aris

METHOD MANUAL

ARIS 10

October 2026

This document applies to ARIS Version 10 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Copyright © 2010-2026 SAG Aris GmbH, Saarbrücken, Germany and/or its subsidiaries and/or its affiliates and/or their licensors.

The name ARIS and all SAG Aris GmbH product names are either trademarks or registered trademarks of SAG Aris GmbH and/or its subsidiaries and/or its affiliates and/or their licensors. Other company and product names mentioned herein may be trademarks of their respective owners.

The Software may contain or be distributed with certain commercial third party components (“Third Party Components”), which may be subject to special license terms and conditions (“Third Party Terms”) located at https://aris.com/terms.

Third Party Terms include important licensing and warranty information and disclaimers of third party licensors. For the avoidance of doubt, if Customer uses the Software in accordance with the Documentation, the Third Party Terms do not restrict the rights granted under this Agreement. In the event that a third party manufacturer no longer provides active support for any Third Party Component, Supplier will use commercially reasonable efforts to respond to any issues with the Software related to the Third Party Components.

Use, reproduction, transfer, publication or disclosure is prohibited except as specifically provided for in your License Agreement with SAG Aris GmbH and/or its subsidiaries and/or its affiliates and/or their licensors.

Table of Contents

1. Introduction 8
2. Architecture of Integrated Information Systems (ARIS) 9
2.1. ARIS architecture concept 9
2.2. Descriptive views 9
2.3. Descriptive levels 12
3. Modeling within the views and levels of the ARIS concept 16
3.1. Function view 16
3.1.1. Requirements definition 16
3.1.1.1. Function tree 16
3.1.1.2. Objective diagram 20
3.1.2. Design specification - Application system type diagram 21
3.1.3. Implementation - Application system diagram 25
3.2. Data view 27
3.2.1. Requirements definition 27
3.2.1.1. The ERM base model 28
3.2.1.2. ERM - eERM extensions 32
3.2.1.3. Design operators added 32
3.2.1.4. Extension of cardinalities 36
3.2.1.5. Identification and existence dependency 37
3.2.1.6. Modeling technical terms used in a company - Technical terms model 38
3.2.1.7. eERM attribute allocation diagram 39
3.2.1.8. IE data model 40
3.2.1.9. Summary of the main terms and forms of representation of the eERM 40
3.2.1.10. Modeling the Data Warehouse structure 41
3.2.1.11. Project management data - Information carrier diagram 42
3.2.2. Implementation - Table diagram 43
3.2.3. Role assignment diagram (RAD) 44
3.3. Organization view 45
3.3.1. Requirements definition 45
3.3.1.1. Organizational structure of companies 45
3.3.1.2. Organizational chart 47
3.3.2. Design specification - Network topology 51
3.3.3. Implementation 52
3.3.3.1. Network diagram 52
3.3.3.2. Material flow modeling - Technical resources 54
3.4. Process view 56
3.4.1. Requirements definition 56
3.4.1.1. Linking functions with organization - EPC 56
3.4.1.2. Event control - Event-driven process chain (EPC) 57
3.4.1.3. Function allocation diagram (I/O) 67
3.4.1.4. Event diagram 69
3.4.1.5. Value-added chain diagram 70
3.4.1.6. End-to-end business scenario diagram 71
3.4.1.7. Object-oriented modeling 71
3.4.1.8. Process selection matrix 71
3.4.1.9. Material flow modeling 72
3.4.1.10. EPC (material flow) 72
3.4.1.11. EPC (column/row display) 73
3.4.1.12. Business controls diagram 75
3.4.1.13. E-Business scenario diagram 75
3.4.1.14. Structuring model 77
3.4.1.15. Role diagram 78
3.4.1.16. Quick model 79

3.4.1.17. Screen design ... 79
3.4.1.18. Screen navigation ... 82
3.4.1.19. Business segment matrix ... 82
3.4.2. Design specification ... 84
3.4.2.1. Access diagram ... 84
3.4.2.2. Linking functions with data ... 84
3.4.2.3. Linking organization with function ... 85
3.4.2.4. Program flow chart ... 86
3.4.2.5. Program flow chart (PF) ... 86
3.4.2.6. Screen diagram ... 87
3.4.3. Implementation - Access diagram (physical) ... 89
3.4.3.1. Linking functions with data ... 89
3.4.3.2. Linking organization with data ... 90
3.4.3.3. Linking organization with functions ... 91
3.5. Product/Service modeling ... 93
3.5.1. Product/Service exchange diagram ... 94
3.5.2. Product/Service tree ... 95
3.5.3. Product allocation diagram ... 96
3.5.4. Product tree ... 98
3.5.5. Product selection matrix ... 99
4. Unified Modeling Language (UML) in ARIS ... 100
4.1. Introduction ... 100
4.2. ARIS UML Designer - Supported UML standard ... 100
5. Methods for knowledge management ... 101
5.1. Introduction ... 101
5.2. Object types for modeling knowledge processing ... 101
5.2.1. Knowledge category ... 101
5.2.2. Documented knowledge ... 102
5.3. Model types for modeling knowledge processing ... 103
5.3.1. Knowledge structure diagram ... 103
5.3.2. Knowledge map ... 103
5.3.3. Representation of knowledge processing in business processes ... 105
6. Balanced Scorecard method ... 107
6.1. Introduction ... 107
6.2. The Balanced Scorecard concept ... 107
6.2.1. Key elements of the BSC approach ... 107
6.2.2. Strategic management process and Balanced Scorecard ... 108
6.2.2.1. Formulation and realization of vision and strategy ... 108
6.2.2.2. Standard perspectives of a Balanced Scorecard ... 109
6.2.2.3. Cause-and-effect chain ... 110
6.2.2.4. Definition of leading and lagging indicators ... 111
6.2.2.5. Communication and derivation of further scorecards ... 111
6.2.2.6. Planning and targets ... 111
6.2.2.7. Strategic learning and feedback ... 112
6.2.3. Advantages and benefits of the Balanced Scorecard ... 112
6.3. Developing a Balanced Scorecard with ARIS BSC ... 113
6.3.1. Terms and abbreviations ... 113
6.3.2. Creating Balanced Scorecards with ARIS BSC ... 114
6.3.2.1. Specification of perspectives ... 114
6.3.2.2. Balanced Scorecard system structure specification ... 114
6.3.2.3. Cause-and-effect relationship specification ... 114
6.3.2.4. Specification of initiatives and KPIs to monitor objectives ... 116
6.3.2.5. Description of KPIs and their relationships ... 120
6.3.3. Relationships to other models ... 120
7. E-Business scenario diagram ... 121
7.1. Introduction ... 121

7.2. E-Business scenario diagram method ... 122
7.2.1. The idea ... 122
7.2.2. The model and its objects ... 123
7.2.3. 'Transmission type' attribute group ... 124
7.3. Evaluations using reports ... 124
7.3.1. Data security check ... 124
7.3.2. System support ... 125
7.3.3. Information flow ... 125
7.3.4. Collaborative business maps ... 125
7.4. Connection to other methods and components ... 125
8. IT City Planning ... 129
8.1. Enterprise Architecture and IT City Planning ... 129
8.2. Which companies may benefit from IT City Planning? ... 129
8.3. IT City Planning with ARIS ... 130
8.4. Service view ... 131
8.5. Service types and their data ... 134
8.6. Detail description of service types ... 135
8.7. Chronological-logical operational sequences of IS elements ... 136
8.8. IT view ... 136
8.9. IT elements and their data ... 137
8.10. Detail description of IT elements ... 137
8.11. Organizational aspects ... 138
8.12. Chronological-logical operational sequences of IT elements ... 139
8.13. Chronological-logical operational sequences within the architecture ... 139
8.14. Possible evaluations ... 140
9. Business process modeling ... 142
9.1. Process classes and the business process diagram ... 142
9.2. Implementation of BPMN in ARIS ... 144
9.3. Elements of the business process diagram ... 144
9.3.1. Pools and lanes ... 144
9.3.2. Modeling guidelines for pools and lanes ... 145
9.3.3. Sequence flow ... 145
9.3.4. Modeling guidelines for sequence flow connections ... 146
9.3.5. Message flow ... 146
9.3.6. Modeling guidelines for message flow connections ... 146
9.3.7. Association ... 147
9.3.8. Events ... 147
9.3.9. Modeling guidelines for events ... 148
9.3.10. Activities ... 148
9.3.11. Modeling guidelines for activities ... 149
9.3.12. Gateway ... 150
9.3.13. Modeling guidelines for gateways ... 151
9.3.14. Artifact ... 152
9.3.15. Sources of figures ... 154
10. Modeling BPMN 2.0 ... 155
10.1. Introduction ... 155
10.1.1. Initial situation and objective ... 155
10.1.2. Purpose of this chapter ... 155
10.2. BPMN Core elements and their implementation in ARIS ... 155
10.2.1. Infrastructure ... 156
10.2.2. Foundation ... 156
10.2.3. Common Elements ... 157
10.2.3.1. Artifacts ... 157
10.2.3.2. Associations ... 159
10.2.3.3. Groups ... 159
10.2.3.4. Text Annotations ... 160

10.2.3.5. Callable Elements ... 160
10.2.3.6. Events ... 161
10.2.3.7. Expressions ... 161
10.2.3.8. Flow Elements ... 161
10.2.3.9. Flow Elements Container ... 162
10.2.3.10. Gateways ... 162
10.2.3.11. Messages ... 163
10.2.3.12. Message Flows ... 163
10.2.3.13. Participants ... 165
10.2.3.14. Resources ... 166
10.2.3.15. Sequence Flows ... 166
10.2.3.16. Elements not included in the current implementation ... 169
10.3. BPMN diagrams and ARIS model types: An overview ... 169
10.4. Processes ... 170
10.4.1. Activities ... 171
10.4.1.1. Resource Assignments ... 172
10.4.1.2. Performers ... 173
10.4.1.3. Activity type: Task ... 173
10.4.1.4. Human Interactions ... 175
10.4.1.5. Activity type: Sub-Process ... 175
10.4.1.6. Sub-Process type: Sub-Process ... 176
10.4.1.7. Sub-Process type: Event Sub-Process ... 177
10.4.1.8. Sub-Process type: Transaction ... 178
10.4.1.9. Sub-Process type: Ad Hoc Sub-Process ... 179
10.4.1.10. Sub-Process type: Call Activity ... 180
10.4.1.11. Global Tasks ... 181
10.4.1.12. LoopCharacteristics ... 181
10.4.1.13. LoopCharacteristics: Representations ... 181
10.4.1.14. StandardLoopCharacteristics, MultiInstanceLoopCharacteristics, and ComplexBehaviorDefinition ... 182
10.4.2. Items and Data ... 183
10.4.2.1. Data Objects ... 184
10.4.2.2. Data Stores ... 185
10.4.3. Events ... 186
10.4.3.1. Catch Events and Throw Events ... 187
10.4.3.2. Start Events ... 188
10.4.3.3. Intermediate Events ... 189
10.4.3.4. End Events ... 189
10.4.3.5. Event Definitions ... 190
10.4.4. Gateways ... 193
10.4.4.1. Exclusive Gateways ... 194
10.4.4.2. Inclusive Gateways ... 195
10.4.4.3. Parallel Gateways ... 195
10.4.4.4. Complex Gateways ... 196
10.4.4.5. Event-Based Gateways ... 196
10.4.5. Lanes ... 196
10.5. Collaborations ... 198
10.5.1. Pools and Participants ... 199
10.5.2. Object types and connection types reused from a process ... 199
10.5.3. Message Flows ... 199
10.6. Conversations ... 200
10.6.1. Conversation Nodes ... 200
10.6.2. Conversation Symbols ... 200
10.6.3. Participants ... 201
10.6.4. Artifacts ... 202
10.6.5. Conversation Links ... 202

10.6.6. Message Flows in a Conversation ... 202
10.6.7. Model assignments ... 203
10.7. Enterprise BPMN collaboration diagram ... 203
11. Customer Experience Management (CXM) ... 205
11.1. Customer journey landscape ... 205
11.2. Customer journey map ... 206
11.3. Customer touchpoint allocation diagram ... 207
11.4. Customer touchpoint map ... 208
11.5. Linking CXM and BPM ... 209
11.5.1. Analysis capabilities ... 210
11.5.1.1. Report ... 210
11.5.1.2. Queries ... 211
11.5.1.3. Get full customer journey overview ... 211
11.5.1.4. Find customer touchpoints clustered by associated risk ... 212
11.5.1.5. Find customer touchpoints clustered by associated ownership ... 213
11.5.1.6. Find customer touchpoints clustered by associated channel ... 213
11.5.1.7. Find risks and initiatives for all customer touchpoints ... 214
11.5.1.8. Find risks and initiatives for bad customer touchpoints only ... 215
11.5.1.9. Find all processes related to customer journeys ... 215
12. Use cases ... 217
12.1. General company documentation ... 218
12.2. Database management/Data warehousing ... 219
12.3. PC hardware and network management ... 220
12.4. Process cost management ... 221
12.5. Quality management ... 221
12.6. Reorganization measures ... 222
12.7. SAP R/3 implementation ... 223
12.8. Software development and implementation ... 224
12.9. Knowledge management ... 225
12.10. Workflow management ... 226
13. Bibliography ... 227
13.1. General literature list ... 227
13.2. Topic-related bibliography ... 228
13.2.1. Unified Modeling Language in ARIS ... 228
13.2.1.1. UML specification ... 228
13.2.1.2. Using UML ... 228
13.2.1.3. UML and business process modeling ... 228
13.2.2. Methods for knowledge management ... 228
13.2.2.1. General knowledge management ... 228
13.2.2.2. Using ARIS for knowledge management ... 229
13.2.3. Balanced Scorecard method ... 229
13.2.4. IT City Planning ... 229
13.2.5. Business process modeling ... 229
14. Support and legal information ... 230
14.1. Documentation scope ... 230
14.2. Support ... 230
14.3. Legal notices ... 231

1. Introduction

In the past, system design and system integration had the greatest potential for optimization. In recent years, however, the focus has shifted more and more towards creating solutions for the special demands of individual sectors. The fact that decentralized information systems became available and that it was possible to incorporate them into integrated information system infrastructures created new cost saving potential regarding the organizational design of companies.

Organizational structures were formerly broken down functionally and established centrally because they were mostly based on centralized host environments with only limited capabilities. As a consequence, companies suffered a loss of flexibility. In the beginning, few people realized or paid attention to the new prospects resulting from the increase in decentralization of computer services and parallel development of new information system architecture concepts (for example, client/server, workflow management).

Today, steadily intensifying competition has turned this potential into the number one topic for every single company. Flexible structures that persistently focus on internal business processes are becoming the decisive competition factor for companies. However, only a holistic view of all business processes enables a company to recognize, streamline, and support interconnected processes through optimized information system infrastructures. Compared with the management of centralized business environments, the management of these new structures is significantly more complex. Facing this challenge requires unequivocal assignment of responsibilities, maximum transparency of structures, homogeneous communication throughout all company levels, and streamlined project management based on defined business objectives.

Enterprise modeling methods assist business managers in accomplishing these complex tasks. Enterprise models are a crucial prerequisite for analyzing business processes, bringing projects in line with the overall business objectives, and using information system infrastructures in the form of composite distributed and integrated systems to optimally support these lean organizational structures.

Thus, modeling the company's actual situation - and, in doing so, examining holistic business processes - is becoming more and more the focus of the discussion. The diversity and increasing multitude of modeling methods used to result in complexity and confusion. Consequently, efforts were made to define standardized framework concepts (architectures) for development and modeling methods.

One of these architectures is the **Architecture of Integrated Information Systems** (**ARIS** ©) developed by Scheer (see Scheer, Architecture of Integrated Information Systems, 1992). This architecture concept enables methods to be evaluated and organized by focusing on their main points, and it serves as an orientation framework for complex development projects because due to its structural elements, it contains an implicit procedure model for developing integrated information systems.

An architecture of this kind naturally leads towards standardization in the use of methods. Based on this architecture, existing and new modeling methods were combined to create a holistic method for modeling business processes.

In addition, the ARIS architecture integrates products such as ARIS Architect within the product range of ARIS. These products support consultants and companies in creating, analyzing, and evaluating business processes in terms of business process re-engineering. Convenient recording and modeling of the relevant business processes in the operating departments is enabled by ARIS Designer.

This manual gives a first introduction to the relevant modeling methods. In addition, approaches and methods are presented that make use of the full range of ARIS products including the system add-ons they offer. This manual also provides excellent support for users who deal with modeling methods without the intention of considering questions or problems regarding the use of tools.

2. Architecture of Integrated Information Systems (ARIS)

2.1. ARIS architecture concept

The **AR**chitecture of integrated **I**nformation **S**ystems (ARIS) is based on an integration concept derived from a holistic view of business processes. The first step in creating the architecture is to develop a business process model containing all basic features for describing business processes. The result is a highly complex model, which is broken down into individual views so that its complexity is reduced. Due to this breakdown, it is possible to describe the content of individual views by special methods suitable for a specific view without having to pay attention to the numerous view interrelationships. The relationships between the views are incorporated in a final step and combined to form an overall overview of process chains without any redundancies.

A second approach that also reduces complexity is a differentiation via descriptions. Following the lifecycle concept, the various description methods for information systems are classified based on their proximity to information technology. This ensures a consistent description, from business management problems through to technical implementation.

Thus, the ARIS concept is a framework for developing and optimizing integrated information systems and for describing their implementation. As the emphasis lies on the technical descriptive level, the ARIS concept serves as a model for creating, analyzing, and evaluating business management process chains. Scheer describes the Architecture of integrated Information Systems in more detail (see Scheer, Architecture of Integrated Information Systems, 1992, and Scheer, ARIS - Business Process Frameworks, 1998).

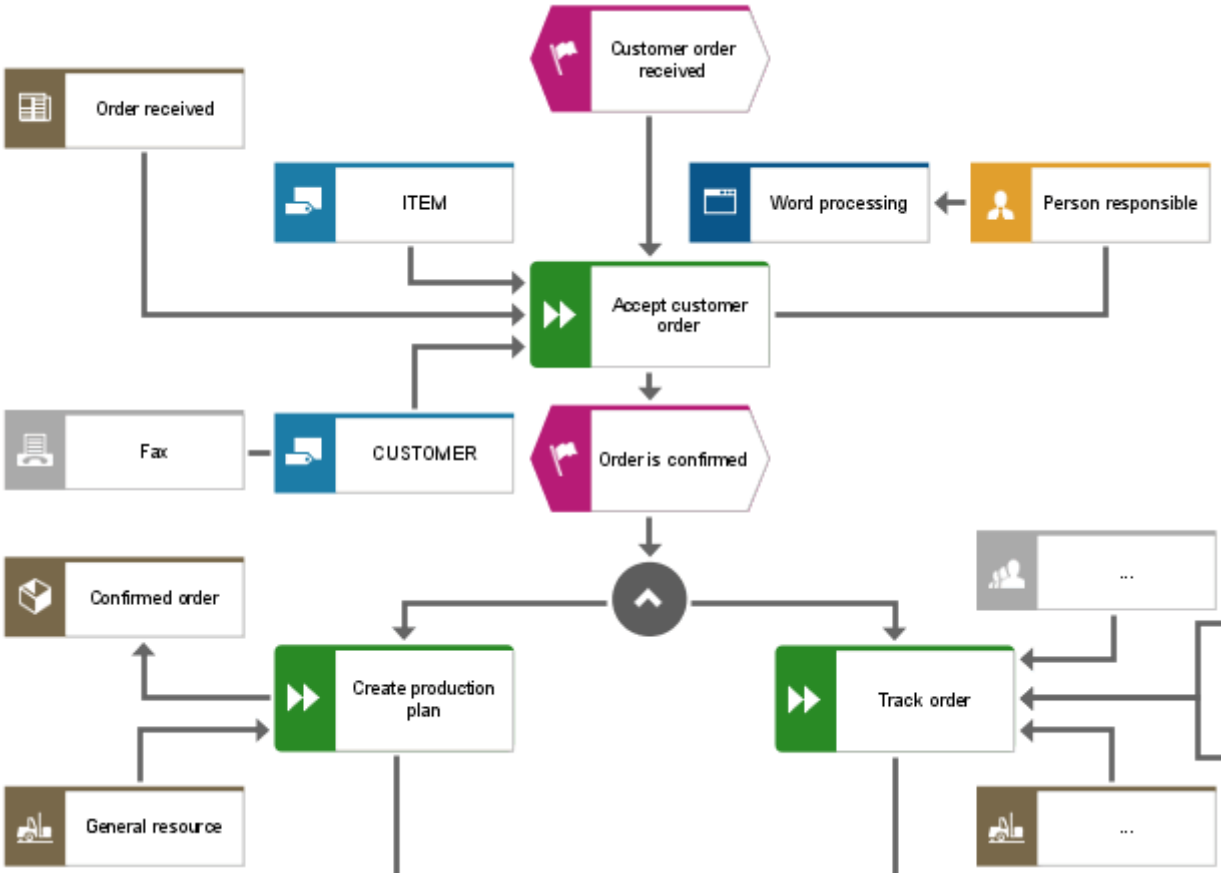
2.2. Descriptive views

The focus of this approach is on a business process, like the one shown in the following figure.

The process is triggered by the **Customer order received** event. In turn, this event activates the **Accept customer order** function (procedure). For this procedure to be performed, the current state of the relevant process environment must first be described. In particular, this includes data relating to customers and items. The state of associated objects may change during workflow processing, for example, when the items' inventory data is updated with the new reservation data.

The procedures are performed by sales employees who can be assigned to departments. Departments use specific information technology resources (personal computers, printers, etc.) to perform their tasks.

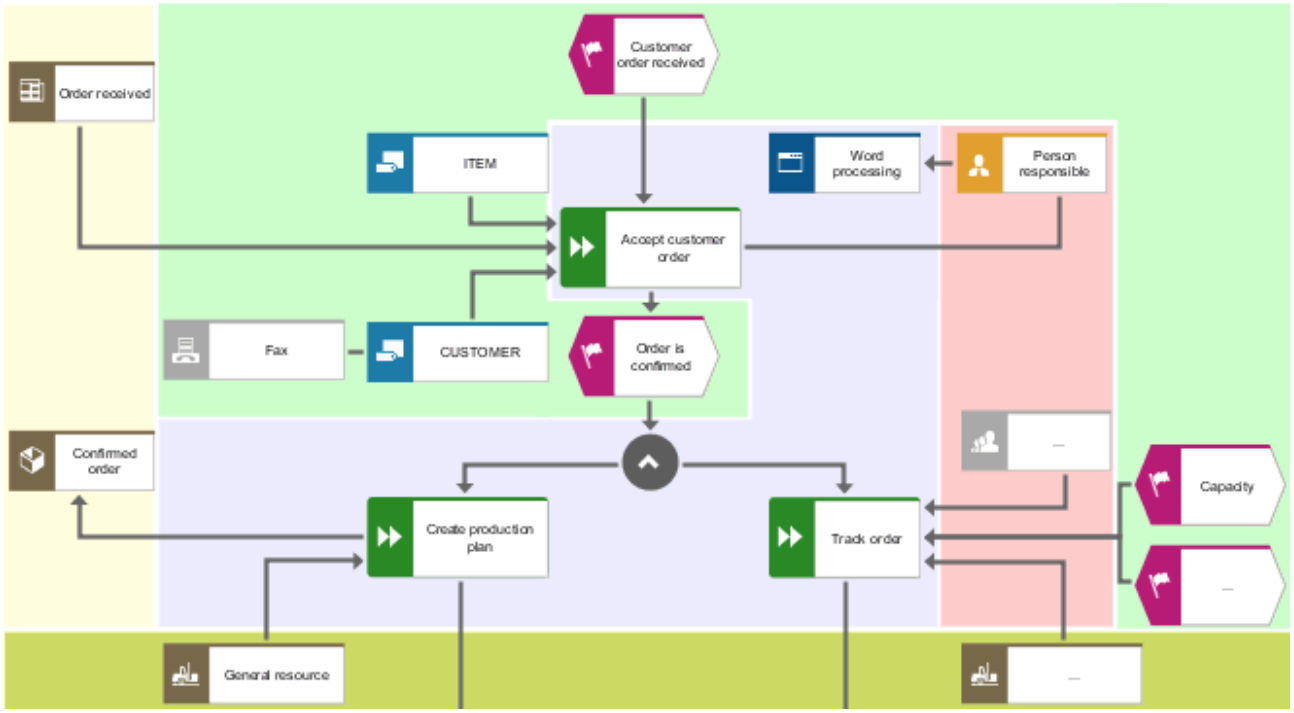
Once the **Accept customer order** procedure is completed, the **Order is confirmed** event occurs that in turn may trigger other procedures, such as **Track order** or **Create production plan**. The **Order** object is now in a new state because the **Order received** object has become an **Order confirmed** object. Carrying out the **Accept customer order** function has generated a product/service that is used - in combination with human and technical resources - as an input for processing subsequent procedures.



Business process model

The components required to provide a full description of a business process include procedures, events, products/services (states), users, organizational units, and information technology resources. Covering all effects on all elements of every procedure under consideration would result in a rather complex model and lead to redundancies in the description.

To reduce complexity, the general context is broken down into individual views (see the following figure) that represent specific modeling and design aspects (see Scheer, Architecture of Integrated Information Systems, 1992, p. 13 et sqq.). These can be processed independently. The views are broken down in such a way that relationships between the components are rather numerous within a single view, while there are only relatively few relationships between the various views.



Process model views

Events, such as **Customer order received** or **Invoice created**, represent the fact that the state of information objects (data) changes. Events are described in the data view of the ARIS architecture.

The states that exist in the objects' environment, for example, within the scope of the customer order, are represented by products/services. The term product/service refers to the supply of either goods or services. Services that create and provide information are information services. Products/Services also include the provision of financial resources. Relationships between products/services are described in the ARIS architecture's **Product/Service view**.

The functions to be carried out (procedures) and their interrelationships form a second view, the **Function view**. It contains the description of the function, an enumeration of individual subfunctions that are part of the overall context, and the relationships that exist between the functions.

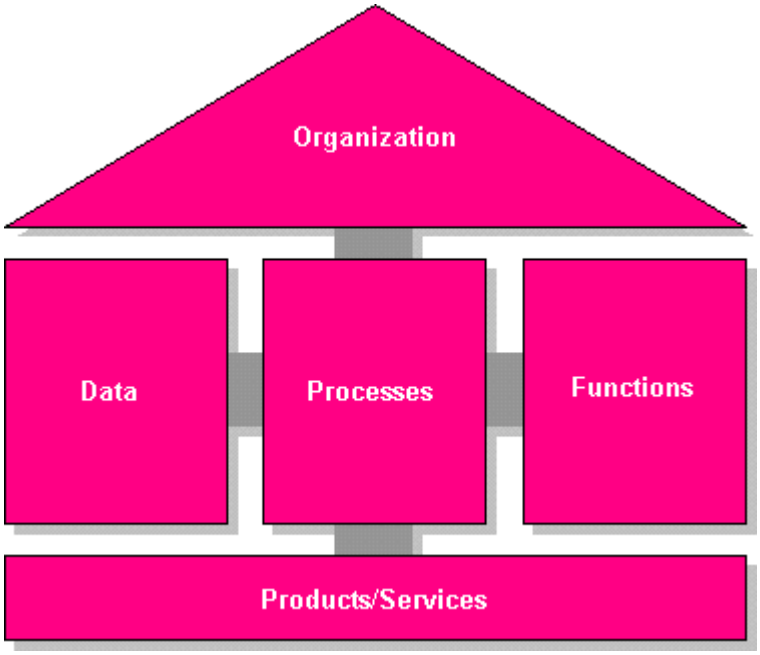
The **Organization view** subsumes users and organizational units, as well as their relationships and structures.

Information technology resources constitute the fourth area of consideration, the so-called **Resource view**. However, this view is significant for the technical consideration of business processes only insofar as it provides general conditions for describing other components that are more directly geared toward business management. For this reason, the components of the other views (Data, Function, and Organization view) are described in terms of their proximity to the information technology resources. Thus, resources are dealt with at the design specification and implementation level of the other views (see chapter Descriptive levels [12]). The lifecycle model that is defined as a result of the descriptive level approach replaces the resource view as an independent object of consideration.

While breaking down the process into individual views reduces complexity, the process component relationships across the views are lost. For this reason, the **Control view** is provided as an additional view for describing the relationships between views. Combining these relationships in a separate view allows for systematic and redundancy-free recording of all relationships.

The control view is an essential component of ARIS. It is the fundamental feature that distinguishes the ARIS concept from other architecture approaches (for comparison with other architecture approaches see Scheer, Architecture of Integrated Information Systems, p. 24 et sqq.).

Thus, there is a total of five ARIS views, which form the basis of the following method descriptions.



Views of a process model

2.3. Descriptive levels

As mentioned earlier, the ARIS resource view is replaced by a lifecycle concept of an information system's descriptions.

Lifecycle models in the form of level or phase concepts describe the lifecycle of information systems. However, the ARIS lifecycle model is not to be understood as a procedure model for developing an information system. It rather serves to define the various descriptions that differ in their proximity to information technology.

ARIS uses the three-tier division shown in the following figure (see Scheer, Architecture of Integrated Information Systems, 1992, p. 16 et sqq.).

The focus of this approach is on the **business management problem**. The description lists rough facts that focus very closely on technical objectives and technical language. The options that information technology provides for supporting business management processes and decisions are also included. Therefore, only semi-formal description methods are used for representation purposes. Because of their lack of detail and their highly technical vocabulary, these description methods cannot serve as a starting point for a formalized implementation.

Thus, a **Requirements definition** has a rather formalized description language to describe the business management approach to be supported, so that it can be used as the basis for consistent transformation into information technology. This procedure is also referred to as (semantic) modeling. The requirements definition is closely associated with the business management problem, as indicated by the width of the arrow in the following figure.

Applying the requirements definition's concept to the design of IT systems leads to the **Design specification** level. Here, the modules or transactions that carry out technical functions are defined, not the functions themselves. At this level, the requirements definition is aligned with general concepts used in information technology. However, the requirements definition and the design specification are only loosely coupled. This

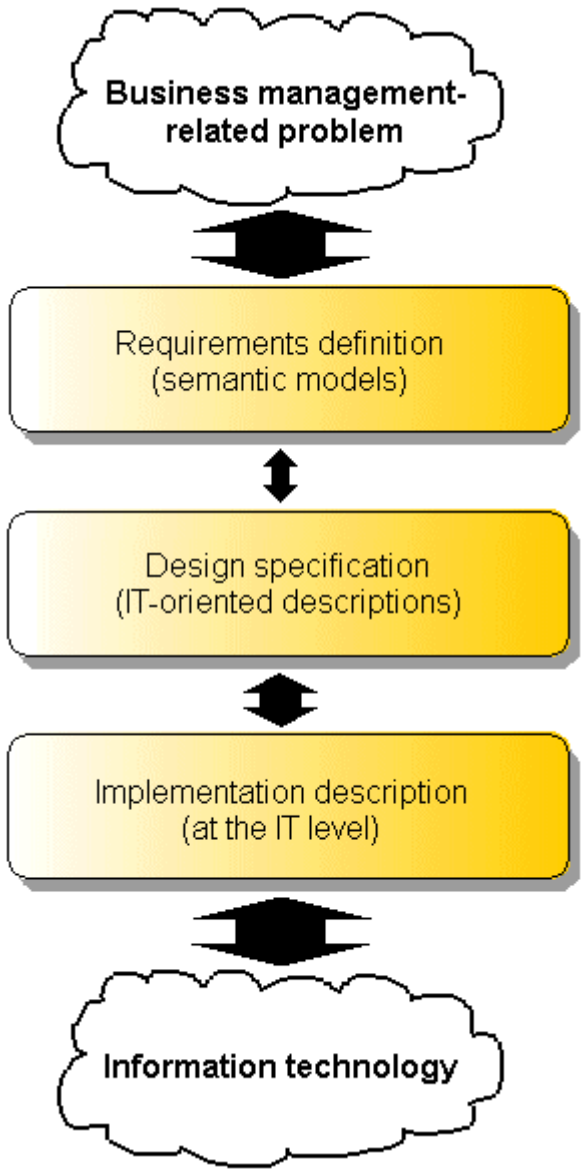
means that a design specification can be changed without affecting the requirements definition. However, this does not imply that requirement definitions and design specifications can be developed separately from each other. In fact, once a requirements definition is complete, the business management-related topics should be specified in such a way that purely IT-specific considerations, such as the information system performance, do not have any impact on the technical content.

At the **Implementation** level, the design specification is transformed into functional hardware and software components. This establishes the link to information technology.

The descriptions have individual update cycles. The update frequency is lowest at the requirements definition level and highest at the implementation level.

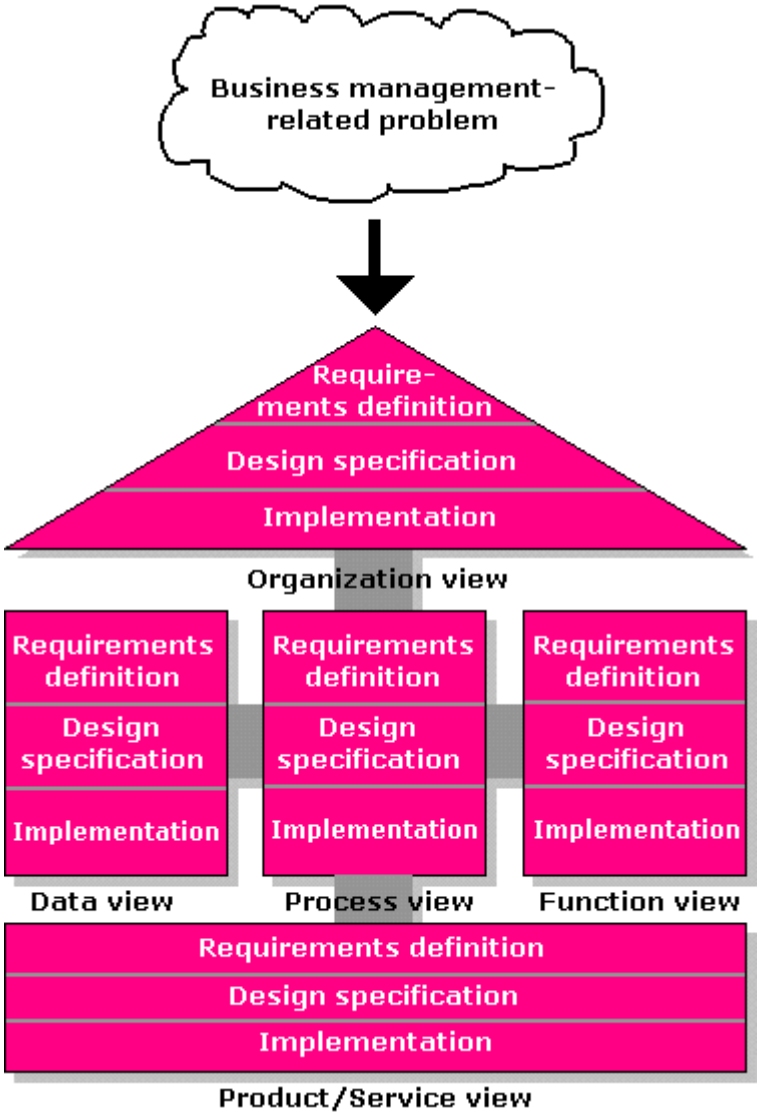
The implementation level is closely coupled with information technology development and is subject to continuous change as a result of rapid innovation cycles in information technology.

The requirements definition level is of particular importance because it serves as a repository for the long-term business management approach and, at the same time, is the starting point for further steps towards technical implementation. Requirement definitions have the longest lifecycle and – due to their close proximity to the business management problem – also document the technical benefits of the information system. For this reason, the view that deals with developing requirement definitions or semantic models is the one with the highest priority. Semantic models build the bridge between users and the initial translation of their business management problem into an IT-related language.



Descriptions of an information system

The combination of views, descriptions, and business management solutions forms the essence of the ARIS concept. As shown in the following figure, each descriptive view is broken down into the **Requirements definition**, **Design specification** and **Implementation** level.



ARIS concept

The ARIS concept sums up the relevant objects or areas of consideration as defined by the architecture's descriptive views and levels. Including the business management problem, which is the focus of this approach, this gives rise to a total of thirteen components. In a next step it is necessary to select and explain suitable description methods for each object or area of consideration.

The criteria for selecting these methods (see Scheer, Business Process Engineering, 1994) include:

- simplicity and understandability of the means of representation,
- suitability for the content to be expressed,
- ability to use consistent methods for all applications to be represented,
- existing or expected level of familiarity with the methods, and
- independence of the methods from technical developments in information technology.

Individual methods applied to the objects or areas of consideration are described in the following chapters.

3. Modeling within the views and levels of the ARIS concept

3.1. Function view

3.1.1. Requirements definition

Modeling methods often look at functions in the context of objects from other descriptive views of ARIS. For example, the relationship between data and functions is displayed to illustrate how the input/output data affect the process of transforming a function.

In contrast, the ARIS architecture strictly separates the various areas of consideration (see Scheer, Architecture of Integrated Information Systems, 1992, p. 62). Consequently, the function view covers only those means of representation that show the interconnections between functions. Relationships between functions and data are displayed in the ARIS process view.

A function is a technical task or activity performed for an object to support one or more business objectives (see Scheer, Architecture of Integrated Information Systems, 1992, p. 63).

Functions are displayed as rectangles with rounded corners:

'Verify customer inquiry' function

Usually, the criterion for establishing such a function is an information object, such as a customer inquiry or a production order. This should also be expressed in the function name. This is shown in the above figure. **Customer inquiry** defines the object while **Check** indicates the operation that is performed for this object. At a higher level, however, mostly a noun is used as the function name (for example, Procurement logistics, Production, Sales).

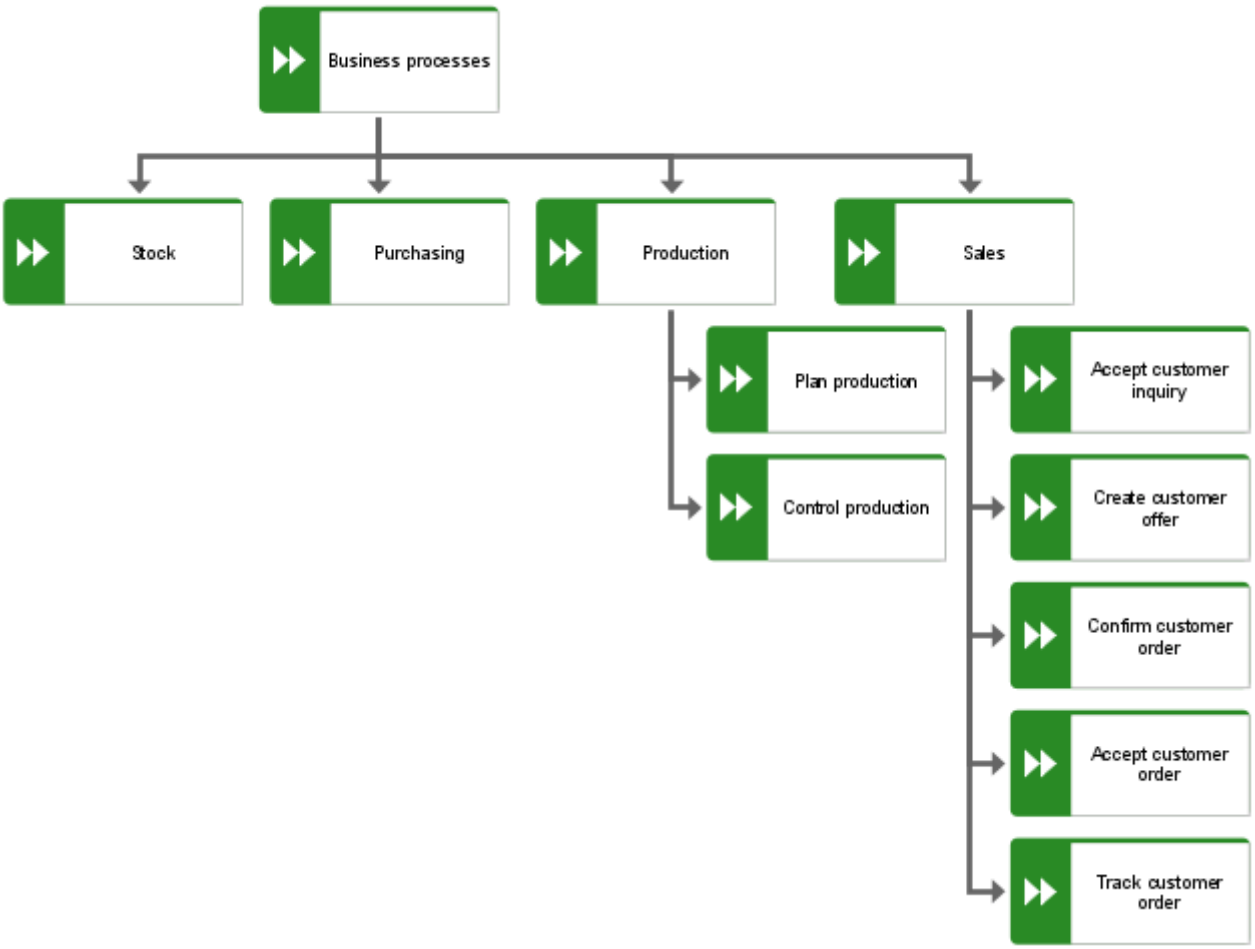
3.1.1.1. Function tree

Functions can be described at different aggregation levels. Accumulations of functions in the form of business processes or process chains form the top level of aggregation. An example may be the processing of a customer order, from customer inquiry through to shipping. A business process thus represents a complex function that can be broken down into subfunctions to reduce its complexity. The term 'function' can be used at all hierarchy levels. However, other terms, such as procedure, process, subfunction, or elementary function, are also used to indicate the hierarchy level.

Breaking down functions can be done across multiple hierarchy levels. Elementary functions represent the lowest level in semantic function trees.

Elementary functions are functions that, from the business management point of view, cannot be broken down any further.

