACHMENT
  - ***cardinality:*** 0 .. n
- attribute **status:** development status of the object
  - ***attribute type:*** string
  - ***cardinality:*** 0,1

Note: *The Object status depends on company-specific development processes. Therefore it must be agreed upon between the partners of the engineering processes.*

E.2.43 Object_relationship

OBJECT_RELATIONSHIP specifies a relationship between a part and a part instance or a document.

- attribute **relating_object:** identifies the part in the OBJECT_RELATIONSHIP
  - ***attribute type:*** Part
  - ***cardinality:*** 1
- attribute **related_object:** identifies the instance of a part or of a document that is related to the part
  - ***attribute type:*** object_select
  - ***cardinality:*** 1
- attribute **relation_type:** specifies the type of relationship
  - ***attribute type:*** string
  - ***cardinality:*** 1

Use one of the following values:

  - o **'assembly'**: the 'related_object' is of data type PART_INSTANCE and component of the assembly specified by 'relating_object'
  - o **'documentation'**: the 'related_object' is of data type Document and documentation of a part specified by 'relating_object'

- attribute **description:** any annotations
  - ***attribute type:*** string_select

- ***cardinality:*** 0,1

E.2.44 object_select

OBJECT_SELECT is of data type SELECT and represents the data types PART_INSTANCE or DOCUMENT.

E.2.45 Organization

ORGANIZATION identifies a group of people who are involved in a defined business process.

- attribute **id:** identifies the organization
  - ***attribute type:*** string
  - ***cardinality:*** 1
- attribute **name:** name of the organization
  - ***attribute type:*** string
  - ***cardinality:*** 1
- attribute **organization_type:** describes the type of organization
  - ***attribute type:*** string
  - ***cardinality:*** 0,1

  If possible, use one of the following values:
  - **'company'**: The organization represents a company.
  - **'department'**: The organization represents a department.
  - **'plant'**: The organization represents a plant or a location.
- attribute **address:** address or component of address (e.g. phone number)
  - ***attribute type:*** ADDRESS
  - ***cardinality:*** 0,1

E.2.46 Organization_relationship

ORGANIZATION_RELATIONSHIP specifies a relationship between two instances of ORGANIZATION. Typical applications are, for example, the description of:

- a organizational structure
- a reorganization of the organizational structure or
- a change of the responsibilities
- attribute **relating_organization:** specifies the first organization in the relationship
  - ***attribute type:*** ORGANIZATION
  - ***cardinality:*** 1
- attribute **related_organization:** specifies the first organization in the relationship

- ***attribute type:*** ORGANIZATION
- ***cardinality:*** 1

- attribute **relation_type:** describes the type of organization
  - ***attribute type:*** string
  - ***cardinality:*** 1

  If possible, use one of the following values:

  - **'hierarchy'**: The related_organization is a suborganization of the relating_organization.
  - **'reorganization'**: The related_organization is a successor of the relating_organization due to a change in responsibilities.
  - **'legal succession'**: The related_organization is a legal successor of the relating_organization.

- attribute **description:** any annotations
  - ***attribute type:*** string_select
  - ***cardinality:*** 0,1

E.2.47 Part

PART is a specialization of OBJECT for the specification of parts within an assembly structure which are marked as affected within a change process. They represent single parts, complete assemblies, vehicles or tools.

- attribute **id:** see OBJECT
- attribute **id_owner:** see OBJECT
- attribute **version:** see OBJECT
- attribute **name:** see OBJECT
- attribute **description:** see OBJECT
- attribute **cost_information:** see OBJECT
- attribute **mass_information:** see OBJECT
- attribute **classified_as:** see OBJECT
- attribute **attachments:** see OBJECT
- attribute **status:** see OBJECT

E.2.48 Part_instance

PART_INSTANCE is a specialization of OBJECT specifying the usage of a part within an assembly structure which is marked as affected by a change process.

- attribute **id:** see OBJECT
- attribute **id_owner:** see OBJECT

- attribute **version:** see OBJECT
- attribute **name:** see OBJECT
- attribute **description:** see OBJECT
- attribute **cost_information:** see OBJECT
- attribute **mass_information:** see OBJECT
- attribute **classified_as:** see OBJECT
- attribute **attachments:** see OBJECT
- attribute **status:** see OBJECT
- attribute **definition:** part for which usage is described
  - ***attribute type:*** PART
  - ***cardinality:*** 1

E.2.49 Person

PERSON specifies an employee who is involved in a business process.

Rule: An instance of person may only be referenced by PERSON_IN_ORGANIZATION.PERSON.

- attribute **name:** name of the person
  - ***attribute type:*** string
  - ***cardinality:*** 1
- attribute **address:** address or component of address (e.g. phone number)
  - ***attribute type:*** ADDRESS
  - ***cardinality:*** 0,1

E.2.50 PersInOrg_or_Org

PERSINORG_OR_ORG is of data type SELECT and represents the data types PERSON_IN_ORGANIZATION or ORGANIZATION.

E.2.51 Person_in_organization

PERSON_IN_ORGANIZATION specifies an employee as a member of an organizational unit which is involved in a business process. The context of the organization is completed by a role played by the person in the business process - or a specific component of it, e.g. as commentator.

- attribute **id:** identifier of the person; this identifier should be unique within the organization
  - ***attribute type:*** string
  - ***cardinality:*** 0,1
- attribute **role:** labels the role which a person plays within a usage
  - ***attribute type:*** string

- ***cardinality:*** 0,1

If possible, use one of the following values:

- **'creator'**: creates e.g. the ECR in the ECM application system of the *Coordinator*
- **'EC manager'**: employee responsible for e.g. the ECR at the *Coordinator*
- **'EC manager subst'**: substitute for 'EC_manager'
- **'contact'**: contact person
- **'contact subst'**: substitute for 'contact'
- **'initiator'**: initiator or trigger of the ECR
- **'initiator subst'**: substitute for 'initiator'

- attribute **person:** specifies the person
  - ***attribute type:*** PERSON
  - ***cardinality:*** 1
- attribute **organization:** specifies the organization to which the person is related
  - ***attribute type:*** ORGANIZATION
  - ***cardinality:*** 1
- attribute **location:** specifies the relevant address of the PERSON_IN_ORGANIZATION
  - ***attribute type:*** ADDRESS
  - ***cardinality:*** 0,1

E.2.52 Person_in_organization_relationship

PERSON_IN_ORGANIZATION_RELATIONSHIP specifies the relationship between two instances of PERSON_IN_ORGANIZATION. Typical applications are the description of, for example:

- structure of responsibilities
- procuration or
- changes in human resources

PERSON_IN_ORGANIZATION_RELATIONSHIP is an additional representation of person-specific roles and their use in contacts and/or specific assignments in PERSON_ORGANIZATION_ ASSIGNMENT.

- attribute **relating_person:** specifies the first person in the relationship. The role is determined by 'relation_type'
  - ***attribute type:*** PERSON_IN_ORGANIZATION
  - ***cardinality:*** 1
- attribute **related_ person:** specifies the second person in the relationship. The role is determined by 'relation_type'

- ***attribute type:*** PERSON_IN_ORGANIZATION
- ***cardinality:*** 1

- attribute **relation_type:** describes the type of relation between two persons
  - ***attribute type:*** DATE
  - ***cardinality:*** 1

  If possible, use one of the following values:
  - **'superior'**: 'related_person' is a superior of the 'relating_ person'
  - **'deputy'**: 'related_person' is a deputy of the 'relating_ person'
  - **'successor'**: 'related_person' is a successor of the 'relating_ person'
- attribute **description:** any annotations
  - ***attribute type:*** string_select
  - ***cardinality:*** 0,1

E.2.53 Person_organization_assignment

PERSON_ORGANIZATION_ASSIGNMENT connects an organization or a person in an organization with product data if they are not directly referenced by this data. The allocation assigns a special usage role.