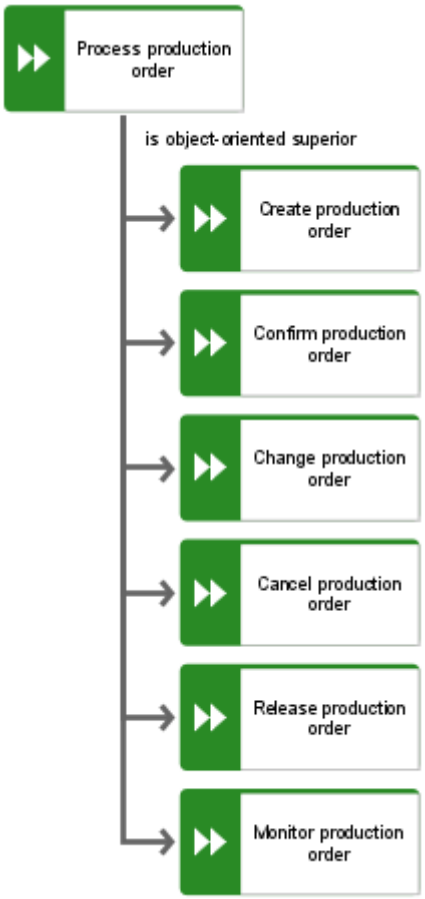
Hierarchical structures are best represented using function trees or hierarchy models.



Function tree (extract)

Grouping functions within a function tree can be performed according to different criteria (see Brombacher/ Bungert, 'Praxis der Unternehmensmodellierung' [Enterprise modeling practise], 1992). Criteria frequently used for this purpose include: processing of the same object (object-oriented), breakdown according to process affiliation (process-oriented), or grouping of functions in charge of the same operations (execution-oriented).

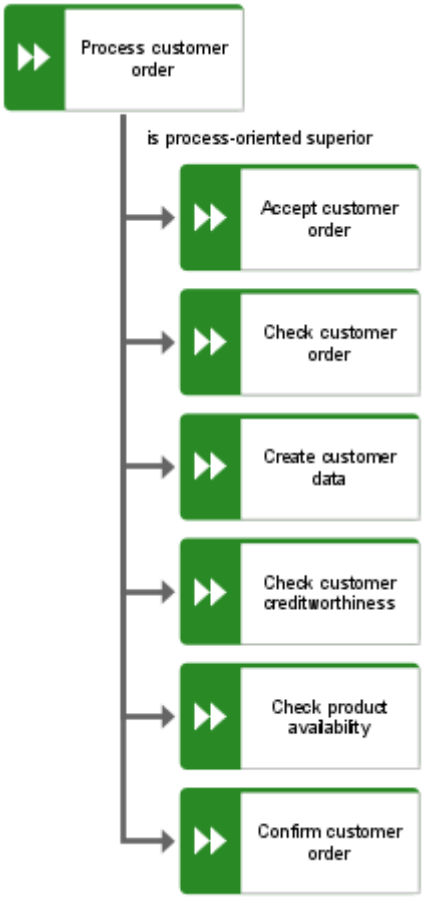
The next figure shows an example of an object-oriented breakdown. The superior **Process production order** function is subdivided into the functions **Create production order**, **Confirm production order**, **Update production order**, **Cancel production order**, **Release production order** and **Monitor production order**. These functions describe different operations (create, update, cancel, etc.) that are performed for one and the same object, which is **Production order**.



Object-oriented function tree

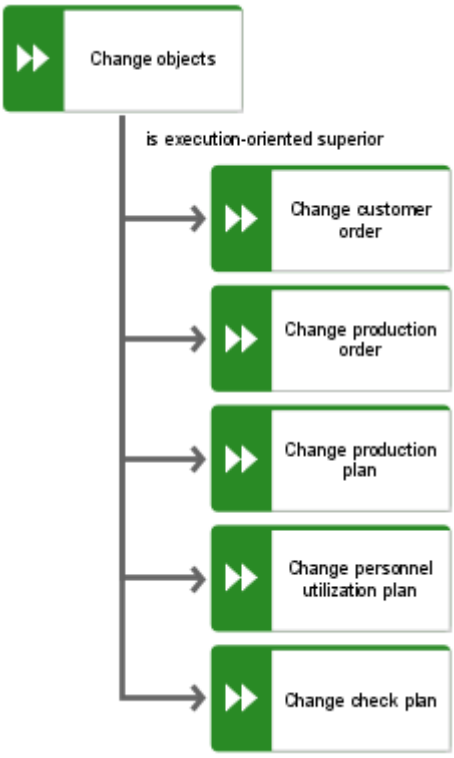
A process-oriented representation is recommended for function trees that represent the results of business process modeling. The following figure shows an example of process-oriented function breakdown.

The functions **Accept customer order**, **Check customer order**, **Create customer data**, **Check customer creditworthiness**, **Check product availability**, and **Confirm customer order** are part of the **Process customer order** business process. Unlike the object-oriented breakdown, the operations here are performed for different objects (customer order, product availability, etc.).



Process-oriented function tree

Execution-oriented grouping means that all functions performing the same operation (check, create, delete) for different information objects are grouped together. An example of the **Change** operation is shown in the following figure. The functions shown may occur in different processes and are involved in processing different objects. However, the type of operation they perform for the various objects is always the same.



Execution-oriented function tree

Representing functions in a function tree reduces complexity, but the representation is static. Besides the static representation, the chronological sequence of functions may also be of interest. Chronological-logical operational sequences are represented in so-called event-driven process chains (EPCs). These contain not only functions, but also events linking the functions. Events belong to the data view in ARIS. In line with the principle of separation of views stipulated by ARIS (see chapter Requirements definition [16]), event-driven process chains are described in the ARIS control view.

Describing functions from a requirements definition-related point of view involves not only the principle of breaking down functions into subfunctions, but also other function properties, especially those that can influence the design of business processes.

For example, it is recommended that functions always include information on whether or not user intervention is required for carrying out the function. Functions of similar type that can be carried out automatically may be bundled and processed in a batch run.

Information on the quantity structure of a function (for example, number of inquiries processed in a day) and on the total amount of time it takes to carry out the function provide further data that can serve as a basis for decision-making with regard to the redesign of business processes. The total amount of time can be further divided into individual units of time (orientation time, processing time, wait time). In ARIS, this information can be saved in the attributes of the **Function** object type. A list of all attribute types that are available is provided in the **ARIS Method Reference** help.

3.1.1.2. Objective diagram

Before a company starts modeling, analyzing, or optimizing business processes (Business Process Reengineering), it should define the objectives it wants to achieve by modeling the company's business processes.

In the objective diagram, companies can define their (business) objectives, arrange them in an objective hierarchy, etc.

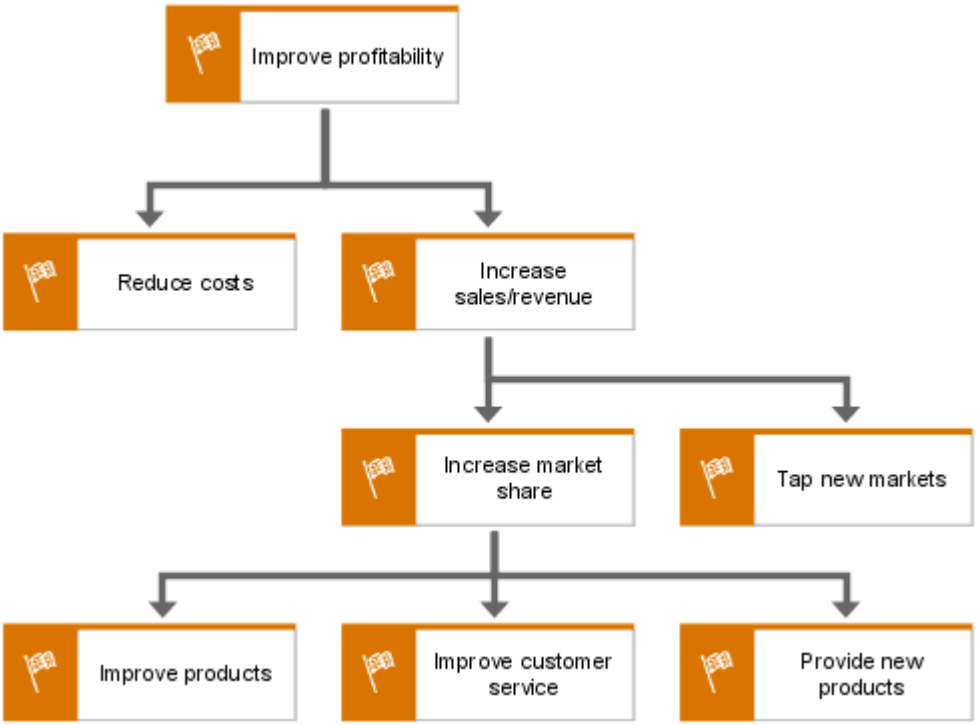
An objective defines future business objectives that are to be achieved by promoting success factors and implementing new business processes.

Factors that may be critical for achieving objectives can be specified, arranged in a hierarchy, and assigned to the goals they help accomplish.

Success factors specify the aspects that need to be considered in order to achieve a specific business objective. They are assigned to business objectives in the objective diagram.

This model type is linked to other model types of the requirements definition via the **Function** object type. The functions (business processes) contributing to the achievement of an objective can be displayed for every single objective. When establishing the procedure model in the business process modeling and optimizing phase, both the prioritization of objectives as well as the assigned functions should be taken into account.

The following figure shows an example of an objective diagram.



Objective diagram

3.1.2. Design specification - Application system type diagram

The design specification of the function view contains the specification for the application system and module types, as well as the modular structure of the application system type, an outline of individual transaction steps, and the definition of input and output presentations in the form of draft lists and screen designs.

Key questions to which the design specification of the function view provides answers are:

- How can application system types, module types, or IT functions support the functions defined in the requirements definition?
- What is the modular structure of application system types or module types?
- Which lists and screens are required to carry out a function?
- Which lists can be created with an application system type or a module type, and which screens do application system types and module types use?
- What technology (operating system, user interface, database management system) is an application system type based on?
- Which business objectives are pursued when a specific application system type is used?

Thus, the **Application system type** is the key object type of the function view's design specification.

Unlike concrete application systems that come into play only at the implementation level of the function view and that represent specific, identifiable (for example, by a license number) application systems within a company, application system types are generated as the result of typifying all application systems that are based on precisely the same technology.

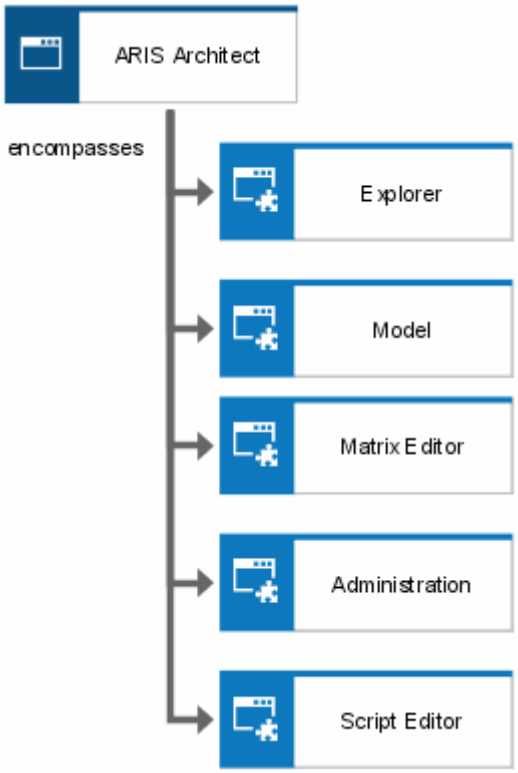
An application system type typifies individual application systems that are based on precisely the same technology.

Example: ARIS Architect is an application system type. You can purchase several licenses for this application system type and thus obtain various individual application systems.

Application system types are represented by the following graphic symbol:

Application system type diagram (1) design specification

Application system types are mostly modular in structure. The application system type diagram is a means of representing this modular structure. Application system types are broken down into module types. The following figure shows an example:



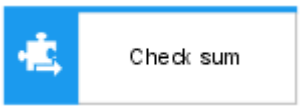
Application system type diagram (2) design specification

In the above example, ARIS Architect consists of the module types **Explorer**, **Model**, **Matrix Editor**, **Administration**, and **Script Editor**. As with application system types, module types typify individual modules that are based on precisely the same technology. Module types are components of application system types. They are capable of autonomous operation.

A module type is a component of an application system type, which is capable of autonomous operation. Module types typify individual modules that are based on precisely the same technology.

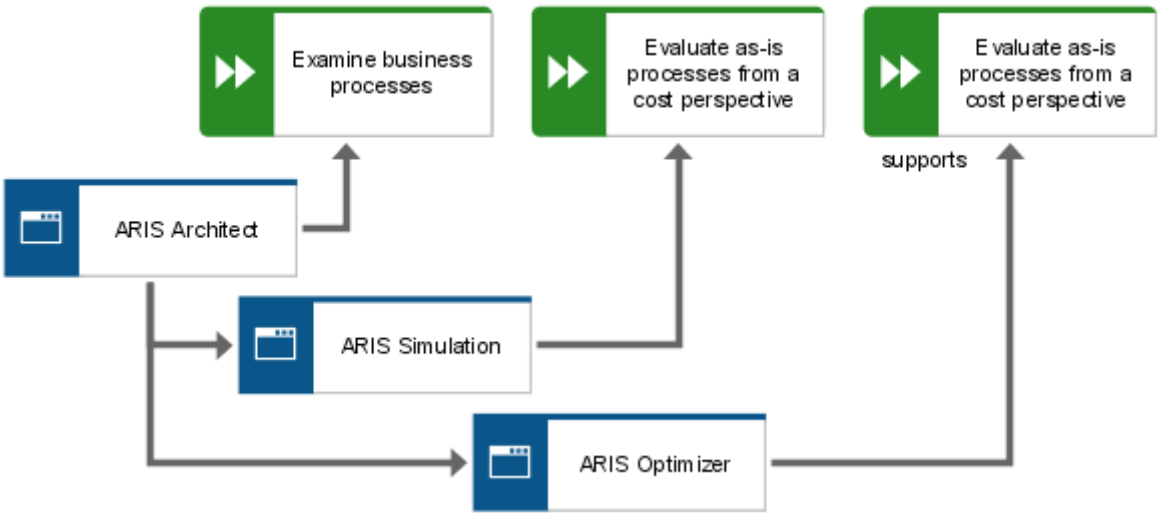
Application system types and module types can be arranged in any hierarchy. At the lowest level, module types can be divided into IT function types.

An IT function type, in the sense of a transaction, is the smallest unit of a module type. IT function types are realized as individual program modules and must always be carried out completely to process an individual work step.



Application system type diagram (3) design specification

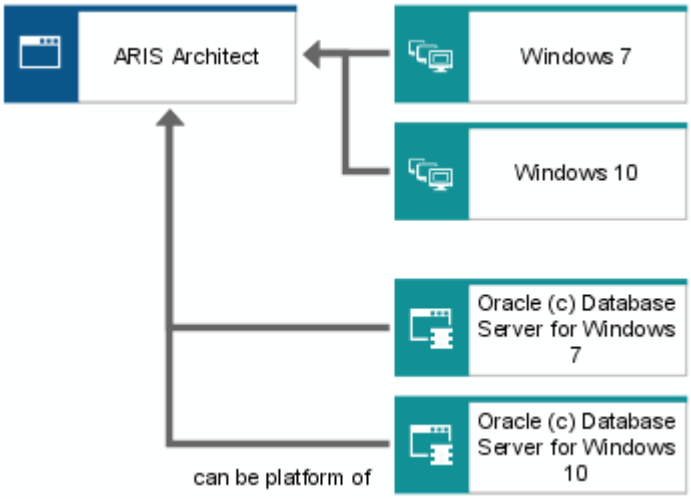
The application system type diagram is also a means of defining the functions of the requirements definition that are supported by the specified application system types and module types. This assignment forms the link between the requirements definition and the design specification of the function view. The following figure shows an example.



Application system diagram (4) design specification

To obtain a more detailed specification of the technology that application system types and module types are based on, it is possible to allocate to them the types of user interfaces, database management systems, and operating systems under which they can run, as well as the programming languages that are used to implement them. As this concerns types and not concrete specimen, multiple relationships are possible. For example, an application system type can be assigned the **Windows 7** and **Windows 8** user interface, which means that the application system type can run under both user interfaces. A unique relationship is required only when the graphical user interface is assigned to a concrete specimen (that is, a specific application system) at the implementation level of the function view. This relationship describes the exact configuration of the application system type license that the company purchased.

An example of possible assignments in an application system type diagram is shown in the following figure.

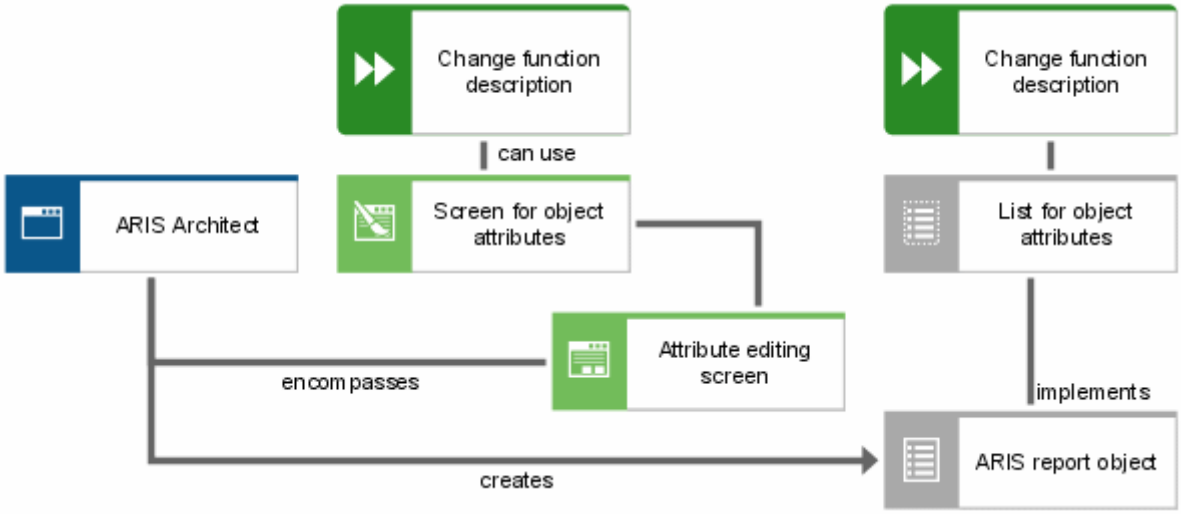


Application system diagram (5) design specification

Processing a technical function with the support of an application system involves the use of various screens and the creation or use of various lists provided by the corresponding application system. For this purpose, the **List** and **Screen** objects are available and can be assigned to either the technical function or the application system types and module types.

If, in a first step, general operational procedures are to be defined without reference to specific application system types, the **Draft list** and **Screen design** objects can be used to specify the required screens and lists. First, both object types specify in general which type of list or screen is to be used (for example,

Enter customer data), without establishing a specific reference to application system type lists or screens. Subsequently, these draft lists and screen designs can be assigned to specific lists and screens. Existing assignments determine possible implementation scenarios. The following figure illustrates an example.



Application system diagram (6) design specification

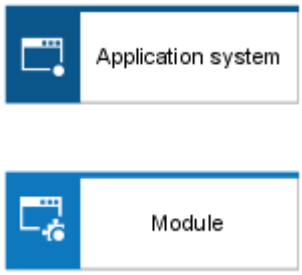
A list of object types and relationships that are available in an application system type diagram is provided in the **ARIS Method Reference** help.

3.1.3. Implementation - Application system diagram

In the application system type diagram, you can assign specific application systems and modules to the application system types and module types described in the design specification. Application systems are specimens of an application system type that exist in the company and can be uniquely identified, for example, by the license numbers.

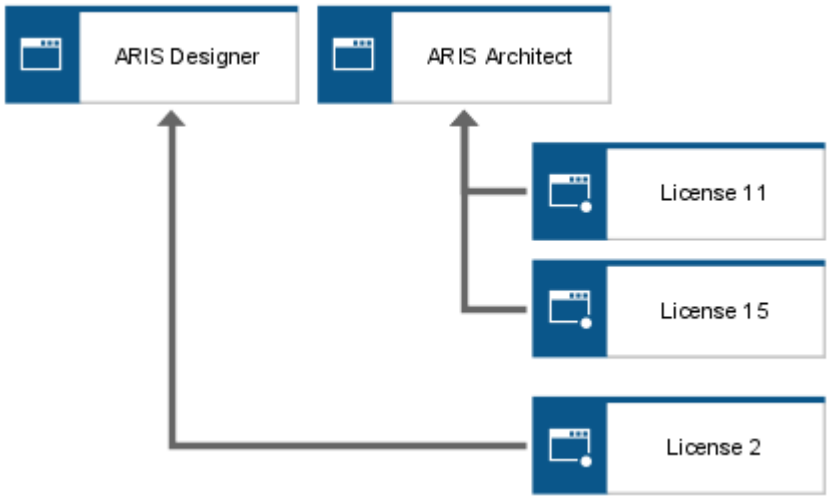
An application system (module) is an individual specimen of an application system type (module type), which can be uniquely identified, for example, by the license number.

Application systems and modules are displayed graphically as follows.



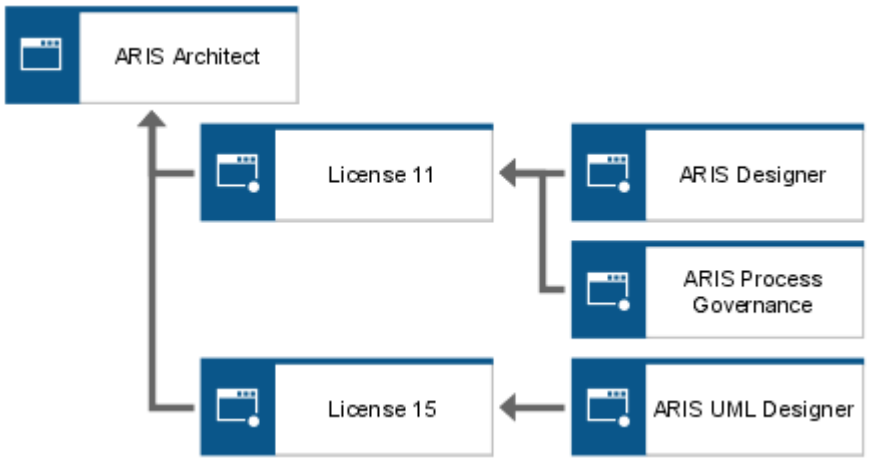
Application system type diagram (7) implementation

As a company may have more than one license for an application system type (module type), more than one application system (module) can be assigned to an application system type (module type) in the application system diagram.



Application system type diagram (8) implementation

The application system diagram shows the actual modular structure of an application system. While the design specification lists all modular components that an application system type may have, here we are dealing with a single application system license so that the modular components can be uniquely defined for each license. Therefore, a company may have multiple application systems of the same application system type, but with completely different modular structures.



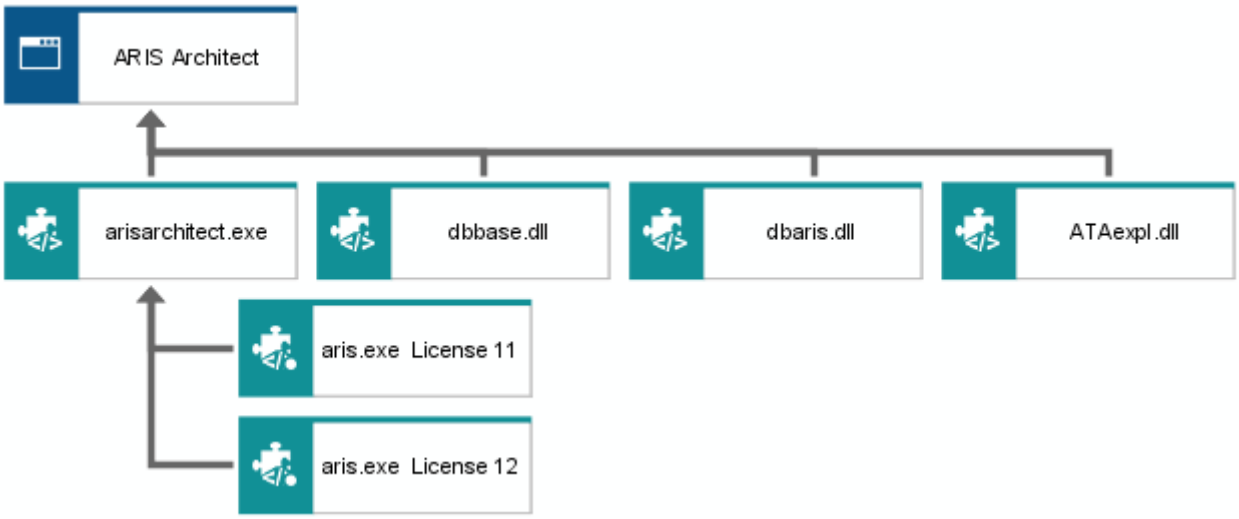
Application system type diagram (9) implementation

The implementation level not only represents all existing application systems and modules, but also enables the technical (physical) implementation of application systems to be defined in the form of individual program files.

The application system type diagram illustrates which program module types are required to implement an application system type or module type.

A program module is a program file on a storage medium obtained by purchasing a license (for example, an EXE file or COM file). A program module type is created by typifying program modules that are based on precisely the same technology.

The following figure illustrates the assignment of program module types to an application system type and of individual program modules to program module types.



Application system type diagram (10) implementation

The **ARIS Architect** application system type consists of the **arisarchitect.exe**, **dbbase.dll**, **abaris.dll**, and **ATAexpl.dll** program module types. Multiple specimens (program modules) of each program module type may exist in the company if several licenses were purchased or if backup copies were created.

Program module types and program modules can be arranged in any hierarchy. For a more precise technological specification of the program it is possible to also represent the access of program module types to program libraries in the application system type diagram.

A list of object types and relationships that are available in an application system type diagram is provided in the **ARIS Method Reference** help.

3.2. Data view

3.2.1. Requirements definition

The requirements definition of the data view includes a description of the semantic data model of the area of consideration. In line with the breakdown approach stipulated by ARIS, the description covers both the objects that specify the start and end events of a process chain and the current state of the relevant process chain environment.

Unlike function modeling, data modeling is particularly demanding as far as the method is concerned. In the function view, the only object examined is the function. Furthermore, relationships between functions simply illustrate superordination or subordination.

Chen's Entity-Relationship Model (ERM) is the most widely used designing method for semantic data models (see Chen, The Entity-Relationship Model, 1976). This modeling method uses a variety of specialized terms, such as entity type, relationship type, attribute, etc. The relationships that exist between these objects are numerous and - compared with function modeling - significantly more difficult to classify.

The following pages introduce modeling with entity relationship models (ERM). First, the objects and relationships of Chen's base model are explained. Then, another chapter describes several rules that were added later to the base model.

3.2.1.1. The ERM base model

The base model distinguishes between entities, attributes, and relationships. Basically, a distinction is made between type level and occurrence level.

Entities are real or abstract objects that are relevant for the business management tasks being examined.

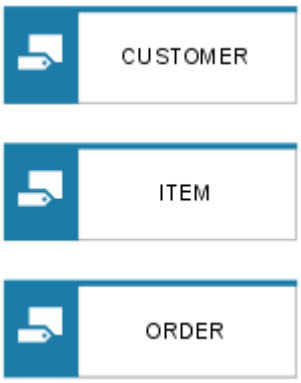
For example, a business process can be a considered object. According to the ARIS breakdown model, the data objects of interest are objects of the environment and objects specifying events. Examples of entities in the **Customer order processing** process are:

- Customer 1235,
- Item 471,
- Order 11.

Entities are described in more detail by specific attributes (properties). For example, a customer can be specified more precisely by his name, first name, and address.

If similar entities are grouped into sets, these are referred to as entity types, the individual occurrences of which are the entities.

Entities of a similar type can be described by the same attributes. For example, customer **Moore** and customer **Miller** are grouped under the **Customer** entity type; item **4710** and item **4712** are grouped under the **Item** entity type. Entity types are displayed as rectangles in the ERM (see the figure below). In the following, entity types are indicated by capitalized text.



Examples of entity types

Attributes are properties describing entity types.

Attribute occurrences are specific values of attributes of individual entities. For example, customer **1235** can be described by attribute occurrences such as **Miller**, **Peter**, and **Munich**. The relevant attributes are **Name**, **First name**, and **City**.

Attributes are usually represented by an oval or a circle. In the following, attributes are represented by ovals. The following figure shows examples of attributes for the CUSTOMER entity type.

Attributes of the 'Customer' entity type

Entity types and attributes are often hard to distinguish and can sometimes only be determined from the context of the modeling procedure. For example, a customer address can be understood as an entity and not as an attribute of CUSTOMER. In this case, a separate entity type ADDRESS would be modeled with a relationship to CUSTOMER. A helpful criterion for specifying whether you are dealing with an entity type or an attribute is the fact that entities have attributes, while attributes cannot have their own attributes. An attribute that is created in an ERM and is to be described by further attributes later on thus becomes an entity type. Whether or not an object will have relationships with other entity types is another helpful question. If yes, the object under consideration is an entity type as well.

A relationship is a logical link between entities.

Therefore, the existence of relationships directly depends on the existence of entities.

If similar relationships are grouped into sets, these are referred to as relationship types.

For example, a possible relationship type between SUPPLIER and PART is SUPPLIES. In the following text, relationship types are also indicated by capital letters. In an ERM, relationship types are displayed as diamonds and are linked with entity types using connections (see the following figure).

Relationship type

Often, only one direction of reading relationship type names results in viable connections. The example above illustrates the relationship **Supplier supplies Part**. In the opposite direction, this would read **Part supplies Supplier**, which is not suitable. If you cannot determine the particular read direction, carefully select superior terms.

Various kinds of relationship types can be distinguished. The distinguishing criteria are the number of entity types linked by relationship types, and the complexity of a relationship.

Thus, unary, binary, or n-ary relationships may exist between entity types.

The complexity or cardinality indicates how many entities of one entity type may be connected with an entity of another entity type.

The relationships to be distinguished are illustrated in the following figure (see Scheer, Business Process Engineering, 1994, p. 34).

There are four different types of relationships (cardinalities):

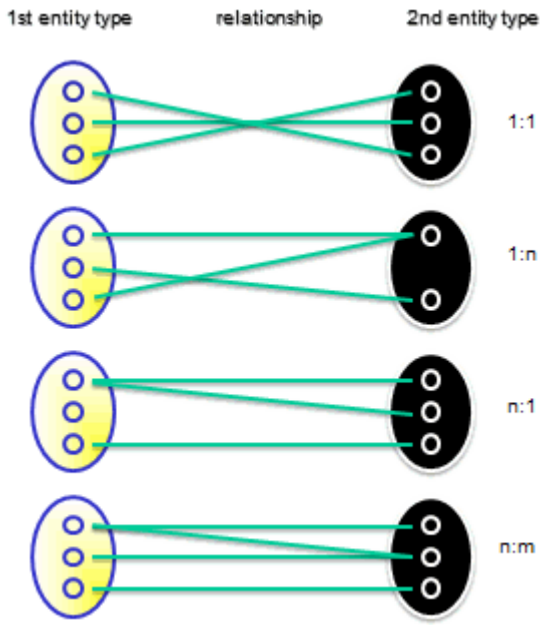
- 1:1 relationship,
- 1:n relationship,
- n:1 relationship,
- n:m relationship.

In a 1:1 relationship, each entity of the first set is assigned to exactly one entity of the second set.

In a 1:n relationship, each entity of the first set is assigned to exactly one entity of the second set, but each entity of the second set may be connected with multiple entities of the first set.

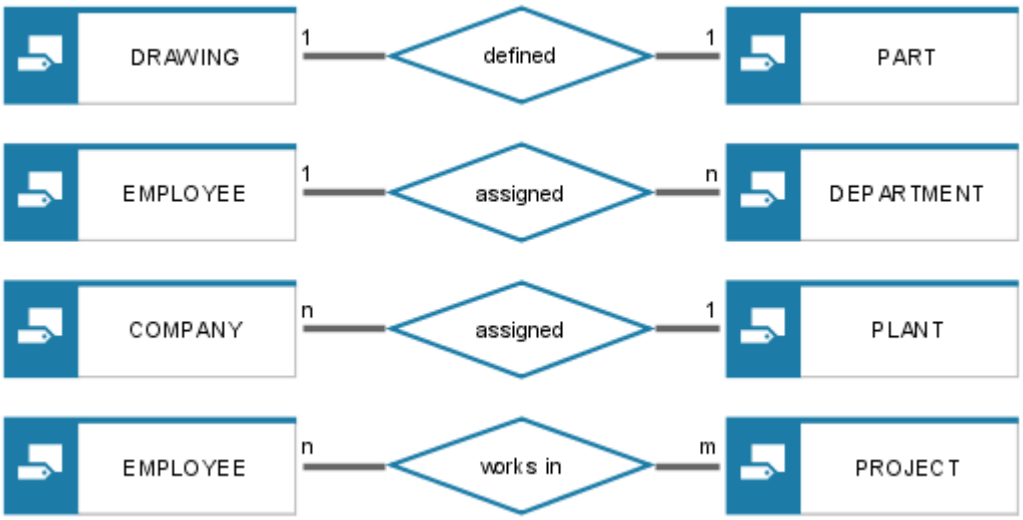
An n:1 relationship means the same, but in reverse order.

In an n:m relationship, multiple entities of the second set are assigned to each entity of the first set and vice versa.



Cardinalities of relationships between entity types

The cardinalities of the relationship type (**Complexity** attribute type) are shown at the connections of the entity relationship model.

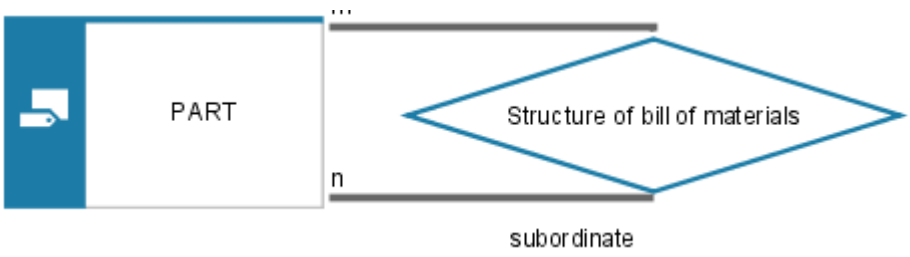


Cardinalities in the ERM

The cardinality on an entity specifies for the entity type in question the maximum number of relationships of a specific relationship type it may have. In the n:1 relationship shown in the above figure, this means that a company of the COMPANY entity type may have multiple ASSIGNED relationships because a company consists of multiple plants, whereas a specific plant may only have a maximum of one ASSIGNED relationship as it must be uniquely assigned to a company.

Chen's original work poses a different interpretation of cardinality. However, the notation used in this manual allows for clearer specifications, particularly when illustrating relationships between multiple entity types. In order to avoid confusion, Chen's original work is not discussed in detail here.

Due to the fact that relationships between entities of one entity type are allowed, two parallel connections may exist between an entity type and a relationship type. These connections can be distinguished by assigning role names. The following figure illustrates recursive relationships. A superior part consists of various subordinate parts. A subordinate part, in turn, may also be used as a component in various superior parts.



ERM for a bill of materials

Both entity types and relationship types can be described by attributes (see the figure below).

The value ranges of attributes are called domains.

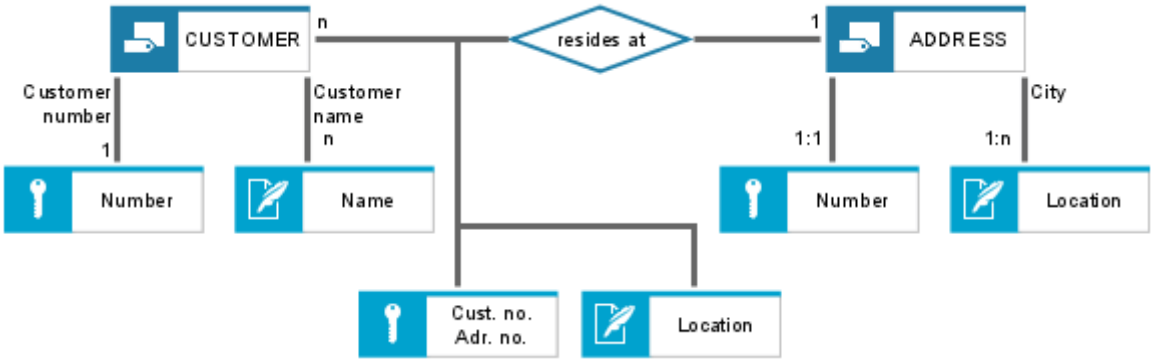
Assignments of domain elements to elements of entity or relationship types are also relationships and can be represented by a connection named accordingly.

A 1:1 relationship must exist between an entity type and at least one domain. The values of this domain uniquely identify individual entities. Therefore, they are called the key attributes of the entity type.

In the example shown in the figure below (see Scheer, Business Process Engineering, 1994, p. 33), the entities of CUSTOMER are uniquely identified by the **Customer ID** key attribute.

Relationships are identified by merging the key attributes of all linked entities. Thus, the key attributes of the RESIDES AT relationship type are Customer ID and Address ID.

The descriptive attributes of the relevant data objects are defined by values derived from domains having a 1:n relationship to entity types or relationship types.



Assignment of attributes in the ERM

3.2.1.2. ERM - eERM extensions

In the last few years, Chen's base model has been substantially extended. This manual will discuss only those extensions that are significant for modeling the data view in the ARIS concept.

3.2.1.3. Design operators added

Design operators provide formal support in creating a data model. Their use ensures systematic creation of data structures and provides the person who looks at an existing data structure with insights into the design process. Based on existing terms, new terms are developed by using design operators. This process is an intellectual procedure running at the level of business management knowledge. The examination of business management facts in terms of data structures either entails that known structures are changed based on new approaches, or that entirely new conclusions are drawn.

Of the numerous and various approaches for extending ERM modeling, four basic design operators have become accepted (see Scheer, Business Process Engineering, 1994, p. 35 et sqq.):

- Classification,
- Generalization,
- Aggregation, and
- Grouping.

Classification

Through classification, objects (entities) of the same type are identified and assigned to a term (entity type). Two objects are identical if the same properties (attributes) are used to describe them.

Classification thus results in the previously described identification of entity types.

Classification of customers

Generalization/Specialization

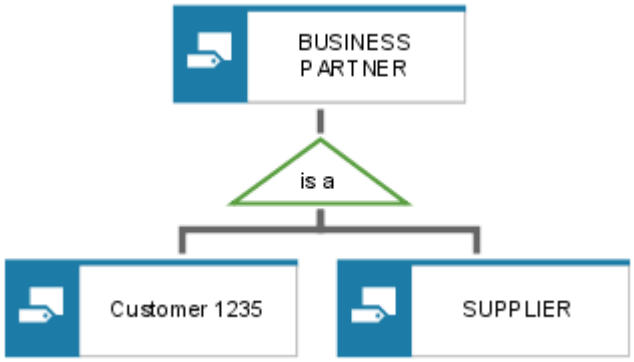
Generalization means that similar object types are grouped under a superior object type.

As shown in the following figure, the **Customer** entity type and the **Supplier** entity type are generalized to form the generic term **Business associate**. Properties (described by attributes) that both source objects share are transferred to the generalized object type. Thus, only those attributes in which the source object types differ are left to be described. The formation of the new entity type **Business associate** is graphically represented by a triangle, also called an 'is a' relationship.

Specialization is the breakdown of a generic term into subterms (**Business associate** is split into **Customer** and **Supplier**).

Specialization is the reverse of generalization. The specialized objects inherit the properties of the generalized object. Apart from these inherited attributes, the specialized object types may have their own attributes. Graphically, specialization and generalization are represented in the same way.

For this reason, the links in the illustration are not drawn as arrows indicating a direction.



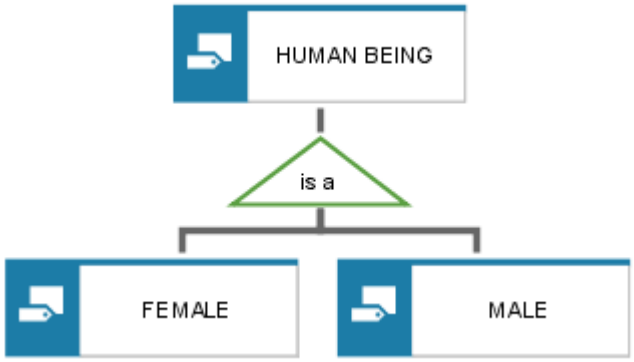
Generalization/Specialization

While specialization primarily supports a top-down procedure when creating a data model, generalization is used to support a bottom-up procedure.

Within the scope of specialization, the completeness and disjunction (alternative) of the subsets formed can be specified as they are created.

Non-disjoint subsets are characterized by the fact that the occurrence of an object may exist in one subset as well as in another. In the above example, a customer may also be a supplier. If an occurrence can be allocated to precisely one subset only, these sets are disjoint.

Complete specialization describes the fact that all specialized object types that meet a specific specialization criterion are listed for a generalized object type. For example, when specializing the **Human being** entity type the **Female** and **Male** entity types can be listed. Thus, specialization in terms of Gender is complete.



Complete specialization

Combining these criteria results in the following four occurrences used for specifying a generalization/ specialization in more detail:

- disjoint/complete,
- disjoint/incomplete,
- non-disjoint/complete,
- non-disjoint/incomplete.

Aggregation

Aggregation is the formation of new object types by combining existing object types. The new object type can be carrier of new properties.

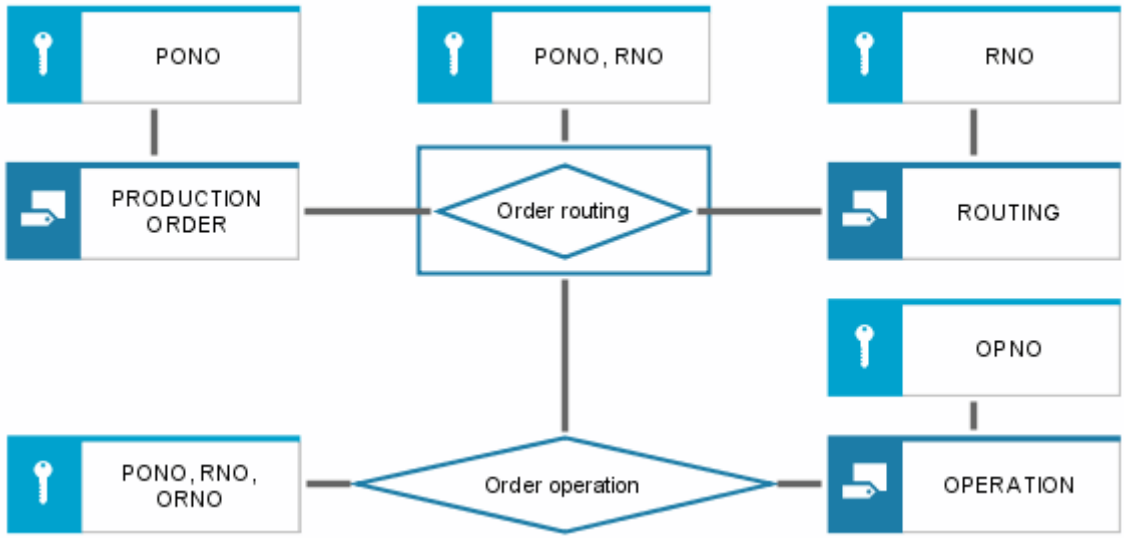
In the ERM, aggregation is expressed by the formation of relationship types (see the following figure). Aggregating the **Production order** and **Routing** entity types forms the new object **Order routing**.

Example of an aggregation

The aggregation operator can also be applied to relationships. An existing relationship type is then treated as an entity type and can thus become the starting point for creating new relationships. An example illustrating this is shown in the following figure.

In a first aggregation step, the **Order routing** relationship type is formed from the **Production order** and **Routing** entity types. The **production order number** (PONO) and **routing number** (RNO) key attributes form the complex key of the order routing. Now, multiple operations can be assigned to the order routing. Therefore, the **Order operation** relationship is formed between the **Order routing** relationship type and the **Operation** entity type. As relationships can be created only between entity types, the original **Order routing** relationship type must be reinterpreted. In the following figure, this is illustrated by a framed diamond. This reinterpreted relationship type thus formed is treated as a 'normal' entity type. To graphically illustrate the formation of the relationship type, the connections of the entity types participating in that formation are drawn to the diamond. The outgoing connections of the reinterpreted relationship type that form new relationships are drawn only to the edges of the rectangle and do not touch the diamond inside the symbol.

Although, as a general rule, it is possible to replace the complex keys with simple keys, maintaining complex keys is useful in terms of tracing the data model's creation.



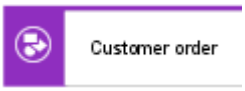
Aggregation with reinterpreted relationship types

In an ERM, a complex structural context is split into a transparent structure. As the relation to the overall structure might become obscured, complex objects in the form of Cluster/Data models are introduced.

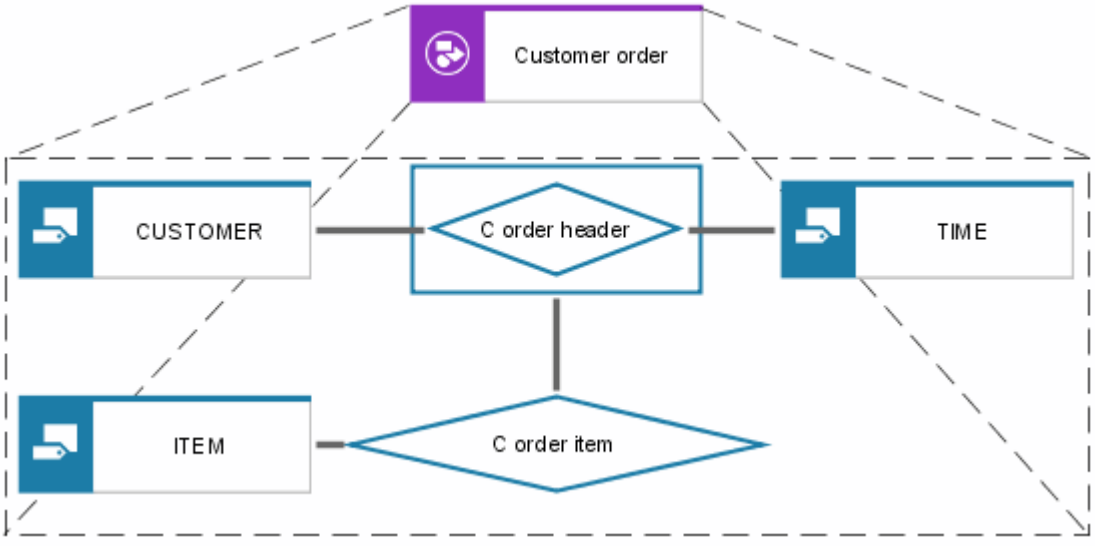
A Cluster/Data model is the logical view of multiple entity types and relationship types of a data model that are required for describing a complex object.

Besides entity types and relationship types, Cluster/Data models themselves can be part of a Cluster/Data model. Unlike entity and relationship types, Cluster/Data models can be arranged in any hierarchy and thus mainly supports a top-down procedure in the process of creating data models. However, forming Cluster/Data models may also be very helpful when combining and consolidating submodels during a bottom-up approach.

The following figure graphically displays a Cluster/Data model.



Cluster/Data model



Cluster/Data model view of multiple objects

The Cluster/Data model represents a logical view of multiple entity types and relationship types. The **Customer**, **Time**, **Customer order header**, **Item**, and **Customer order item** entity types and relationship types are required to describe the complex **Customer order** object.

Grouping

Grouping forms groups from the elements of an entity set.

For example, in the following figure, all Operating resources are combined into an Operating resources group. The operating resources group is an independent object which can be described more precisely by additional attributes (name of the operating resources group, number of operating resources) not contained in the individual operating resources. Other examples are the grouping of workstations into departments or the combination of order line items into orders.

Grouping

3.2.1.4. Extension of cardinalities

When specifying cardinalities, so far only the upper limit for the admitted number of relationship occurrences was indicated. For example, the cardinalities in the following figure indicate that a project can be assigned a maximum number (m) of employees and one employee can participate in a maximum number (n) of projects.

Upper/Lower limit (1)

Besides the upper limit, the lower limit specifying the minimum number of relationship occurrences may also be of interest. For this purpose, the cardinalities can be expressed as a letter pair (a,b), for example, (see Scheer, Business Process Engineering). The letter pair (a1, b1) in the following figure indicates that every project can participate in **at least** a1 and **at most** b1 relationship occurrences of the **works in** type, which means that every project can be assigned **at least** a1 and **at most** b1 employees. The other letter pair (a2, b2) indicates that one employee can participate in **at least** a2 and in **at most** b2 projects.

Upper/Lower limit (2)

Thus, every relationship is defined by two degrees of complexity (minimum, maximum). The lower limit often has the values 0 and 1, whereas the value range for the upper limit is defined as 1 <= max <= * (where * is 'any number').

A lower limit of min = 0 means that an entity may participate in a relationship, but does not necessarily have to. A lower limit of min = 1 indicates that an entity must participate in at least one relationship.

In the following figure, the lower limits indicate that an employee may participate in a relationship, but does not necessarily have to (min = 0), while a project has to participate in at least one relationship (min = 1). What is expressed here is that there can be employees who are not assigned to a project. In turn, however, every project must be assigned at least one employee.

Upper/Lower limit (3)

If minimum values are equal to 0 or 1 and maximum values are equal to 1 or *, the following four cases of a (min,max) notation can be distinguished: (1,1), (1,m), (0,1), and (0,m). Alternatively, the following abbreviated notation can be used (see Schlageter/Stucky, Database systems, 1983, p. 51):

- 1 (corresponds to (1,1))
- c (corresponds to (0,1)),
- m (corresponds to (1,m)),
- cm (corresponds to (0,m)).

The following figure shows the previous graphical example using the abbreviated notation.

Upper/Lower limit (4)

3.2.1.5. Identification and existence dependency

The method of extending cardinalities via the specification of lower and upper limits as discussed in chapter **ERM - eERM extensions [32]** enables certain dependencies between data objects to be defined.

By definition, relationship types and reinterpreted relationship types do not exist autonomously, but come into being due to the existence of the entity types they link. This means that they depend on other entity types in terms of both identification and existence.

In addition, there are entity types that depend on the existence of other entities even though they have their own key attribute. These dependencies may be the result of a grouping operation, for example. Thus, in the example of the following figure, a department requires at least one assigned workstation, while defining a workstation, in turn, implies that it be assigned to a department. As shown in the following figure, these existence-related dependencies are expressed by specifying the complexity. A (min, max) notation uses (1,1) and (1,*). The definition of existence-related dependencies within the data model results in conditions for the referential data integrity when implemented. In other words, complying with these conditions ensures that

the consistency of the database contents is preserved even after certain transactions have been performed. In the example below, this means that a department can be deleted only if all workstations assigned to this department are also deleted.

Existence dependency

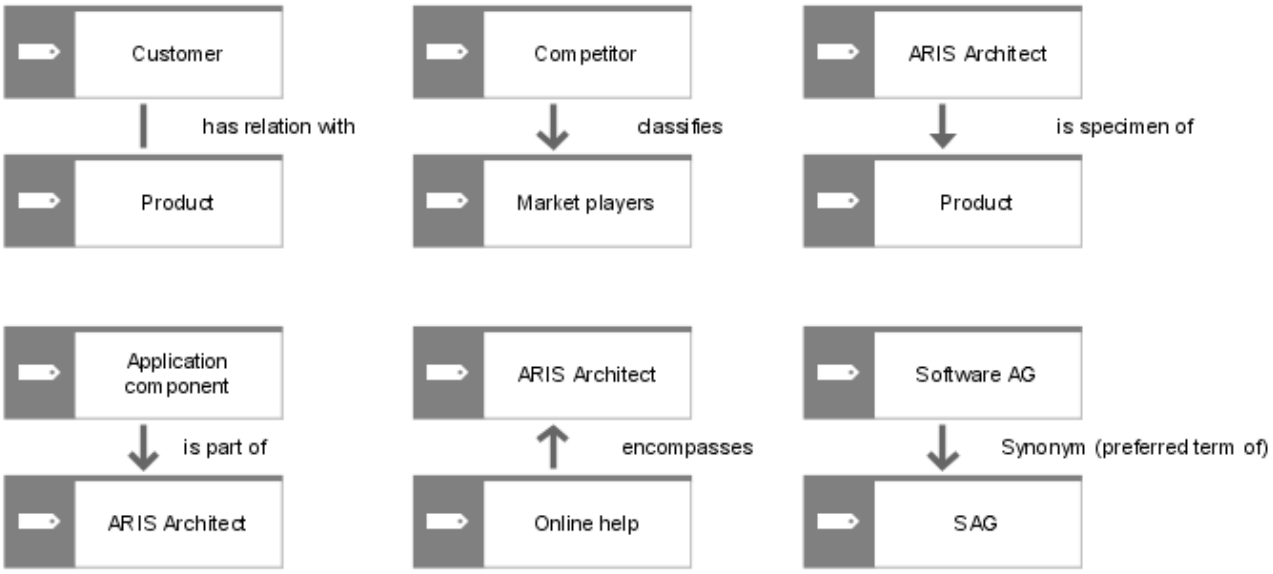
3.2.1.6. Modeling technical terms used in a company - Technical terms model

In modeling, and especially in data modeling, a difficulty that frequently occurs is the variety of terms used for information objects in large companies. For example, the purchasing department's definition of the term **order** differs from the production department's definition of the same term. However, acceptance of the information gathered can be substantially increased by the use of consistent terminology throughout the company or the department. For this reason, ARIS Method provides the so-called Technical terms model that may be used for managing the various terms in the form of a synonym management for data objects, as well as for specifying the relationships that exist between the objects of data models (entity type, relationship type, ...) and the technical terms used within the company.

In order to represent these relationships, the 'Technical term' object type was introduced. You can thus assign multiple technical terms to every information object of the data model. The following figure illustrates an example.

Technical terms (1)

Technical terms can be interrelated and may be arranged in a hierarchy. The following figure illustrates how connection types are used between technical terms.



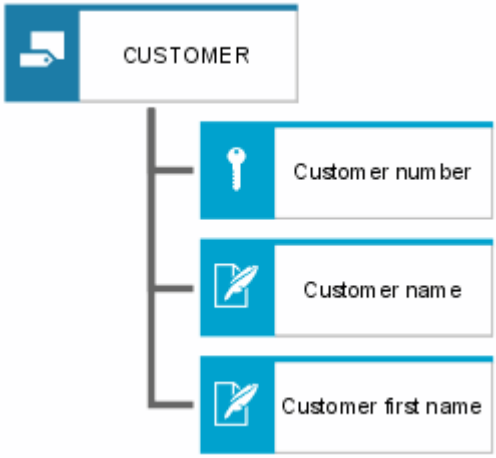
Technical terms (2)

The technical terms defined in the technical terms model can also be used in other model types that contain information objects, for example, in process chains for illustrating a function's input/output data.

3.2.1.7. eERM attribute allocation diagram

Data models in the form of eERMs, even if they display entity types and relationship types only, mostly have a rather complex structure. If ERM attributes were included in these models, they would no longer be legible.

eERM attribute allocation diagrams enable you to assign ERM attribute allocations to every entity type and relationship type in a separate model. The object type of the eERM (entity type or relationship type) can be included in this model as an occurrence copy, and the relationships to the ERM attributes can be modeled. It is possible to distinguish whether the linked ERM attribute is a key attribute, a foreign key, or a descriptive attribute. The following figure illustrates an example.

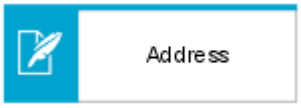


Allocation of ERM attributes to an entity type

This model type not only enables individual ERM attributes to be represented and allocated, but also displays attribute type groups and their allocations.

An attribute type group represents a group of ERM attributes of an entity type that are closely related semantically. For example, ERM attributes of an entity type that, in their entirety, form a secondary key can be combined to form an attribute type group.

Attribute type groups are represented as follows:



Representation of an attribute type group

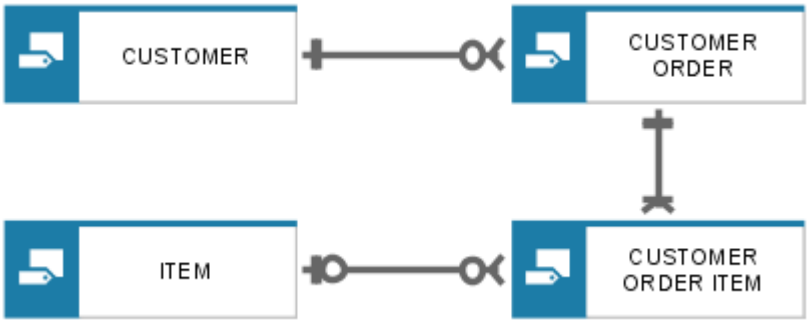
A list of relationships that are available in an ERM attribute allocation diagram is provided in the **ARIS Method Reference** help.

3.2.1.8. IE data model

The IE data model complies with the data modeling notation of the Information Engineering Facility (IEF) CASE tool by Texas Instruments Inc.

It does not provide proprietary object types for relationships between entity types.

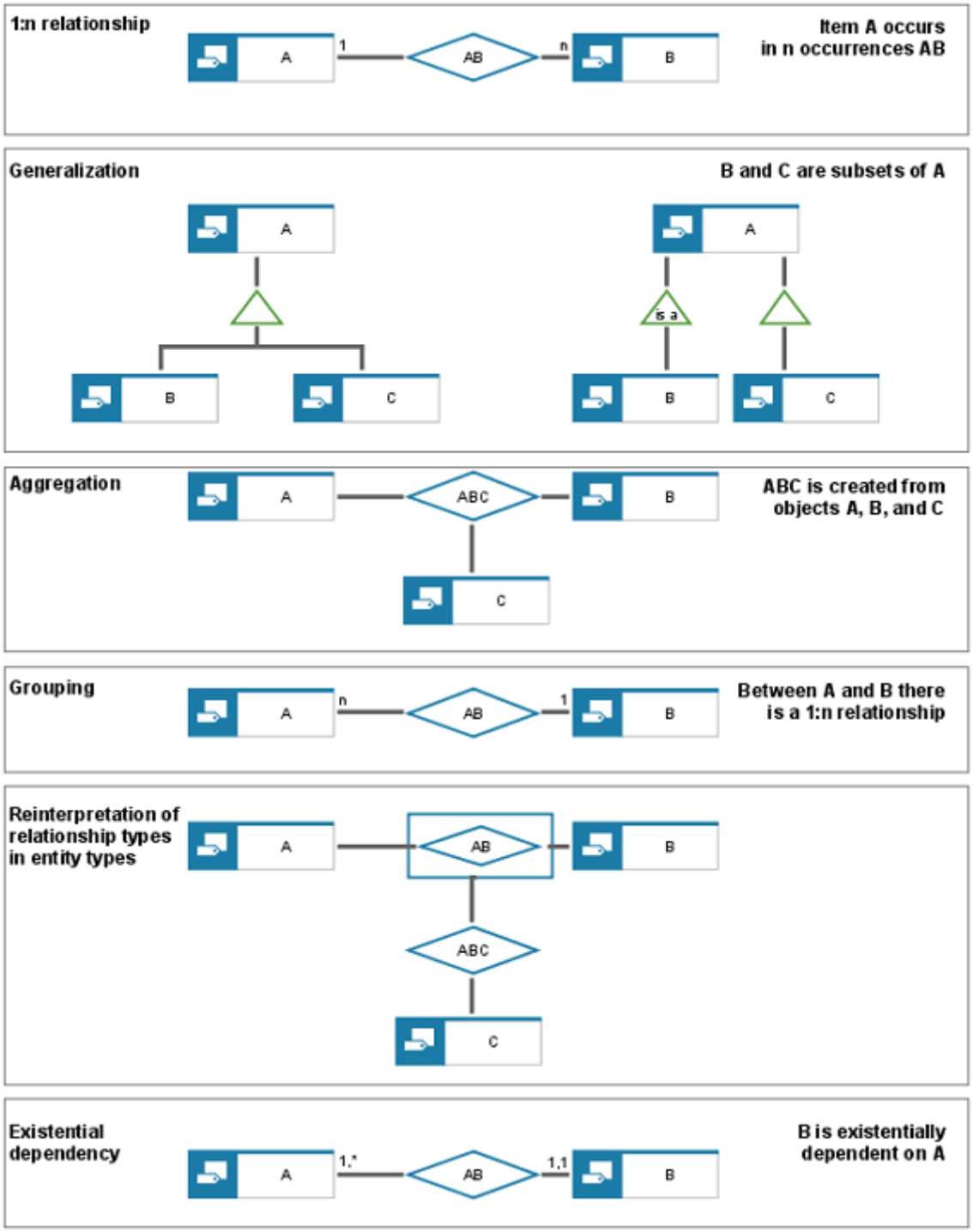
The following figure illustrates an example of a data model in IE notation.



Data model in IE notation

3.2.1.9. Summary of the main terms and forms of representation of the eERM

The terms and representation forms of structural elements and design operators featured by the extended entity relationship model (eERM) are summarized in the following figure (see Scheer, Business Process Engineering, 1994, p. 45).

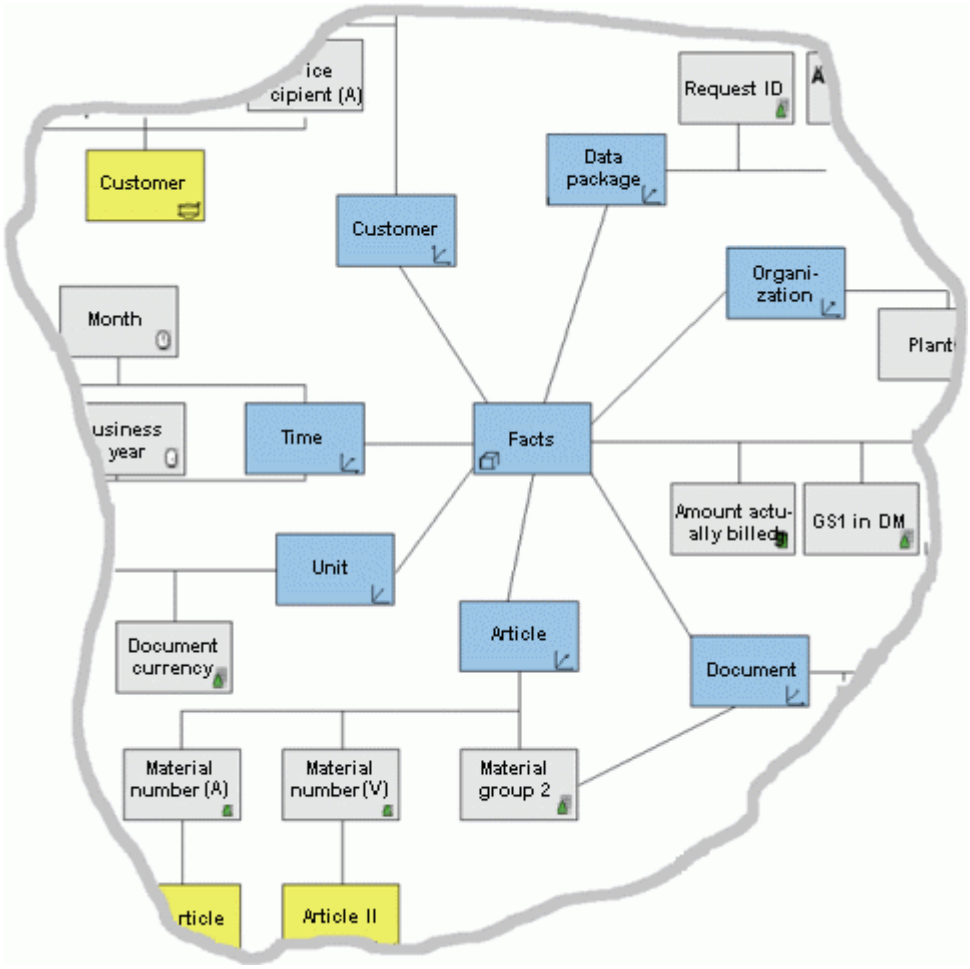


eERM: Terms and forms of representation

3.2.1.10. Modeling the Data Warehouse structure

The Data Warehouse structure diagram describes the structure of a Data Warehouse. Primarily, the diagram is a static description, that is, it illustrates the interrelation of data as well as their locations. In the ARIS architecture this type of description is realized in the data view. The focus lies on the interrelation and arrangement of information. The data dimensions are described by the info cube. The interplay of the dimensions is represented by the star schema (see the following figure). A dimension can serve as a key for connecting other dimensions. The objects of individual dimensions can have specific values, which are

cataloged in fact tables and are exactly defined by KPIs. Dependencies are described in dimension tables listing their key attributes and characteristics. The hierarchical interrelations of the features are described by tree structures. Finally, dimensions can be allocated to master data tables using the structure diagram.



Data Warehouse in the star schema

3.2.1.11. Project management data - Information carrier diagram

The information carrier diagram is an optional component for project management with ARIS. It belongs to the requirements definition of the data view and is used to record incoming and outgoing data, such as documents, logs, or ARIS models.

The required documents (for example, word processing files) can be displayed explicitly and accessed from within ARIS via the **Link 1** to **Link 4** attributes.

Information carrier diagram

3.2.2. Implementation - Table diagram

The table diagram is used to describe the tables and fields of a database system. The following figure shows a graphical representation of tables and fields.

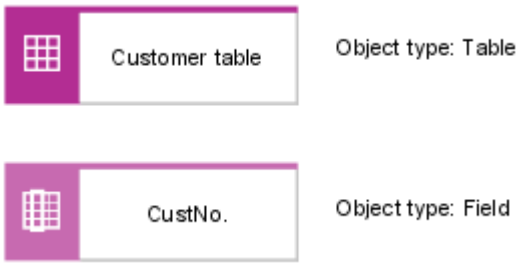


Table diagram (1)

The individual fields assigned to this table can be shown for each table. For further specification, a sorting index and the domain can be assigned to each field. The following figure illustrates an example.

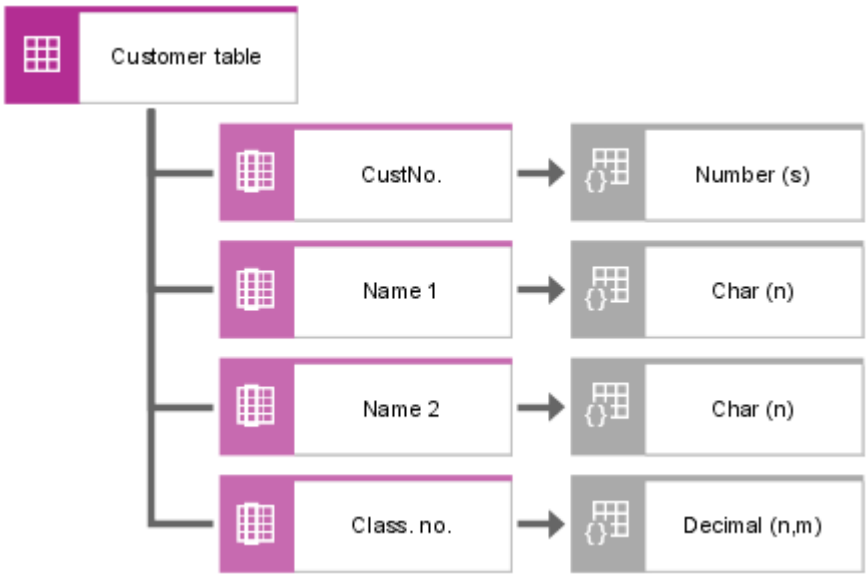


Table diagram (2)

Multilateral relationships between tables and relations or entity types may occur. These relationships can be illustrated in the table diagram by selecting the relevant connections.

Due to the fact that converting or documenting database tables and fields used in a company does not necessarily require the definition of a relational schema, both the realization relationships between relations (or attributes) and tables (or fields) and between entity types (or ERM attributes) and tables (or fields) can be represented.

The representation may focus either on the relations and attributes realized by the tables and fields, or - leaving out the relational definitions – on the entity types, relationship types, and ERM attributes illustrated by the tables and fields. Both types of representation are illustrated in the following figure.

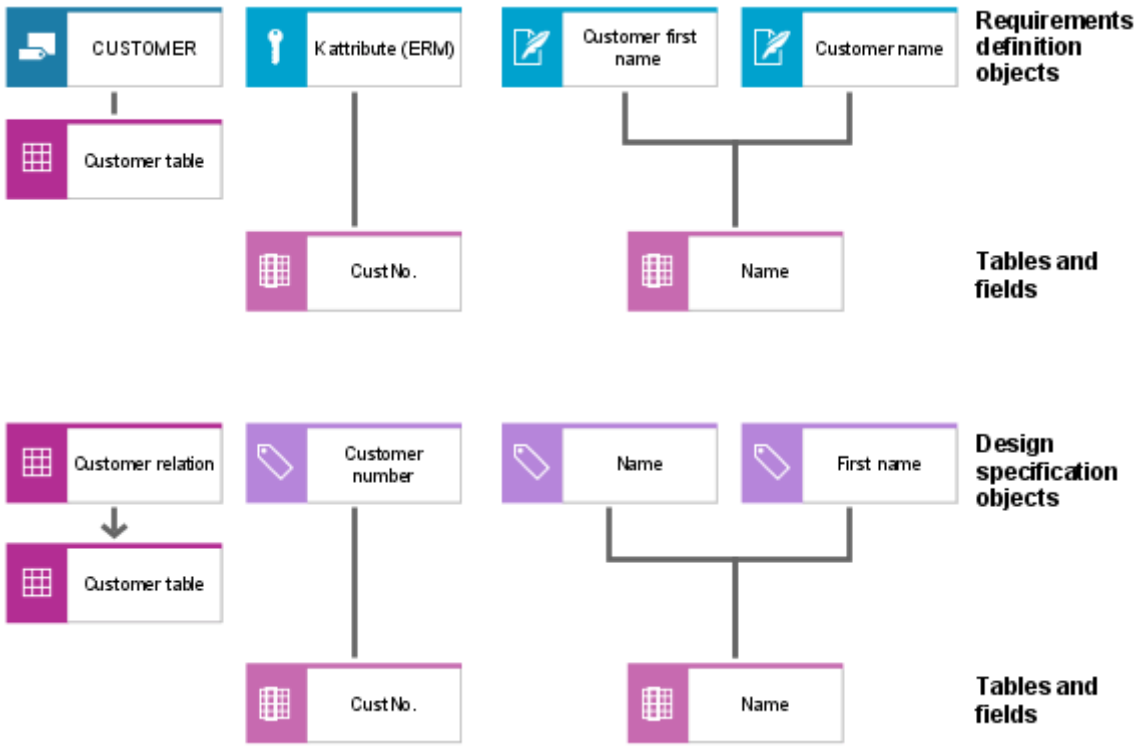


Table diagram (3)

To be able to define the exact location of specific tables and fields in a company, it must be possible to define every single specimen of a table. The same applies when the privileges for accessing tables and fields are to be specified for organizational units. The **Table** object type introduced earlier determines the logical structure of a physical table and its fields at the **Type level**. However, multiple specimens of every table thus defined may exist on different media or at different locations in a company. This fact can be represented using the **Table** (specimen) and **Field** (specimen) object types.

With the help of these objects, the specimen count of a table or a field can be determined exactly. The following figure shows this aspect.

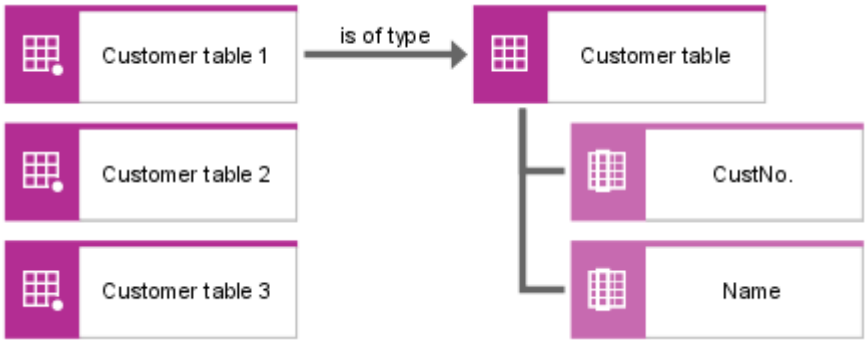


Table diagram (4)

A list of objects and relationships that are available in a table diagram is provided in the **ARIS Method Reference** help.

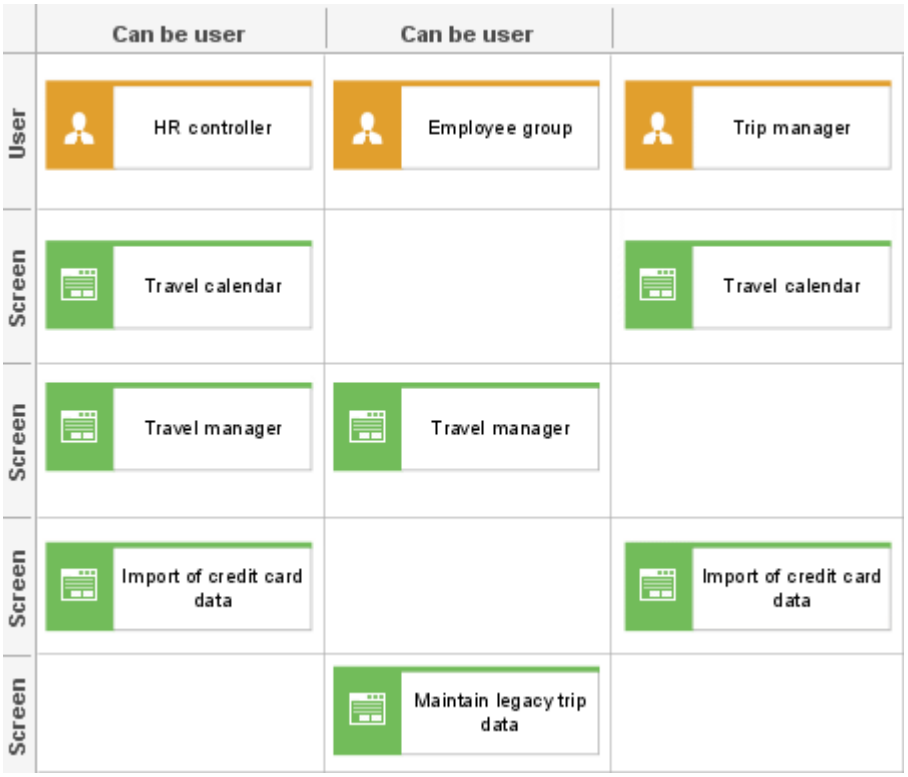
3.2.3. Role assignment diagram (RAD)

The SAP® reference model is represented using EPCs. These EPCs illustrate the business processes at different levels of detail. In the EPCs with the highest level of detail, which the SAP® terminology refers to as

processes, procedures of processing executables in the SAP® system are modeled. These processes can be assigned both roles and executables.

In ARIS, this is done in the function allocation diagram where the EPC containing the modeled process must be assigned to the corresponding function definition. Thus, the function allocation diagram shows which roles are necessary to run the executables. However, since there are no direct relationships between roles and executables, no decision can be made as to which executable a role is responsible for if there are multiple roles. For this reason, the assignment of roles to executables takes place in the role assignment diagram (RAD). One role is displayed per column. Executables are placed in columns, which creates implicit relationships.

The information can be used during R/3 implementation to create the necessary user profiles and authorization concepts for operating the SAP® system.



Role assignment diagram

3.3. Organization view

3.3.1. Requirements definition

3.3.1.1. Organizational structure of companies

Companies are complex social structures that are divided into manageable units. To deal with the given complexity, patterns are defined and rules established. The result of this process is called organization. Until recently, the role of organizational considerations as an aspect of developing information systems has rarely been the object of research. But newer business concepts, such as Lean Production, Lean Management, or

CIM are closely allied with the organizational setup of the area of consideration. For this reason, the ARIS concept provides an independent descriptive view on organization.

In a company's organizational design, a distinction can be made between the organizational structure and the process organization.

The organizational structure encompasses the rules by which the company is statically structured. The process organization contains the rules relating to the tasks to be performed by the company. This task-related structure in the sense of distributing functions to task performers is dealt with in the control view of the ARIS house. The organization view basically looks at a company's organizational structure.

The design of an ideal company organization with the aim of reducing coordination efforts to a minimum depends on the company's business environment and objectives. Therefore, it is not possible to define universally valid ideal organizational structures that may serve as reference structures.

The structure of organizational units depends on various criteria.

A very common criterion is the functional structure. One company function (procurement, production, finance and accounting, sales) is given responsibility for all products and sectors. The advantage of this approach lies in a high level of staff specialization, but it also entails a multiplication of efforts concerning communication and coordination between the functional areas.

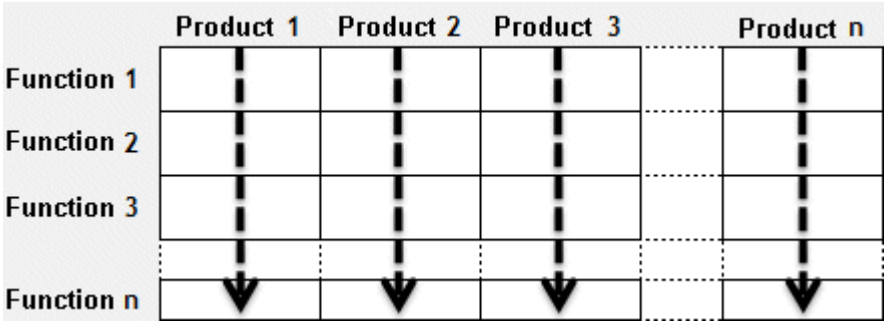
Both the design and use of information systems had their focus on this functional breakdown of companies for a long time. However, looking at integrated process chains in the sense of cohesive processing of similar data objects makes it difficult to establish interrelationships between individual functions for such a structural design.

For this reason, the discussion of integrated data processing resulted in the demand for a consistent base of data aimed to support the various functions. However, the intended integration of functions counteracts the desired effect of reducing complexity via the functional structure.

Hence, when dealing with the aim of achieving functional integration, other criteria of organizational breakdown are frequently applied.

For example, the breakdown may focus on criteria such as sectors or products. The following figure shows a breakdown by product (see Scheer, Business Process Engineering, 1994, p. 26 et sqq.).

In a sector-based organizational structure the organizational units are specified in accordance with the local distribution of the company or corporate division. This kind of structure is particularly suitable for sales functions because regional factors such as varying legislation can be dealt with more appropriately.



Organizational breakdown by product

A product-based organizational structure defines organizational units for products or product groups. Within a product group, a maximum number of functions that are relevant for this particular product group are integrated. The aim of this procedure is to reduce the communication effort that functional structures entail. However, this results in the necessity of information exchange between the product group-based subsystems.

To counteract these effects, hybrid organizational forms are often created. The following figure illustrates this with an example of Purchasing (see Scheer, Business Process Engineering, 1994, p. 26 et sqq.).

	Product group 1	Product group 2	Product group 3
Central purchasing		Supplier selection	
		Contract agreement	
Scheduling			
Order			
Accounting control			

Hybrid organizational forms

Using a purely functional structure would imply that a central purchasing department was responsible for all product groups. In this case, synergy effects between the product groups may be exploited, but major coordination problems would arise from carrying out a purchasing procedure across all subfunctions. When the purchasing functions are split up according to the various product groups, individual purchasing departments must be established for every product group to carry out all purchasing functions. When selecting suppliers or negotiating framework contracts, for example, synergy effects may only be obtained at the expense of high coordination efforts.

In the breakdown illustrated in the above figure, those purchasing functions for which high synergy effects are expected are broken down functionally, that is, they are carried out by a central purchasing department. Functions that must comply with specific requirements and restrictions stipulated by individual product groups are divided by product group in an object-oriented approach. These functions can thus be integrated in the process flow of the individual product groups immediately. This means that the operational handling of processes takes place in decentralized units, while the relationships among the decentralized units are dealt with at the superior, central coordination level.

These flexible organizational forms are given special emphasis in the ARIS concept due to their strongly process-oriented approach. The formation of organizational structures to which various breakdown criteria are applied at the same time is a claim put forward especially by accounting-based approaches, such as the profit center concept.

3.3.1.2. Organizational chart

A typical way of representing organizational structures is the organizational chart. In this chart, organizational units (as task performers) and their interrelationships are represented according to the structuring criteria selected.

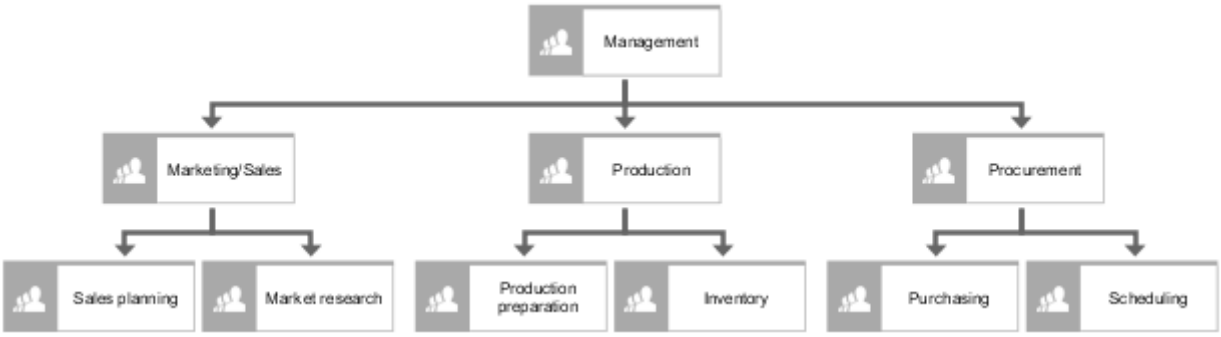
Organizational units are the performers of the tasks that must be carried out in order to achieve the business objectives.

Organizational units are linked via relationships. The following figure shows an example.

For a more precise specification of the hierarchical relationships, a distinction is made between various connection types that may exist between organizational units. In this context, a connection may have one of the following meanings:

- is technical superior to
- is disciplinary superior to
- is a component of

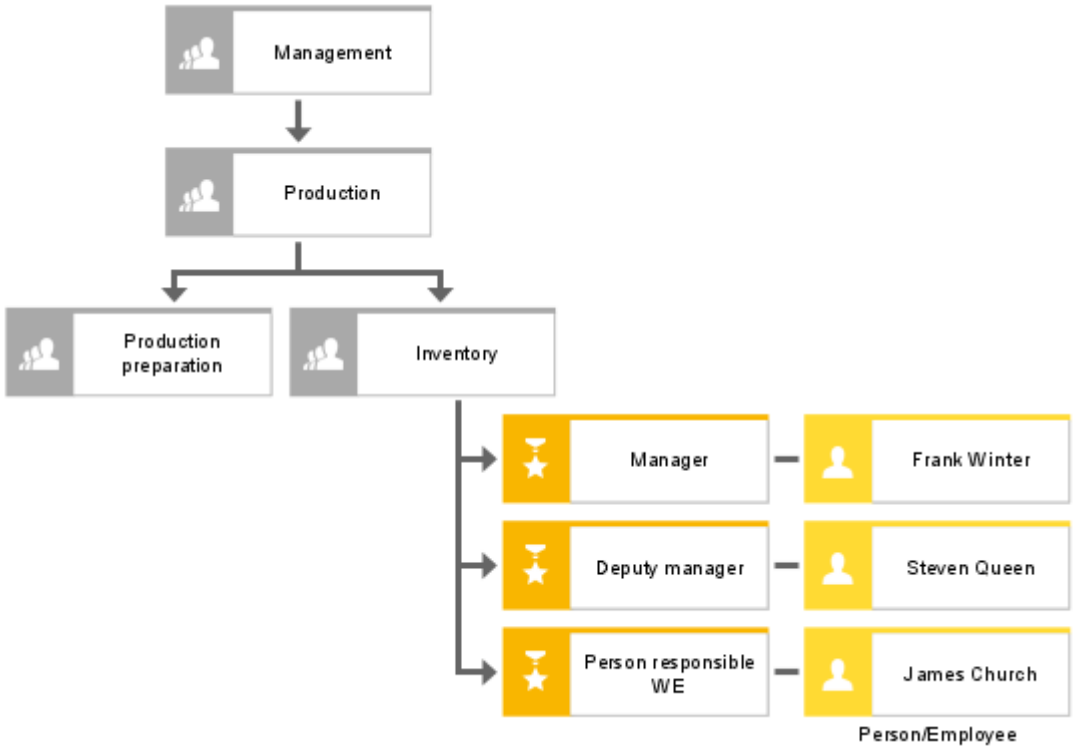
If functional responsibilities are recorded, the organizational chart illustrates the distribution of the business tasks.