This assignment allows the specification of additional role-based information to data objects.

Note: *PERSON_ORGANIZATION_ASSIGNMENT is an abstract data type that has to be completed in an application data model (e.g. the ECM Data Model). In the application model, a subtype of PERSON_ORGANIZATION_ASSIGNMENT specifies all data objects that can be added by role-based information.*

- attribute **person_organization:** specifies an organization or a person in an organization
  - ***attribute type:*** PersInOrg_or_Org (ORGANIZATION or PERSON_IN_ORGANIZATION)
  - ***cardinality:*** 1
- attribute **role:** specifies the role the person or the organization plays
  - ***attribute type:*** string
  - ***cardinality:*** 1

  If possible, use one of the following values:
  - **'creator'**: creates e.g. the initial engineering change in the ECM application system of the *Coordinator*
  - **'EC manager'**: employee responsible for e.g. the engineering change at the *Coordinator*
  - **'EC manager subst'**: substitute for 'EC_manager'
  - **'contact'**: contact person

- **'contact subst'**: substitute for 'contact'
- **'initiator'**: initiator or trigger of e.g. the ECR
- **'initiator subst'**: substitute for 'initiator'

- attribute **is_applied_to:** specifies the object with which the PERSON_ORGANIZATION_ASSIGNMENT is associated
  - ***attribute type:*** person_organization_assignment_select (PART, CLASS_SYSTEM, SCHEDULED_TASK, or DETAIL)
  - ***cardinality:*** 1 .. n

E.2.54 person_organization_assignment_select

PERSON_ORGANIZATION_ASSIGNMENT_SELECT is of data type SELECT and represents the data types PART, CLASS_SYSTEM, SCHEDULE_TASK, or DETAIL.

E.2.55 Person_organization_with_timestamp

PERSON_ORGANIZATION_WITH_TIMESTAMP represents a signature, which an individual person or a structure unit playing a defined role ('role' of PERSON_IN_ORGANIZATION or 'organization_type' OF ORGANIZATION) releases at a certain date.

Note: *An implementation satisfies the requirements of a signature e.g., if the instantiation of PERSON_ORGANIZATION_WITH_TIMESTAMP is generated automatically by a function within the graphical user interface which prevents any kind of manipulation.*

- attribute **person_organization:** specifies an organization or a person in an organization
  - ***attribute type:*** PersInOrg_or_Org (ORGANIZATION or PERSON_IN_ORGANIZATION)
  - ***cardinality:*** 1
- attribute **actual_date:** date of signature instantiation
  - ***attribute type:*** DATE
  - ***cardinality:*** 1

E.2.56 Platform

PLATFORM is a specialization of EXPLICIT_SCOPE and represents a top-level structure category of vehicles.

For more information, see EXPLICIT_SCOPE.

- attribute **id:** see EXPLICIT_SCOPE
- attribute **version:** see EXPLICIT_SCOPE
- attribute **name:** see EXPLICIT_SCOPE
- attribute **description:** see EXPLICIT_SCOPE
- attribute **id_owner:** see EXPLICIT_SCOPE

E.2.57 Project

PROJECT specifies a work program (e.g. for the development of a new vehicle). A project in the ECM context serves to allocate the change, i.e. it describes the range of validity of the change.

- attribute **id:** identifies the project
  - ***attribute type:*** string
  - ***cardinality:*** 0,1
- attribute **id_owner:** organization in which the attribute '.id' is uniquely defined, i.e. the pair (id, id_owner) must be unique for the interpretation of the attribute '.id'
  - ***attribute type:*** ORGANIZATION
  - ***cardinality:*** 0,1
- attribute **name:** name of the project
  - ***attribute type:*** string_select
  - ***cardinality:*** 0,1
- attribute **description:** any annotations
  - ***attribute type:*** string_select
  - ***cardinality:*** 0,1
- attribute **planned_start_date:** planned starting date of the project specified as a date or as an event
  - ***attribute type:*** event_or_date_select
  - ***cardinality:*** 0,1
- attribute **planned_end_date:** planned end date of the project specified as a date, an event, or a period
  - ***attribute type:*** event_period_or_date_select
  - ***cardinality:*** 0,1
- attribute **actual_start_date:** actual starting date of the project specified as a date
  - ***attribute type:*** DATE
  - ***cardinality:*** 0,1
- attribute **actual_end_date:** actual end date of the project specified as a date
  - ***attribute type:*** DATE
  - ***cardinality:*** 0,1
- attribute **leads_to:** references the target of the project as scope or functional element
  - ***attribute type:*** PROJECT_SELECT

- ***cardinality:*** 0 .. n

E.2.58 Project_hierarchy

PROJECT _HIERARCHY defines a hierarchical relationship between two instances of PROJECT. The PROJECT _HIERARCHY is used to create hierarchical structures involving PROJECT objects.

- attribute **relating_project:** specifies the higher level of the hierarchy
  - ***attribute type:*** PROJECT
  - ***cardinality:*** 1
- attribute **related_project:** specifies the lower level of the hierarchy
  - ***attribute type:*** PROJECT
  - ***cardinality:*** 1
- attribute **description:** any annotations
  - ***attribute type:*** string_select
  - ***cardinality:*** 0,1

E.2.59 project_select

PROJECT_SELECT is of data type SELECT and represents the data types EXPLICIT_SCOPE, IMPLICIT_SCOPE, or FUNCTIONAL_ELEMENT.

E.2.60 Property

PROPERTY is the definition of a particular quality.

- attribute **property_type:** type of the attribute
  - ***attribute type:*** string
  - ***cardinality:*** 1

  Attribute values are partner-specific. Examples are:

    - **'units per year'**: as a numerical value
    - **'sample required',**
    - **' sample done',**
    - **'test needed',**
    - **'test done',**
    - **'package relevant',**
    - **'series affected',**
    - **'certification relevant'**: as a logical value

- attribute **description:** any annotations
  - ***attribute type:*** string_select
  - ***cardinality:*** 0,1
- attribute **attachments:** documents or files which complete the indication of the property
  - ***attribute type:*** ATTACHMENT
  - ***cardinality:*** 0 .. n

E.2.61 property_select

PROPERTY_SELECT is of data type SELECT and represents the data types ID, HEADER, DETAIL, COMMENT, ACCEPTANCE, OBJECT, ATTACHMENT, or SCOPE_SELECT.

E.2.62 Property_value

PROPERTY_VALUE is the numerical or textual value of a PROPERTY_VALUE_REPRESENTATION.

Rule: PROPERTY_VALUE may only be instantiated in one of its specializations (NUMERICAL_VALUE, VALUE_RANGE, STRING_VALUE, or LOGICAL_VALUE).

- attribute **value_name:** specifies the word or group of words used to refer to the property value
  - ***attribute type:*** string
  - ***cardinality:*** 0,1

E.2.63 Property_value_association

PROPERTY_VALUE_ASSOCIATION is a mechanism for assigning a PROPERTY_VALUE_ REPRESENTATION to an object.

Note: *Property_value_association is an abstract data type and has to be completed in an application data model (e.g. the ECM Data Model). In the application model a subtype of Property_value_association specifies all data objects that can be added by property values.*

- attribute **describing_ property_value:** specifies the value that is being assigned
  - ***attribute type:*** PROPERTY_VALUE_ REPRESENTATION
  - ***cardinality:*** 1
- attribute **described_element:** specifies the object that is described by the PROPERTY_VALUE_ASSOCIATION
  - ***attribute type:*** property_select (ID, HEADER, DETAIL, COMMENT, ACCEPTANCE, OBJECT, ATTACHMENT, IMPLICIT_SCOPE, EXPLICIT_SCOPE, FUNCTIONAL_ELEMENT, or PROJECT)
  - ***cardinality:*** 1 .. n

E.2.64 Property_value_representation

PROPERTY_VALUE_REPRESENTATION is the representation of Property.