Organizational chart (1)

The **Position** object type is provided to represent individual positions within the company, for example, positions for which descriptions exist. This object type is illustrated in the following figure. Multiple positions can be assigned to an organizational unit. The meaning of the connections corresponds to the interaction between organizational units.

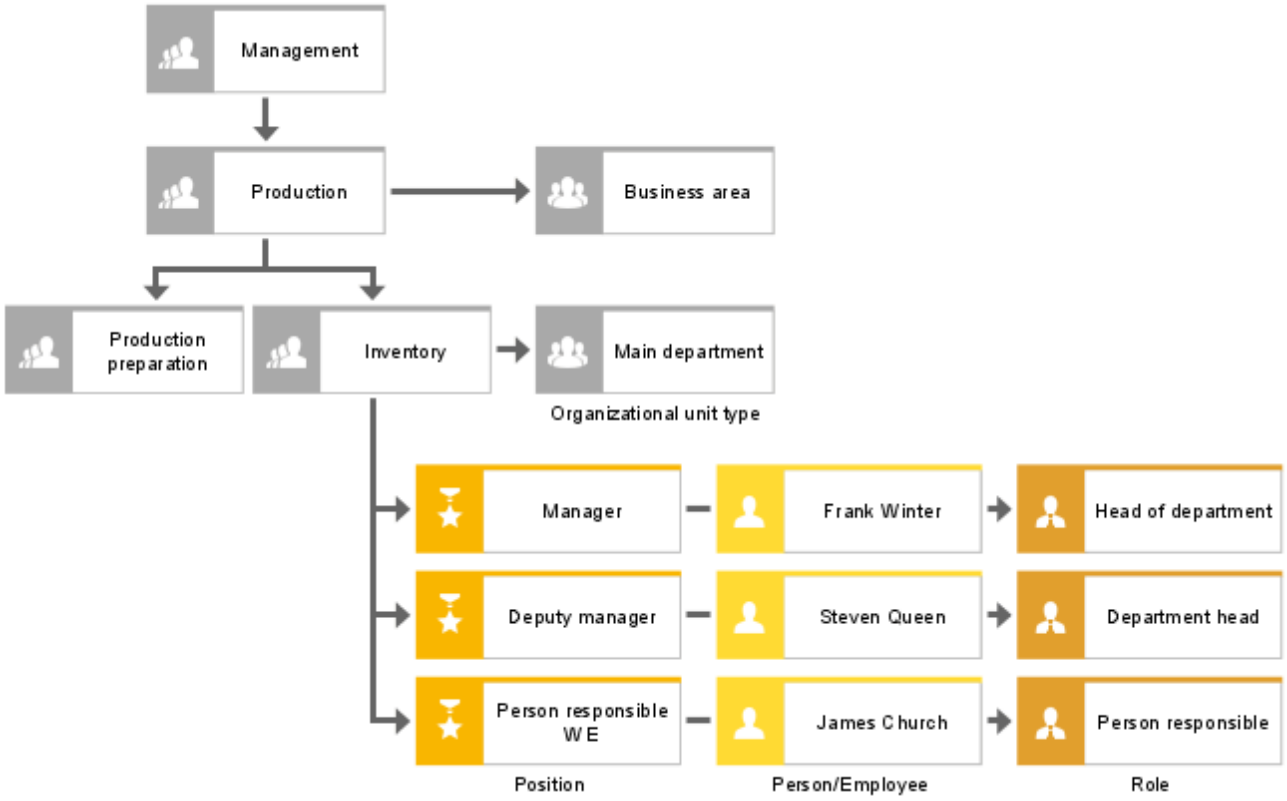
Individual persons in the company can be assigned to the positions and organizational units. ARIS offers separate objects for persons, which is illustrated in the following figure. The assignment of an individual person to an organizational unit shows that this person is assigned as an employee to this organizational unit, whereas the assignment to an individual position defines the current staffing in the company. An example is shown in the following figure.



Organizational chart (2)

Organizational units and persons can be typified. For example, you can define for each organizational unit whether it is a department, a main department, or a group. Persons can be assigned to the **Department head**, **Group leader**, or **Project manager** roles, for example.

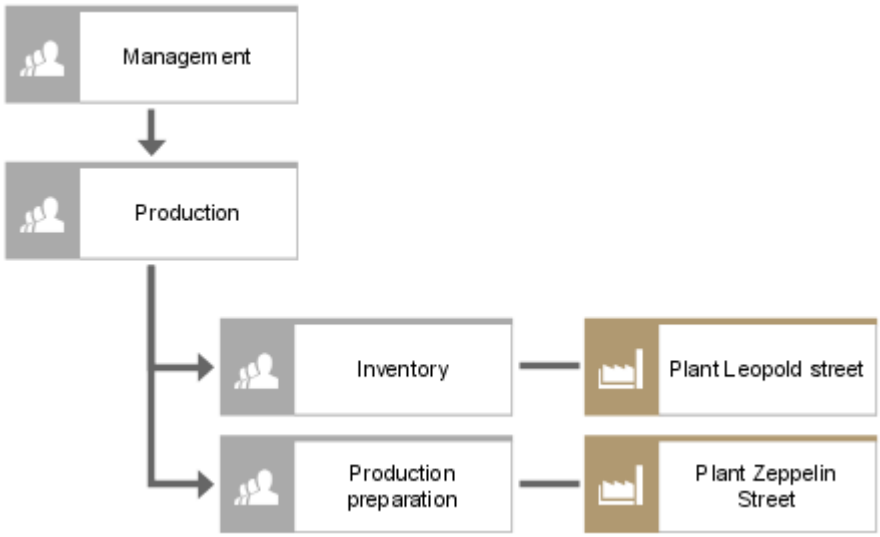
This typification is represented by the **Organizational unit type** and **Role** objects provided for this purpose. An example of the typification of organizational units and persons is shown in the following figure.



Organizational chart (3)

Using these object types enables you to depict general business rules derived from concrete organizational units or employees. Thus, in process chains, it is possible to define that only specific roles may carry out a function or have access to an information object.

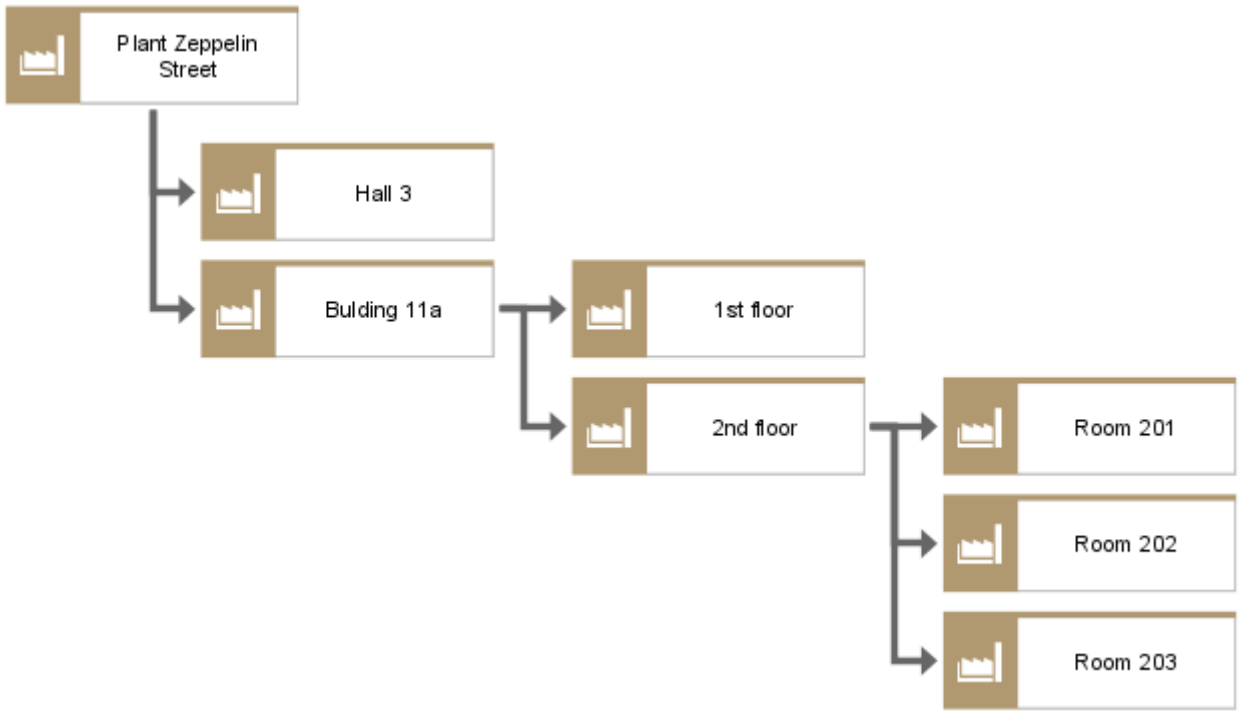
The modeling of the organizational structure of the company is the starting point for the network topologies to be defined at the design specification level, which are to support this organizational structure as best as possible. Included in the definition of the network topology are network connections and network nodes, which may be found at particular locations of the company. The location of an organizational unit is therefore the most important link between requirements definition and design specification of the organization view. Thus, the location of every organizational unit is already defined in the requirements definition. An example is shown in the following figure.



Organizational chart (4)

Locations may be arranged in any required hierarchy. A location can be an entire plant, a building or, for a more detailed examination, an office through to an individual workstation in a room. This makes it possible, in the design specification, to assign network nodes of a network to individual workstations of the organizational unit. For example, the design specification may stipulate that a total of 3 network nodes must be available in a particular office (room 202).

The following figure shows an example of a location hierarchy.



Organizational chart (5)

3.3.2. Design specification - Network topology

The company's organizational structure as represented in the organizational chart can now be supported using communication and information system infrastructures. The structural requirements for these information systems can generally be defined in the design specification in the form of network topologies.

In a first step, various network types can be incorporated in a **Network topology** model.

A network type typifies individual network specimens that are based on precisely the same technology.

The following figure shows an example of a network type:

Network type

Network types can be interlinked and, since they are logical constructs, they can also be arranged in a hierarchy.

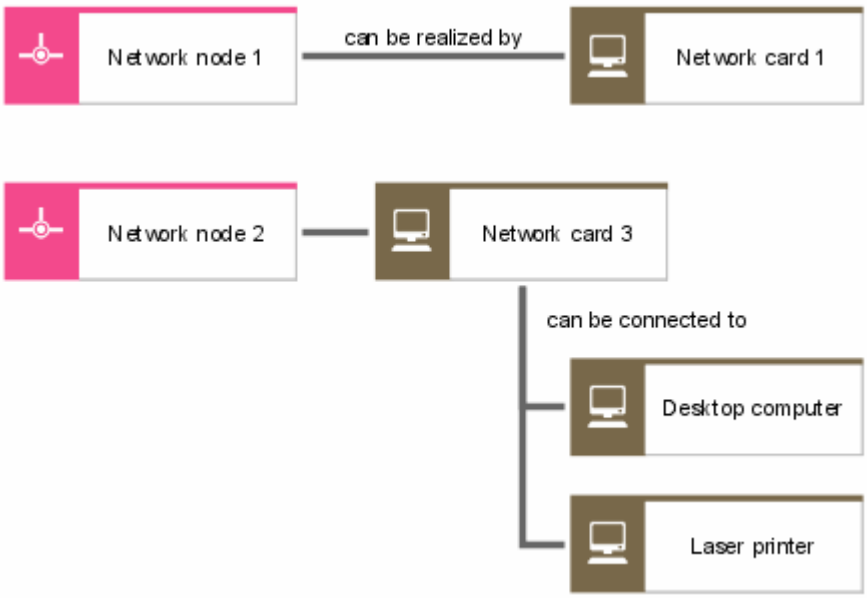
Network node types and network connection types can be assigned to every network type. Thus, technological restrictions resulting from the choice of one particular network type for a company can be identified immediately. It is possible to show for every network connection type which network node types it may end in.

When speaking of hardware component types, the term may refer to either network hardware for implementing the defined network structures, or hardware component types that can be connected to network node types.

As with application system or network types, hardware component types do not represent individual specimens of hardware components that can be identified, for example, by inventory numbers assigned by the company. Instead, they typify all hardware components that are based on the same technology. Hardware component types may be arranged in any required hierarchy.

A hardware component type typifies individual specimens of hardware components that are based on precisely the same technology.

Together with network node and connection types, a kind of reference model of the network topology can now be created. This model indicates which hardware component types can be used for realizing specific network connection types or network node types. An example of a connection type might be a particular type of transmission cable. Besides, it is possible to show which hardware component type can be connected to which network node type. Network node types may also have relationships to hardware component types that are used to create node types. An example is shown in the following figure.



Network topology

The link between network topology and the objects of the requirements definition is established via two approaches.

On the one hand, for every single hardware component type the organizational unit or position responsible for it can be specified.

On the other hand, it is possible to define for each network type, network node type, network connection type, and hardware component type at which location within the company they may be found. Thus, the location is the central link between the requirements definition and the design specification of the organization view.

A list of object and relationship types that are available in a network topology model is provided in the **ARIS Method Reference** help.

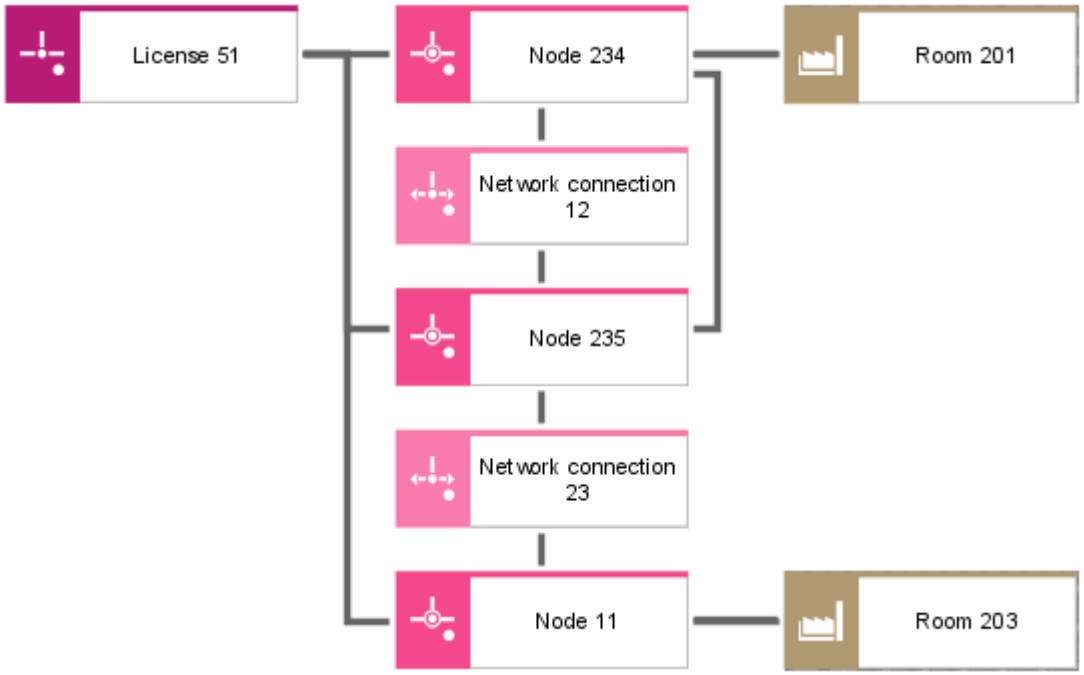
3.3.3. Implementation

3.3.3.1. Network diagram

The network diagram illustrates the realization of the network topology defined in the design specification.

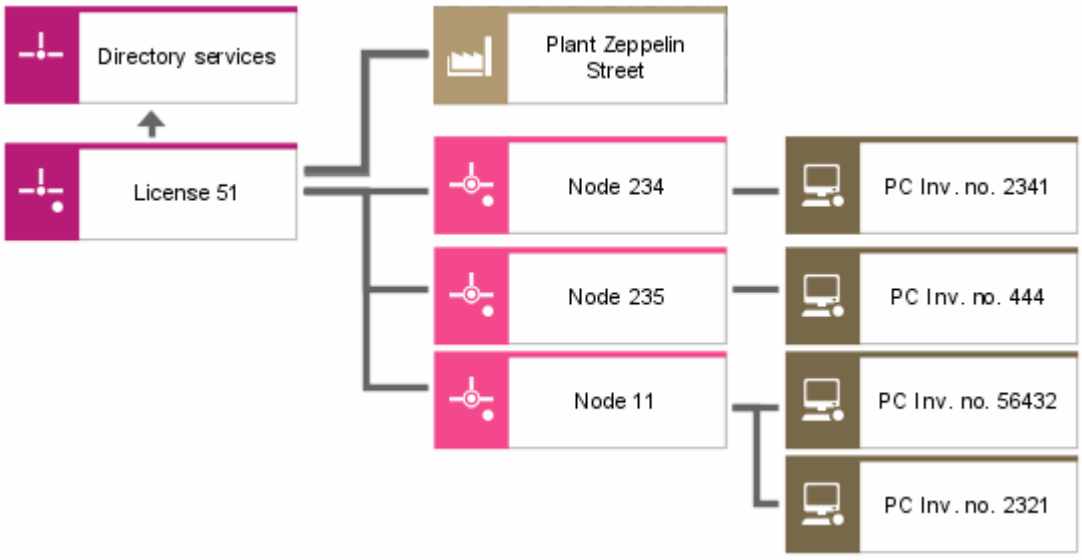
The networks that exist in the company are recorded by means of the **Network** object. It is possible to specify for each network the network nodes and network connections it consists of.

The exact location of every network, network node, and network connection within the company can be indicated. A location can be an entire plant, a specific building, a complex of buildings, an office, or an individual workstation.



Network diagram (1)

The network diagram also records the hardware components used to realize each network connection and network node. Besides, it is possible to illustrate the structure of every single hardware component. On the one hand, hardware components are used to form network connections and network nodes; on the other hand, they can be connected to network nodes. This relationship can be represented in the network diagram, as well. For every object of the specimen level, the relationship to the corresponding object of the design specification level can be modeled. This way it is possible to express, for example, that the network in the **Port St.** plant is of the **FDDI ANSI X3.139** type.



Network diagram (2)

Thus, the network diagram establishes the links to the design specification via type allocations, as well as the links to the requirements definition via the allocation of network components to specific locations.

A list of object and relationship types that are available in a network diagram is provided in the **ARIS Method Reference** help.

3.3.3.2. Material flow modeling - Technical resources

To illustrate the material flow in process models (EPC (material flow), material types are allocated to individual functions of the business process in the form of function input or output. Similar to the allocation of information objects to functions (representation of information transformation by functions), this allocation represents the transformation of input material types to output material types. Additionally, the technical resources required for transforming materials can be recorded in the process chains. In this context, a distinction is made between operating resources, warehouse equipment, transport systems, and technical operating supplies.

In the **Technical resources** model type you can arrange technical resources in a hierarchy, assign a type to them, and classify them. The following object types are available for this purpose:

Operating resource

Operating resources are specimens of various operating resource types that are available for a company to perform its tasks. Operating resources are often identified by inventory numbers (for example, number of a production plant).

Operating resource type

An operating resource type typifies individual operating resources that are based on precisely the same technology.

Operating resource class

Similar operating resource types can be combined to form an operating resource class. The similarity can be based on different classification criteria. Thus, an operating resource type can be assigned to multiple operating resource classes.

Warehouse equipment

Warehouse equipment items are specimens of various warehouse equipment types that are available for a company to perform its tasks. Warehouse equipment items are often identified by inventory numbers.

Warehouse equipment type

A warehouse equipment type typifies individual warehouse equipment items that are based on precisely the same technology.

Warehouse equipment class

Similar warehouse equipment types can be combined to form a warehouse equipment class. The similarity can be based on different classification criteria. Thus, a warehouse equipment type can be assigned to multiple warehouse equipment classes.

Technical operating supply

A technical operating supply is an individual specimen of a technical operating supply type. In general, it can be identified by means of an inventory number.

Technical operating supply type

A technical operating supply type typifies individual technical operating supply items that are based on precisely the same technology.

Technical operating supply class

Similar technical operating supply types can be combined to form a technical operating supply class. The similarity can be based on different classification criteria. Thus, a technical operating supply type can be assigned to multiple technical operating supply classes.

Transport system

A transport system is an individual specimen of a transport system type. In general, it can be identified by means of an inventory number or a plant number.

Transport system type

A transport system type typifies individual transport systems that are based on precisely the same technology.

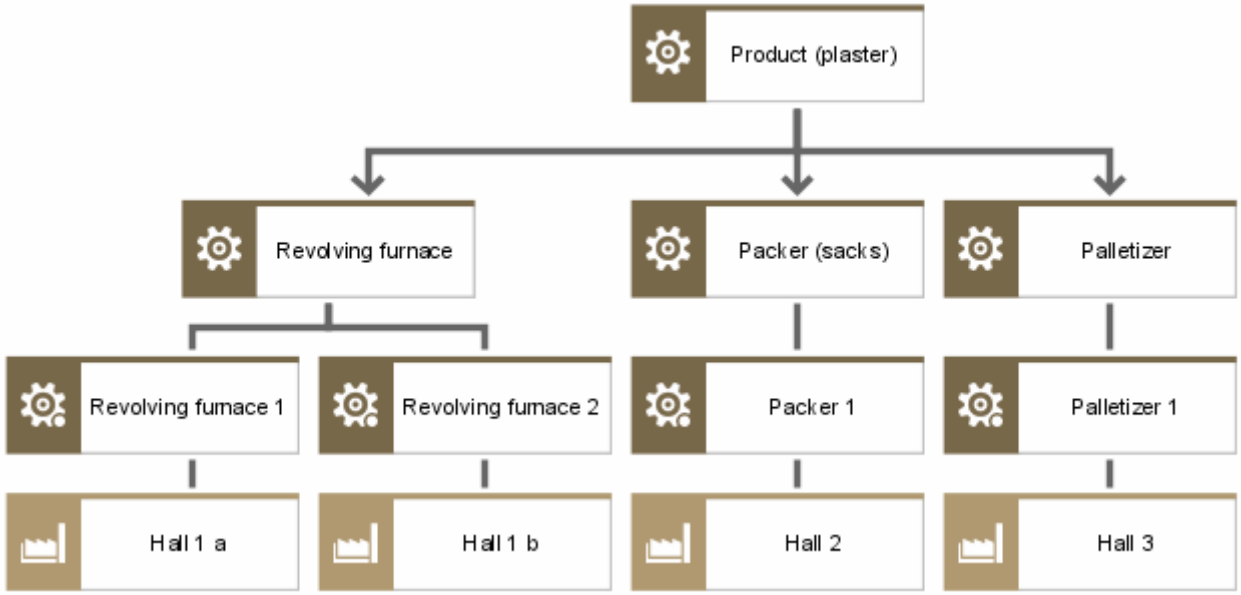
Transport system class

Similar transport system types can be combined to form a transport system class. The similarity can be based on different classification criteria. Thus, a transport system type can be assigned to multiple transport system classes.

Due to the various possibilities of creating hierarchies in the **Technical resources** model type it is possible to describe the structure of complex technical installations. For example, the components of a complex production plant and their interrelationships can be illustrated this way.

In addition to the modeling options mentioned above, there is also the possibility of defining location allocations and organizational responsibilities for technical resources. To do this, the **Location**, **Organizational unit**, **Group**, **Position**, and **Person** object types which you already know from the **Organizational chart** model type are available. These object types can be linked to the **Operating resource**, **Warehouse equipment**, **Technical operating supply**, and **Transport system** object types.

An example of a **Technical resources** model type is shown in the following figure.



Technical resources

3.4. Process view

3.4.1. Requirements definition

The relationships between the objects of the data, organization, and function views are analyzed in the control/process view. The relationships to be analyzed result from the connections between the views.

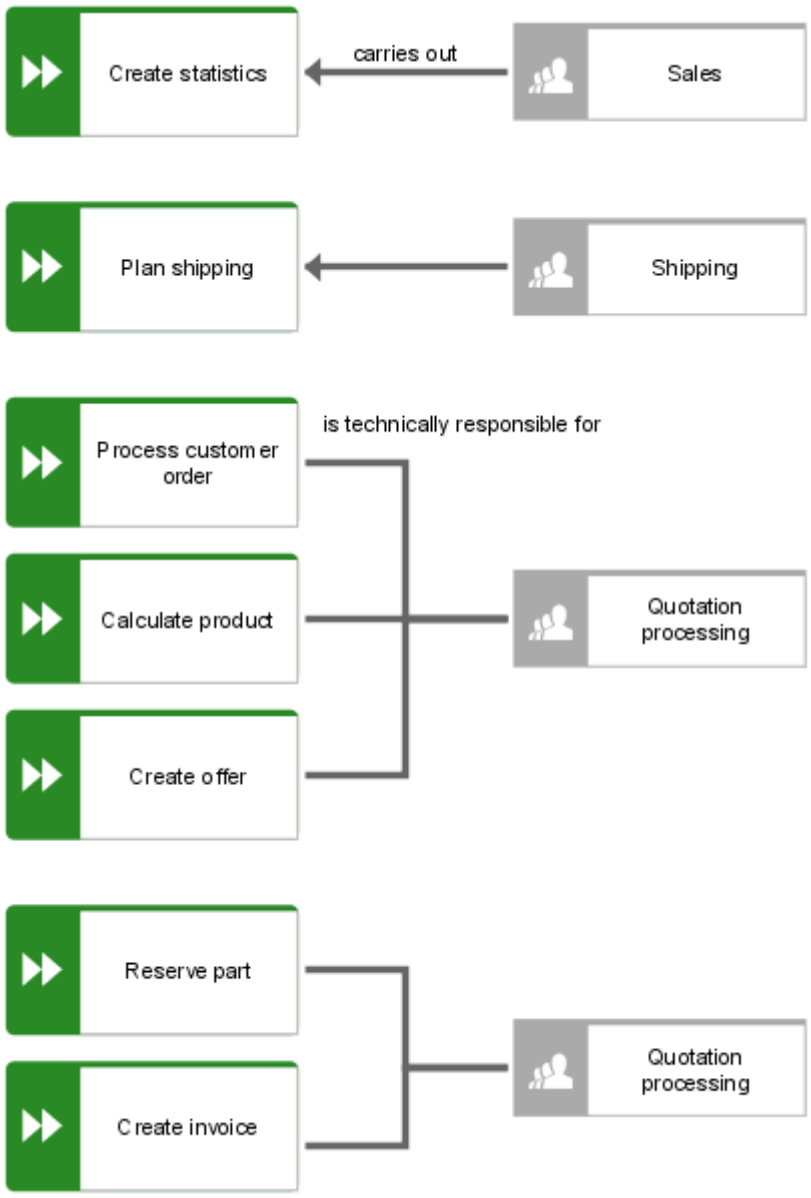
First, the relationships between two views are examined, then diagrams are introduced, illustrating the relationships between all three views.

3.4.1.1. Linking functions with organization - EPC

Linking the function view with the organization view serves to allocate the functions defined in the function tree to the task performers (organizational units) of the organizational chart. This allocation defines an organizational unit's responsibility and decision-making power with regard to its allocated functions. Looking at how these organizational allocations are realized in a process chain (business processes) enables you to identify the degree of functional integration, that is, the number of business process functions that are to be processed by an organizational unit.

The following figure shows an example of the allocation of organizational units to functions. In this figure, the function placed on the left is assigned the organizational unit responsible for its execution.

The functions' superior or subordinate positions in the hierarchy are illustrated in the function view (function tree), and the relationships between the organizational units are shown in the organization view (organizational chart). Therefore, there is no need to define them at this point.



Organizational units

3.4.1.2. Event control - Event-driven process chain (EPC)

The operational sequence of functions in the sense of business processes is represented in process chains. The start and end events of every function can be specified. Not only do events trigger functions, they also represent results of functions.

An event describes the fact that an information object has taken on a business management-relevant state that controls or influences the progression of the business process. Events trigger functions and are the results of functions. Unlike a function, which is a time-consuming task, an event relates to one point in time.

The change in state of an information object may refer to the first occurrence of this information object (for example, **Customer order received**), or to a change in state in the sense of a change in status recorded in an

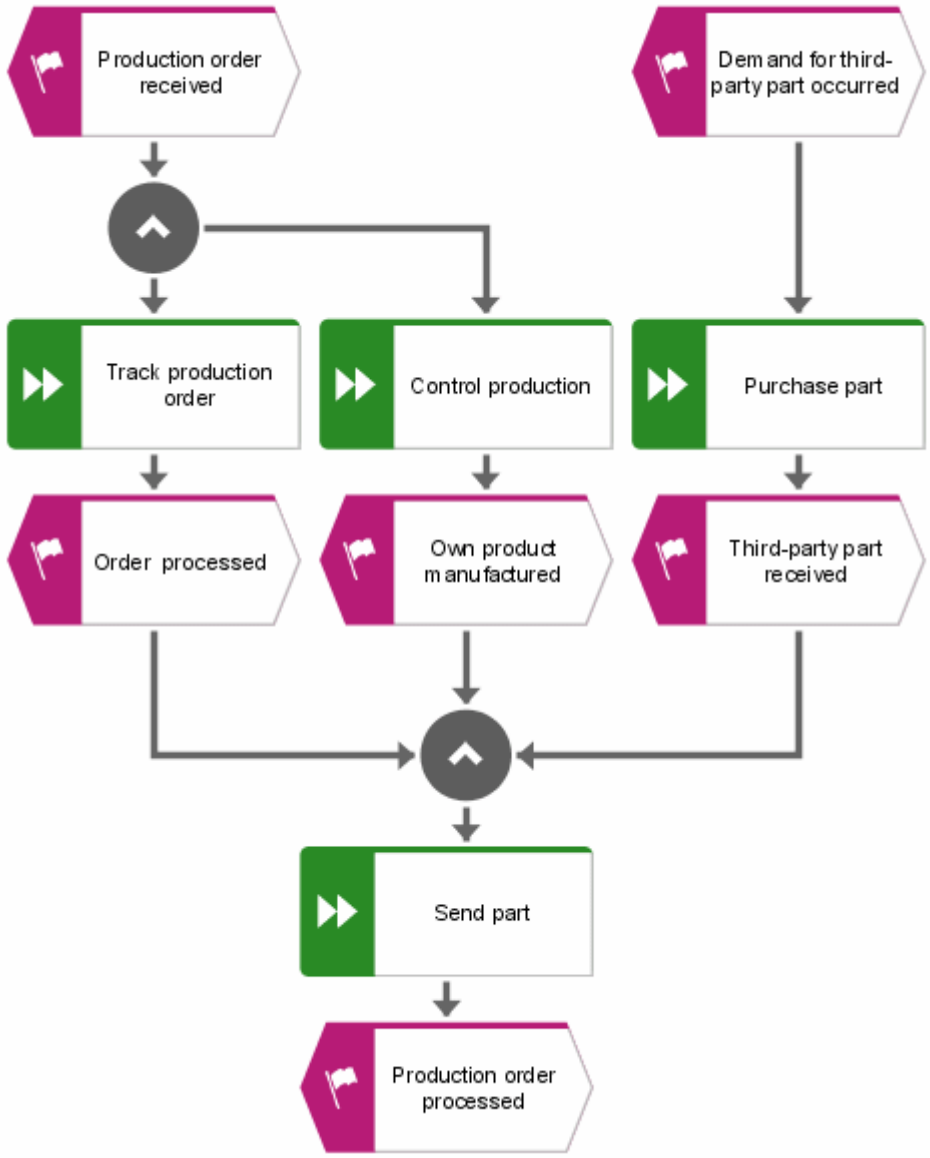
attribute occurrence (for example, **Offer is refused**). Since information objects and attributes are described in the ARIS data view, the event-driven representation of process chains forms a link between the data view and the function view and is thus attributed to the ARIS control view.

Events are graphically represented as hexagons. The name should not only contain the information object itself (**Order**), but also its state change (**received**). The following figure illustrates events.

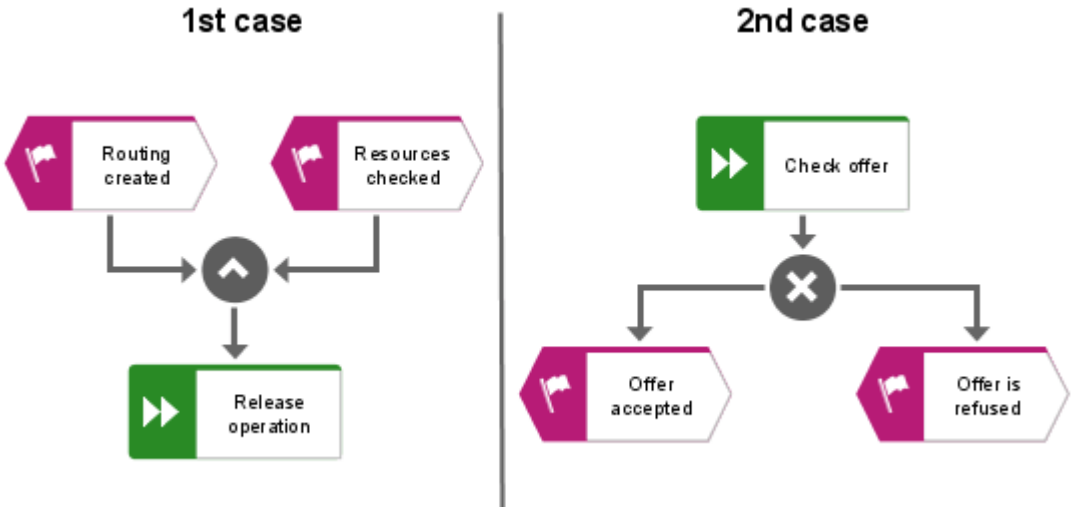
Events

Events trigger functions and are the results of functions. By arranging events and functions in a sequence, so-called Event-driven process chains (EPCs) are created. An event-driven process chain (EPC) shows the chronological-logical operational sequence of a business process.

An example of an EPC is shown in the following figure. Since events determine which state or condition will trigger a function and which state will define the end of a function, the start and end nodes of such an EPC are always events. Multiple functions can originate from one event simultaneously, and a function can have multiple events as its result. A rule that is represented by a circle is used to illustrate branches and processing loops in an EPC. However, these connections do not only serve as graphic operators, but define the logical links between the objects they connect.


Production order received
Demand for third-party part occurred
Track production order
Control production
Purchase part
Order processed
Own product manufactured
Third-party part received
Send part
Production order processed

EPC (1)


1st case
2nd case
Routing created
Resources checked
Release operation
Check offer
Offer accepted
Offer is refused

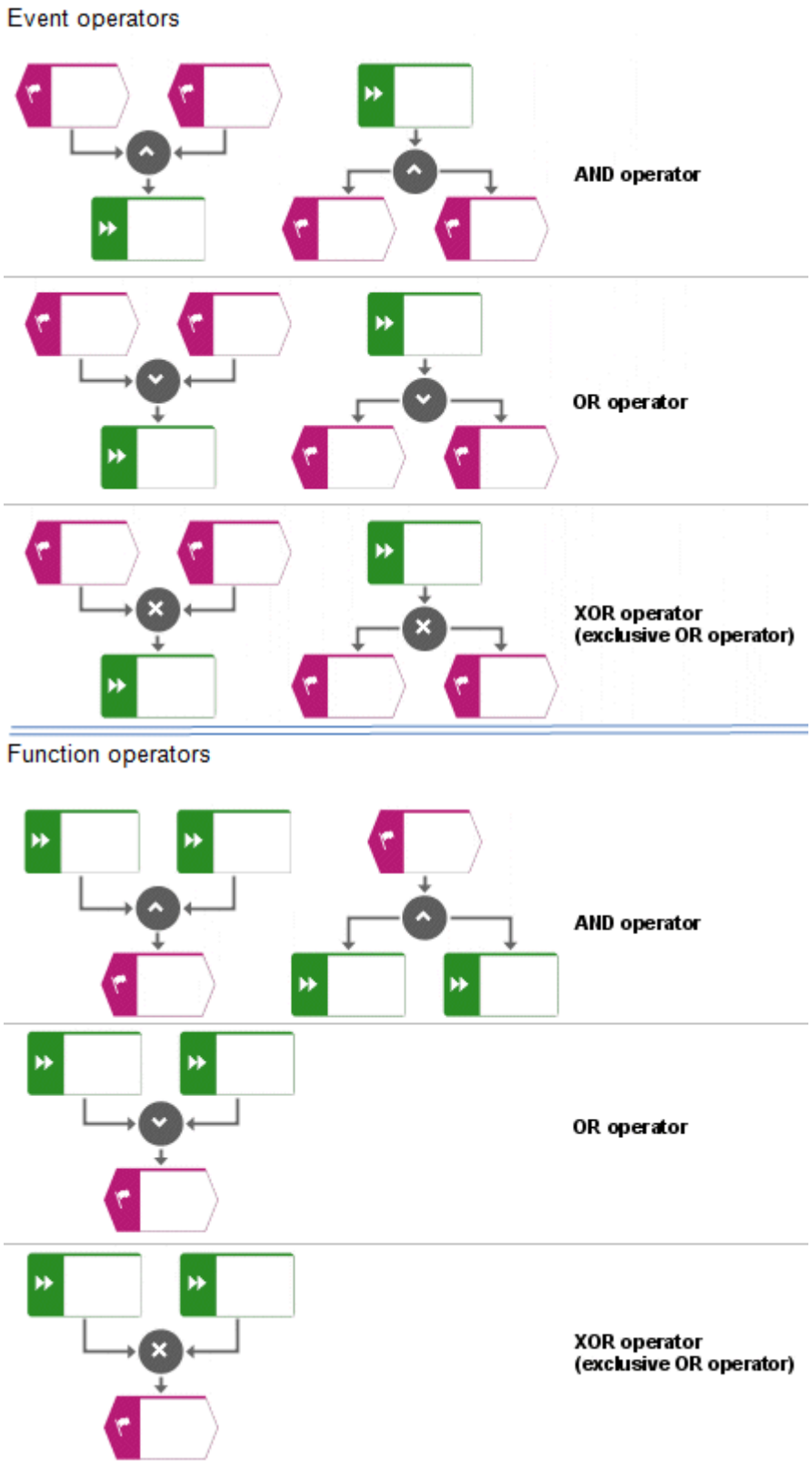
Rules (1)

In the first example of this figure the start events are connected via an AND rule. This means that the **Release operation** procedure is started only if the routing and the required resources are available. Therefore, both events must have occurred before the procedure can begin. The second example shows an exclusive OR connection using an XOR rule. The **Check supplier offer** function may result in either acceptance or rejection of the quote. Both results, however, cannot occur at the same time. Besides these two cases and links of the 'inclusive OR' type, more complex relationships are possible.

Thus, two different types of operators exist:

1. event operators and
2. function operators.

An overview of all possible event and function operators is listed in the following figure (see Hoffmann, Kirsch, Scheer, 'Modellierung mit Ereignisgesteuerten Prozessketten' [Modeling with event-driven process chains], 1993, p. 13).



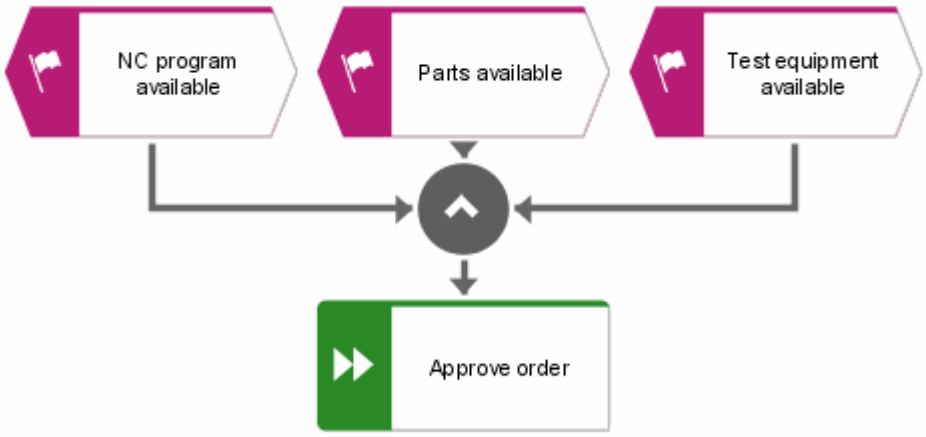
Rules (2)

In this context, special attention must be paid to the restrictions which exist for function operators. Due to the fact that events cannot make decisions (only functions can do this), a triggering event must not be linked using an OR or XOR operator.

Below, possible operators are explained using examples.

Linking triggering events

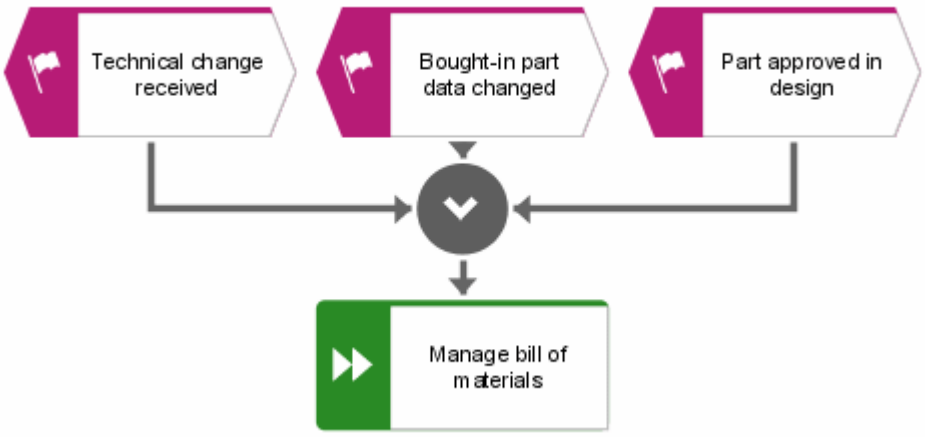
AND rule



Rules (3)

The function can be started only after all events have occurred.

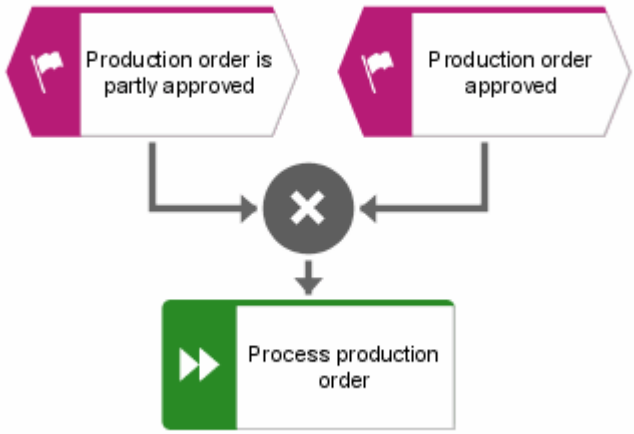
OR rule



Rules (4)

The function is carried out after at least one of the events has occurred.

Exclusive OR rule (XOR rule)

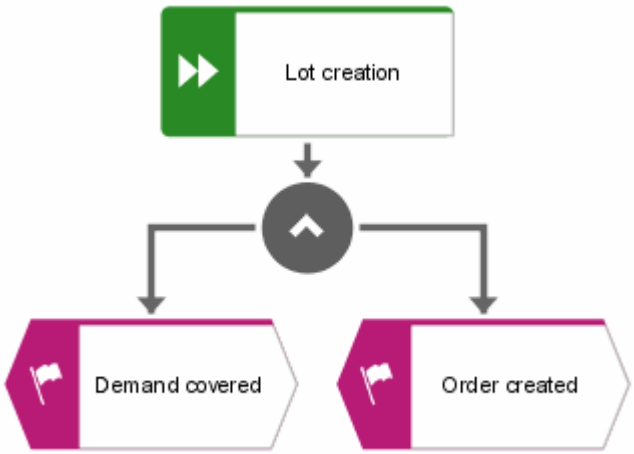


Rules (5)

The function is started after no more than exactly one event has occurred.

Linking created events

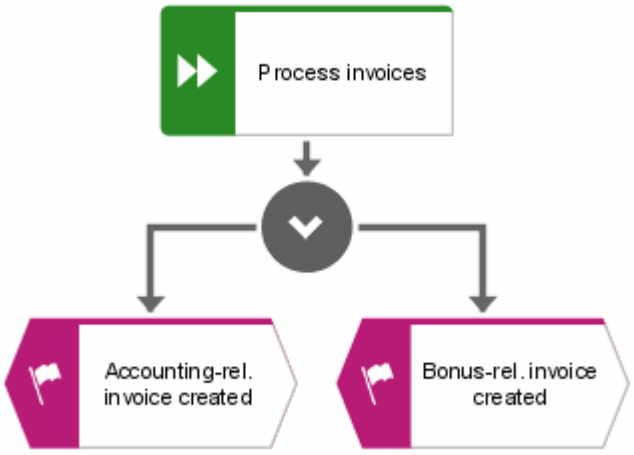
AND rule



Rules (6)

All events will occur after function execution is complete.

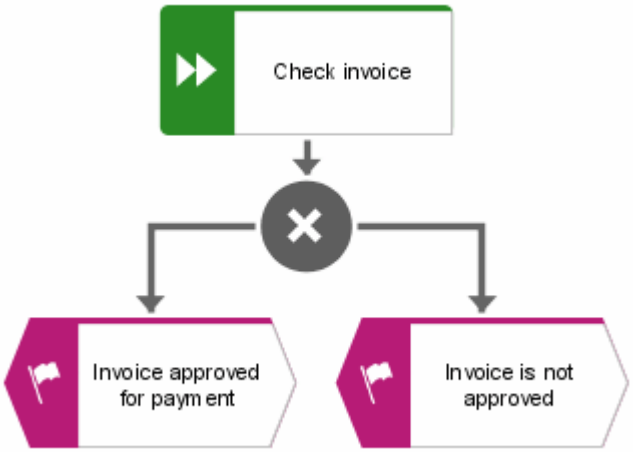
OR rule



Rules (7)

At least one of the events will occur after function execution is complete.

Exclusive OR rule (XOR rule)

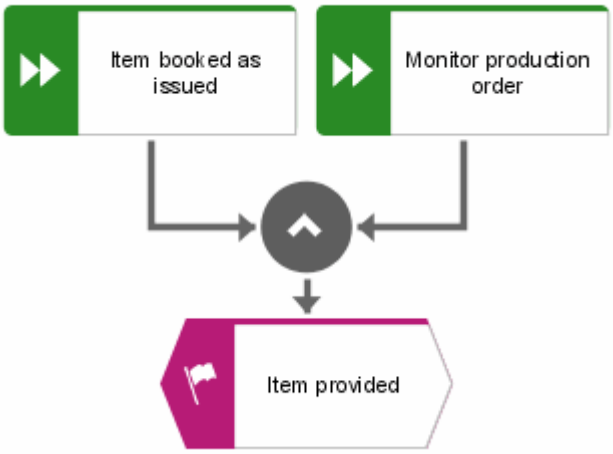


Rules (8)

No more than one event will occur after function execution is complete.

Linking functions with created events

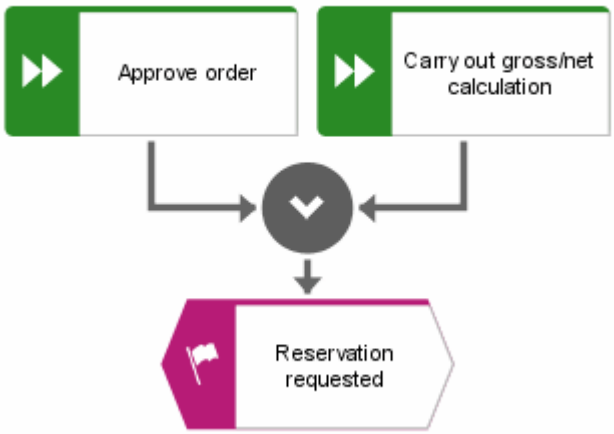
AND rule



Rules (9)

The event occurs only after all functions have been carried out.

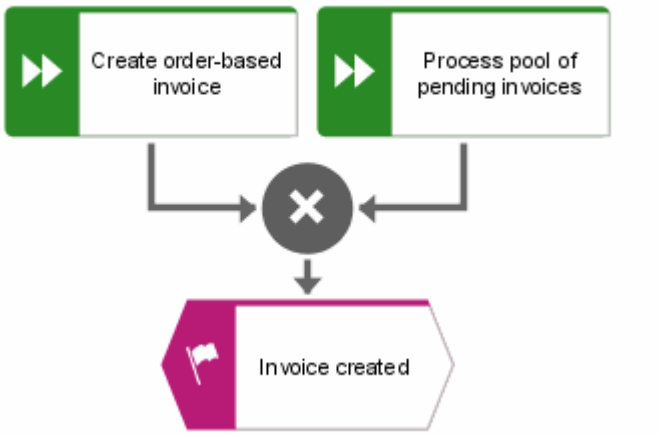
OR rule



Rules (10)

The event occurs after at least one of the functions has been carried out.

Exclusive OR rule (XOR rule)

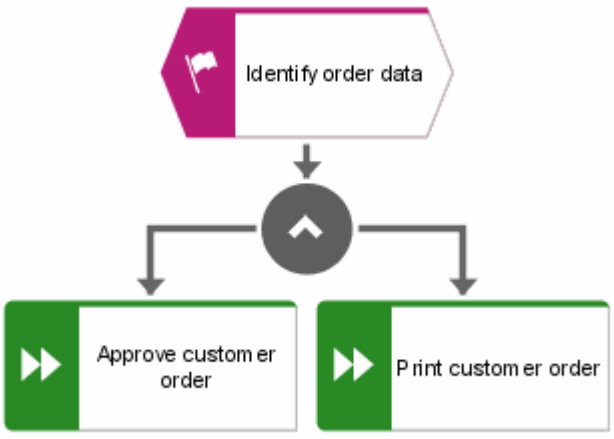


Rules (11)

The event occurs after no more than exactly one function has been carried out.

Linking functions with triggering events

AND rule



Rules (12)

The event triggers all functions.

OR rule

Events have no decision-making power. This operator is invalid.

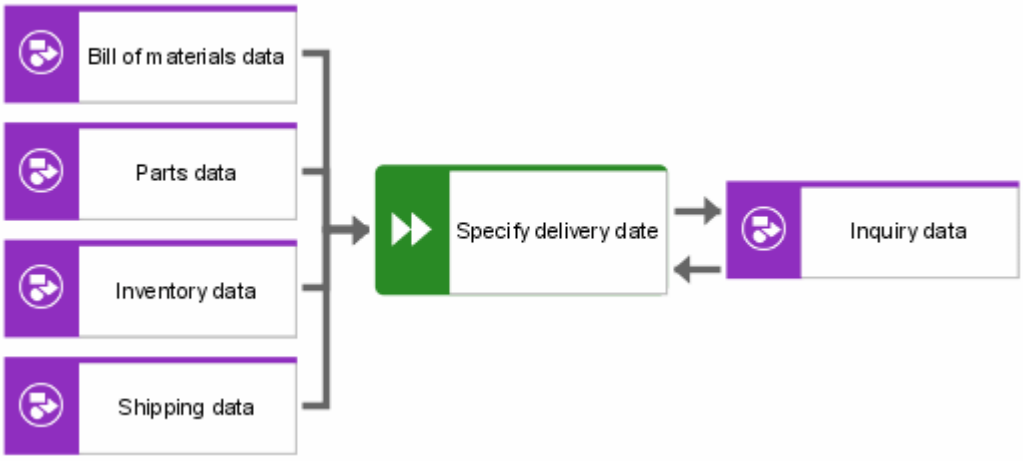
Exclusive OR rule (XOR rule)

Events have no decision-making power. This operator is invalid.

Besides being illustrated in the form of event-driven process chains, these branches can also be represented in table form in the event and function columns of a process chain diagram. Since functions are sorted sequentially in a process chain diagram, branches and processing loops can be represented only in a rather complex and thus confusing manner.

3.4.1.3. Function allocation diagram (I/O)

In addition to the event control representation explained in chapter Event-driven process chain (EPC) [57] , the transformation of input data into output data and the representation of data flows between functions also form a link between the data view and the function view in the ARIS concept. The transformation of input data into output data can be illustrated in so-called Function allocation diagrams (I/O). The figure below illustrates an example of a function allocation diagram (I/O). The input data of the **Determine delivery date** function are **Parts data**, **Inventory data**, **Bill of materials data**, and **Shipping data**. **Inquiry data** serves as both input data and output data. Thus, a function allocation diagram (I/O) consists of functions of the function view and information objects of the data view. The arrows determine whether an information object is used only as input data, output data, or as input/output data. More detailed specifications are possible, indicating, for example, that the function has created or deleted an information object. Depending on the degree of detail, information objects can be Cluster/Data models, Entity types or Relationship types, or attributes of the data view.

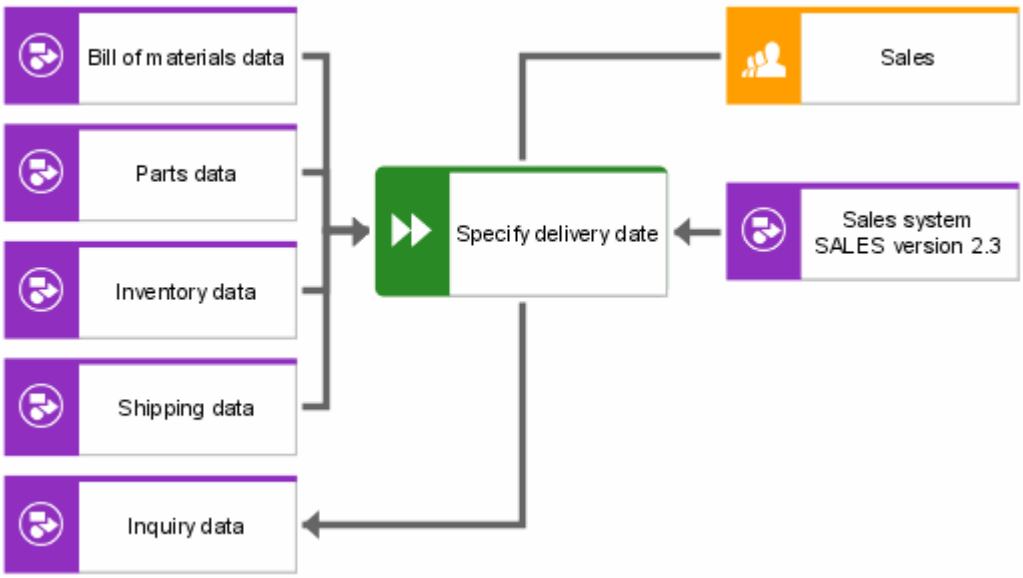


Function allocation diagram I/O (1)

The example shown above illustrates the actual aim of function allocation diagrams (I/O), which is to represent a function's input/output data.

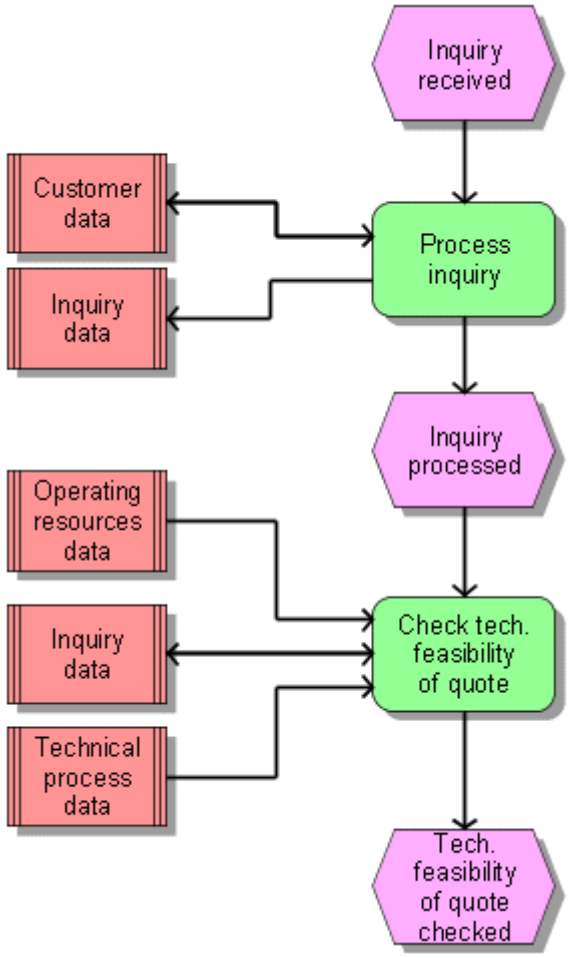
Besides a function's input/output data, events and all other objects that can be allocated to the functions in an EPC are available. Thus, the user is able to restrict the modeling of process chains in EPC diagrams to events and functions, and to assign each function a function allocation diagram (I/O) containing

all additional relationships the function has. This allows for much clearer representations of business processes and also explains the use of a new name for this model type. The figure below illustrates an example of this more detailed representation in a function allocation diagram.



Function allocation diagram I/O (2)

Besides this method of representing data transformation in the form of function allocation diagrams (I/O), it is also possible to include this information in an EPC. The figure below illustrates an example. In this case, the links between functions and information objects play the same role as in function allocation diagrams (I/O). However, including them in a process chain with numerous branches may result in a very complex representation.



EPC I/O (1)

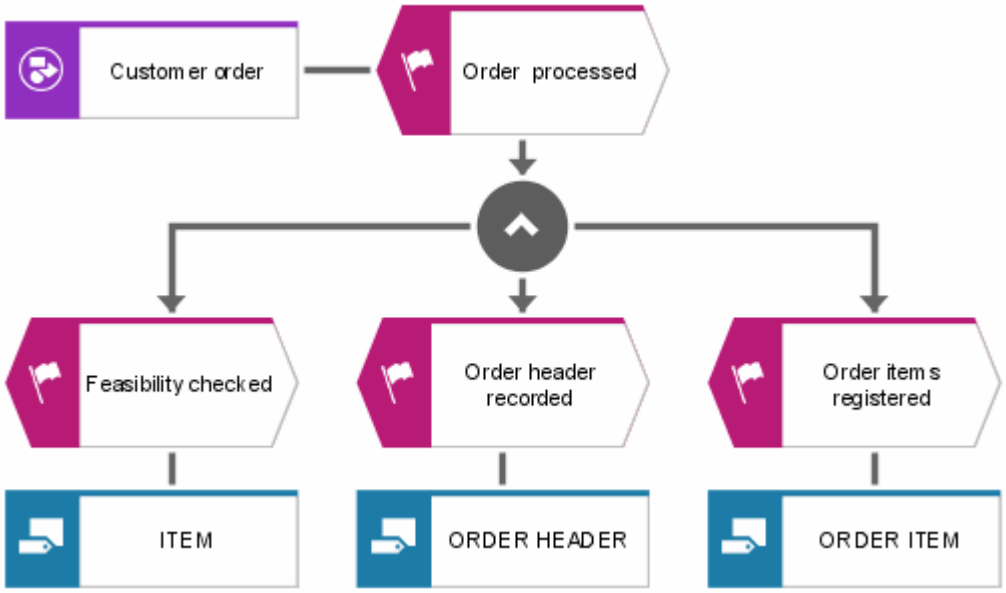
3.4.1.4. Event diagram

Events define the fact that the state of information objects has changed. Thus, every event references particular information objects of the data model and defines the status of this information object at a given point in time.

First of all, events are roughly specified in a top-down procedure (example: **Order processed**). The next level of detail in process modeling involves the specification of detailed events that, in certain combinations, cause the event to occur at the rough level. For example, the occurrence of the events **Feasibility checked**, **Order header registered**, and **Order items registered** can, in sum, define the **Order processed** status.

You can use the event diagram to display these event correlations at the rough and detail modeling levels. For this purpose, an event at the rough level can be assigned an event diagram displaying the events and the corresponding operators at the detail level (which results in the formation of a hierarchy). Moreover, you can include information objects of the data model in this model type and link them to the events. Thus you specify the event which defines the state change of a given information object.

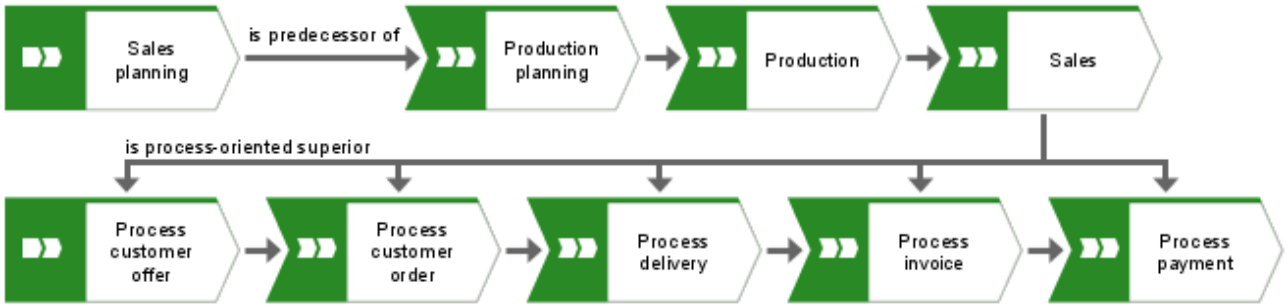
An example is shown in the following figure.



Event diagram

3.4.1.5. Value-added chain diagram

The value-added chain diagram is mainly used to identify the functions within a company that are directly involved in the creation of a company's value-added. These functions can be interlinked as a sequence of functions and thus form a value-added chain. The following figure shows an example of a value-added chain diagram.



Value-added chain

In a value-added chain diagram, functions can be arranged in a hierarchy, similar to a function tree. It always represents process-oriented superiority/subordination.

A value-added chain diagram not only enables you to express a superiority or subordination of functions, it can also display the functions' links to organizational units and information objects. When allocating organizational units to functions, as with process chains we differentiate between a function's technical responsibility, its IT responsibility, and the actual execution of a function.

A list of other relationships that are available in a value-added chain diagram is provided in the **ARIS Method Reference** help.

3.4.1.6. End-to-end business scenario diagram

The **End-to-End business scenario diagram** can be used to describe the business scenario map or the end-to-end processes of an organization. The business scenario map contains only business scenario objects and the **Level (E2E business scenario)** attribute is set to **Level 1**.

The description of an end-to-end process contains value-added chain objects that represent processes at the lowest level of the hierarchy in the process landscape. Those objects are connected using the **is predecessor of** connection to describe the chronological order of the processes. The **Level (E2E business scenario)** attribute is set to **Level 2** and the model is assigned to the relevant business scenario object.

3.4.1.7. Object-oriented modeling

For modeling in a UML environment, UML is available. All method-relevant information on UML diagrams and UML elements is accessible directly via the ARIS UML Designer interface.

3.4.1.8. Process selection matrix

The process selection matrix displays different process scenarios by assigning main processes to individual scenarios.

The user can determine which functions of the scenario processes are to occur in the company. For this purpose, all main functions (scenario functions) of an application system or of an industry reference model need to be included as processes.

The following symbol types are available for modeling a process selection matrix:

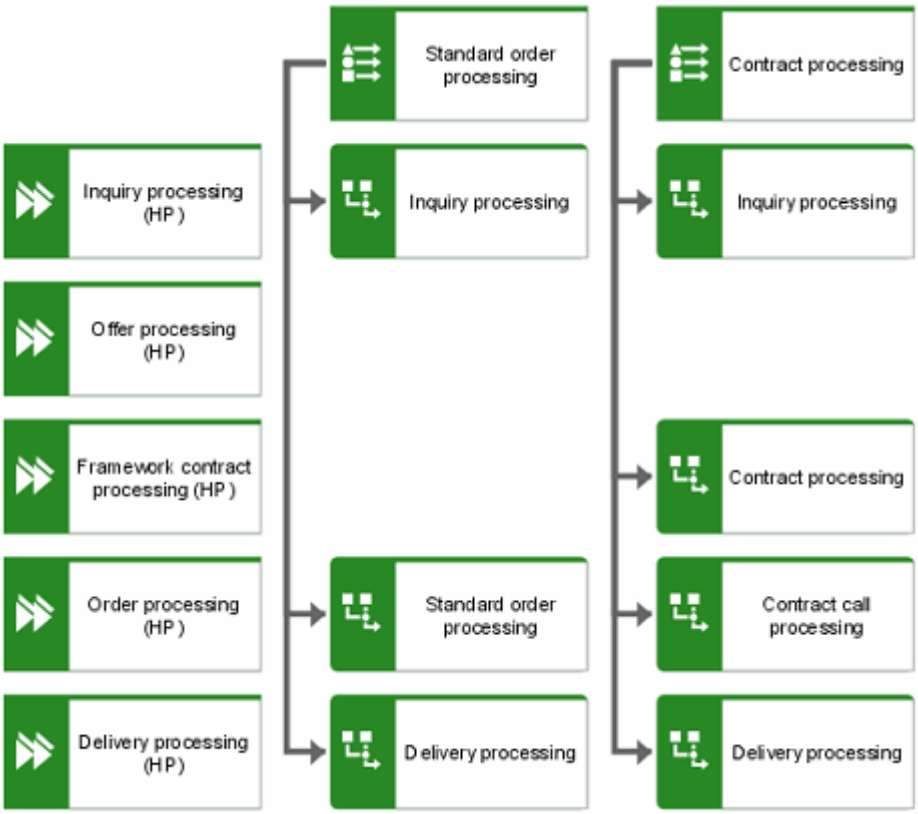
- Scenario
- Process
- Main process

A scenario represents a scenario process in the selection matrix which arranges different main processes in groups.

The process represents functions of the scenario process that are described in more detail in the reference model by process models.

The main process represents the main functions in function trees to which the processes (functions from the scenario processes) are assigned.

The following figure shows an example of a process selection matrix.



Process selection matrix

3.4.1.9. Material flow modeling

You can use EPCs to illustrate not only the information flow, but also the material transformation. To represent a material flow within business processes, ARIS provides the **EPC (material flow)** model type which is an extension of the **EPC** model type.

3.4.1.10. EPC (material flow)

In addition to the object types of an EPC, the following object types are available in the 'EPC (material flow)':

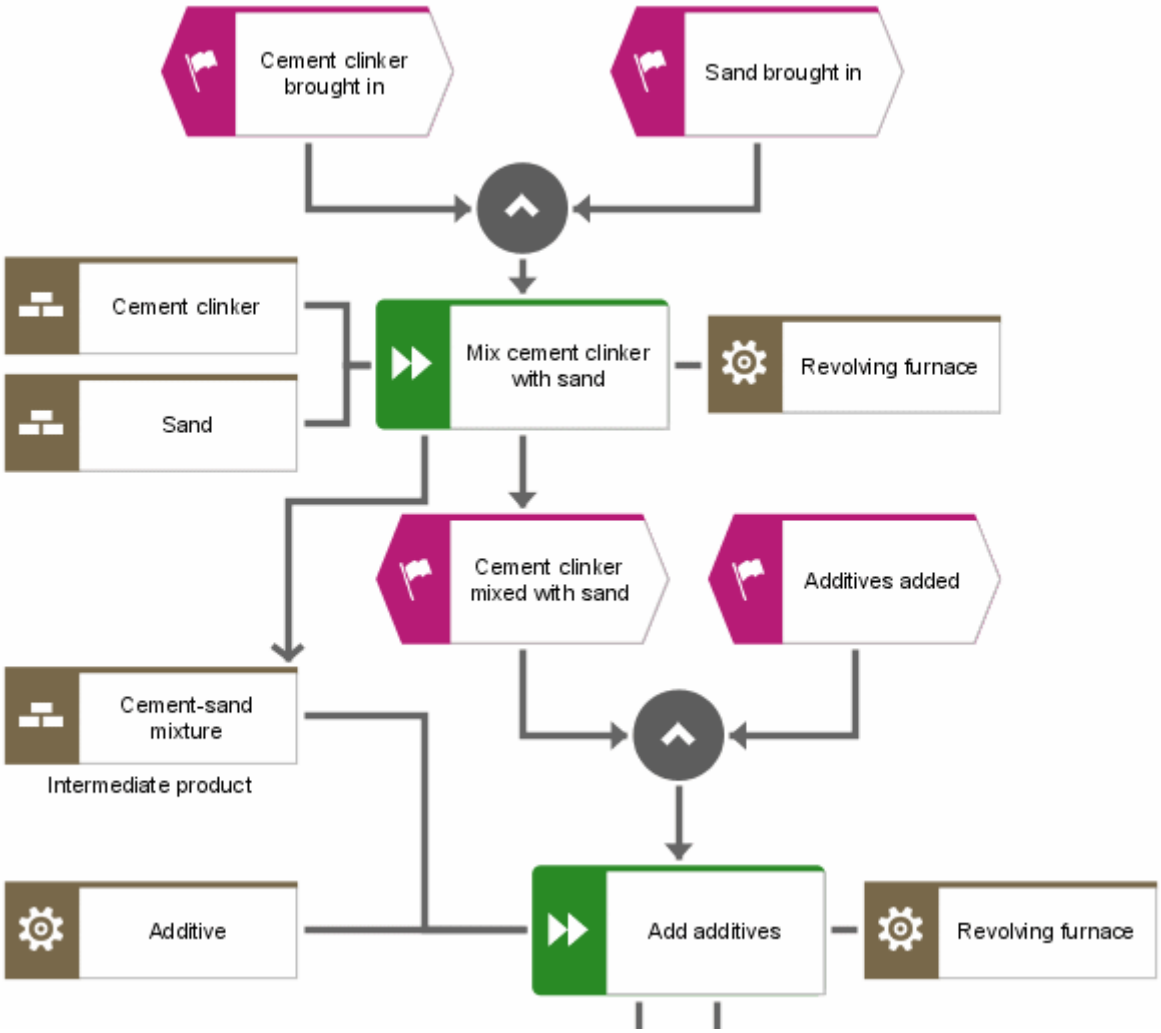
- Material type
- Packaging material type
- Operating resource type
- Operating resource
- Technical operating supply type
- Technical operating supply
- Warehouse equipment type
- Warehouse equipment
- Transport system type
- Transport system

The **Material type** object type can be linked to the **Function** object type by means of an incoming or outgoing connection. In the case of an incoming connection, the materials that a function requires as input are defined. In this context, by selecting the corresponding connection type, you can define whether the function uses none, part, or all of the material. An outgoing connection specifies the material types created by the function.

Technical resources are required for material transformation. In process chains, you can also link them to the **Function** object type. To specify alternative resources that may be available, the **requires alternatively** connection type is provided in addition to the **requires** connection type.

If materials are to be packaged during function execution, packaging material types are needed. In order to specify the corresponding packaging material types, you can model a relationship between the function and the required packaging material types.

The following figure shows an EPC (material flow) and the corresponding technical resource types and packaging material types.



EPC (material flow)

3.4.1.11. EPC (column/row display)

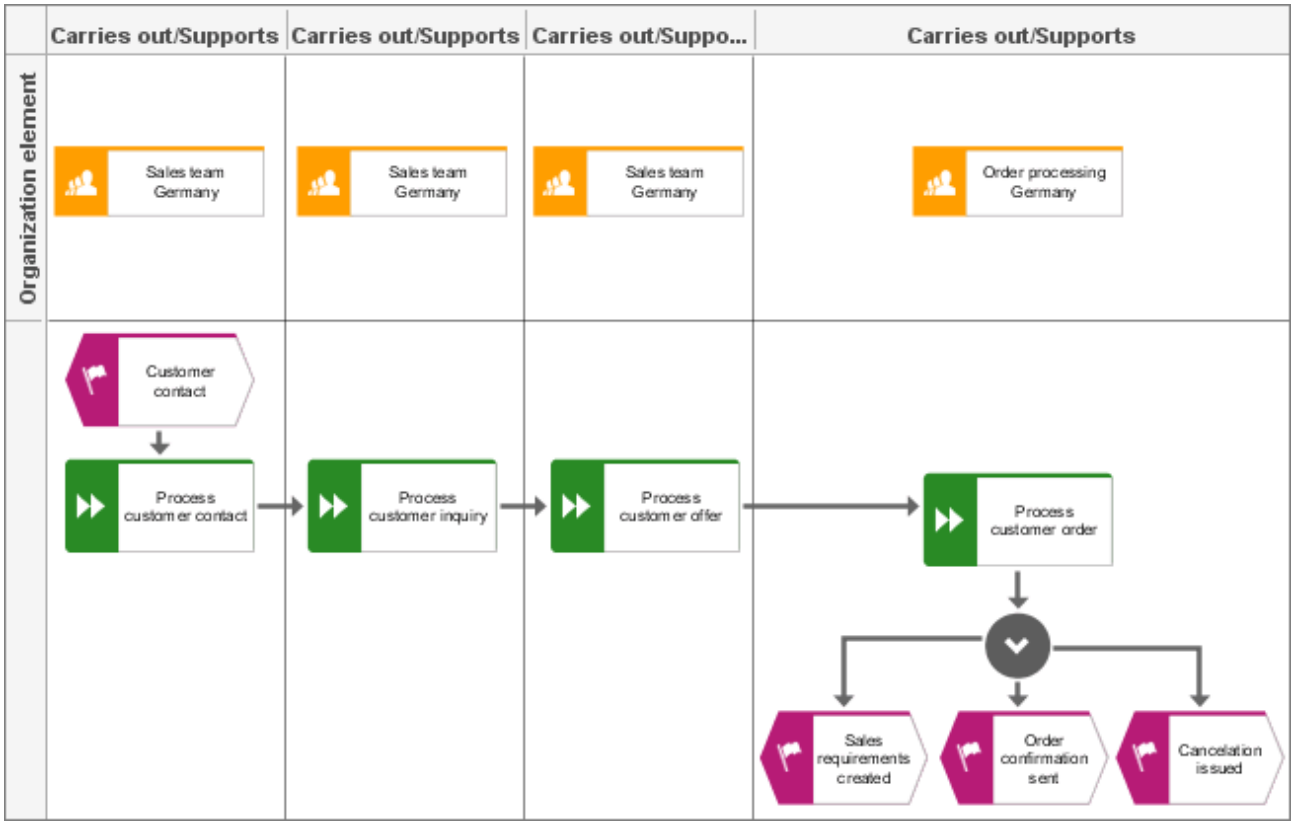
The following description also applies to the EPC (row display).

Most of the explanations on the EPC also apply to the **EPC (column display)** model type, except that all symbols in this model are distributed over various columns. The advantage is that this representation makes the EPC much easier to interpret. Organizational elements and application systems are placed in the diagram header. All other symbols are placed in the second row of each column.

A particular characteristic of all lane models (that is, models that are modeled in columns and/or rows) is the automatic creation of invisible (implicit) relationships. For example, when you model application systems and functions, the implicit relationship 'supports' is automatically created in the default columns of the EPC (column display). Organizational elements and functions are implicitly connected by a 'carries out' relationship. The user may also add the following columns named after the implicit relationships:

- Contributes to
- Decides on
- Is IT responsible for
- Is technically responsible for
- Must be informed on cancellation
- Must inform about result of
- Must be informed about
- Accepts
- Has consulting role in

The following figure shows an example of an EPC (column display).



EPC (column display)

The difference between the EPC (column display) and the EPC (row display) lies in the modeling direction. In the EPC (column display) modeling is performed from top to bottom, in the EPC (row display) from left to right.

3.4.1.12. Business controls diagram

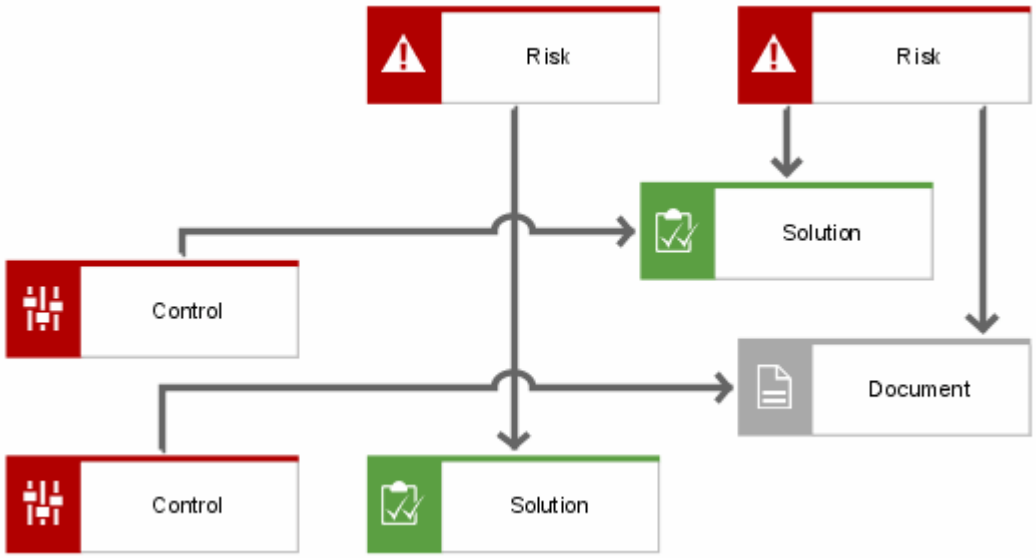
A business controls diagram displays potential risks for a process or function as well as risk control methods.

A risk represents the possible danger of a defined process objective not being achieved.

Risk control is a general way of eliminating or minimizing risks.

Risk solution means implementing risk control for a risk.

The business controls diagram layout corresponds to a matrix or table. The abscissa shows the potential process risks, and the ordinate shows the possible risk control methods. Risk solutions are now inserted as operators between risk and risk control. Additionally, organizational units (in the sense of user requirements) and documents, which also support implementation of risk control for a risk, may be added to the model.



Business controls diagram

This model type is generally used to describe SAP® standard processes. The model shows the risks and risk control methods of an SAP® solution for the process being examined.

3.4.1.13. E-Business scenario diagram

The smooth execution of inter-company business processes is continuously gaining in significance. Here, the focus of interest lies on the operational sequence of specific procedures at the interfaces between companies on the one hand and at the interfaces between companies and their customers on the other. The contacts need to take place in a clear, quick, consistent, and direct manner.

Also, rapidly finding suitable business associates (from a corporate perspective) and providers (from a consumer point of view) is becoming increasingly more important. Maximum optimization of these processes results in a competitive advantage. The ideal platform for supporting these bilateral relationships is the Internet. As the processes in the above-mentioned environment are very complex, it is necessary to define the term **e-business**.

E-business describes all computer-assisted processes involving two economic agents and the attempt to gain added value by using new media.

Thus, e-business can mean simply acquiring an item using the Internet, or a highly complex project involving two companies, or creating a website for a corporate presentation.

Relationships between companies are referred to as Business-to-Business (or B2B), while relationships between companies and consumers are called Business-to-Consumer (or B2C).

The E-Business scenario diagram was developed to support e-business.

The ability to view a value-added chain in its entirety, that is, from the end user to each of the companies involved in a procedure, provides a basis for developing optimization potential. The objective is, for example, to improve the supply chain, to reduce procurement and distribution costs, or to optimize the information system architecture. The contents represented by the objectives can be modeled using this method.

The economic agents are arranged in the upper row of the diagram and referred to as **Business participants**. The participating companies can be assigned using an organizational chart. Here, interest centers on the individual processes that economic agents perform as part of the overall process as well as the interfaces between these subprocesses. An individual process is a business process that plays an important role in inter-company cooperation and that can be assigned to the process model. The business process is supported by application systems (business components), such as the R/3 system.

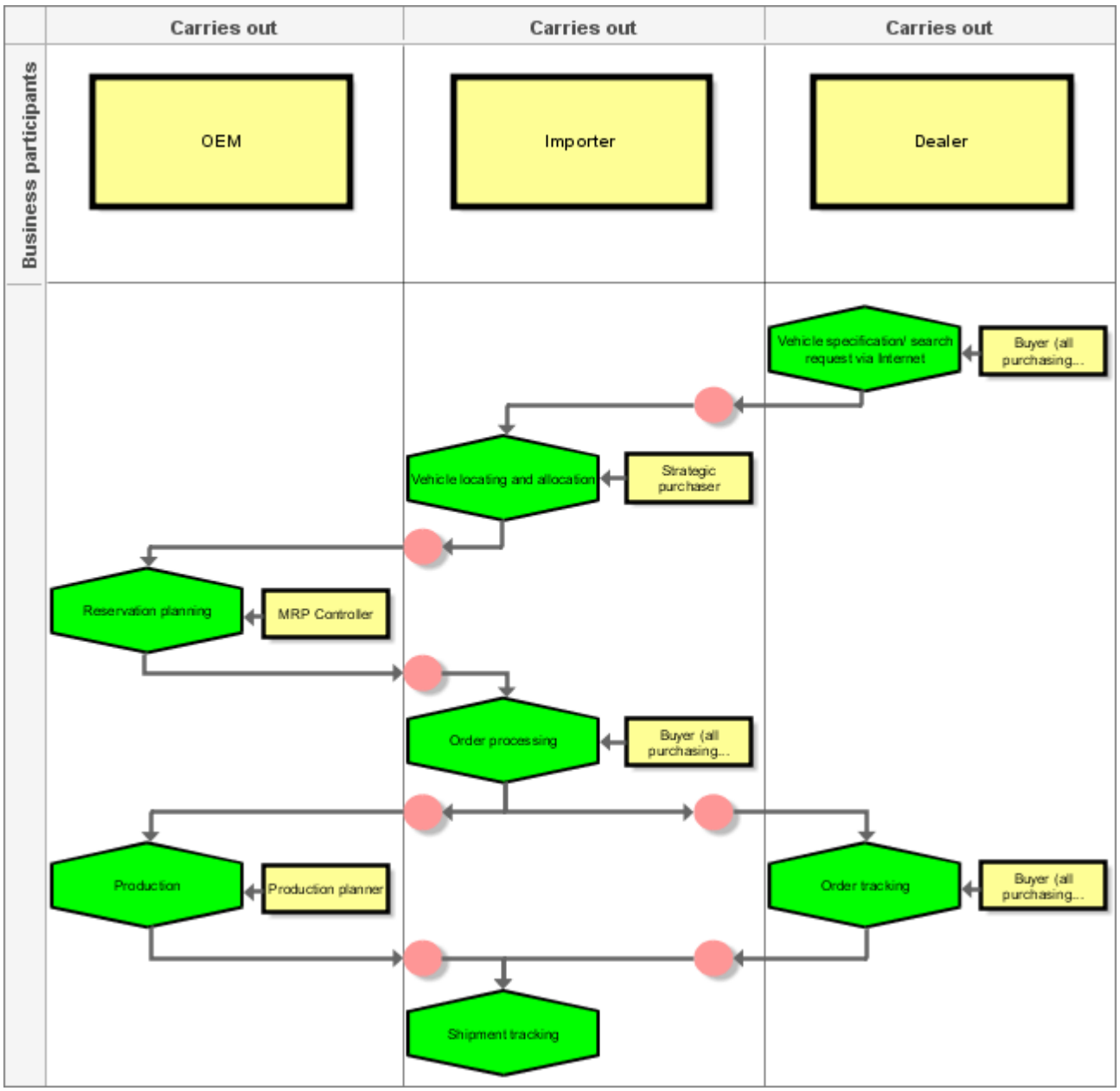
Even the roles of the employees involved in the process can be defined. These are referred to in the model as **Employee role**.

The main feature of the interfaces is the transfer of process-specific information. The information is gathered in business documents and can assume the form of an XML or HTML document. The business document can also be assigned as a data model. As an alternative to this object, the objects **Money transaction** (for representing a cash flow), **Goods shipment** (for representing a flow of goods), as well as **E-mail**, **Internet**, **Intranet**, **Extranet**, and **Mobile phone** (for specifying the technological aspects of the data transfer) can also be used.

All operational procedures relating to a company are modeled in the row below the business participant, but in the same column.

Thus, column borders form abstract interfaces. These merit special attention, as they carry the main potential for optimization, and it is therefore always beneficial to model them.

Explanation of terms: In the sample model below, OEM stands for **Original Equipment Manufacturer** and MRP for **Material Resource Planning Controller**.



E-Business scenario diagram

The sample model shows how a manufacturer, an importer, and a dealer cooperate. Each party has its specific processes in the overall structure and uses business documents to exchange information via the interfaces to processes of other business associates. The persons involved in the business processes are also recorded and assigned with their roles.