- attribute **value_determination:** see PROPERTY_VALUE
  - ***attribute type:*** string
  - ***cardinality:*** 0,1
- attribute **definition:** specifies the property that has to be characterized
  - ***attribute type:*** PROPERTY
  - ***cardinality:*** 1
- attribute **specified_value:** specifies the property value that qualifies the representation
  - ***attribute type:*** PROPERTY_VALUE
  - ***cardinality:*** 1
- attribute **qualifier:** specifies the kind of the representation
  - ***attribute type:*** string
  - ***cardinality:*** 0,1

  If possible, use one of the following values:
  - **'nominal'**: The value is the nominal value.
  - **'specified'**: The value is specified.
  - **'typical'**: The value is a typical value.
- attribute **value_determination:** specifies information on how the representation is to be interpreted
  - ***attribute type:*** string
  - ***cardinality:*** 0,1

  If possible, use one of the following values:
  - **'calculated'**: The value has been calculated.
  - **'designed'**: The value represents a value intended by the design.
  - **'estimated'**: The value has been estimated.
  - **'measured'**: The value has been measured.
  - **'required'**: The value represents a requirement.
  - **'set point'**: The value is used as the initialization value.

E.2.65 Property_value_representation_relationship

PROPERTY_VALUE_REPRESENTATION_RELATIONSHIP is a relationship between two PROPERTY_VALUE objects.

- attribute **relating:** specifies the first of the two objects in the relationship
  - ***attribute type:*** PROPERTY_VALUE_ REPRESENTATION
  - ***cardinality:*** 1
- attribute **related:** specifies the second of the two objects in the relationship
  - ***attribute type:*** PROPERTY_VALUE_ REPRESENTATION
  - ***cardinality:*** 1
- attribute **relation_type:** specifies the meaning of the relationship
  - ***attribute type:*** string
  - ***cardinality:*** 1

  If possible, use one of the following values:
  - **'decomposition'**: defines a relationship in which the related object is one of what may be several subvalues of the relating object
  - **'dependency'**: defines a derived relationship where the related object is based on the relating object
  - **'equivalence'**: defines a relationship in which the related object represents the same thing as the relating object
  - **'substitution'**: defines a relationship in which the related object replaces the relating object
- attribute **description:** any annotations
  - ***attribute type:*** string_select
  - ***cardinality:*** 0,1

E.2.66 Scheduled_task

SCHEDULED_TASK describes a time-phased task in the context of ECM time management and is a component detailing the engineering process.

- attribute **task_name:** name of the task
  - ***attribute type:*** string
  - ***cardinality:*** 1

Note: *Because of company-specific naming conventions, values specified for .task_name are partner specific.*

- attribute **description:** any annotations
  - ***attribute type:*** string_select
  - ***cardinality:*** 0,1
- attribute **responsible:** person or organization responsible for performing the task

- ***attribute type:*** PersInOrg_or_Org
- ***cardinality:*** 0,1

• attribute **performer:** person or organization performing the task
- ***attribute type:*** PersInOrg_or_Org
- ***cardinality:*** 0,1

• attribute **planned_start_-date:** date or event at which the task should be started
- ***attribute type:*** event_or_date_select
- ***cardinality:*** 0,1

• attribute **planned_end_-date:** date, event or duration at/after which the task should be executed
- ***attribute type:*** event_period_or_date_select
- ***cardinality:*** 0,1

• attribute **actual_start_date:** date at which the task was started
- ***attribute type:*** DATE
- ***cardinality:*** 0,1

• attribute **actual_end_date:** date at which the task was finished
- ***attribute type:*** DATE
- ***cardinality:*** 0,1

E.2.67 Scope_relationship

SCOPE_RELATIONSHIP is a relationship between two SCOPE objects (EXPLICIT_SCOPE or IMPLICIT_SCOPE).

• attribute **relating_scope:** specifies the first scope in the relationship
- ***attribute type:*** SCOPE_SELECT
- ***cardinality:*** 1

• attribute **related_scope:** specifies the second scope in the relationship
- ***attribute type:*** SCOPE_SELECT
- ***cardinality:*** 1

• attribute **relation_type:** specifies the meaning of the relationship
- ***attribute type:*** string_select
- ***cardinality:*** 1

If possible, use one of the following values:

- o **'hierarchy'**: defines a hierarchical relationship in which the related scope is a subordinate of the relating scope
- o **'decomposition'**: specifies a relationship in which the related scope is one of what may be several subvalues of the relating scope

- attribute **description:** any annotations
  - ***attribute type:*** string
  - ***cardinality:*** 0,1

E.2.68 scope_select

SCOPE_SELECT is of data type SELECT and represents the data types IMPLICIT_SCOPE or EXPLICIT_SCOPE_SELECT.

E.2.69 Series

SERIES is a specialization of Explicit_SCOPE and specifies a production program. A series in the ECM context serves to allocate the change, i.e. describe the range of validity of the change.

- attribute **id:** see EXPLICIT_SCOPE
- attribute **version:** see EXPLICIT_SCOPE
- attribute **name:** not allocated
- attribute **description:** see EXPLICIT_SCOPE
- attribute **id_owner:** see EXPLICIT_SCOPE

E.2.70 Simple_property_association

SIMPLE_PROPERTY_ASSOCIATION is a association of PROPERTY_VALUE instances to some data objects.

- attribute **specified_values:** References a set of property values.
  - ***attribute type:*** PROPERTY_VALUE
  - ***cardinality:*** 1 .. n
- attribute **value_type:** specifies the defined property
  - ***attribute type:*** STRING
  - ***cardinality:*** 1
- attribute **described_element:** specifies the data type the property is associated to
  - ***attribute type:*** SIMPLE_PROPERTY_SELECT
  - ***cardinality:*** 1

E.2.71 simple_property_select

SIMPLE_PROPERTY_SELECT IS OF DATA TYPE SELECT AND REPRESENTS THE DATA TYPES ID, HEADER, DETAIL, COMMENT, ACCEPTANCE, OBJECT, ATTACHMENT, OR SCOPE_SELECT.

E.2.72 Specific_detail

SPECIFIC_DETAIL is a specialization of DETAIL which describes only the detailed information which, for example,

- differs in a subset of the affected objects ('(input/output_)affected_objects') from the details of the associated GENERAL_DETAIL (date, cost center, etc.),
- specifies a relationship between the 'input_affected_objects' and the 'output_affected_objects' (replacement of objects, etc.),
- assigns specific information to single affected objects (e.g. when an affected object has been identified by a partner (Figure 24).