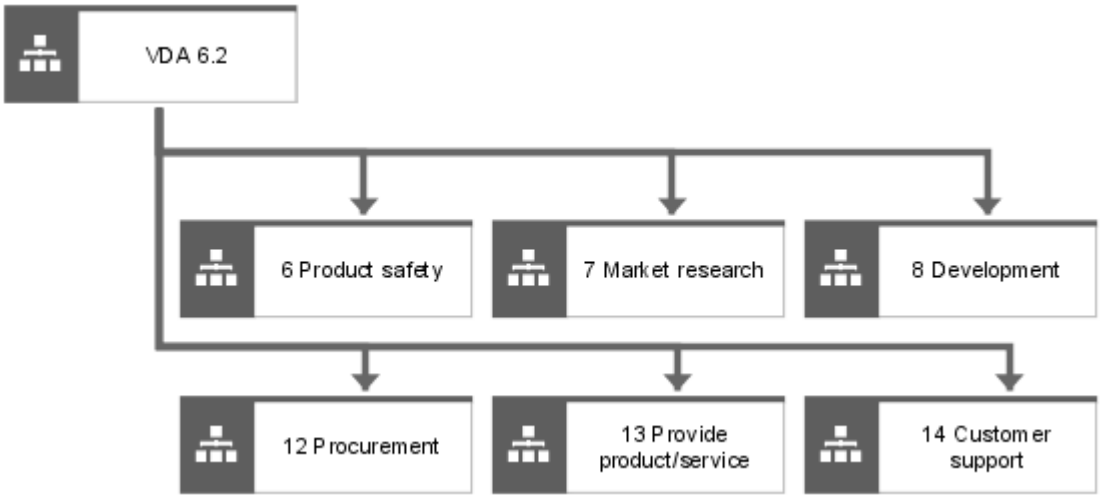
3.4.1.14. Structuring model

The structuring model is generally used to express the hierarchy or systematization (specialization or generalization) of facts.

A structural element represents a fact (in the direction of the intended systematization).

Models relating to the facts can be assigned to individual structural elements of the fact hierarchy.

Structuring models are most frequently used in quality management, particularly for certification purposes. There, the structuring model divides a norm into its individual components, and models which help meet the quality criterion are assigned to individual structural elements.



Structuring model

By means of a report, these facts can easily be evaluated or used for documentation purposes.

3.4.1.15. Role diagram

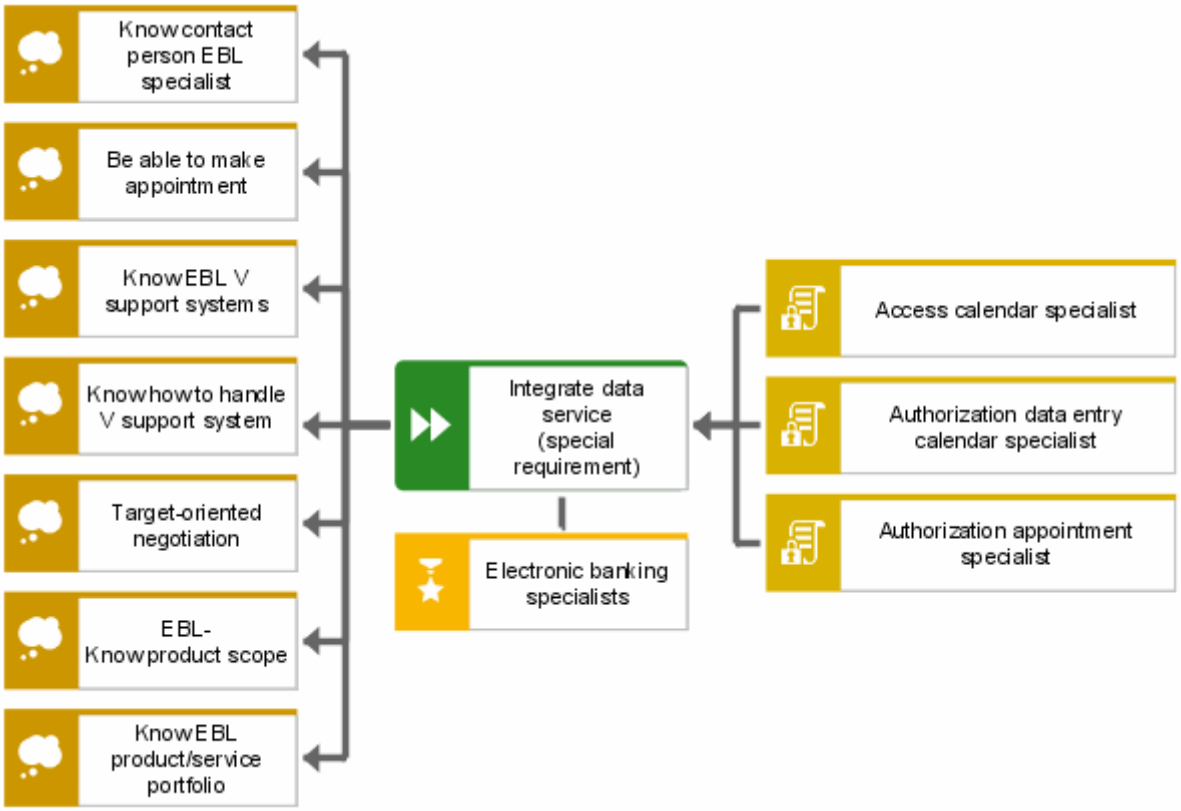
In general, the role diagram is used to describe processes in more detail. The focus is put on the organizational units participating in the processes as well as on their roles. Objects and their relationships have the following properties:

A role participates in processes with due consideration of authorizations. The specification of the authorization type in the process (a role is 'involved in the execution') is just as significant as executability itself. During process execution with a specific authorization, the Role - Participation - Process relationship chain (including both Participation - Authorization condition and Participation - Authorization value) is established.

A role can be occupied by persons, positions, or information systems. The role forms the link between the processes and the resources involved in them. It is defined by an aggregation of expectations of the resources involved in the processes.

Executing a process requires skills that the participating role or the allocated resource must have. To be able to define roles in a process-oriented manner, the processes must be assessed and the process requirements of the persons/systems involved be specified. More precisely, requirements of persons or systems constitute the knowledge and capabilities (skills) these persons or systems have. The evaluation of a skill is standardized by an assigned evaluation scale.

Therefore, in the role diagram it is possible to depict processes and specific elementary processes, represent the resources involved, record their skills or required skills, and show their authorizations.



Role diagram

The sample model displays the elementary process' requirements of the role (capabilities and authorizations), as well as the elementary role's requirements of the resource with regard to capabilities and authorizations.

The diagram is assigned to the relevant elementary process and elementary role. By assigning the diagram to the elementary process, the requirements of the underlying process EPC (corresponds to the process reference model) can be viewed. Assigning it to the elementary role shows the elementary role's requirements of the resource with regard to its capabilities and authorizations in the role structure diagram.

3.4.1.16. Quick model

The **Quick model** model type enables you to model without method restrictions. The Quick model contains the **Quick object** object type with over 30 different symbols. Relationships of the **has relation with** type can be created between quick objects. Multiple connections of this type are allowed between two objects.

The corresponding default attributes can be specified for models, objects, and connections.

You can assign multiple quick models to all objects of any object type provided by ARIS Method. In addition, you can assign any number of models provided by ARIS Method to a quick object regardless of the model type.

3.4.1.17. Screen design

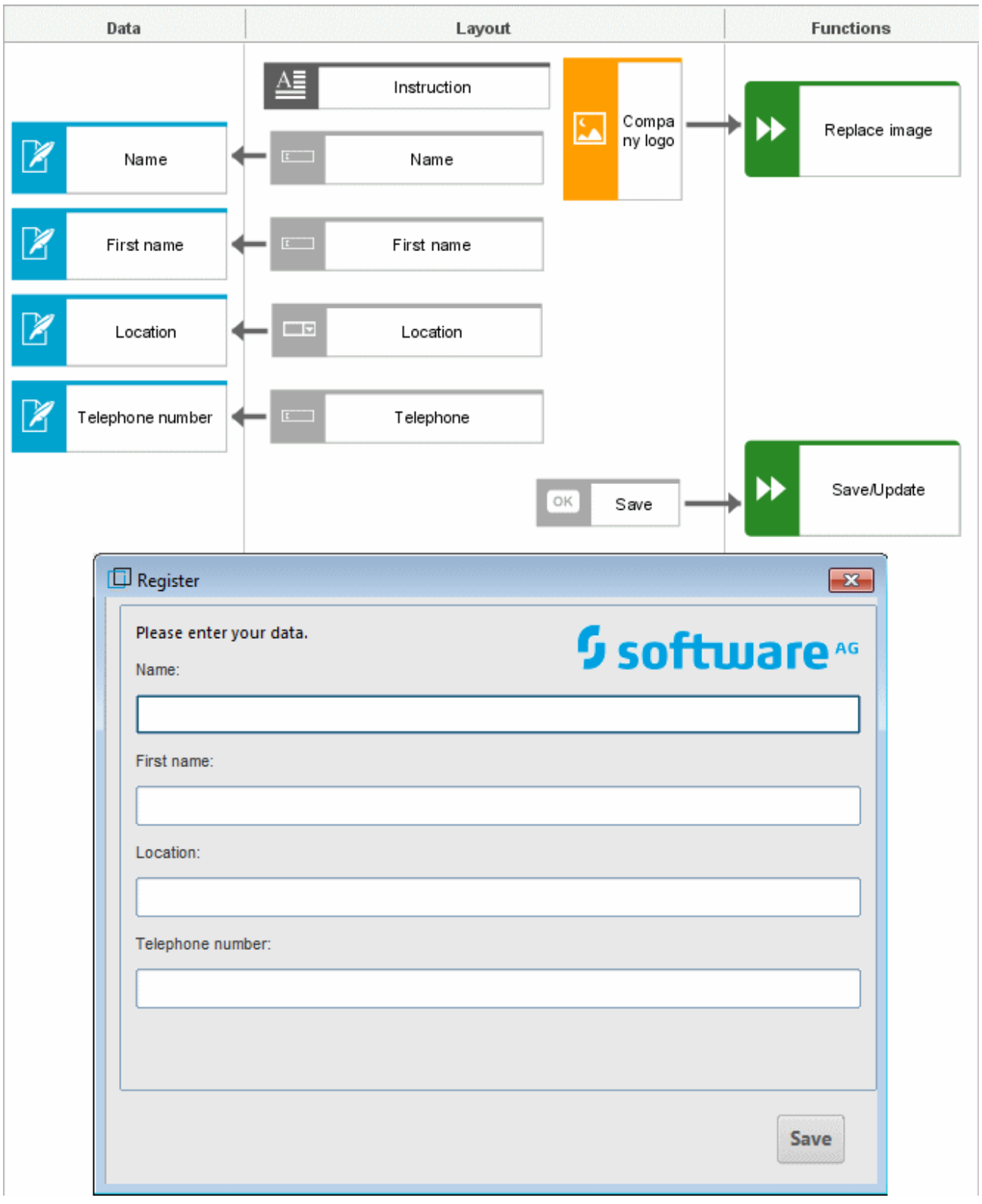
When designing software, you can use a 'Screen design' in ARIS to specify the technical requirements of a dialog or Web form.

In the **Layout** column, you specify the structure of the dialog or Web form. The procedure for designing a dialog, for example, is similar to working with a resource editor in a development environment.

The graphic elements that can be placed in the **Layout** column include text boxes, spin boxes, option buttons and check boxes, combo boxes, buttons, tree and list controls, as well as bitmaps and static text. You can use the **Tabindex** attribute type to specify the order in which the Tab key moves the cursor to the individual screen elements.

You can place various data elements and function objects in the **Data** and **Functions** columns. Using a connection of the **represents** type, the objects can be associated with the data elements and functions they are editing.

Each screen design can be assigned to the corresponding screen object that is used, for example, in an EPC or a model of the **Screen navigation** type. In addition, a screen design can also be assigned to the entity type, cluster, complex object type, class, or function/IT function edited via the screen.



Screen design

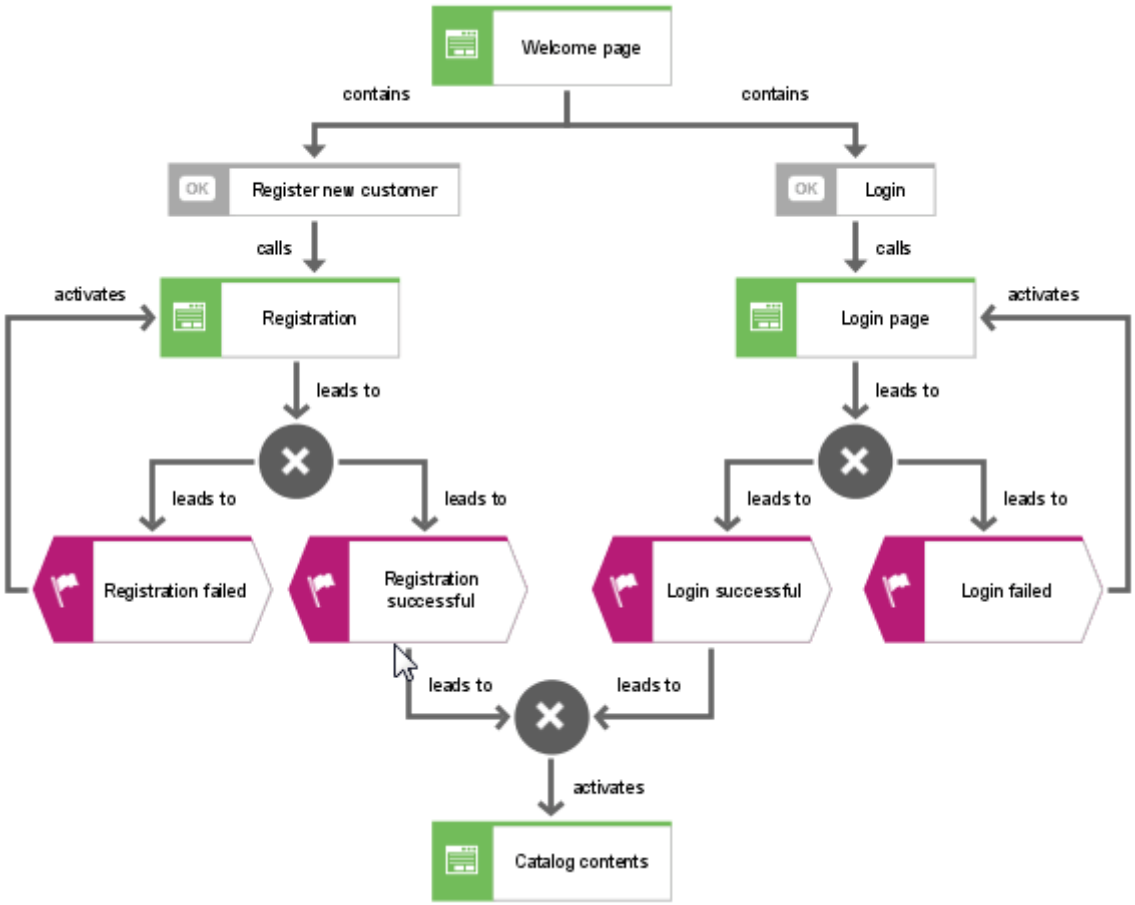
3.4.1.18. Screen navigation

In a model of the **Screen navigation** type, you can either specify the structure of a screen that comprises multiple subscreens (for example, a website with several form fields or frames), or you can describe the transition between various screens. The transition between the screens can be described in detail.

Example 1.

You want to illustrate that a screen element has to be active before the next screen can be accessed. Assign the triggering screen element (of the **Screen design** model) to the screen using the **contains** connection. Then draw a connection of the **calls** type from the screen element to the screen that follows.

It is also possible to show that navigation depends on events. When you exit a screen various events can occur. For example, if a user has completed the registration page of an online shop, the registration can either be completed successfully or it can fail. This will determine whether the user moves to the contents page of the catalog or is returned to the registration page.



Screen navigation

3.4.1.19. Business segment matrix

In the business segment matrix, the various markets in which a company is active are shown in an overview and their significance for the success of the company is shown in visual terms.

Each market is described by

- the product or service offered and
- the customer group that the offer addresses.

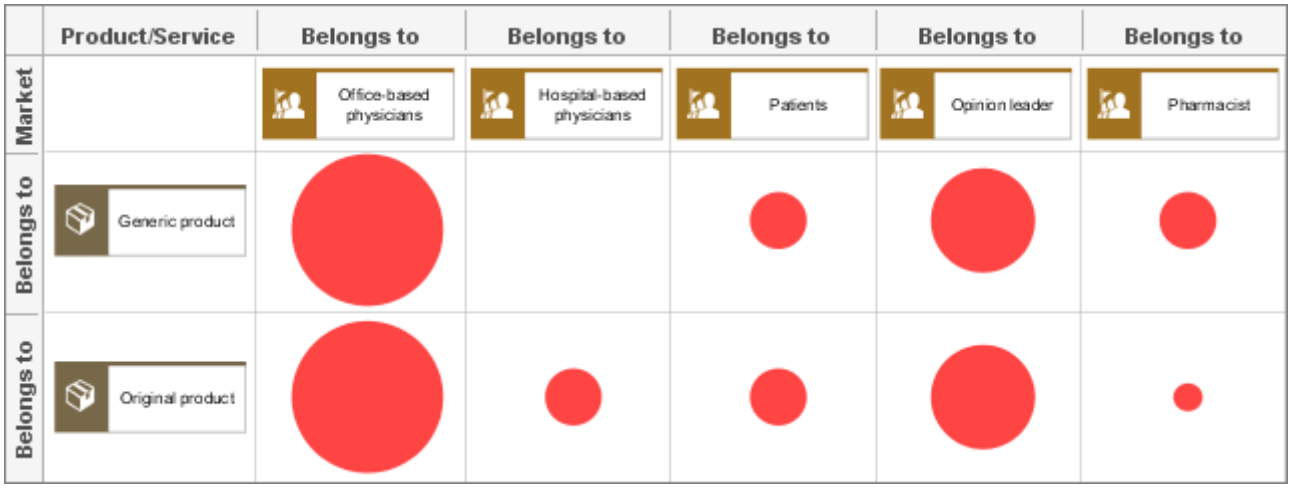
Products and services (objects of the **Product/Service** type) are placed in the cells of the first column of the business segment matrix. The target group (various organizational elements) is placed in the cells of the first row. You define the market by placing a business segment object in the cell where product row and target group column intersect. Implicit relationships of the **belongs to business segment** type are established between the product/service and the organizational element.

To emphasize the significance of a business segment, five different symbols are available from **almost unimportant** to **very important**.

When modeling, you need to observe that each business segment can be placed within the matrix only once.

For each business segment, you can indicate its importance in terms of the corporate strategy. A strategy describes long-term procedures that the company employs to achieve its objectives and to gain competitive edge.

The following figure shows a business segment matrix from the pharmaceutical field.



Business segment matrix

Business segments can also be assigned an objective diagram. The objective diagram contains the objectives set for the business segment as well as the processes and success factors supporting the achievement of these objectives.

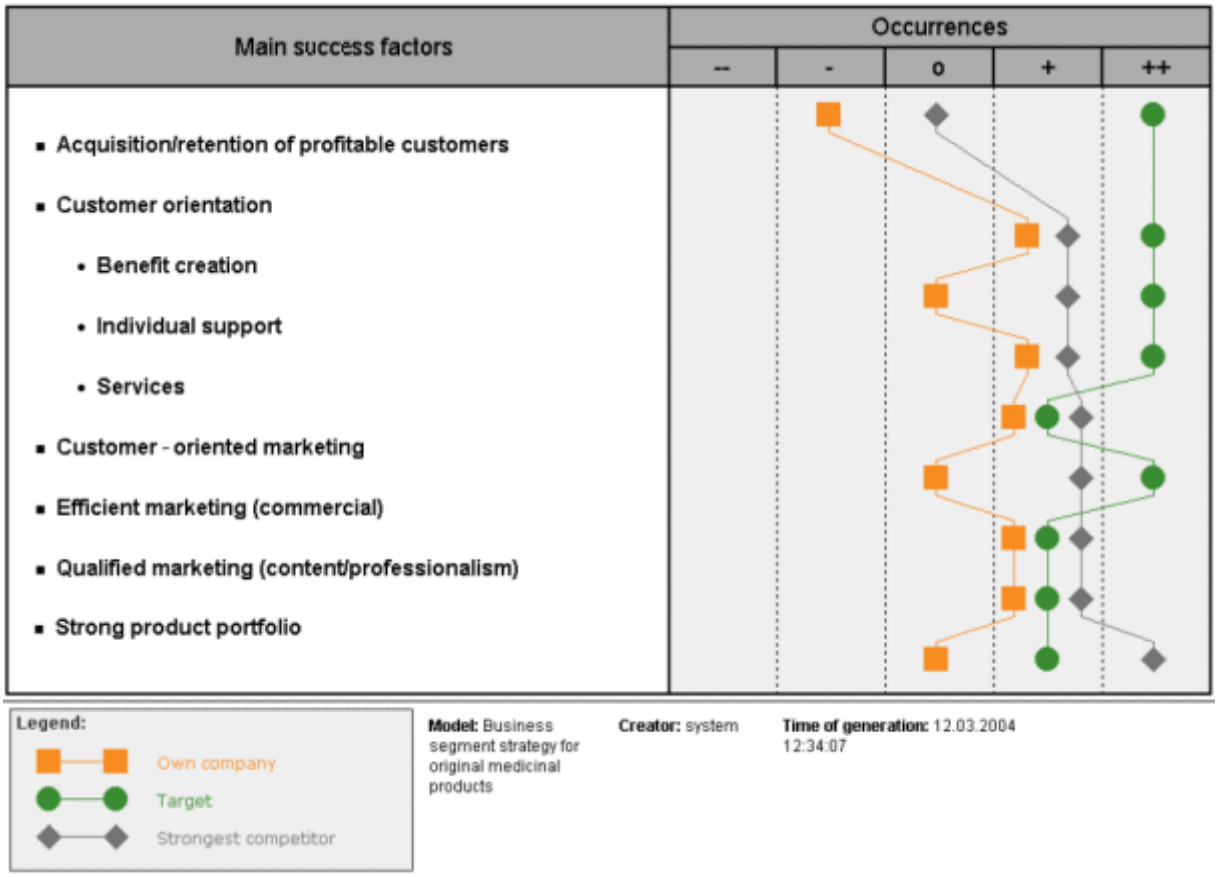
The success factors in the objective diagram can be the basis of success factor analysis if the **Success - Actual**, **Success - Target**, and **Success - Competitor** attributes are specified in the attribute type group of the same name. Success is evaluated by means of a five-step scale from **very low** to **very high**.

Procedures

To perform a success factor analysis,

1. use the pop-up menu of the business segment to start ARIS report (**Evaluate > Report**),
2. and select the **MSF_Analysis(Object).rso** report script of the **BPM** group in the default path of the Report Wizard.

The report is output in HTML format.



Report

Alternatively, you can start the success factor analysis via the pop-up menu of the objective diagram. Select the **MSF_Analysis(Model).rsm** report script.

3.4.2. Design specification

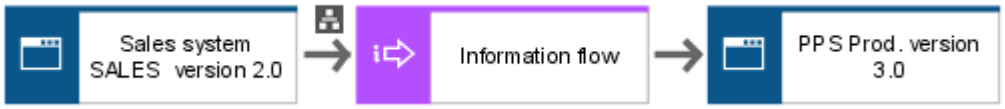
3.4.2.1. Access diagram

The relationships illustrated below between the objects explained in the design specification descriptions of the other views can be included in the access diagram of the control view. In order to make the illustration clearer, the individual dual relationships are dealt with separately.

3.4.2.2. Linking functions with data

First, you can define the information flows between application system types, module types, or IT functions. For this purpose, an information flow object is created between the corresponding application system types or module types. In order to specify the information flow between system types in more detail, an eERM diagram or a table diagram is linked with the information flow object. Thus, the information flow objects may be located either at the requirements definition level, design specification level, or implementation level.

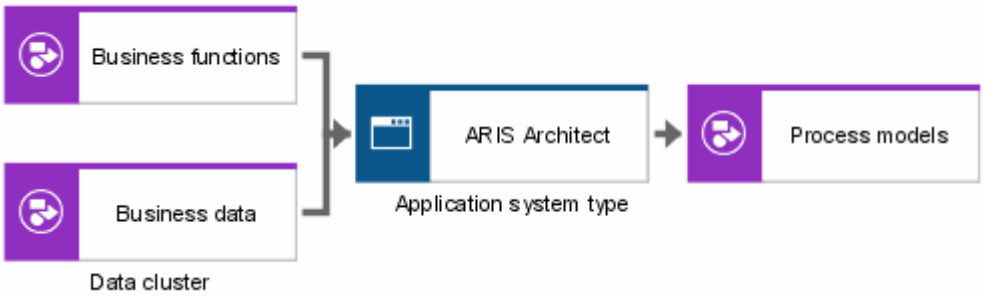
An example is shown in the following figure.



Information flow between application system types

Besides the information flows, input and output data of every application system type, module type, and IT function type can be represented as data objects of the requirements definition or design specification. The direction of the arrows indicates whether it is an incoming (input) or outgoing (output) data flow.

An example is shown in the following figure.



I/O data at the design specification level

3.4.2.3. Linking organization with function

The fact that the organizational aspects are linked to the functional aspects defined at the design specification level basically answers the following questions:

- Who (which organizational unit, position, person, etc.) is responsible for the application system types and module types specified in the function view at the design specification level, or who uses these systems?
- Which locations (organization view) within the company use which application system types or module types?
- Which platforms available in the company (hardware component types (organization view)) are suitable to run which application system types?

In order to answer the first question, connections can be drawn in the access diagram between the organizational units of the organizational chart (organizational units, positions, and persons) and the objects of the application system type diagram (application system type, module type, IT function, etc.). While doing this, the significance of this relationship can be specified more precisely. We distinguish the following:

- An organizational unit may be **responsible for** an application system type as far as the **technical aspects** are concerned.
- An organizational unit may be **responsible for** the **development** of an application system type.
- An organizational unit may be a **user** of an application system type.

The question of location may be solved by assigning locations from the organization view to application system types, module types, and IT function types.

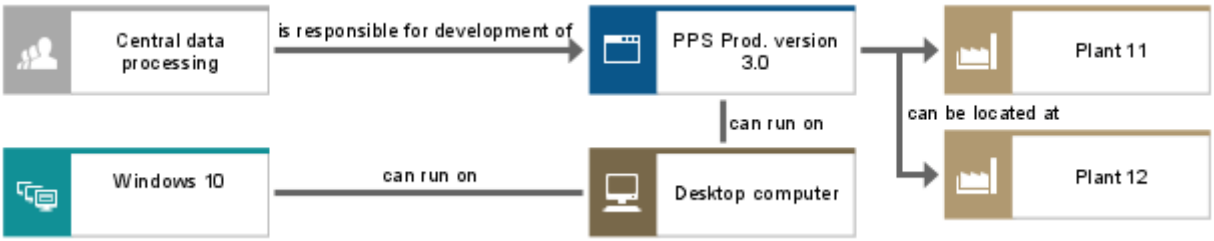
In the design specification we are not dealing with individual application systems in the sense of individual licenses, but with application system types. Therefore, no actual locations of application systems are

defined by means of this relationship (this is realized at the implementation level), but possible locations of a particular application system type are pointed out.

The design specification of the organization view defines the hardware component types available in a company. In the control view, the relationship between these hardware component types and the application system types can be established. This is how the hardware platforms for running certain application system types, module types, or IT function types are determined. At this stage, the graphical user interface types, operating system types, and DBMS types included in the function view can be assigned to the hardware component types, as well.

A list of relationships that are available in an access diagram is provided in the **ARIS Method Reference** help.

The following figure shows examples of relationships.



Access diagram (excerpt)

3.4.2.4. Program flow chart

In the access diagram, you can establish the relationships between the object types of the organization view and data view and the application system types, module types, and IT function types specified in the application system type diagram (see chapter **Access diagram [84]**). In this model type, you cannot directly represent the allocation of functions of the requirements definition. This allocation is realized in the application system type diagram. Similarly, possible chronological-logical operational sequences of application system types, module types, and IT function types cannot be illustrated directly. Strictly following the ARIS architecture, you can trace these links only by navigating through a number of model types.

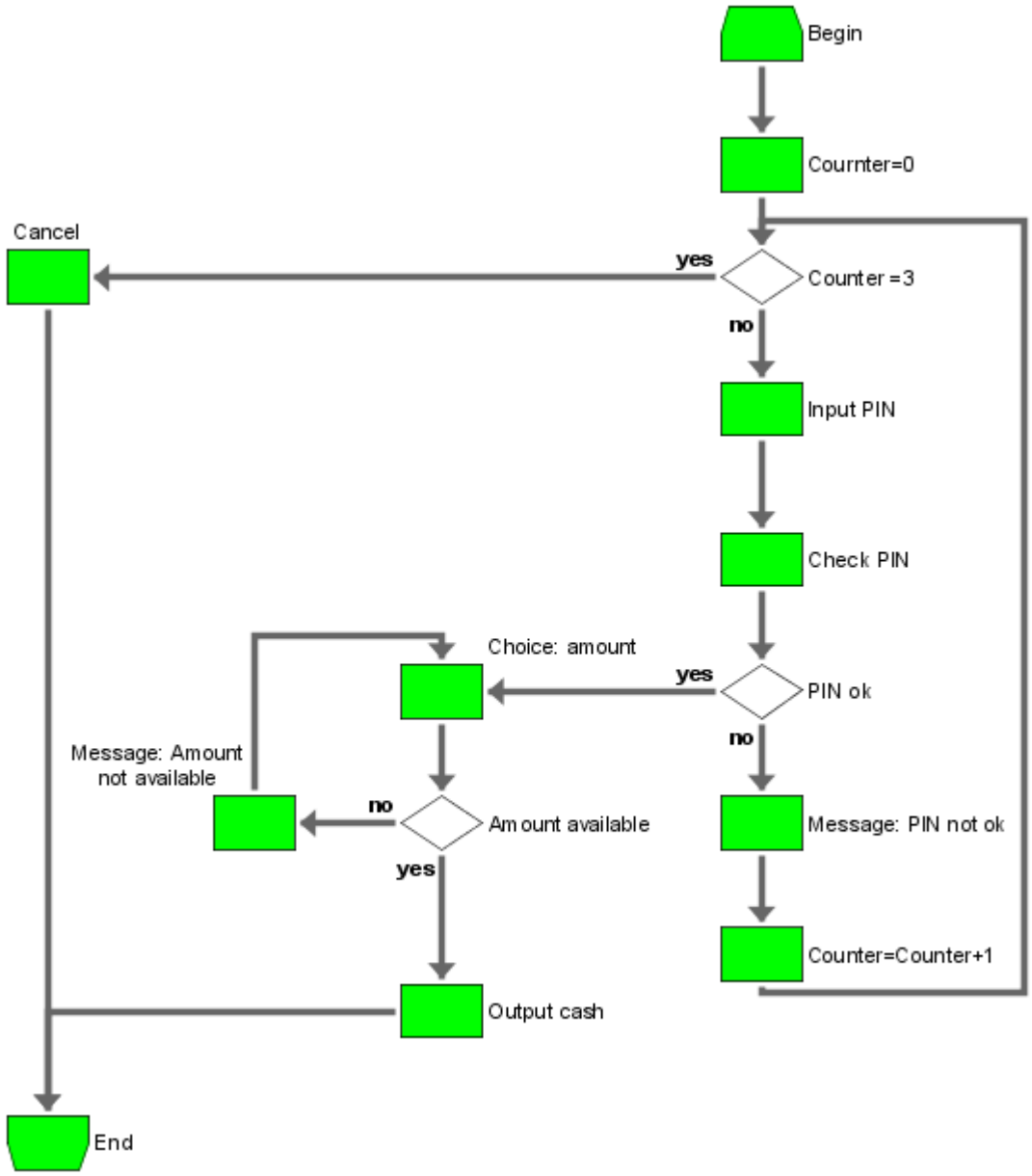
However, in the system design environment, model types (for example, program flow charts (PF) [86]) have been established that allow an integral view of all aspects of the system design.

For this reason, **ARIS** provides the **Program flow chart** model type. It enables you to model all relationships to application system types, module types, and IT function types available in other **ARIS** model types, regardless of the **ARIS** breakdown into views. Moreover, you can represent the chronological-logical operational sequence of the object types mentioned. For this purpose, events are provided in this model type. Similar to arranging functions and events in the EPC, you can define module sequences in the program flow chart. In this context, the event is seen as a trigger that activates module types or application system types. Branches can be represented by the rules known from the EPC. Unlike in an EPC, you can also define operational sequences in the program flow chart without the use of events.

3.4.2.5. Program flow chart (PF)

The program flow chart (PF) is used to show processing sequences of a program. The sequence of the processing steps is illustrated by the relationships between the objects. This diagram does not represent any data.

The following figure shows a simplified example of the processing sequences of an automatic teller machine. The illustration of the processing sequences reveals a strong focus on implementation.



Example of a program flow chart (PF)

3.4.2.6. Screen diagram

A screen diagram describes screens used in software development. The aim is automatic derivation of screens from the screen diagram.

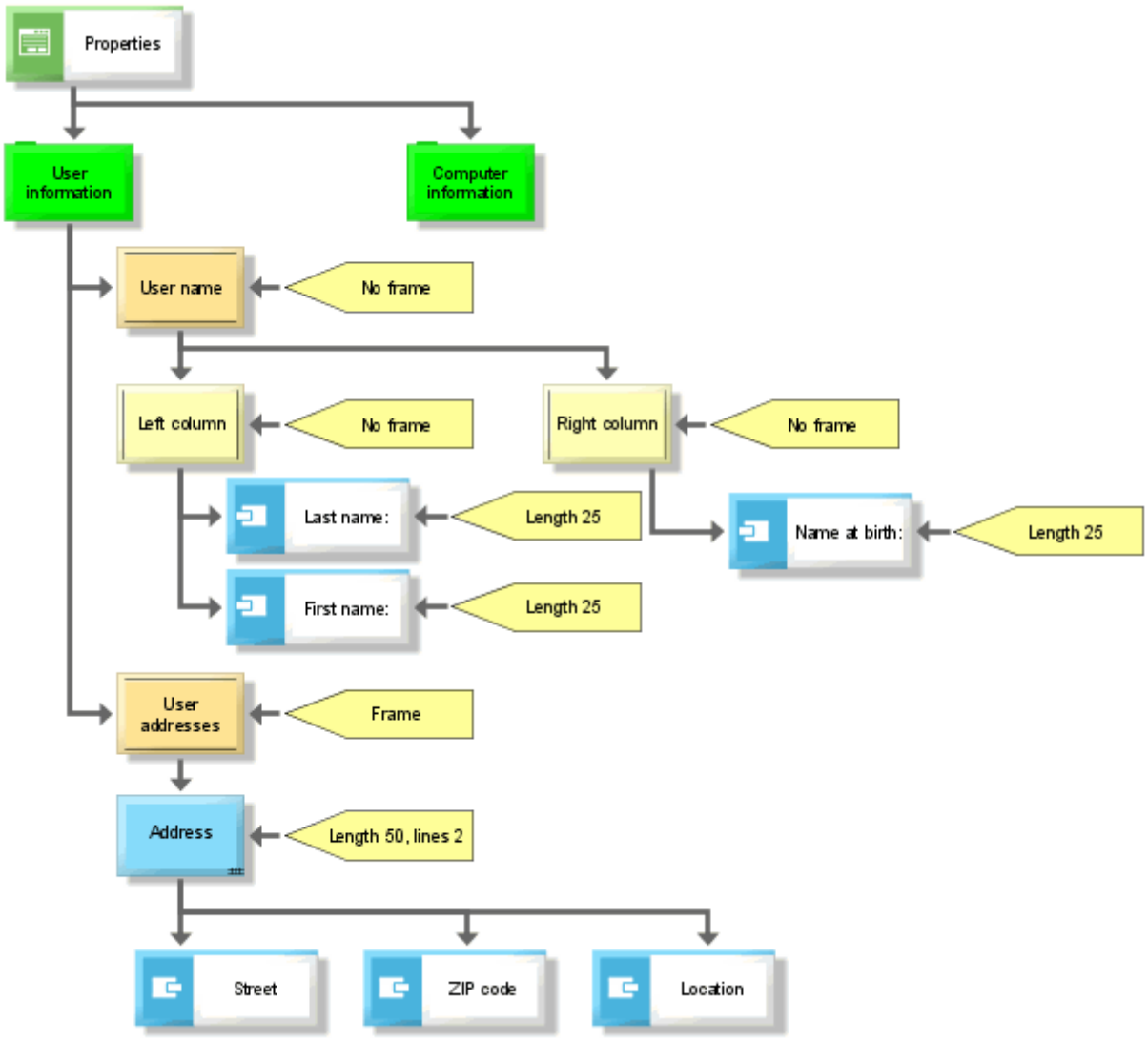
Thus, screen diagrams display the structure and to a certain degree the functionality of screens. From left to right and top to bottom, the screen diagram's structure corresponds to the geometry of the interface described.

The central symbol is the 'screen', which corresponds to a window in Windows terminology. This window can have multiple tabs (**Page** symbol). In general, the interface can be divided into areas using a table format

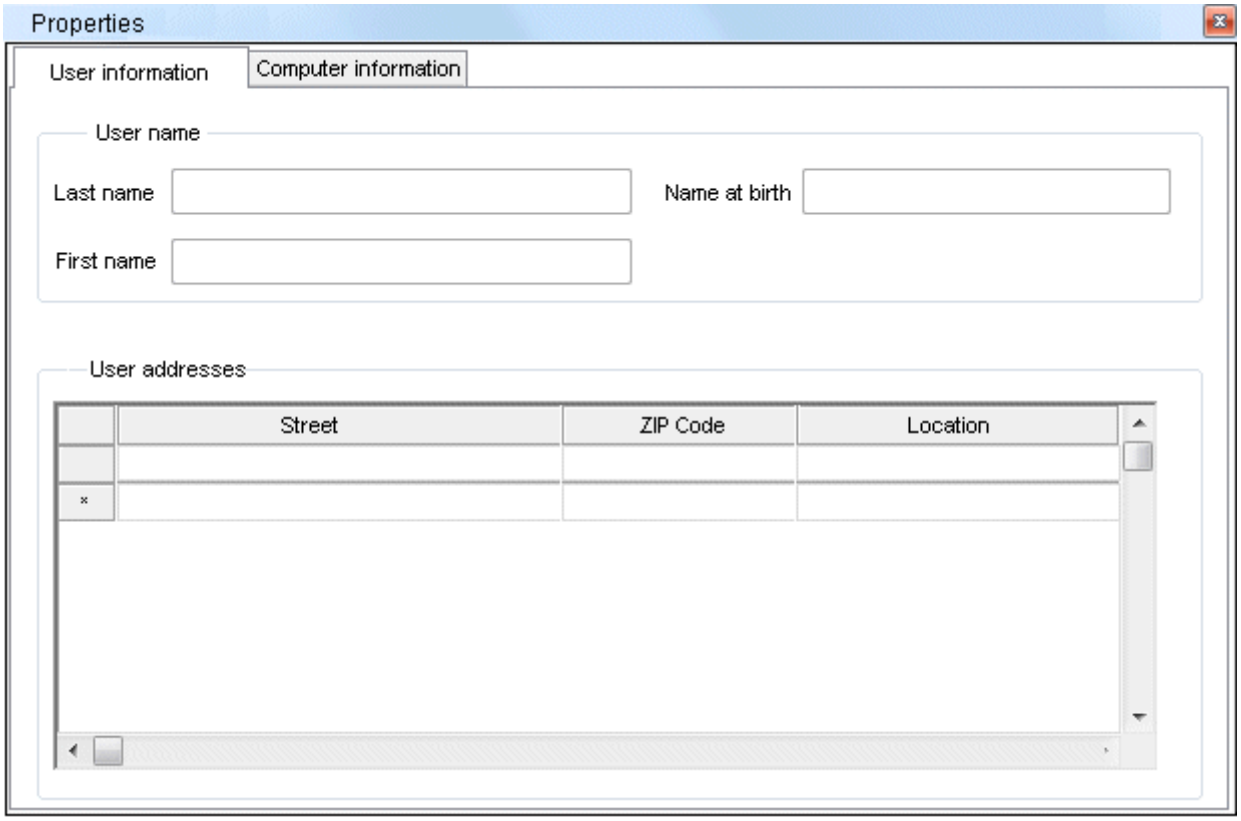
(**Section** symbol for a row, and **Column** symbol for a column). The **Section** and **Column** symbols can be nested as required in order to form complex interfaces. You can place tables (**Screen table** symbol), text entry boxes (**COT attribute** symbol), graphics (**Bitmap** symbol), and text descriptions (**Text** symbol) on the interface. Using the **Layout** symbol you can assign representation properties to the **Screen**, **Page**, **Section**, **Column**, **Screen table**, **COT attribute**, and **Text** objects.

Additional symbols can be used to describe the screen interface.

The following figure shows an example of a screen diagram. The second figure illustrates the screen derived from the diagram.



Screen diagram (1)



Screen diagram (2)

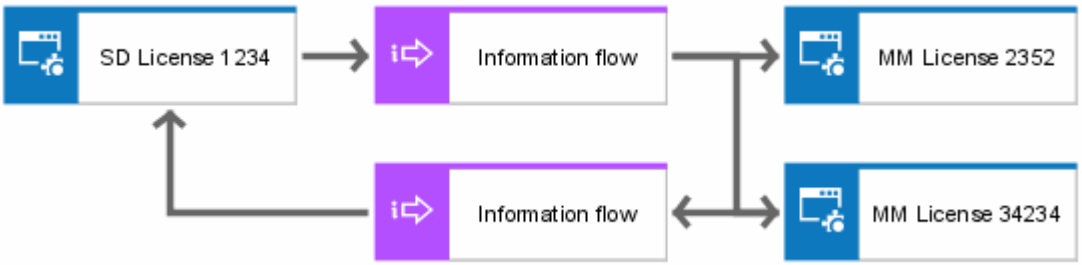
3.4.3. Implementation - Access diagram (physical)

The questions considered in the design specification of the control view are also relevant for the implementation level. However, in contrast to the design specification level, concrete specimens of individual objects are examined, not object types. For example, the focus may be on relationships between concrete application systems and organizational units, not on relationships between application system types and organizational units.

The relationships illustrated in the following are modeled in the access diagram (physical).

3.4.3.1. Linking functions with data

To show which data is exchanged between application systems, information flow objects can be created between application system objects of the function view. Unlike application system objects at the design specification level, these application system objects are no application system types, but concrete specimens (individual licenses). This means that application systems, modules, and program module types can be interlinked by data flow connections. If you defined at the design specification level that the **Sales system SD version 2.1** module type can exchange data with the **Material management system MM version 1.2** module type, the implementation level shows that the **SD license no. 1234** module exchanges data with the **MM license no. 2352** and **MM license no. 34234** modules. Both MM modules are of the **Material management system MM version 1.2** module type. This is illustrated in the following figure.