Rule: *One ECM can have any number of ECM_SPECIFIC_DETAIL. Every ECM_SPECIFIC_DETAIL is implicitly (in the form of the same ECM) assigned to exactly one ECM_GENERAL_DETAIL.*

- attribute **status:** see DETAIL
  - ***attribute type:*** string
  - ***cardinality:*** 0,1
- attribute **creator:** see DETAIL
  - ***attribute type:*** PERSON_ORGANIZATION_WITH_TIMESTAMP
  - ***cardinality:*** 1
- attribute **input_affected_objects:** see DETAIL
  - ***attribute type:*** OBJECT
  - ***cardinality:*** 0 .. n
- attribute **output_affected_objects:** see DETAIL
  - ***attribute type:*** OBJECT
  - ***cardinality:*** 0 .. n
- attribute **description:** see DETAIL
  - ***attribute type:*** string
  - ***cardinality:*** 0,1
- attribute **cost_information:** see DETAIL
  - ***attribute type:*** COST_PROPERTY
  - ***cardinality:*** 0 .. n
- attribute **mass_information:** see DETAIL
  - ***attribute type:*** MASS_PROPERTY
  - ***cardinality:*** 0 .. n
- attribute **attachments:** see DETAIL
  - ***attribute type:*** ATTACHMENT
  - ***cardinality:*** 0 .. n
- attribute **further_change_scope:** see DETAIL

- ***attribute type:*** STATUS
- ***cardinality:*** 1

- attribute **tasks:** see DETAIL
  - ***attribute type:*** SCHEDULED_TASK
  - ***cardinality:*** 0 .. n
- attribute **id:** see DETAIL
  - ***attribute type:*** ECM_ID
  - ***cardinality:*** 1
- attribute **position:** identifies change positions
  - ***attribute type:*** string
  - ***cardinality:*** 0,1
- attribute **change_type:** describes the type of change which has to be carried out with regard to the affected objects
  - ***attribute type:*** CLASS
  - ***cardinality:*** 0,1

The following can be used to define the kind of a comment (Instantiation example see Part 0 of this recommendation, Figure 17):

- CLASS.id = **'specific detail classification'**
- CLASS_ATTRIBUTE.id = **'specific detail change type'**
- CLASS_ATTRIBUTE.allowed_values S[1] = PROPERTY_VALUE_REPRESENTATION
- PROPERTY_VALUE_REPRESENTATION.specified_value = STRING_VALUE
- STRING_VALUE.value_component = one of the predefined values below
- PROPERTY_VALUE_REPRESENTATION.definition = PROPERTY
- PROPERTY.property_type = **' specific detail change type '**

If possible, use one of the following values for STRING_VALUE.value_component:

- **'design release':** authorization for the development of a part or a document.
- **'manufacturing release':** authorization for a manufacturing process of a part.
- **'new':** additional (new) part for an assembly

  Rule: *The number of 'input_affected_objects' must be zero.*
- **'revise':** changes to a part which is in use
- **'unchanged':** no change required
- **'delete':** part is void for the assembly
- **'replace':** 'output_affected_objects' will replace the 'input_affected_objects' with the engineering change

E.2.73 Specific_detail_relationship

SPECIFIC_DETAIL_RELATIONSHIP specifies the relationship between two SPECIFIC_DETAIL instances (Figure 24).

- attribute **relating_specific_detail:** specifies the first instance of SPECIFIC_DETAIL
  - ***attribute type:*** SPECIFIC_DETAIL
  - ***cardinality:*** 1
- attribute **related_specific_detail:** specifies the second instance of SPECIFIC_DETAIL
  - ***attribute type:*** SPECIFIC_DETAIL
  - ***cardinality:*** 1
- attribute **relation_type:** describes the type of relationship
  - ***attribute type:*** string
  - ***cardinality:*** 1

Use the following value:

- **'sequence':** defines a sequence in which 'relating_specific_detail' is the preceding object of the relationship and 'related_specific_detail' is the succeeding object
- attribute **description:** any annotations
  - ***attribute type:*** string_select
  - ***cardinality:*** 0,1

E.2.74 Status

STATUS specifies the processing status of the change process with regard to the phase-specific Reference Process.

- attribute **status:** describes the processing status
  - ***attribute type:*** string
  - ***cardinality:*** 1

Note: *The values of 'status' depend on the process they are used in. Therefore it is specified in the phase-specific Data Models using STATUS.*

- attribute **approved_by:** person and/or organizational unit responsible for determining the status
  - ***attribute type:*** PERSON_ORGANIZATION_WITH_TIMESTAMP
  - ***cardinality:*** 0,1
- attribute **planned_date:** date at which it was planned that the respective status be achieved
  - ***attribute type:*** DATE
  - ***cardinality:*** 0,1
- attribute **actual_date:** date at which the status was actually achieved

- ***attribute type:*** DATE
- ***cardinality:*** 0,1

E.2.75 Status_relationship

STATUS_RELATIONSHIP is a relationship between two STATUS objects.

- attribute **relating_status:** describes the first status in the STATUS_RELATIONSHIP
  - ***attribute type:*** STATUS
  - ***cardinality:*** 1
- attribute **related_status:** describes the second status in the STATUS_RELATIONSHIP
  - ***attribute type:*** STATUS
  - ***cardinality:*** 1
- attribute **relation_type:** specifies the meaning of the relationship
  - ***attribute type:*** string
  - ***cardinality:*** 1

  If possible, use one of the following values:
  - **'sequence'**: defines a relationship in which the relating STATUS must be completed before the related STATUS is specified
- attribute **description:** any annotations
  - ***attribute type:*** string_select
  - ***cardinality:*** 0,1

E.2.76 string_select

STRING_SELECT is of data type SELECT and represents the data types MULTI_LANGUAGE_-STRING or DEFAULT_LANGUAGE.

E.2.77 String_with_language

STRING_WITH_LANGUAGE specifies text information in a specific language together with an identification of the language used.

- attribute **contents**: the textual information stored in the language identified by the language_specification attribute
  - ***attribute type:*** string
  - ***cardinality:*** 1
- attribute **language_specification**: the language in which the contents is given
  - ***attribute type:*** LANGUAGE
  - ***cardinality:*** 1

E.2.78 String_value

STRING_VALUE is a specialization of PROPERTY_VALUE and defines an alphanumerical text.

- attribute **value_name:** see PROPERTY_VALUE
- attribute **value_component:** any text
  - ***attribute type:*** string
  - ***cardinality:*** 1

E.2.79 Unit

UNIT specifies the unit of measurement for a value.

- attribute **name:** name of the unit of measurement
  - ***attribute type:*** string
  - ***cardinality:*** 1
- attribute **prefix:** factor for the measured value
  - ***attribute type:*** string
  - ***cardinality:*** 0,1

  If possible, use one of the following values:
  - **'milli',**
  - **'kilo',**
  - **'mega'**

E.2.80 Value_range

VALUE_RANGE is a specialization of PROPERTY_VALUE and defines a numerical range of values.

- attribute **value_name:** see PROPERTY_VALUE
- attribute **upper_limit:** upper limit of the range of values
  - ***attribute type:*** NUMBER
  - ***cardinality:*** 1
- attribute **lower_limit:** lower limit of the range of values
  - ***attribute type:*** NUMBER
  - ***cardinality:*** 1
- attribute **unit:** unit of the numerical value
  - ***attribute type:*** UNIT
  - ***cardinality:*** 1

Appendix F: ECM Conformity Requirements

This appendix outlines the conformity requirements for any implementation of this recommendation.

F.1 Introduction

Conformity is the fulfillment of defined and documented requirements. Therefore, conformity is the basis for making definitive statements about the degree of fulfillment of a given specification/standard by an implementation with respect to the specification/standard itself. The definition of ECM Conformity follows these rules.

F.2 General conformity statement

A compliant implementation of the recommendation can only be realized using the XML schema provided in Appendix I of this part of the Recommendation.

F.3 ECM conformity classes

An implementation of this recommendation is considered to be compliant if the implementation conforms to the following conformity classes

- ECM Data conformity
- ECM Message conformity
- ECM Interaction Scenario conformity

ECM Interaction Scenario conformity builds upon ECM Message conformity, and ECM Message conformity builds upon ECM Data conformity. These conformity classes are outlined in the following subsections

F.3.1 ECM Data conformity

ECM Data conformity exists if the payload of each communicated ECM Message conforms to the data model defined in Section 7.2 and Appendix E of this part of the recommendation, i.e.

- all explicit constraints, e.g. referential integrity constraints, of that data model must be satisfied
- all implicit constraints of the data model specified in Appendix E of this part of the recommendation must be satisfied

In addition, all rules and restrictions implied by any ECM Reference Sub-Process-specific recommendation (e.g. Part 1: ECR) must be observed by any compliant implementation.

F.3.2 ECM Message conformity

ECM Message conformity exists if for each communicated ECM Message the following applies:

- The message header fields and structure of the ECM Message conform to section 7.1.3 as well as Appendix H ("ECM Message headers and control information") of this recommendation.
- The name of each ECM message is one of the message names defined in the respective section of this series of recommendations.
- The parameters of each ECM Message conform to the parameters defined in the respective section of this series of recommendations.
- There is a valid correlation between the selected objects of the payload of each ECM message (e.g. ID.id) and the header fields of that ECM message (e.g. InteractionId)

Note: *The details of a valid correlation are defined in the ECM Implementation Guide.*

- The payload of each ECM messages conform to the criteria specified for the ECM Data conformity

F.3.3 ECM Interaction Scenario conformity

ECM Interaction Scenario conformity exists if the ECM messages used in any interaction instance conform to the criteria specified for ECM Message conformity

Note: *The ECM Interaction Scenarios specified by an ECM Reference Sub-Process-specific recommendation (e.g. Part 1: ECR) are not binding in the sense of conformity but provide a few of potentially many Interaction Scenarios.*

Appendix G: Handling of Errors, Warnings and Acknowledgements

G.1 Introduction, scope and overview

This appendix focuses on providing messages for errors, warnings and acknowledgements in order to deal with acknowledgements, errors and warnings during cross-partner ECM message communication and the cross-partner ECM process.

- Acknowledgements, warnings and errors in the context of a communication: Errors or unexpected states occurring during message processing between the local ECM applications, i.e. errors during transmission, conversion or import of an ECM message need to be communicated back to the sender. Acknowledgements should be provided to ensure and track reliable end-to-end communication in the event of unreliable message transmission.
- Acknowledgements, warnings and errors in the context of messages issued by the process or its users and tools: Some errors and unexpected states occur only after the message has been successfully imported. For example, incorrect data may be entered by users at the sender's end but are only detected by users at the receiver's end or during a later phase of the process at the receiver's end. A distinction must be made between two kinds of scenarios:
  - Message usage errors that can be detected early: The error is detected before the sender's and receiver's process switched the state in their respective private ECM process to the next action according to the ECM Sub-Process specific IS Protocol Specification to beyond the message send or receive action (see e.g. Part 1 - ECR, Appendix A). In this case, acknowledgements and error messages can be used to allow the sender to generate the message again, maybe if necessary with a different content. If agreed upon by the partners, the progress in the private ECM process in the state according to the ECM IS Protocol Specification of sender and receiver can be delayed until a correct message is exchanged and acknowledged.
  - Message usage errors that are detected late: In general, when a message or its content that has been used in the process is found to be errored, this state of affairs can be signaled using the error message described below. However, this type of scenario must be handled at process level and is not covered by the scope of this appendix. The ECM Reference Sub-Processes are intended to deal with this kind of error, e.g. by providing iterative interactions, rollbacks; or the administrator needs to reset the state of the ECM applications accordingly, which is outside the scope of the ECM Reference Process.

The described error handling is to be applied specifically to ECM-related issues. Implementation-specific errors such as Web service communication errors may also be communicated using this recommendation but these are not mentioned specifically.

As mentioned earlier, the error handling defined in this appendix basically refers to message processing. Therefore, it can be applied to process-oriented integration as well as data-oriented integration.

The following provides an introduction to a generic error handling message and its use. This is followed by a section on the message header for the generic error handling message. The last section provides recommendations regarding the implementation of error handling.

G.2 Message header and control information

The table below provides information on the message header data for messages of the type *messagestatus*. This type of message can be used to specify messages that describe an error, a warning or an acknowledgement. The phrase "causing message" indicates the message which triggered the error and to which therefore the error, warning or acknowledgement refers.

Table 6: Message headers and control information for ECM Messages

Data field	Description (types/format, sample value, meaning)
ORGANIZATIONIDTYPE	As for messages of the type *request, respond or notify*
ORGANIZATIONID	As for messages of the type *request, respond or notify*
INTERACTIONID	As for messages of the type *request, respond or notify*
INTERACTIONSCENARIO-NAME	As for messages of the type *request, respond or notify*
MESSAGENAME	***Type:*** string; values: "ACKNOWLEDGEMENT", "WARNING", "ERROR" Describes whether this message indicates an acknowledgement, a warning or an error. An acknowledgement signals the successful completion of one or more processing steps (see ProcessingPhase) of an ECM message to its sender. A warning signals that an unexpected state occurred during the processing of an ECM message, but processing seems to have been completed successfully. An error signals that a step involved in the processing of an ECM message failed.
KINDOFSUBDIALOG	As for messages of the type *request, respond or notify*
CLASS	Type : string ; values: "SYSTEM/COMMUNICATION", "IMPLEMENTATION", "USAGE" 1. In the case of an error or a warning, this value provides a classification of the error with regard to the supposed cause. "SYSTEM/COMMUNICATION" - indicates an error assumed to be caused by a system (e.g. a system or component is down) or the message transport layer (e.g. a message is distorted). This error may be transient; i.e. another transmission attempt for the same message may succeed. "IMPLEMENTATION" – indicates an error assumed to be caused by the ECM implementation. This error is probably permanent; i.e. a retransmission is not expected to succeed.

	"USAGE" – indicates an error assumed to be caused by using the ECM system (e.g. in incorrect data element was entered by the user generating the message). This error is probably transient; correct usage of the ECM system allows the error to be avoided. 2. In the case of an acknowledgement, no value should be provided.
PROCESSINGPHASE	Type: string; values: "TRANSFER", "CONVERT"," IMPORT", "USER-RECEPTION", "USER-READING", "USER-CHECKING" 1. In the case of an error or a warning, this value describes the processing phase during which the error or warning was detected. 2. In the case of an acknowledgement, this value describes which processing phase of message processing was successfully completed. It is assumed that the following processing phases are executed in the sequence given here. Therefore, with regard to acknowledgements in particular, an acknowledgement of a later processing phase (e.g. convert) implies the successful completion of an earlier processing phase (e.g. transfer). "TRANSFER" – indicates that communication from the sender's to receiver's network domain caused an error or a warning or was completed successfully. "CONVERT" - indicates that conversion from the common format into the receiver's internal format caused an error or a warning or was completed successfully. "IMPORT" - indicates that conversion from the common format into the RECEIVER'S internal format caused an error or a warning or was completed successfully. "USER-RECEPTION" – indicates that the originating message (or its corresponding counterpart in the receiver's system) was received by the user. This value is only to be used for an acknowledgement. "USER-READING" – indicates that the originating message (or its corresponding artifact in the receiver's system) was read by the user. This value is only to be used for an acknowledgement. "USER-CHECKING" – indicates that the causing message (or its corresponding artifact in the receiver's system) caused an error or a warning when checked by the user. In the case of an acknowledgement, the value indicates that the message was checked successfully. Checking by a user can be performed to detect message usage errors early (see 0). "OTHER" – indicates that the causing message (or its corresponding artifact in the receiver's system) caused an error or a warning during another processing phase. This value cannot be used for acknowledgements. ***Note:*** The processing phases from "TRANSFER" up to "USER-CHECKING" are considered part of the send/receive message action (extended by user reading and user checking of a message). Especially if "user checking" is used and in order to support error handling, the send/receive action in the ECM IS Protocol Specifica-