Data flow

To specify the data objects that are exchanged between systems in more detail, corresponding model types of the data view are assigned to the information flow objects.

Apart from the data flows between application systems, input/output data can also be specified for every application system. There are two reasons for the relationships to be represented in an access diagram (physical). In the first case, the data objects are objects of the table diagram (table, field, view (physical)) located in the data view of the implementation level. These data objects can be linked to application system objects of the design specification level or implementation level via input/output relationships. In the second case, the application system objects are concrete application systems or modules of the implementation level, which are linked to objects in the data view.

Therefore, the following general rule can be defined:

If one of the object types participating in an input/output relationship originates from the implementation level of the relevant view, the relationships in the process view are represented at the implementation level (access diagram (physical)), as well.

An example is shown in the following figure.

Input/Output relationships

3.4.3.2. Linking organization with data

The focus is on the same questions dealt with in the design specification:

- Which organizational units are responsible for data objects?
- Who has access to which data objects?
- Which data objects are stored on which hardware components?

Unlike the relationships in the design specification, the relationships established here form a link to the data objects shown at the implementation level of the data view.

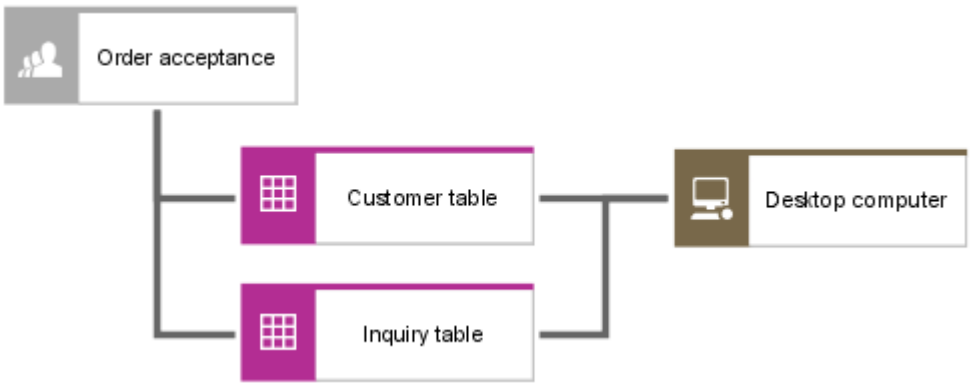
This means that the responsibility for data objects is defined for relations and attributes for the physical structures, that is, tables, fields, and their specimens [table (specimen), field (specimen)].

To represent these dependencies, connections are drawn in the access diagram (physical) between the objects of the organization view (organizational unit, position, person, etc.) and the table diagram's objects mentioned earlier (table, field, view (physical), etc.).

When a connection is drawn between organizational units and tables and fields, the meaning of each relationship must be defined individually. For example, **is responsible for** means that an organizational unit is responsible for the contents of the relevant table or field, while **accesses** means that a position or person has access privileges for the data objects shown.

In addition to defining access privileges and responsibilities, you can use the hardware component object (organization view/implementation) to define on which of the available hardware components - which can be uniquely identified by the inventory number, for instance - certain information objects of the company are located. For this purpose, the **Hardware component** object may be linked to information objects of the implementation level (tables, fields, etc.), the design specification level (relations, attributes), or the requirements definition level (entity types, cluster/data models, etc.) in the access diagram (physical).

The figure below illustrates an example.



Assignments to hardware component

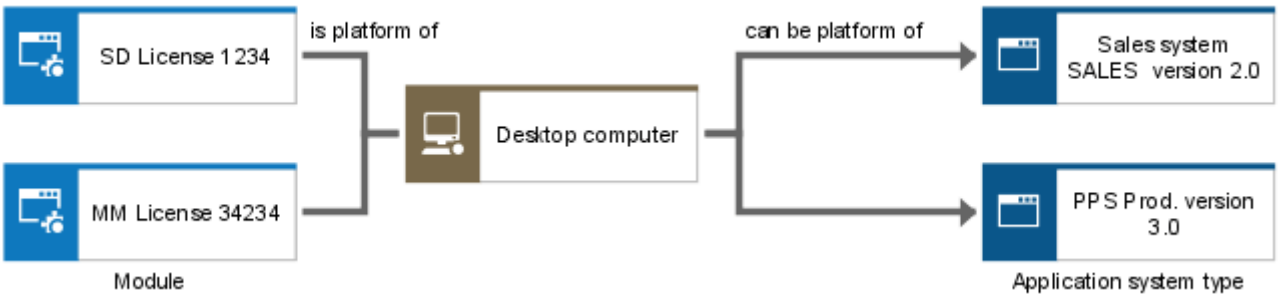
3.4.3.3. Linking organization with functions

The relationships defined in the access diagram (physical) between objects of the organization view and the function view answer the following questions:

Which application systems run on which hardware components, and which application system types can run on them?

In order to illustrate these dependencies, the **is platform of** and **can be platform of** relationships can be modeled between the application system objects of the implementation level (application system, module, program module, etc.) or the design specification level (application system type, module type, etc.) and the **Hardware component** object type of the organization view.

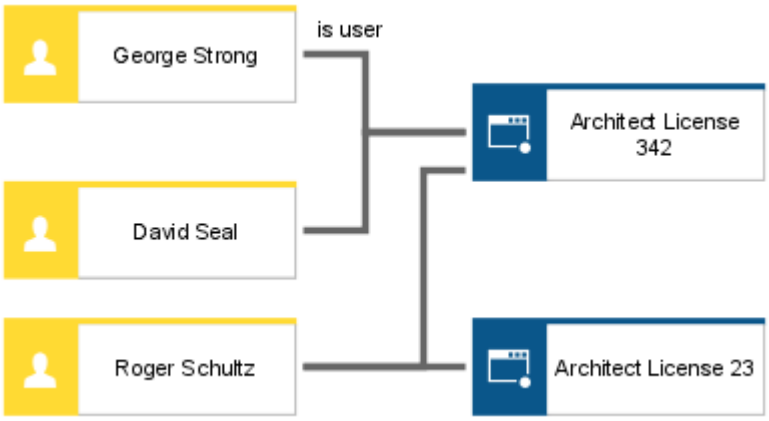
An example is shown in the following figure.



Hardware component as platform

Which organizational unit uses a specific application system?

While the design specification level is the level for defining the users that access certain application system types, the implementation level allows for defining this relationship for specific application systems (individual licenses). For example, it is possible that in one company multiple licenses of the **ARIS Architect** application system type are available with different configurations. An access diagram (physical) can show which user is using which license. For this purpose, the **Organizational unit**, **Position**, and **Person** object types may be linked with the **Application system** and **Module** object types via the **uses** connection. The following figure illustrates an example.

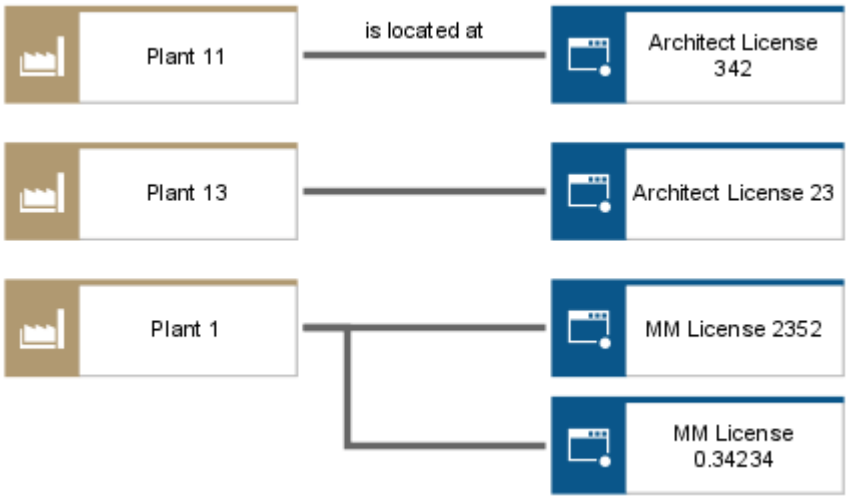


Users and application systems

Which company locations have application systems?

The design specification can use the **Application system type - Location** relationship to define which application system types may be situated at specific locations in the company. For the purpose of specifying exactly where in the company individual licenses assigned to an application system type are used, the access diagram (physical) may serve to link locations with the **Application system**, **Module**, and **IT function** object types.

An example is shown in the following figure.



Location assignments

The **ARIS Method Reference** help all relationships available in the access diagram (phys.).

3.5. Product/Service modeling

ARIS provides various model types for describing a company's products and services.

Products or services are generated or provided in the course of a value creation process. They are the result of a human act or a technical procedure. The term product/service refers to the supply of either services or goods.

Goods can be consumable products, material types, operating resource types, technical operating supply types, or packaging material types. The trigger for creating a product or service is always the demand of an organizational unit or a customer. Goods are offered to the customer in the form of tangible merchandise.

Services are intangible products standing out for the fact that they are simultaneously produced and consumed.

For example, typical providers of pure services are banks, insurance companies, or public authorities.

The stronger the customer orientation in the market segment of a product provider, the more important it is for that provider to closely observe and improve the services in the product environment.

Therefore, the various model types available in ARIS are designed for describing both individual products or services and a combination of products and services.

You can use the following model types for product/service modeling:

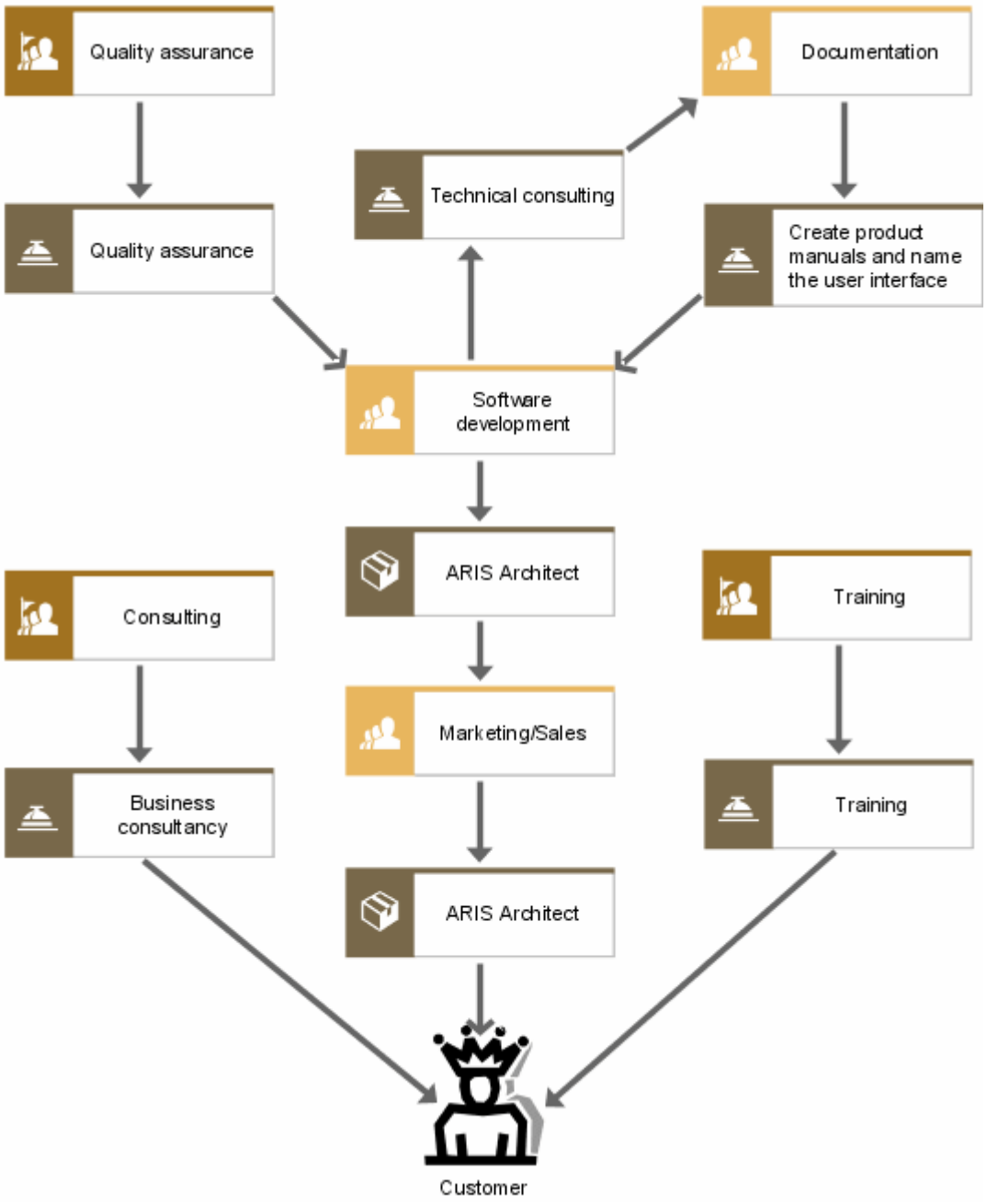
- Product/Service exchange diagram
- Product/Service tree
- Product allocation diagram
- Product tree
- Product selection matrix

3.5.1. Product/Service exchange diagram

The product/service exchange diagram illustrates the creation of products/services and their exchange within the company. The term product/service refers to the supply of either services or products, each of which is represented by the corresponding symbol. Products can be material types, operating resource types, technical operating supply types, and/or packaging material types, all of which are familiar from the EPC (material flow), for example. Products/Services that are the input for and/or output of functions can be connected with the start and/or end events of these functions.

This product/service exchange between business management functions can be used to advantage at an abstraction level that ranges between the value-added chain diagram and the EPC. The product/service exchange relationships can be illustrated not only from a functional viewpoint, but also from an organizational viewpoint. The product/service exchange diagram provides various modeling options to serve this purpose.

The following figure illustrates an example of a product/service exchange diagram.



Product/Service exchange diagram

3.5.2. Product/Service tree

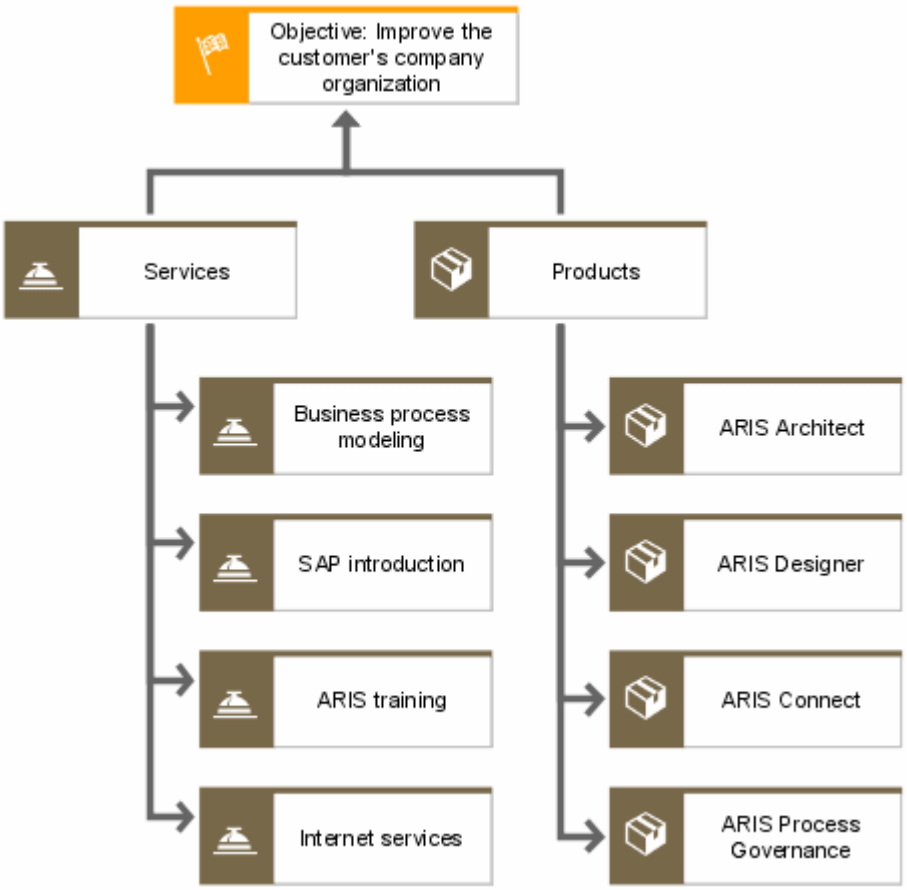
Products or services can be looked at from different levels of abstraction. Therefore, it is useful to store the relationships in a model showing which product or service components are making up a complete product or service. This static aspect is represented in the product/service tree. For example, a complex product often contains several modules, each of which has various component parts. Each of these elements can be understood as a product or service.

The **has relation with** connection, which is also permitted between products or services in the product/service tree, can be used to describe other kinds of dependencies. These include the relationship between a consumer credit and the checking account through which the payments are effected.

Substitution relationships to other products or services such as (potential) replacement products or services can also be represented.

This static model also represents the interrelationship between products or services and the (business) objectives.

The following figure illustrates an example of a product/service tree.

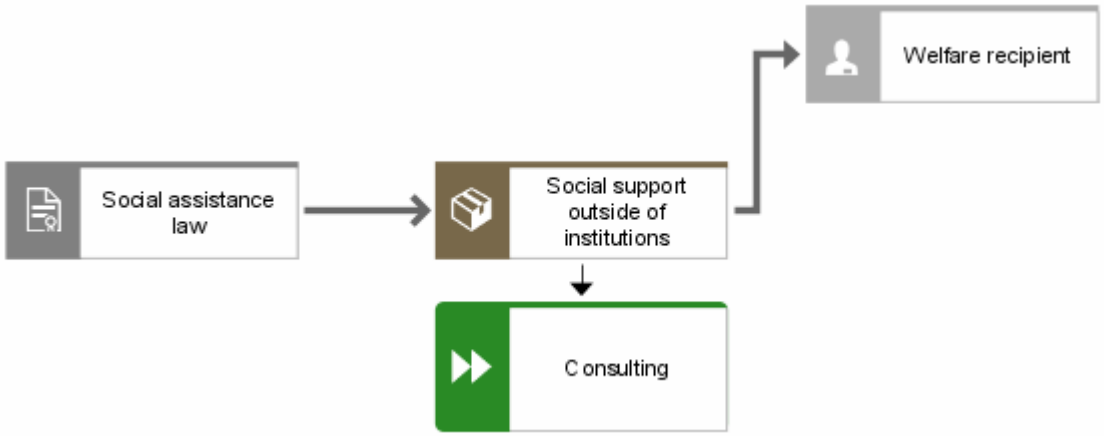


Product/Service tree

3.5.3. Product allocation diagram

Besides the general product/service diagrams, which belong to the graphic models, product models are recommended for realizing abstract representations. The product allocation diagram is primarily used to analyze product creation in public administration. Like the product/service exchange diagram, this model type can be used to show which organizational units provide or use which products, and which functions are required for the creation of the products, or for which functions the products provide an input. In addition, the (legal) order basis of each product is shown here. The objectives to be achieved with the various products can be represented as well.

The following figure shows part of a product allocation diagram for public authorities.



Product allocation diagram (1)

Finally, this model type can be used to describe aspects relating to product marketing.

In the following, a simplified example of banking products describes these aspects.

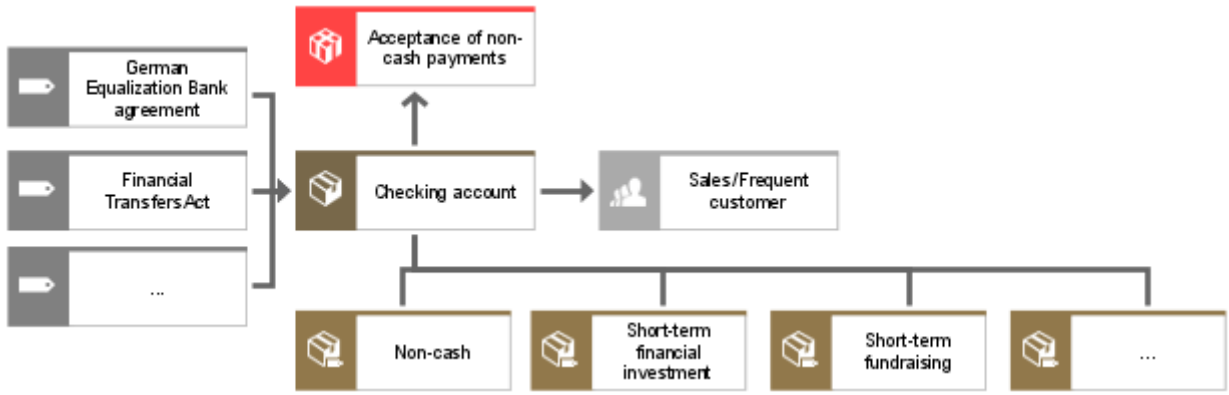
The growth of the Internet and the rising number of private Internet users over the past decades has been accompanied by the spread of online banking. At the same time, the financial power of adolescents has increased, making them more important as a target group.

As a result, the **checking account** service is now being offered in different forms:

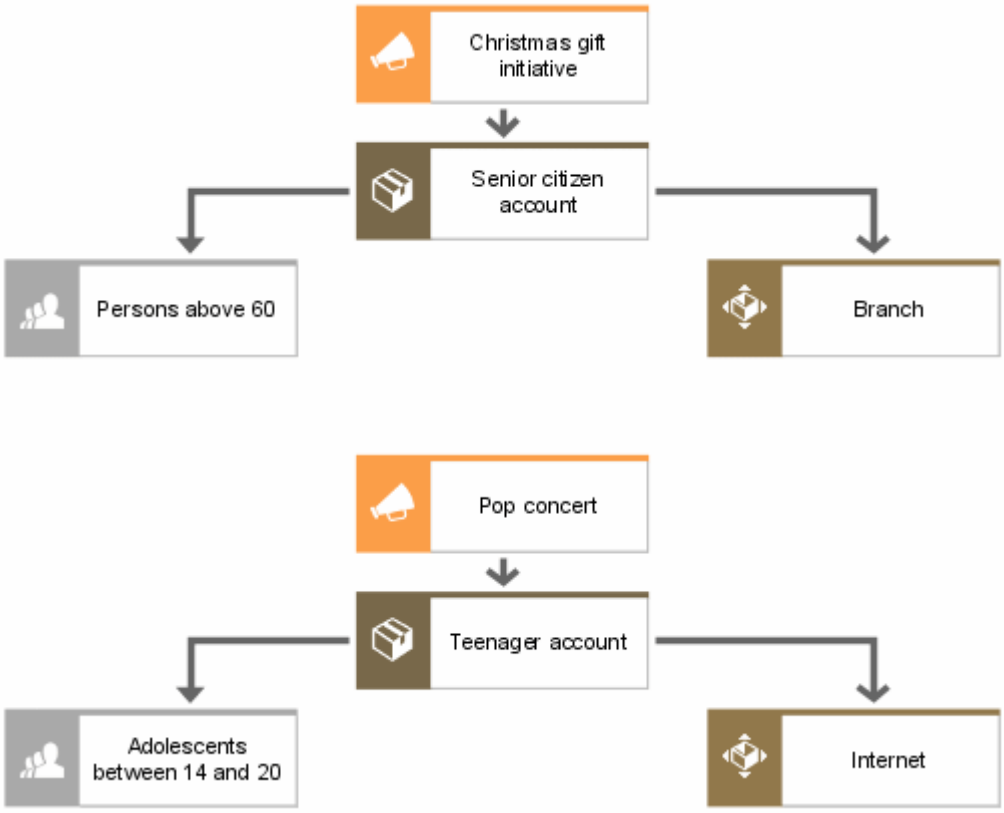
For example, it can be offered as a 'senior citizen account', with the holder being supported by the staff at a branch of the bank. This product is geared particularly to older customers who are less familiar with the new technologies, attach importance to personal support and advice from people they know, and whose mobility is restricted due to their age. The fees charged for such an account may be above average.

Another variant of a checking account may be a low-fee online 'teenager account'. This product is aimed at adolescents aged 12 to 20 who are familiar with Internet technology, but have a lower budget. The fees should therefore be at the lower end of the range.

The following figures show product allocation diagrams for these two product variants:



Product allocation diagram (2)



Product allocation diagram (3)

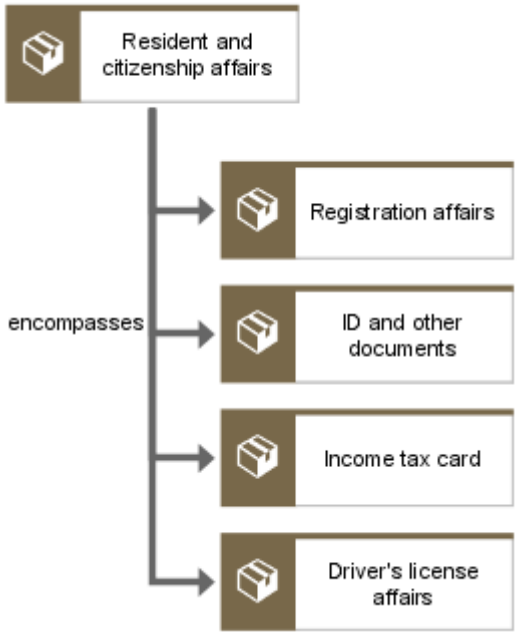
The **Teenager account** and **Senior citizen account** services have been created as object variants of the checking account and are identified by the **Sales product** attribute. A sales product is a product or service rendered by a company. It is offered under different names in different market segments. Generally, different marketing instruments are used for different sales products.

The ARIS variants component can be used to develop any number of sales products from a given product.

3.5.4. Product tree

The purpose of the product tree is to analyze the composition of products in public administration. This model essentially corresponds to the product/service tree, but does not provide the option of modeling replacement products. The product tree is located at the requirements definition level of the product/service view.

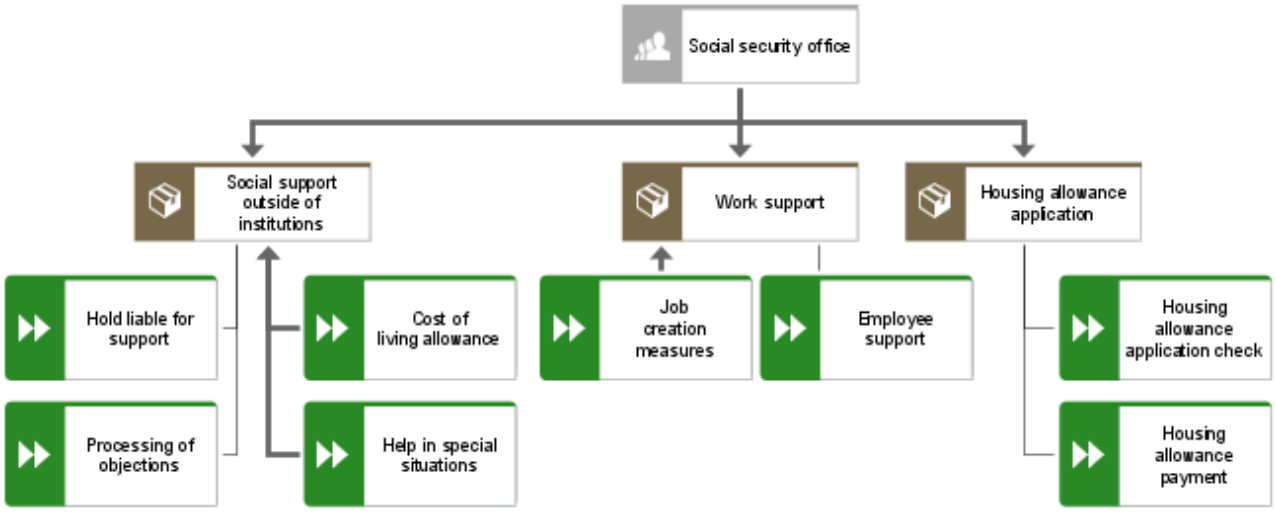
The following figure illustrates an example of a product tree.



Product tree

3.5.5. Product selection matrix

In the product selection matrix, the focus is on an organizational unit and the products within its responsibility. The functions required for the products' creation can be allocated to the products. The model is suitable as a starting point enabling navigation to organizational charts, product trees, and processes relevant to the creation of products. The following figure shows an example of a product selection matrix.



Product selection matrix

4. Unified Modeling Language (UML) in ARIS

4.1. Introduction

UML (Unified Modeling Language) is an object-oriented modeling language whose language constructs are standardized by a working group of the OMG (Object Management Group). UML is based on the object-oriented approaches of OMT, Booch, and OOSE.

4.2. ARIS UML Designer - Supported UML standard

ARIS UML Designer 9.x supports the entire UML standard 2.5.

The UML specification is available at **https://www.omg.org/spec/UML/2.5/Beta2/PDF/**.

5. Methods for knowledge management

5.1. Introduction

The purpose of knowledge management is the systematic control of knowledge, an increasingly important company resource. It encompasses development, monitoring, support, and improvement of strategies, processes, organizational structures, and technologies for effective knowledge processing within a company. This includes all activities relating to acquisition, preparation, transmission, and utilization of knowledge. These knowledge management activities generally do not occur in isolation; they occur primarily in the operational and scheduling business processes of the company. Hence, an integrated view of business processes, knowledge processing, organizational structures, information systems, etc. is needed.

Most of these aspects can be depicted using established ARIS methods (for example, EPCs, organizational charts, function allocation diagrams, eERMs, etc.). However, accurate representation, analysis, and improvement of knowledge processing requires additional means of representation to identify and structure the content of relevant knowledge categories, to describe the distribution of knowledge within an organization, and to model knowledge creation and utilization in business processes.

For this reason, two new object types, **Knowledge category** and **Documented knowledge**, and two new model types, **Knowledge structure diagram** and **Knowledge map**, have been added. Furthermore, existing model types used for the representation of business processes (EPC, etc.) were extended to include constructs for handling knowledge creation and utilization. The new object and model types are methodically integrated into the main model types of the requirements definition (for example, eERM, organizational chart, function tree), thus ensuring an integrated perspective. For example, this would enable models from a business process optimization project to be used for the purpose of analyzing and improving knowledge processing.

The knowledge structure diagram is located in the data view of the requirements definition. The knowledge map, like the extended model types for business process modeling, belongs to the control view of the requirements definition.

5.2. Object types for modeling knowledge processing

5.2.1. Knowledge category

The **Knowledge category** object type, represented by an oval thought bubble (see figure Knowledge structure diagram [103]), illustrates an object with content referring to specific knowledge. Examples of **knowledge categories** include project management knowledge, specific industry knowledge, specific technology knowledge, customer and competitor knowledge, etc. These categories assist in classifying a company's existing or required knowledge.

Knowledge assigned to a particular knowledge category can be either implicit knowledge, that is, knowledge that cannot be fully documented as it is available in the form of employee or group skills, or explicit knowledge that can be documented in the form of descriptions or technical drawings. Knowledge categories often contain both. For example, project management knowledge can include project managers' experiences on the one hand and information provided in a project management manual on the other.

In addition to general attributes like Description, Remark, Source, etc., the following specific attributes serve to describe knowledge categories in more detail:

Attribute name	Value range	Description/Example
Updating frequency	Enumeration type: hourly, daily, weekly, monthly, annually, seldom, never	The updating frequency describes how often the knowledge of the relevant category must be refreshed to be up-to-date. For example, basic trigonometry knowledge needs to be updated rarely or, for practical purposes, never, whereas knowledge of certain stock prices must be updated daily or even hourly.
Significance	Percentage: 0..100	The significance of the knowledge category for the company can range from 0% (totally unimportant) to 100% (extremely important).
Degree of coverage	Percentage: 0..100	The current degree of coverage for the relevant knowledge in the company can range from 0% (not covered at all) to 100% (maximum possible coverage). If the degree of coverage of a knowledge category is to be represented by a particular organizational unit or person, the corresponding attribute of the **has at disposal** connection type can be used to specify this in a knowledge map.
Knowlegde advantage	Percentage: 0..100	The relative advantage of your company over the competition in terms of knowledge can range from 0% (the competition has the greatest possible advantage over your company) to 100% (your company has the greatest possible advantage over the competition).
Knowledge usage	Percentage: 0..100	The degree of utilization of a particular knowledge category can range from 0% (relevant knowledge is not utilized at all) to 100% (optimal utilization of relevant knowledge).
Desired degree of coverage	Percentage: 0..100	The desired degree of coverage for relevant knowledge can range from 0% (not covered at all) to 100% (maximum possible degree of coverage).
Future significance	Enumeration type: sharply falling, falling, stable, rising, sharply rising	**Future significance** depicts the expected tendency of a knowledge category to change in significance for the company.
Structural change speed	Percentage: 0..100	The structural change speed is a measure of how fast the methods applied to acquire relevant knowledge must change (0%: no change, 100% maximum change speed).

These attributes are used to assess the significance of the relevant **knowledge category** for the company. They can therefore serve as the basis for identifying important or urgent measures aimed at improving the company's knowledge management. It is often helpful to display such values graphically. Copying and pasting the values from the **Attributes** window into a table calculation program that can create the desired models is a simple way to do so. For example, it is possible to compare the current and desired **degree of coverage** in a bar chart for the **knowledge categories** under consideration.

5.2.2. Documented knowledge

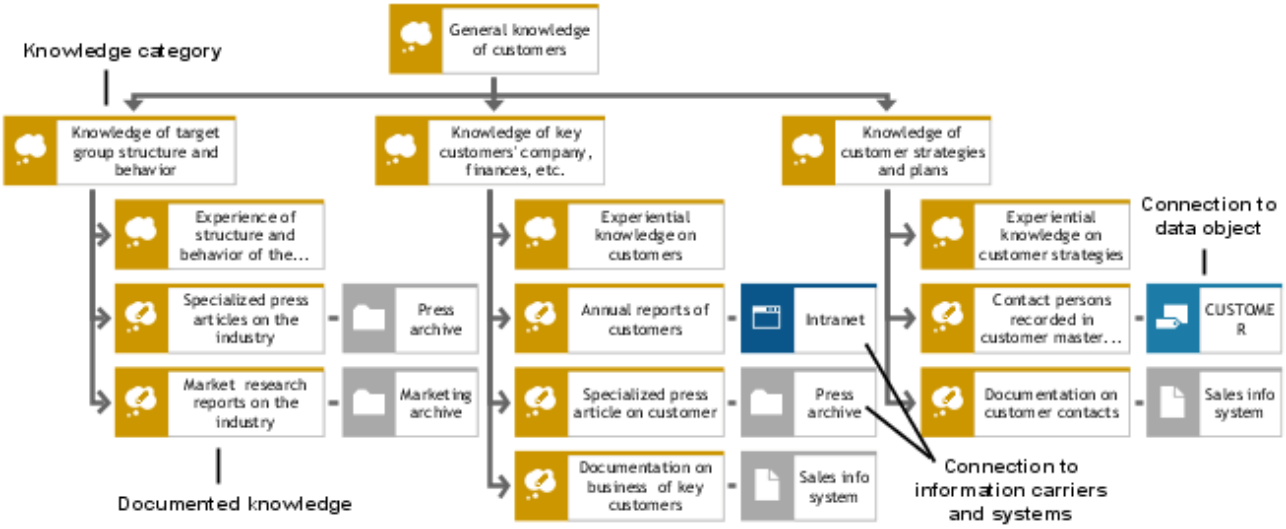
Unlike the **Knowledge category** object type, which can include implicit and explicit knowledge, the **Documented knowledge** object type relates exclusively to knowledge categories that are explicitly documented, or are, in principle, capable of being documented. An example of this type of knowledge is knowledge on using software that is documented in a manual. When assigning knowledge to knowledge categories, differentiating between general knowledge categories and documented knowledge helps to identify the possibilities and limitations of information system support for knowledge processing, as only documented knowledge can be electronically stored, transmitted, and processed.

The **Documented knowledge** object type is represented by a rectangular thought bubble. It contains the same specific attribute types as the Knowledge category [101] object type.

5.3. Model types for modeling knowledge processing

5.3.1. Knowledge structure diagram

Using a knowledge structure diagram, knowledge categories can be broken down based on their content. An example of this is shown in the following figure. A knowledge category may encompass other knowledge categories as well as documented knowledge. Documented knowledge can also be divided into several documented knowledge subcategories. However, it cannot encompass any general knowledge categories.

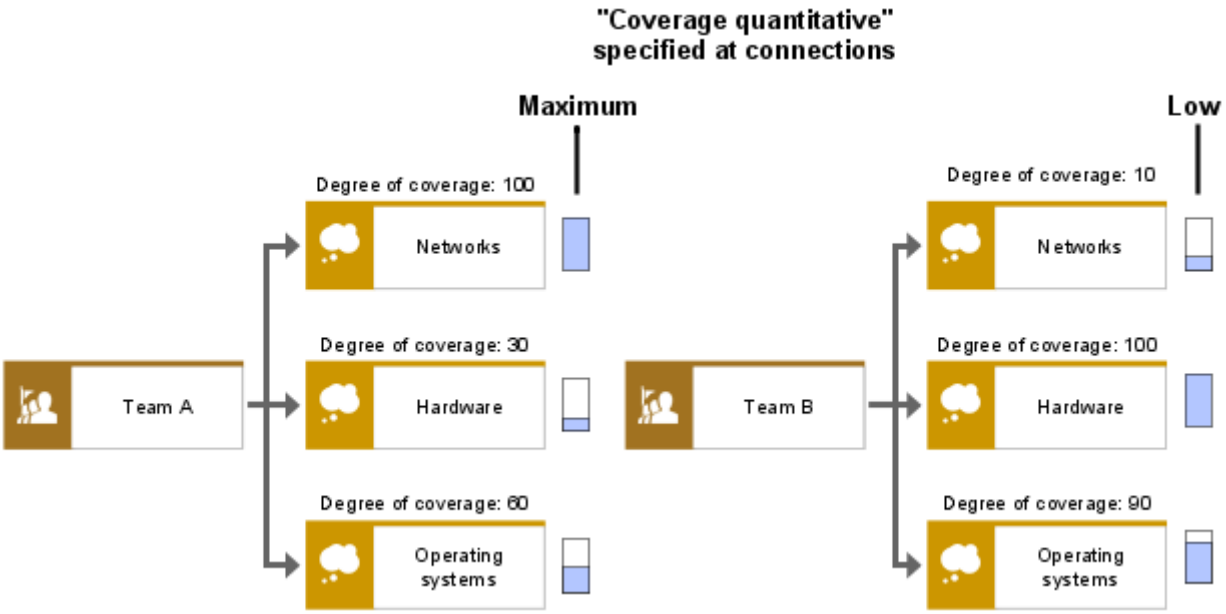


Knowledge structure diagram

For documented knowledge, a knowledge structure diagram can show which information carrier stores the knowledge, or which information objects of a data model or which classes of an object-oriented system are used for knowledge documentation purposes. Finally, the types or classes of application systems that are used to manage the knowledge can also be modeled.

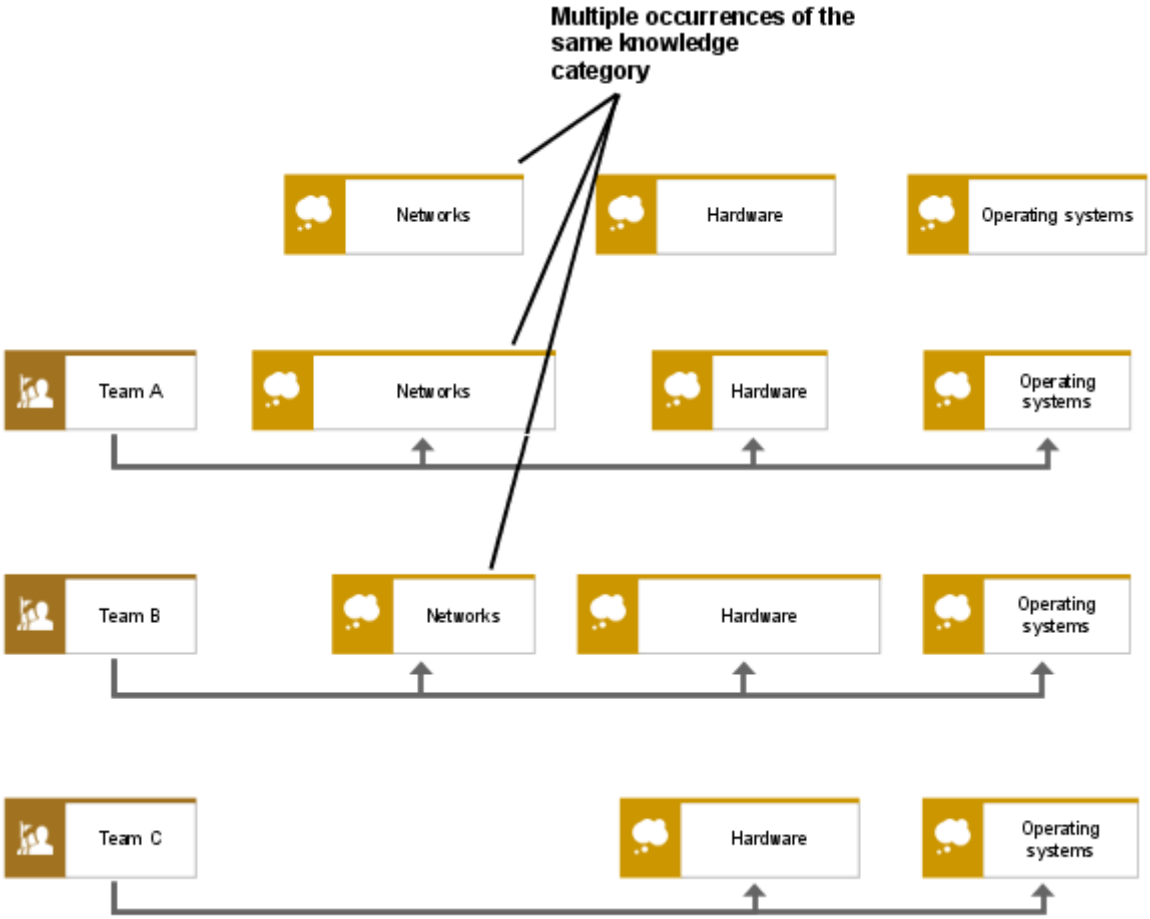
5.3.2. Knowledge map

A knowledge map depicts the organizational distribution of knowledge categories. Various object types of the organization view (for example, Organizational unit, Position, Person, Location, Group) can be connected to knowledge categories using **has at disposal** connections. In addition to the fact that a particular person or organizational unit has knowledge in a particular category, the degree of coverage can also be specified. The **has at disposal** connection contains the **Degree of coverage** attribute, which can express the degree of knowledge coverage in the selected category for the relevant organizational unit as a percentage. A value of 100% stands for maximum coverage, while a value of 0% means that absolutely no knowledge exists for the category. This is equivalent to the absence of the above-mentioned connection. In addition to this quantitative assessment, a qualitative assessment that can be made that can be displayed in the form of a graph. This is what the **Coverage quality** connection attribute is for, which can have the values **Low**, **Average**, **High**, and **Maximum**. This information can be visualized by graphic symbols at the connections as shown in the following figure. There is no direct relation between the values of the **Degree of coverage** and **Coverage quality** attributes. If both attributes are used, it is advisable that the qualification **Low** be used for a degree of coverage of up to 25%, **Average** for 26-50%, **High** for 51-75%, and **Maximum** for 76-100%.