	tion is considered to start when the message is composed by a user at the sender's end and to have been completed only after the last acknowledgment regarding these processing phases is received by the sender. This means that if an error is detected at the receiver's end, the message can be composed and sent again without any need to reset the state in the Interaction Scenario according to the ECM IS Protocol Specification. ***Note:*** If some of these processing phases are used several times, e.g. because of intermediary systems at the receiver, an error or warning may refer to any instance of the processing phase, and an acknowledgement should only be used when the last instance of the processing phase has been completed successfully.
CODE	***Type:*** string; values: "MESSAGE-UNEXPECTED", "MESSAGE-MISSING", "MESSAGE-HEADER-CONTENT-UNEXPECTED", "MESSAGE-HEADER-FORMAT-UNEXPECTED", "MESSAGE-BODY-CONTENT-UNEXPECTED", "MESSAGE-BODY-FORMAT-UNEXPECTED", "SECURITY-PROBLEM", "SYSTEM-PROBLEM", "TRANSMISSION-PROBLEM", "OTHER" 1. In the case of an error or a warning, this value indicates the type of error that was detected. The error can be described in more detail using DESCRIPTION. "MESSAGE-UNEXPECTED" – indicates that the causing message was unexpected; e.g. because a different message is specified in the ECM Interaction Scenario. "MESSAGE-MISSING" – indicates that a message is missing; e.g. because a timeout occurred. "MESSAGE-HEADER-CONTENT-UNEXPECTED" – indicates a problem with the content of the message header; e.g. one identifier has an unknown value. "MESSAGE-HEADER-FORMAT-UNEXPECTED" – indicates a problem with the format of the message header; e.g. one value has an irregular type. "MESSAGE-BODY-CONTENT-UNEXPECTED" – indicates a problem with the content of the message body; e.g. one identifier has an unknown value. "MESSAGE-BODY-FORMAT-UNEXPECTED" – indicates a problem with the format of the message body; e.g. one value has an irregular type. "SECURITY-PROBLEM" – indicates a problem related to authentication, authorization, integrity or confidentiality of a message. "SYSTEM-PROBLEM" – indicates trouble with a system involved, e.g. a required system is down. "TRANSMISSION-PROBLEM" – indicates trouble transmitting the message, e.g. a network problem. "OTHER" – indicates an error of another kind.

	2. In the case of an acknowledgement, no value should be provided.
DESCRIPTION	***Type:*** string; example: "A message of the type "*respond*" was expected." 1. In the case of an error or a warning, this value gives a text-based explanation and specific error or warning message. 2. In the case of an acknowledgement, no value should be provided.
DATE	***Type:*** string; format: "YYYY-MM-DD"; example: "2006-12-31" Indicates the date on which the message was created.
TIME	***Type:*** string; format: "HH:MM:SS:mm"; example: "10:05:33:01" Indicates the time at which the message was created in hours, minutes, seconds and milliseconds.
CAUSINGMSG _ORGANIZATIONIDTYPE	Copy of the value of *ORGANIZATIONIDTYPE* in the causing message of the type *request*, *respond* or *notify*. Optional.
CAUSINGMSG _ORGANIZATIONID	Copy of the value of *ORGANIZATIONIDTYPE* in the causing message of the type *request*, *respond* or *notify*. Optional.
CAUSINGMSG _INTERACTIONID	Copy of the value of *ORGANIZATIONIDTYPE* in the causing message of the type *request*, *respond* or *notify*. Optional.
CAUSINGMSG _NEWINTERACTIONID	Copy of the value of *INTERACTIONSCENARIONAME* in the causing message of the type *request*, *respond* or *notify*. Optional.
CAUSINGMSG _INTERACTIONSCENARIONAME	Copy of the value of *MESSAGENAME* in the causing message of the type *request*, *respond* or *notify*. Optional.
CAUSINGMSG _MESSAGENAME	Copy of the value of *SEQUENCENO* in the causing message of the type *request*, *respond* or *notify*. Optional.
CAUSINGMSG _SEQUENCENO	Copy of the value of *ISCOMPLETE* in the causing message of the type *request*, *respond* or *notify*. Optional.
CAUSINGMSG _ISCOMPLETE	Copy of the value of *KINDOFSUBDIALOG* in the causing message of the type *request*, *respond* or *notify*. Optional.
CAUSINGMSG _REQUESTOR _ORGANIZATIONIDTYPE	Copy of the value of *REQUESTOR_ORGANIZATIONIDTYPE* in the causing message of type *respond*. If the causing message was of the type *request* or *notify*, no value should be provided here. Optional.
CAUSINGMSG _REQUESTOR _ORGANIZATIONID	Copy of the value of *REQUESTOR_ORGANIZATIONID* in the causing message of the type *respond*. If the causing message was of the type *request* or *notify*, no value should be provided here. Optional.
CAUSINGMSG _REQUESTOR _INTERACTIONSCENARIO-NAME	Copy of the value of *REQUESTOR_INTERACTIONSCENARIONAME* in the causing message of the type *respond*. If the causing message was of the type *request* or *notify*, no value should be provided here. Optional.
CAUSINGMSG _REQUESTOR _MESSAGENAME	Copy of the value of *REQUESTOR_MESSAGENAME* in the causing message of the type *respond*. If the causing message was of the type *request* or *notify*, no value should be provided here. Optional.
CAUSINGMSG _REQUESTOR _SEQUENCENO	Copy of the value of *REQUESTOR_SEQUENCENR* in the causing message of the type *respond*. If the causing message was of the type *request* or *notify*, no value should be provided here. Optional.
CAUSINGMSG	Copy of the value of *REQUESTOR_ISCOMPLETE* in the causing

_*REQUESTOR* _*ISCOMPLETE*	message of the type *respond*. If the causing message was of the type *request* or *notify*, no value should be provided here. Optional.
CAUSINGMSG _*REQUESTOR* _*KINDOFSUBDIALOG*	Copy of the value of *REQUESTOR_KINDOFSUBDIALOG* in the causing message of the type *respond*. If the causing message was of the type *request* or *notify*, no value should be provided here. Optional.
CAUSINGMSG _*BINARY*	Copy of the binary representation of the causing message (including header and body). Optional.

Notes: *If an acknowledgement, warning or error message is to include the body of the causing message, this is done in the same way as for the messages* request*,* respond *or* notify*.*

The fields CAUSINGMSG_* *may be omitted if an error can no longer be associated with a specific message.*

The fields CAUSINGMSG_REQUESTOR_ORGANIZATIONIDTYPE *to* CAUSINGMSG_REQUESTOR_KINDOFSUBDIALOG *may be omitted if the causing message was of the type request or notify.*

It may not be possible to use some of the values due to the error that occurred. In this case, the special value "UNKNOWN" should be used for these fields.

G.3 Error handling

G.3.1 Message usage and error handling

The following are guidelines for using the messages. These guidelines give due consideration to the requirements mentioned in section G.1.

G.3.1.1 Signaling of errors, warnings, acknowledgements

Error messages, warnings and acknowledgements can be sent to the sender (see G.2) An error message or warning can also be used for errors not directly related to a message.

G.3.1.2 Handling of communication errors

When using unreliable message transport, communication errors can be handled using acknowledgements and retransmission-based algorithms. The sender repeatedly sends one specific message until he receives acknowledgement of the processing phase "IMPORT" for that message from the receiver. The receiver waits for the message with the next higher sequence number, discarding duplicates of earlier acknowledged messages as well as any messages with a number higher than the next expected sequence number[12].

G.3.1.3 Handling message usage errors that can be detected early

Like G.3.1.2, this kind of error can be detected and handled by enhancing the mechanism for errors and acknowledgements with the processing phase "USER-CHECKING". The send/receive message action in the ECM IS Protocol Specification is only considered to have been completed after this type of acknowledgement has been received by the sender of the

[12] J. Gray, A. Reuter: "Transaction processing: concepts and techniques". Morgan Kaufmann Publishers, San Francisco; 1993.

causing message. If a user at the receiver's end indicates an error detected while checking the message, the message must be composed again by the sender and resent to the receiver.

Note: *This basic mechanism requires that a user at the sender's end composes the message and its content (as a part of the send action) – and that the receiver checks the message and its content after receiving it (as a part of the receive action).*

G.3.2 Cooperation-specific input parameters

The following Boolean parameters must be agreed upon in order to regulate error handling:

- IsErrorSignallingUsed – describes whether error messages are exchanged.
- IsWarningSignallingUsed – describes whether warning messages are exchanged.
- IsAckTransferEnabled – describes whether acknowledgement messages are exchanged if transfer of the message to the receiver is completed successfully.
- IsAckImportEnabled – describes whether acknowledgement messages are exchanged if the message is successfully imported by the receiver.
- IsAckConvertEnabled – describes whether acknowledgement messages are exchanged if conversion of the message by the receiver is completed successfully.
- IsAckUserReadEnabled – describes whether acknowledgement messages are exchanged if the message is read successfully by a user at the receiver's end.
- IsAckUserCheckingEnabled – describes whether acknowledgement messages are exchanged if the message is successfully checked by a user at the receiver's end.
- IsMessageBodyAppendedToErrors – describes whether the message body is appended to error messages.
- IsMessageBodyAppendedToWarnings – describes whether the message body is appended to warning messages.
- IsMessageBodyAppendedToAcknowledgements – describes whether the message body is appended to acknowledgement messages.
- IsMessageBinaryIncludedInErrors – describes whether the message binary is included in error messages.
- IsMessageBinaryIncludedInWarnings – describes whether the message binary is included in warning messages.
- IsMessageBinaryIncludedInAcknowledgements – describes whether the message binary is included in acknowledgment messages.
- IsCommunicationErrorHandlingEnabled – describes whether the handling of communication errors according to G.3.1.2 is to be used. This implies that IsAckTransferEnabled is TRUE.
- IsMessageUsageErrorHandlingEnabled – describes whether the handling of communication errors according to G.3.1.3 is to be used. This implies that IsAckUserCheckingEnabled and IsErrorSignallingUsed are TRUE

Appendix H: Message Headers and Control Information

This chapter describes an implementation-form-independent data model for the Message header and control information. This data model formalizes the textual description of the requirements in Section 7.1.2 and Appendix G.

H.1 EXPRESS-G representation

The result of the formalization of the textual description of the requirements is presented as EXPRESS-G representation in Figure 38. A brief introduction to EXPRESS-G notation can be found in Appendix B.

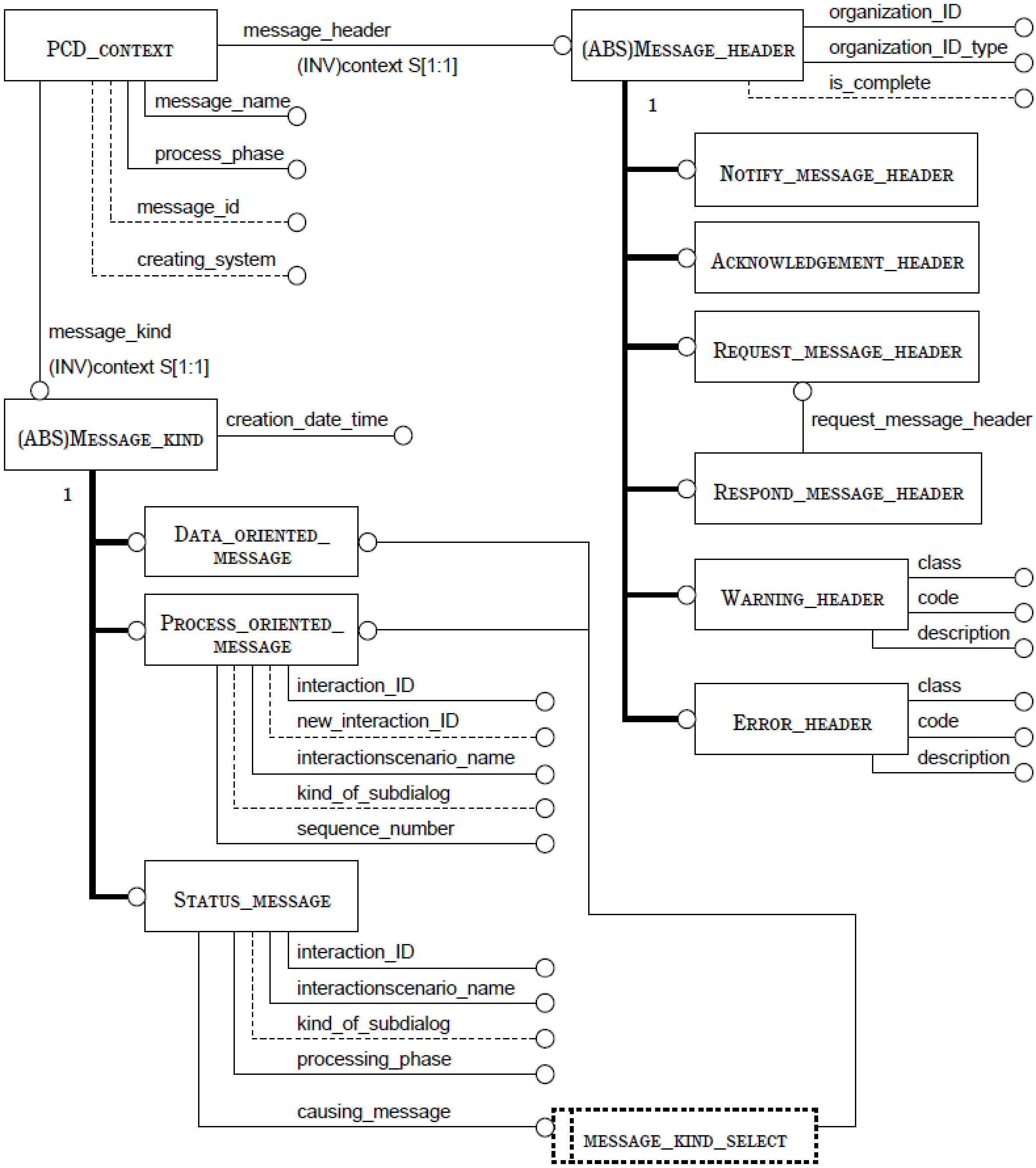


Figure 38: EXPRESS-G representation of the message header and control information

H.2 Data dictionary

H.2.1 ECM_context

An ECM_context combines exactly one Message_header with one Message_kind and provides information about the name of the message as well as the ECM Sub-Process context information of that message.

Each ECM_context is an ECR_context or a ECO_context.

The data associated with an ECM_context comprises the following:

- message_name.

message_name

The message_name specifies the name of the message. See Section 7.1.2 for details.

H.2.2 ECR_context

An ECR_context indicates that the message was sent in the context of the ECR process

H.2.3 ECO_context

An ECO_context indicates that the message was sent in the context of the ECO process

H.2.4 Message_kind

A Message_kind object describes the kind of message.

Each Message_kind is a Data_oriented_message, a Process_oriented_message or a Status_message.

The data associated with an Error_header comprises the following:

- creation_date_time.

creation_date_time

The creation_date_time indicates the creation date and time of the message at the sender of the message. See Appendix G for details.

H.2.5 Data_oriented_message

A Data_oriented_message indicates that the message was sent in a data-oriented or combined scenario. See Section 2.2 for further details.

Each Data_oriented_message references – via the inherited inverse attribute .context – an ECM_context, which itself references a Message_header of the type Notify_message_header, Request_message_header or Respond_message_header via the attribute .message_header.

H.2.6 Process_oriented_message

A Process_oriented_message indicates that the message was sent in a process-oriented or combined scenario. See Section 2.2 for further details.

Each Process_oriented_message references – via the inherited inverse attribute .context – an ECM_context, which itself references a Message_header of the type Notify_message_header, Request_message_header or Respond_message_header via the attribute .message_header.

The data associated with a Process_oriented_message comprises the following:

- interaction_ID
- new_interaction_ID (optional)
- interactionscenario_name
- kind_of_subdialog (optional)
- sequence_no

interaction_ID

The interaction_ID uniquely identifies the interaction scenario dialog of a message. See Section 7.1.2 for details.

new_interaction_ID

The new_interaction_ID uniquely identifies the new interaction scenario dialog of a message. The new_interaction_ID need not be specified for a particular Process_oriented_message. See Section 7.1.2 for details.

interactionscenario_name

The interactionscenario_name specifies the name of the interaction scenario in the context of a message. See Section 7.1.2 for details.

kind_of_subdialog

The kind_of_subdialog specifies the kind the interaction in the subdialog. The kind_of_subdialog need not be specified for a particular Process_oriented_message. See Section 7.1.2 for details.

sequence_no

The sequence_no specifies the current number count of messages from the sender of this message in the current interaction scenario dialog. See Section 7.1.2 for details.

H.2.7 Status_message

A Status_message indicates that the message was sent in a data-oriented scenario. See Appendix G for further details.

Each Status_message references – via the inherited inverse attribute .context – an ECM_context, which itself references a Message_header of the type Acknowledgment_header, Warning_header or Error_header via the attribute .message_header.

The data associated with a Status_message comprises the following:

- interaction_ID
- interactionscenario_name
- kind_of_subdialog (optional)
- processing_phase

interaction_ID

The interaction_ID uniquely identifies the interaction scenario dialog of a message. See Appendix G for details.

interactionscenario_name

The interactionscenario_name specifies the name of the interaction scenario in the context of a message. See Appendix G for details.

kind_of_subdialog

The kind_of_subdialog specifies the kind the interaction in the subdialog. The kind_of_subdialog need not be specified for a particular Status_message. See Appendix G for details.

processing_phase

The processing_phase specifies the processing phase of the receiver of this message. See Appendix G for details.

H.2.8 Message_header

A Message_header provides an indication of the type of message.

Each Message_header is a Notify_message_header, a Request_message_header, a Respond_message_header, an Acknowledgement_header, a Warning_header or an Error_header.

The data associated with an Error_header comprises the following:

- organization_ID
- organization_ID_type
- is_complete (optional)

organization_ID

The organization_ID uniquely identifies the organization sending this message in the context of communication between *coordinator* and *participant*. See Section 7.1.2 for details.

organization_ID_type

The organization_ID_type specifies the method used to identify the sending organization in the data element organization_ID. See Section 7.1.2 for details.

is_complete

The is_complete specifies the completion status (TRUE, FALSE) of a message. The is_complete need not be specified for a particular Message_header. See Section 7.1.2 for details.

H.2.9 Notify_message_header

A Notify_message_header indicates that the message is of the type Notify. See Section 7.1.2 for details.

Each Notify_message_header references – via the inherited inverse attribute .context - an ECM_context, which itself references a Message_kind of the type Data_oriented_message or Process_oriented_message via the attribute .message_kind.

H.2.10 Request_message_header

A Request_message_header indicates that the message is of the type Request. See Section 7.1.2 for details.

Each Request_message_header references – via the inherited inverse attribute .context – an ECM_context, which itself references a Message_kind of the type Data_oriented_message or Process_oriented_message via the attribute .message_kind.

H.2.11 Respond_message_header

A Respond_message_header indicates that the message is of the type Respond. See Section 7.1.2 for details.

Each Respond_message_header references - via the inherited inverse attribute .context - an ECM_context, which itself references a Message_kind of the type Data_oriented_message or Process_oriented_message via the attribute .message_kind.

The data associated with an Error_header comprises the following:

- request_message_header

request_message_header

The request_message_header identifies the Request_message_header of the previous request message of this particular respond message. See Section 7.1.2 for details.

H.2.12 Acknowledgement_header

An Acknowledgement_header indicates that the message is of the type acknowledgement. See Appendix G for details.

Each Acknowledgment_header references – via the inherited inverse attribute .context –an ECM_context, which itself references a Message_kind of the type Status_message via the attribute .message_kind.

H.2.13 Warning_header

A Warning_header indicates that the message is of the type warning. See Appendix G for details.

Each Warning_header references – via the inherited inverse attribute .context – an ECM_-context, which itself references a Message_kind of the type Status_message via the attribute .message_kind.

The data associated with a Warning_header comprises the following:

- class
- code
- description (optional)

class

The class provides a classification of the warning with regard to the supposed cause. See Appendix G for details.

code

The code provides an indication of the kind of warning that was observed. See Appendix G for details.

description

The description provides an additional textual explanation of the specific warning message. The description need not be specified for a particular Warning_header. See Appendix G for details.

H.2.14 Error_header

An Error_header indicates that the message is of the type error. See Appendix G for details.

Each Error_header references – via the inherited inverse attribute .context – an ECM_-context, which itself references a Message_kind of the type Status_message via the attribute .message_kind.

The data associated with an Error_header comprises the following:

- class
- code
- description (optional)

class

The class provides a classification of the error with regard to the supposed cause. See Appendix G for details.

code

The code provides an indication of the kind of error that was observed. See Appendix G for details.

description

The description provides an additional textual explanation of the specific error message. The description need not be specified for a particular Error_header. See Appendix G for details.

Appendix I: ECM XML Schema

This appendix references a listing of the XML schema representing the ECM Data Model specified in Appendix E without comments or other explanatory text.

The listing is available in computer-interpretable form and can be found at the following URL:

http://sasig.com/ecm/ecm-xsd.zip

An annotated listing is included.

If you have difficulties accessing this site, please contact:

Thomas Springmann, Daimler AG
phone: +49 7031 90-82043
e-mail: thomas.springmann@daimler.com

Maintenance Form

If you find an error or other changes that should be made to this publication, please complete this form and return it to the proper address below.

Name of Submitter:		Date:
Company:		
Company Address:		
Phone:	Fax:	E-mail:
CHANGE REQUEST (Use additional sheets if necessary)		
Page Number of Change:		
Document Currently Reads:		
Recommended Changes/Should Read:		
Recommended Additions:		
Reason for Change:		
Signature of Submitter:		

Submit this Change Request by email to ecm-issue@sasig.com or to your organization at one of the addresses listed below:

Automotive Industry Action Group
26200 Lahser Road, Suite 200
Southfield, MI 48034
USA
Phone: (248) 358-3570
Fax: (248) 358-3253
Web: www.aiag.org

Odette International Limited
Forbes House
Halkin Street
London SW1X 7DS
UK
Phone: +44 207 344 0277
Fax: +44 207 245 6093
Web: www.odette.org

Japan Automobile Manufacturer's Association, Inc. (JAMA)
Otemachi Building, 6-1
Otemachi 1-chome Chiyoda-ku
Tokyo 100-0004
Japan
Phone: +81 3-5405-6130
Fax: +81 3-5405-6136
Web: www.jama.or.jp

ProSTEP iViP Association
Dolivostr. 11
64293 Darmstadt
Germany
Phone: +49 6151 9287-336
Fax: +49 6151 9287-326
Web: www.prostep.org