Knowledge map - Relating to organizational units

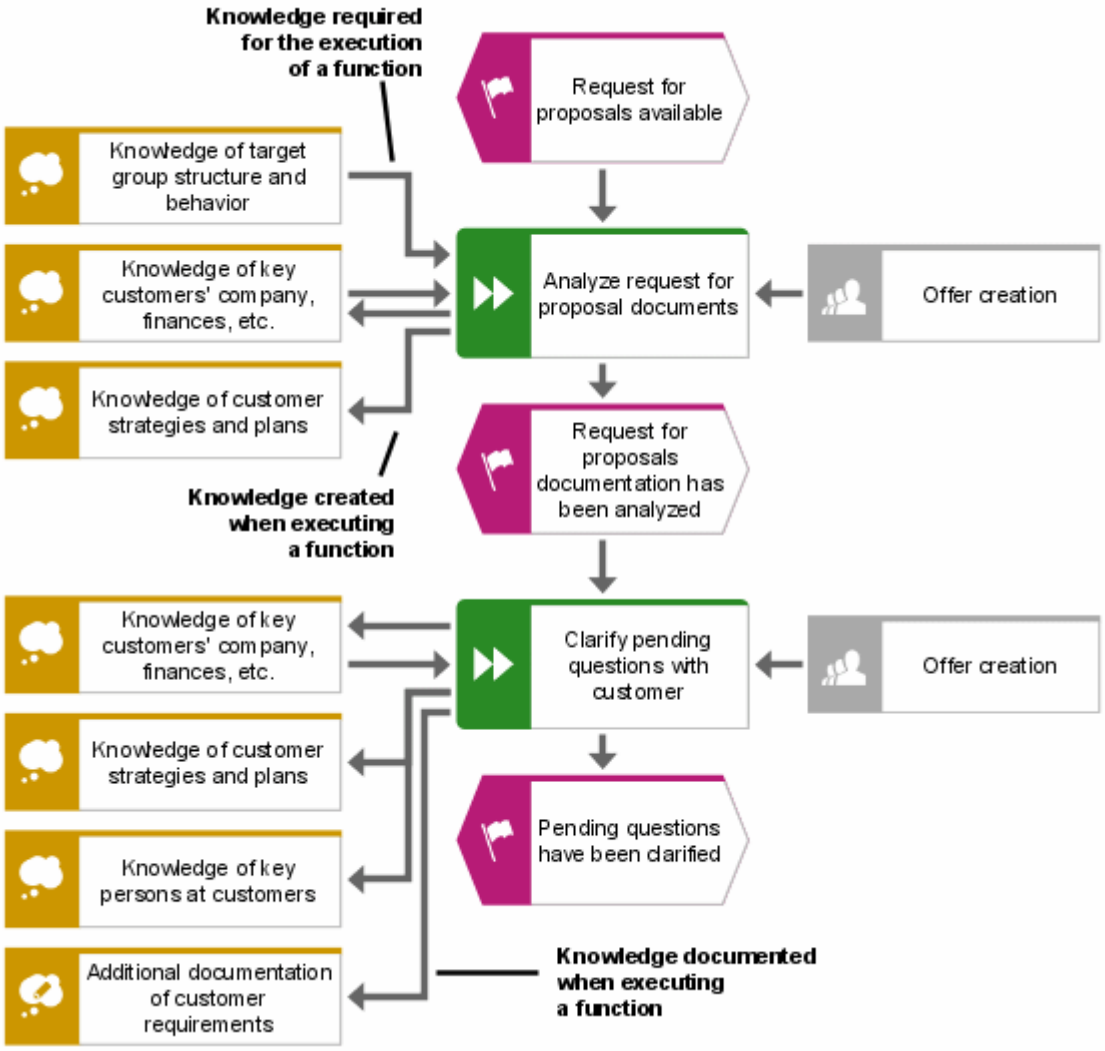
The knowledge map shown in the following figure focuses on organizational units, that is, all relevant **knowledge categories** are specified for each organizational unit. You can also start with **knowledge categories** as central objects and add all relevant organizational units to them. The navigation options in ARIS (**Relationships** tab in the **Properties - Object** dialog) make it easy to find other existing relationships of an organizational unit or a knowledge category in both cases. Knowledge maps are often represented in the form of a matrix. The matrix representation can be achieved by arranging several occurrences of the same knowledge category in column format as shown in the following figure. In this example, the names are visible only for the **knowledge categories** displayed at the top, which look similar to the header of a table. For the other occurrences, the names were removed using the attribute placement function. This figure also shows an alternative visual representation option for differing degrees of coverage: the **knowledge categories** are scaled in different sizes.



Knowledge map - Matrix representation

5.3.3. Representation of knowledge processing in business processes

The utilization and creation of knowledge in the company's business processes is modeled using the model types available for representing business processes (EPC, EPC (material flow)). The **Knowledge category** and **Documented knowledge** object types are now available in these model types. It is possible to specify for a function which kind of knowledge (general or documented) is required for its execution and which knowledge is created and/or documented during its execution. This type of representation enables business processes to be examined in terms of the knowledge processing involved. For example, gaps in required knowledge can be revealed. Besides, the qualification profile needed to carry out a function can be determined.



Knowledge processing in an EPC

6. Balanced Scorecard method

6.1. Introduction

If companies want to survive in the turbulent business environment and global competition of today's markets, they need the best possible business processes. However, it is just as important for them to be able to react to new developments in the business environment, quickly and in line with the strategic business objectives. This requires efficient management processes that enable consistent implementation of corporate strategies and achievement of strategic objectives in day-to-day business by means of operational initiatives.

Many traditional management approaches do not establish the connection between the formulation of corporate strategies and their implementation based on strategy-oriented initiatives, or consistent control of the achievement of strategic objectives.

In addition, many companies are still run purely on the basis of financial KPIs which are of limited use since they relate mainly to past performance and thus cannot provide much information relevant to future management. Only by looking at the reasons for financial success (customers, processes, innovation, etc.) it will be possible, at an early stage, to identify factors that might prevent the achievement of strategic objectives.

The Balanced Scorecard approach offers a methodology with a structure that is easy to understand and implement, and that helps to avoid weak points.

6.2. The Balanced Scorecard concept

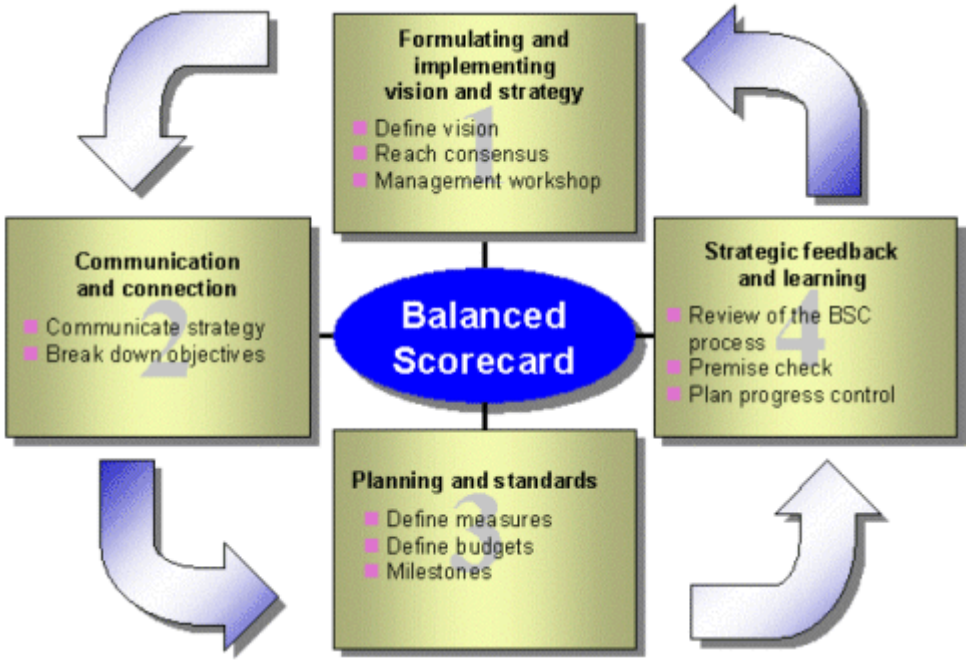
6.2.1. Key elements of the BSC approach

The Balanced Scorecard approach is a strategic management system first proposed by Robert Kaplan and David Norton in 1992 ('The Balanced Scorecard - Measures that drive Performance', Harvard Business Review, January/February 1992). It was developed from the results of research on the subject of **Approaches to performance measurement**. This research found that performance measurement systems geared exclusively to financial KPIs are likely to impede value-adding activities in many companies. Built on these findings, Kaplan and Norton set out to collaborate with innovative businesses to create a system of KPIs that would make it possible to optimally measure the realization of a company's visions and strategies.

The Balanced Scorecard approach associates KPIs with various views of the company (so-called perspectives). These include internal performance perspectives (for example, learning and growth perspective, process perspective) and external performance perspectives (for example, customer perspective, economic/financial perspective). Due to this arrangement of KPIs, a certain balance between short-term and long-term goals, financial and non-financial KPIs, leading and lagging indicators, and internal and external views can be achieved. The integration of industry-specific KPIs adds a further benchmarking component to the concept.

The pure performance measurement approach has evolved into a comprehensive management system for goal-oriented corporate management, starting from corporate vision and individual competitive strategies to the formulation and control of initiatives using balanced KPIs. Therefore, the Balanced Scorecard approach is more than just a system of performance measurement KPIs. It assists companies in communicating and implementing their corporate strategy and supports the resulting strategic learning process (double-loop learning).

6.2.2. Strategic management process and Balanced Scorecard



BSC: Strategic management process

The strategic management process in the context of the Balanced Scorecard approach comprises two phases:

1. In the first phase, the company's strategy must be established on the basis of strategic analysis. Within the scope of this analysis, all data relating to competition are being recorded. The aim of this analysis is to identify and assess trends, opportunities, and risks pertaining to the development of the business environment, and to determine the company's own core competencies. At the end of this phase, the company's individual corporate strategy is defined.
2. The implementation of the corporate strategy takes place in the second phase. In this phase, the BSC method can help. The corporate strategy may be refined with partial strategies (for example, business segment-specific strategies). This leads to further strategic objectives. These are concretized on the basis of targets for particular measurement categories. An action program in the form of instructions on how to proceed determines how these objectives are to be achieved. Within the scope of business planning, this action program is broken down into various corporate divisions or departments. Budgeting can be used to put actions into more concrete terms. With the help of individual strategy-related scorecards, the achievement of objectives can be measured by the KPIs generated. Based on the scorecards defined, a review process takes place during which further initiatives are specified or the strategy is 'reworked' due to possible deviations.

6.2.2.1. Formulation and realization of vision and strategy

The strategy of a company results from its vision.

The vision represents a superior mission statement for the company under consideration; as a general, but striking statement it is vague, yet serves as the prime principle of action and as such expresses a sort of short version of the company's philosophy.

Individual strategies are determined for different strategic business units. These strategies must be geared to the performance goals of the company. Therefore, it is necessary that prior to introducing a BSC,

senior management holds a workshop to determine the vision and resulting strategies for the strategic business units. In most cases, this involves setting a financial target (financial perspective). The focus may vary: Possible objectives are particular values for return on capital employed, ROI, shareholder value, sales revenue, or cash flow. The financial objectives need to be realized through specific behavior on the relevant market (customer perspective). Therefore, after defining the objectives, appropriate market and customer segments are selected. This customer perspective also involves the definition of strategic objectives and the identification of relevant KPIs. For example, they may relate to market shares or growth rates in a given customer segment.

Appropriate company resources are required to implement specific market-oriented strategies. When establishing a Balanced Scorecard, resources are divided into two categories as follows:

1. Once the financial and customer objectives are defined, the main business processes (process perspective) are looked at, objectives determined, and initiatives and KPIs generated. The focus is usually on process times and costs.
2. As part of the so-called learning and growth perspective, strategic objectives for human resources development, information technology, and innovation are derived from the strategic objectives of the financial, customer, and process perspectives.

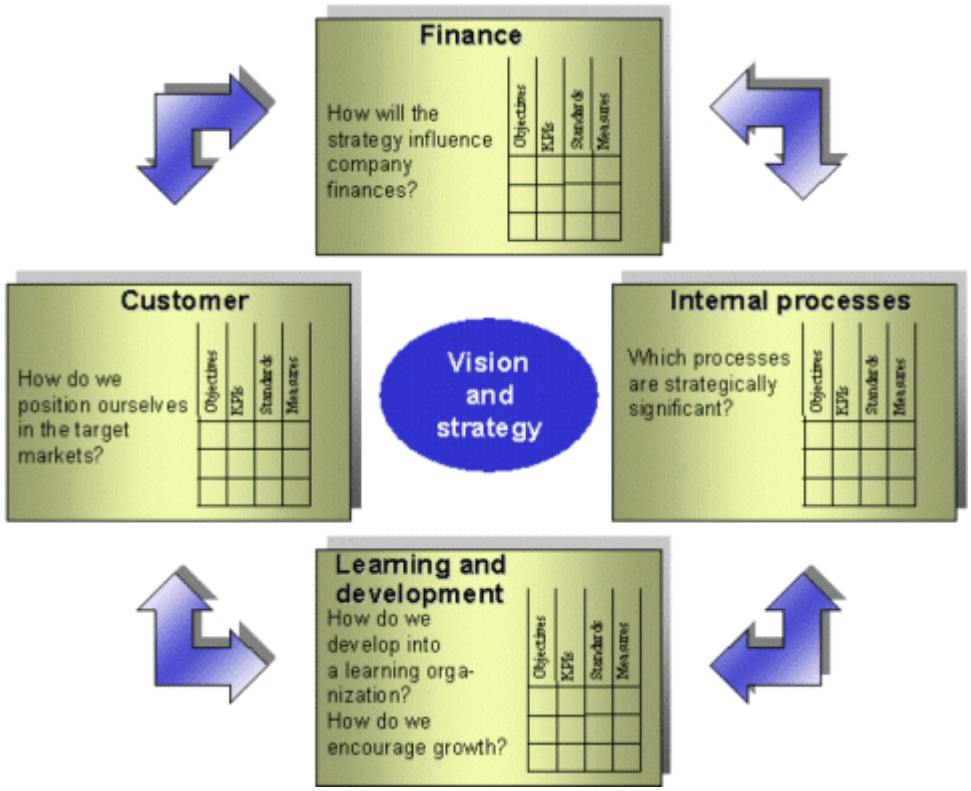
When setting up a Balanced Scorecard, all strategic objectives defined are mutually interactive, which is known as the cause-and-effect chain.

For the first step to be effective it is important that a general consensus on the vision, strategies, and resulting strategic objectives be reached at the management level.

6.2.2.2. Standard perspectives of a Balanced Scorecard

Kaplan and Norton propose four standard perspectives for the structure of a Balanced Scorecard:

1. Financial perspective - Shareholder expectations: 'What effect on finances does the strategy have?'
2. Customer perspective - Customer expectations: 'How do we position ourselves in the target markets?'
3. Process perspective - Process requirements: 'Which processes are of strategic importance?'
4. Learning perspective - Requirements of organizational learning and innovation: 'How can we develop into a learning organization? How can we promote growth?'



BSC: Perspectives

The standard perspectives have an implicit logic that helps implement strategy and formulate specific cause-and-effect relationships. However, in principle, it is also possible to define other perspectives for a company (for example, environment perspective), which may be of relevance to the corporate strategy, or whose objectives and KPIs have a direct relation to the standard perspectives.

6.2.2.3. Cause-and-effect chain

The cause-and-effect chain defines the cause-and-effect relationships between individual objectives of a strategy, that is, across the defined perspectives of the KPIs. In other words, it describes how the objectives of the perspectives can be achieved. The perspectives themselves form the general conditions for the cause-and-effect chain ('force field of the company'). It can be assumed that improvements in the learning and growth perspective environment have a direct and positive effect on the objectives and KPIs of the internal process perspectives. Furthermore, developments in the process perspective also have a positive impact on the objectives and KPIs of the customer perspective, which in turn leads to an improvement in the financial objectives.

Within the scope of developing a Balanced Scorecard system that is to be valid throughout the company, objectives and KPIs are defined for each strategy, and the relationships between them are represented within and across the perspectives. Based on the objectives and KPIs specified in the financial perspective, corresponding 'positive agents' (performance drivers) are determined in the customer perspective. In a next step, these are broken down to the learning and growth perspective via the internal process perspective. This procedure allows strategies of a company or division to be split into subobjectives and KPIs, and corresponding operational initiatives to be derived from them.

By establishing a cause-and-effect chain and defining the KPIs, a balance is implicitly created between leading and lagging indicators.

6.2.2.4. Definition of leading and lagging indicators

The establishment of cause-and-effect chains also produces a temporal connection between individual objectives and KPIs. Generally, performance drivers - so-called leading indicators - are part of the **learning and growth perspective** and the **process perspective** and come before the effects of the customer perspective and financial perspective. Most KPIs in the customer and financial perspectives are result indicators that measure business success; as so-called lagging indicators they focus on the past.

Both leading and lagging indicators should be defined in every Balanced Scorecard perspective so that the interrelationships between initiatives and the achievement of objectives can be represented. The leading indicators enable you to detect deviations from given objectives at an early stage.

6.2.2.5. Communication and derivation of further scorecards

The effectiveness of a corporate strategy and its realization using the Balanced Scorecard concept depends on the corporate scorecard's recognition and acceptance throughout the company. For this reason it is vital that corporate strategies and the corporate scorecard are promoted at all hierarchy levels of the company through a comprehensive communication campaign.

In a top-down approach, targets are derived from the superior corporate strategy (corporate scorecard) for the subordinate hierarchy levels of the company, and superior strategic objectives are adapted to match specific departmental goals. KPIs measuring the degree of goal accomplishment are generated for these objectives, too. Various initiatives are required in the different corporate divisions in order to achieve the objectives, and these initiatives are recorded in the scorecards of the underlying hierarchy levels. All targets must be based on long-term strategic considerations reaching beyond short-term results that may be inconsistent with the overall strategy. An example of a short-term result is a reduction in costs based on a short-term increase in the level of production.

6.2.2.6. Planning and targets

The BSC can be used to combine the process of strategy implementation at the various corporate hierarchy levels with the budgeting process. The purpose of this combination is to align all resources with the corporate strategy.

The BSC is integrated in four steps into the long-term strategic planning and budgeting process.

1. Ambitious targets are set that are communicated to and accepted by the employees. The relevant KPIs relating to the results in the client and financial perspectives are identified by the cause-and-effect relationships between the KPIs.
2. Strategic initiatives must set out exactly at the KPIs with the greatest discrepancy between current value and target value. In the long term, this procedure aligns capital investments and action programs with strategically significant target values.
3. Critical, company-wide initiatives are identified to create synergies with the strategic objectives.
4. Based on budgets of varying maturities (5, 3, and 1-year plan), KPI plan values are linked to business planning.

6.2.2.7. Strategic learning and feedback

The Balanced Scorecard concept contributes to strategic learning in the company and offers the possibility to give feedback on strategy realization.

Strategic learning is aimed at every single employee and at the entire company as a learning organization. Employees should be acquainted with the strategy and align their actions with it. During a defined feedback process, performance data is gathered to assess whether the strategy has been realized. Assumptions are checked by verifying previously established hypotheses on cause-and-effect relationships of strategic objectives.

This is how double-loop learning comes into being, where strategic targets are subjected to critical review in a bottom-up process, after which new cause-and-effect relationships are generated. These result in a more efficient pursuit of strategy due to a continuously growing information base.

Feedback is therefore extremely important as it represents the counterprocess to breaking down BSCs. Feedback comprises not only the reporting of KPIs in the form of plan progress figures, but also suggestions for further initiatives and new interrelations. It reveals potential inconsistencies in the pursuit of strategy.

6.2.3. Advantages and benefits of the Balanced Scorecard

The advantage of the Balanced Scorecard method lies in its consistent, strategy-oriented corporate management, which includes the lowest levels of a company, where strategy-based operational initiatives are derived. All resources and all employees of a company are aligned with the corporate strategy. The strategy is consistently communicated and monitored through Balanced Scorecards.

The Balanced Scorecard method overcomes the disadvantages of traditional KPI systems in corporate management:

- No limitation to financial KPIs
- No pure focus on the past
- Qualitative data also considered
- Reduction of complexity, consensus building
- Strategic objectives as the starting point of the Balanced Scorecard approach
- Focus on management bottlenecks, not on existing data
- Acceleration of required changes

The Balanced Scorecard method is ideal for strategy-oriented corporate management as it overcomes the following barriers:

- Concretization barrier: In traditional management approaches, vision and strategy are theoretical formulations; the derivation of specific (operational) action plans is not consistently pursued.
- Vision barrier: The strategy is often not understood by those employees who have to implement it.
- Commitment barrier: Strategies are not associated with specific departmental or individual objectives.
- Implementation barrier: Reporting is geared to operational-monetary objectives, rather than strategic objectives.
- Operational barrier: The budgeting process is separated from the strategic planning process.

6.3. Developing a Balanced Scorecard with ARIS BSC

6.3.1. Terms and abbreviations

Vision: A vision indicates the future strategic thrust and mission of a company and is realized through various strategies.

Strategy: Strategies are developed based on the vision. Strategies can be split into specific substrategies.

Strategic objective: Each strategy consists of a certain number of strategic objectives. Generally, management determines strategic objectives in workshops. Objectives are interconnected according to cause-and-effect relationships and chronology.

Success factor: A success factor influences the success of a company's business operations. It may interact with other success factors and direct them towards a specific behavior with a particular degree of effectiveness.

View/Perspective: A perspective puts a certain view of the company in more concrete terms. In principle, the selection of perspectives should include the following views of a company: human-oriented view, internal view, process-oriented view, and external view.

KPI: Each strategic objective or success factor is allocated KPIs to measure performance and achievement of objectives.

Actual value: Current value of a KPI, strategic objective, or success factor. Based on the actual value, the current degree of goal accomplishment can be derived by means of a planned-actual comparison.

Plan value: Plan value of a KPI, strategic objective, or success factor set as the target for a particular period. In the case of strategic objectives, this is usually a percentage derived from the weighted achievement of objectives of individual KPIs. A plan value can be broken down into individual periods and takes account of periodic fluctuations. Plan values represent a degree of goal accomplishment of 100 %.

Target value: Value set as the target for a period in the future. Generally, this value becomes the plan value once the target period is reached.

Minimum value: Smallest possible value that an objective, critical success factor, or KPI can have. In ARIS, the minimum value is set to 0 by default. However, it can be modified.

Maximum value: Highest possible value (upper limit) a strategic objective, a success factor, or a KPI can have. In ARIS, it is generally used to standardize the representation of multiple KPIs so that they can be compared.

Warning limit: The warning limit corresponds to the plan value, that is, the plan value corresponds to the limit at which the value of a KPI, strategic objective, or success factor falls short of an expected target.

Tolerance range: The tolerance range is the negative deviation from the plan value for a strategic objective, KPI, or success factor that can still be accepted.

Alarm limit: The alarm limit denotes the plan value minus the tolerance range. All values of a strategic objective, KPI, or success factor that fall below the alarm limit are no longer acceptable.

Initiative: An initiative influences a single objective or a number of strategic objectives. Generally, initiatives are assigned a person responsible for them, and they have a starting point, an end point, resources, etc.

Indicator type: A KPI can be of the **Leading indicator** or **Lagging indicator** type. A leading indicator measures a performance driver and is an indicator pointing to the future. A lagging indicator measures a result (result indicator) and is retrospective. Economic/Financial targets are generally lagging indicators, while the targets

of process, learning, and growth perspectives are increasingly leading indicators. It is recommendable to include lagging indicators in the perspectives in order to represent cause-and-effect relationships between KPIs.

Data source: Each KPI has a data source from which it can be transferred to the Balanced Scorecard system.

6.3.2. Creating Balanced Scorecards with ARIS BSC

6.3.2.1. Specification of perspectives

At the start of a Balanced Scorecard project, the perspectives should be specified within the strategic plan. These perspectives are valid for all corporate scorecards.

The **Perspective** object type is available for this purpose. Perspectives can be modeled in the BSC cause-and-effect diagram.

Perspectives are linked via the **is influenced by** connection. There is no need to define a specific sequence. However, establishing a logical structure (with the individual perspectives arranged in a sequence) considerably simplifies automatic generation of a model. Furthermore, the logical structure of the cause-and-effect chain becomes more transparent.

6.3.2.2. Balanced Scorecard system structure specification

The following models provide the framework for creating a Balanced Scorecard system:

- Organizational chart

  A Balanced Scorecard system can be structured in line with the organizational structure of a company. Any number of BSC cause-and-effect diagrams can be assigned to the objects of the organizational chart (for example, for variant creation, history management, etc.). This assignment establishes a connection between the objectives required for strategy implementation and the corresponding organizational units. Based on the organizational chart, the Balanced Scorecard system of a company can be broken down from the highest corporate level to the level of individual departments or employees.
- Structuring model

  If a company prefers to create its Balanced Scorecard system based on strategic business segments rather than the organizational chart, the structuring model is available as a starting model for the balanced scorecard structure. In this case, any number of BSC cause-and-effect diagrams can be assigned to a structuring object.
- Value-added chain diagram or function tree

  As the Balanced Scorecard is meant to be an instrument for performance management and performance measurement, the Balanced Scorecard system can be established in ARIS with a view to value-added criteria. The value-added chain diagram or the function tree can be used for this purpose.

6.3.2.3. Cause-and-effect relationship specification

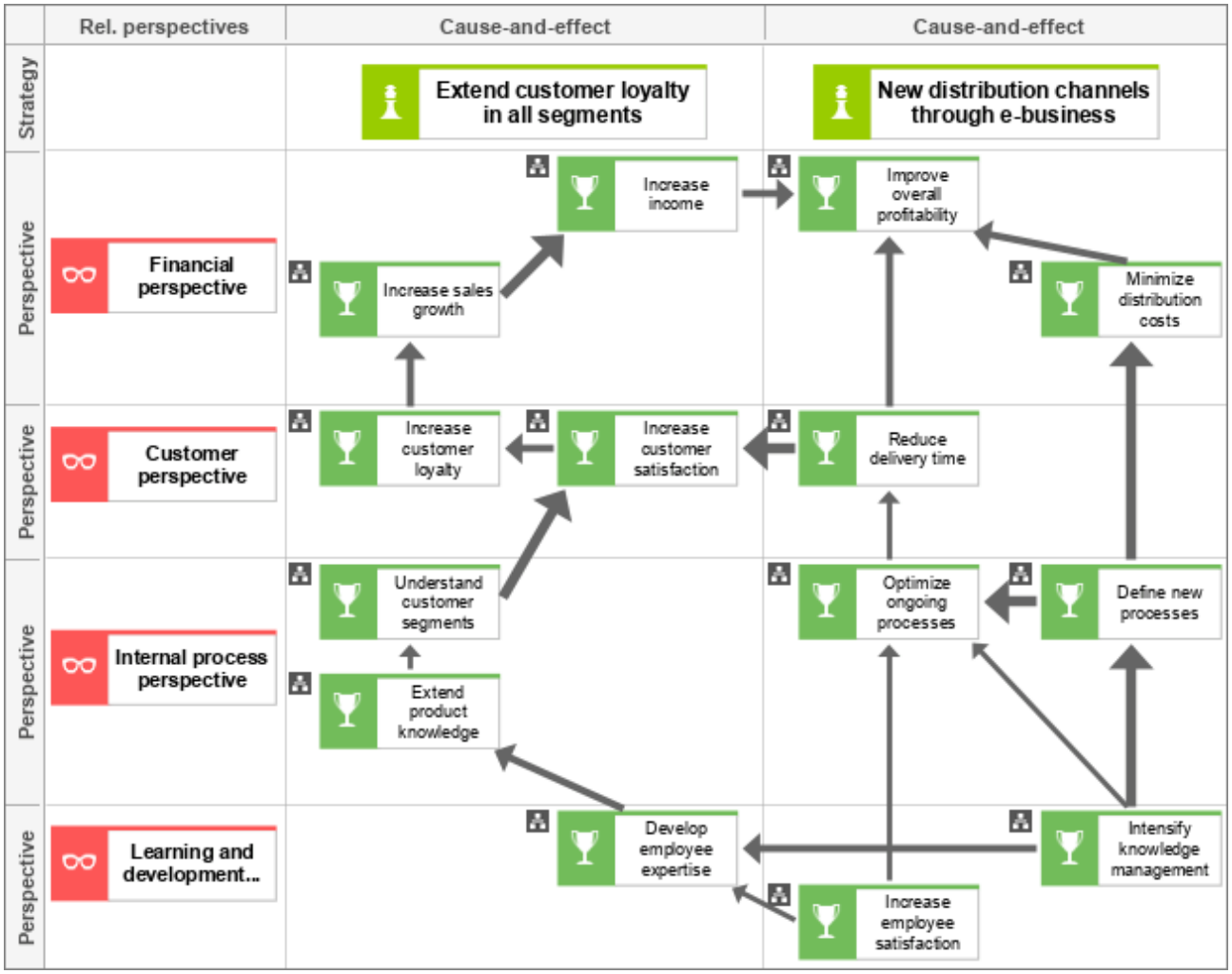
The objects of the organizational chart and the functions or structural elements of the structuring model, which determine the company-wide structure of the Balanced Scorecard, are assigned BSC cause-and-effect diagrams.

In the BSC cause-and-effect diagrams, the strategic objectives and success factors required for successful strategy implementation are defined, and their mutual influence is illustrated by means of the various Balanced Scorecard perspectives. First, the individual perspectives used in all cause-and-effect chains of the company are created as objects in ARIS.

The BSC cause-and-effect diagram is a lane model which allows the **Perspective** object types to be stored in the **Relevant perspectives** column and the **Strategy** object types to be stored in the **Strategy** row. The **Objective**, **Success factor**, and **KPI** object types can be modeled in the cause-and-effect columns. The degree of effectiveness of the strategic objectives and their success factors are described in the cause-and-effect column. The greater the effectiveness, the thicker the arrow that represents it.

Furthermore, success factors for a Balanced Scorecard can be defined in the BSC cause-and-effect diagram. By using the **Success factors** object type, you can include other factors on which the company has no direct influence and their KPIs in the Balanced Scorecard (for example, market growth).

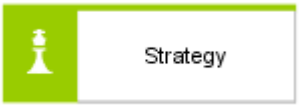
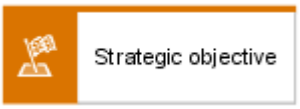
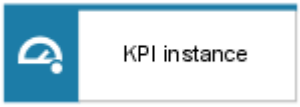
The BSC cause-and-effect diagram can be found at the requirements definition level of the control view.



BSC: Cause-and-effect diagram

The **Objective** and **Success factor** object types in the modeling column of the BSC cause-and-effect diagram can be assigned a BSC KPI allocation diagram only.

The following symbols are used in the BSC cause-and-effect diagram:

Symbol	Object type
Perspective	Perspective
 Strategy	Strategy
 Strategic objective	Objective
Success factor	Success factor
 KPI instance	KPI instance

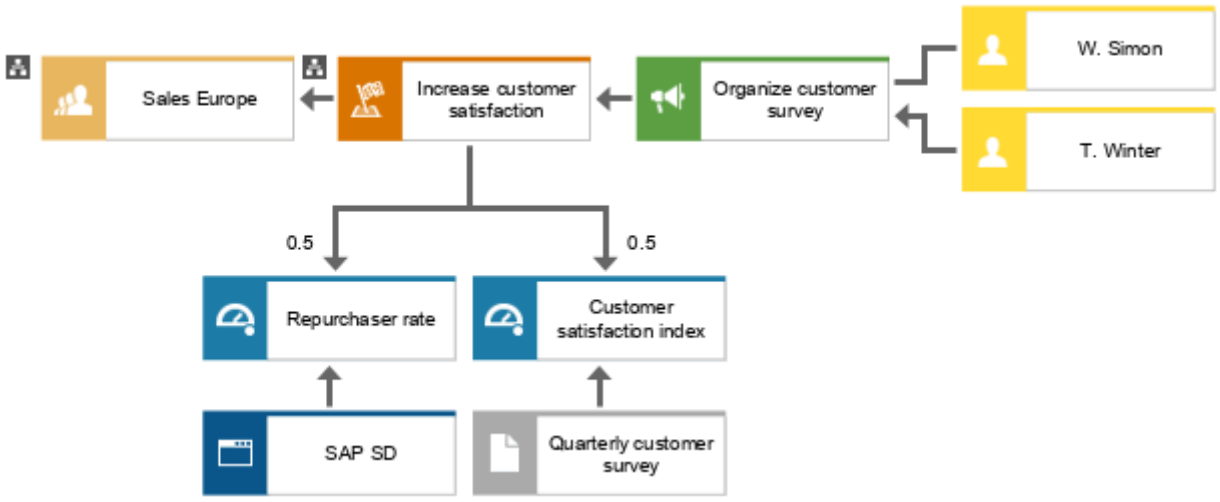
6.3.2.4. Specification of initiatives and KPIs to monitor objectives

After strategic objectives and success factors have been defined in the BSC cause-and-effect diagram, a BSC KPI allocation diagram is assigned to each of the individual objects. In the diagram, objectives and success factors are allocated KPIs to serve as a benchmark for the relevant objective or success factor. These KPIs are subsequently identified and channeled. If an object is allocated multiple KPIs, a connection of the **is measured by/measures** type can be used to specify the weighting of any KPI. This indicates how important the KPI is for the degree of goal accomplishment of the objective or success factor. For the analysis script of ARIS BSC to run successfully, the **Weighting** attribute must be specified for all connections of the **is measured by/measures** type.

Apart from KPIs, objectives and success factors in the BSC KPI allocation diagram may also be allocated objects to define the data source (for example, entity type, file, document, database, storage medium, etc.). This allocation establishes a link to the ARIS Data Warehouse method. It allows an accurate definition of which data elements in the Data Warehouse method each Balanced Scorecard KPI is connected with.

Since is possible to depict the interaction of KPIs using the **influences/is influenced by** connection type, the effects that leading indicators have on lagging indicators can be described in the BSC KPI allocation diagram. For this we recommend that an additional BSC KPI allocation diagram be designed, which does

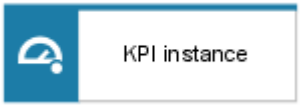
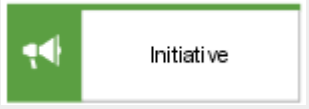
not focus on the allocation of individual KPIs to objectives or success factors, but on the consolidation and definition of KPIs used in other BSC KPI allocation diagrams and on the description of their effects.

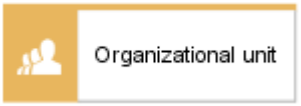
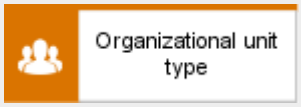
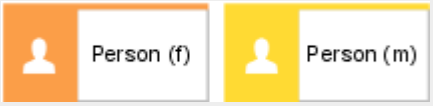
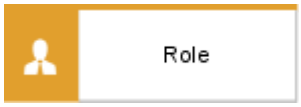
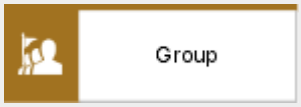


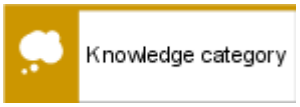
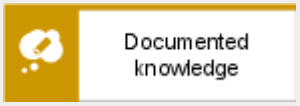
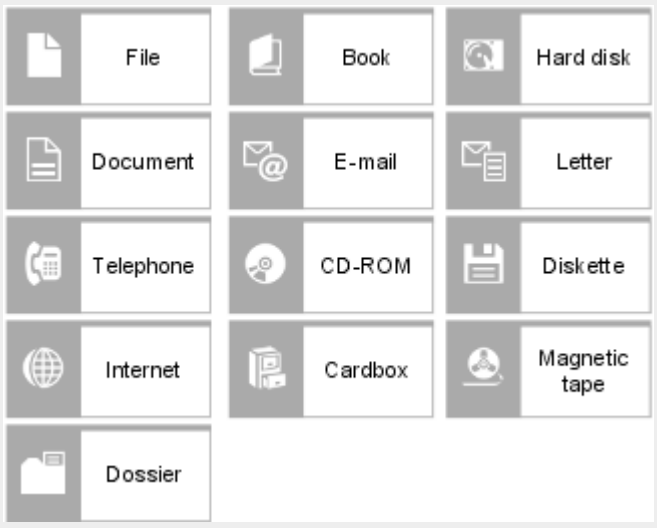
BSC: KPI allocation diagram

The BSC KPI allocation diagram can be found at the requirements definition level of the process view.

The following symbols are used in the BSC KPI allocation diagram:

Symbol	Object type
Strategic objective	Objective
Success factor	Success factor
 KPI instance	KPI instance
 Initiative	Task

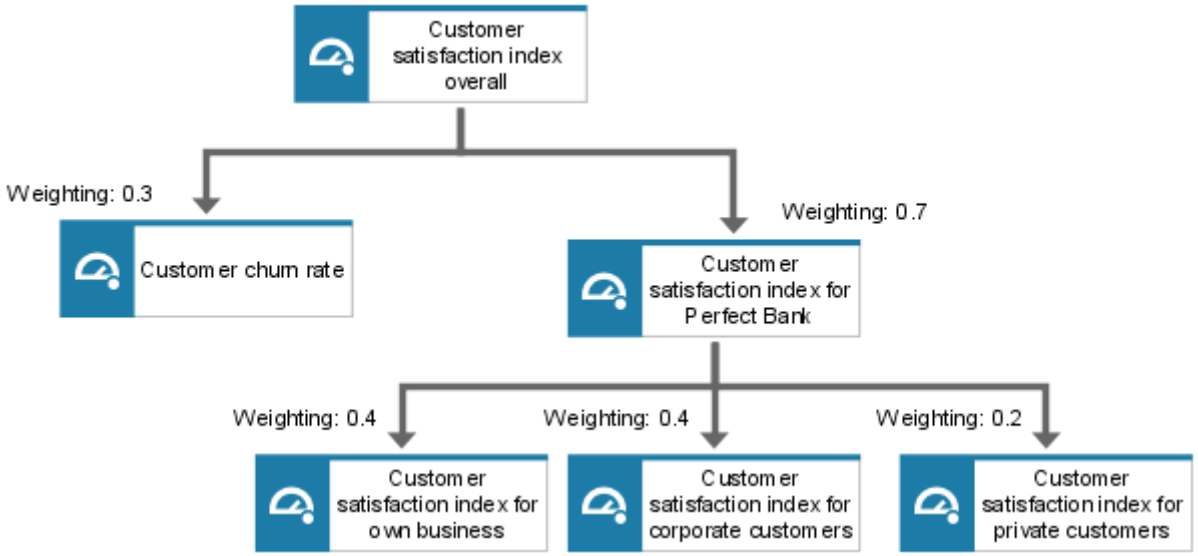
Symbol	Object type
 Organizational unit	Organizational unit
 Organizational unit type	Organizational unit type
Position	Position
 Person	Person
 Role	Role
 Group	Group
Computer	Application system type
Technical term	Technical term
Entity type	Entity type

Symbol	Object type
Relationship type	Relationship type
ERM attribute	ERM attribute
Cluster	Cluster/Data model
 Knowledge category	Knowledge category
 Documented knowledge	Documented knowledge
Class	Class
 Information carrier	Information carrier

6.3.2.5. Description of KPIs and their relationships

After KPIs for measuring strategic objectives and critical success factors have been selected, these indicators and their relationships can be defined in more detail using the **KPI tree** model type.

A KPI tree arranges various KPIs in a hierarchy by means of the **influences** connection type. The **Weighting** attribute can be specified for these connections, thus enabling the calculation of an overall KPI within a KPI tree with the help of weights. The KPI tree is assigned to the KPI instance representing the overall KPI. This assignment is included in the **Create BSC management view** and **Perform BSC planned-actual comparison** evaluations.



BSC: KPI tree

The **KPI tree** uses the **KPI instance** object type only.

The KPI tree can be found at the requirements definition level of the data view.

6.3.3. Relationships to other models

The objects of the Balanced Scorecard method can be associated with the following ARIS models:

- EPC
- Objective diagram
- DW structure

7. E-Business scenario diagram

7.1. Introduction

The smooth execution of inter-company business processes is increasingly gaining in significance. Here, the operational sequence of specific procedures at the interfaces between companies on the one hand and at the interfaces between companies and their customers on the other hand is in the center of interest. The contacts need to take place in a clear, quick, consistent, and direct manner.

Rapidly finding suitable business associates (from a corporate perspective) and providers (from a consumer point of view) is also of great relevance. Maximum optimization of these processes results in a competitive advantage. The ideal platform for supporting these multilateral relations is the Internet. As the processes in the above-mentioned environment are very complex, it is necessary to define the term 'e-business'.

E-business is a generic term for the use of information and communication technologies in support of a company's business activities. It includes the use of electronic media for the purpose of supporting relationships and processes involving business associates, employees, and customers (Herrmans, Sauter, 1999).

Thus, e-business can mean the creation of a website for a corporate presentation, the acquisition of an item via Internet, a highly complex project shared by two companies, or multilayered relationships between any number of business associates meeting in a marketplace.

It can be subdivided into the following concepts:

B2B (Business to Business)

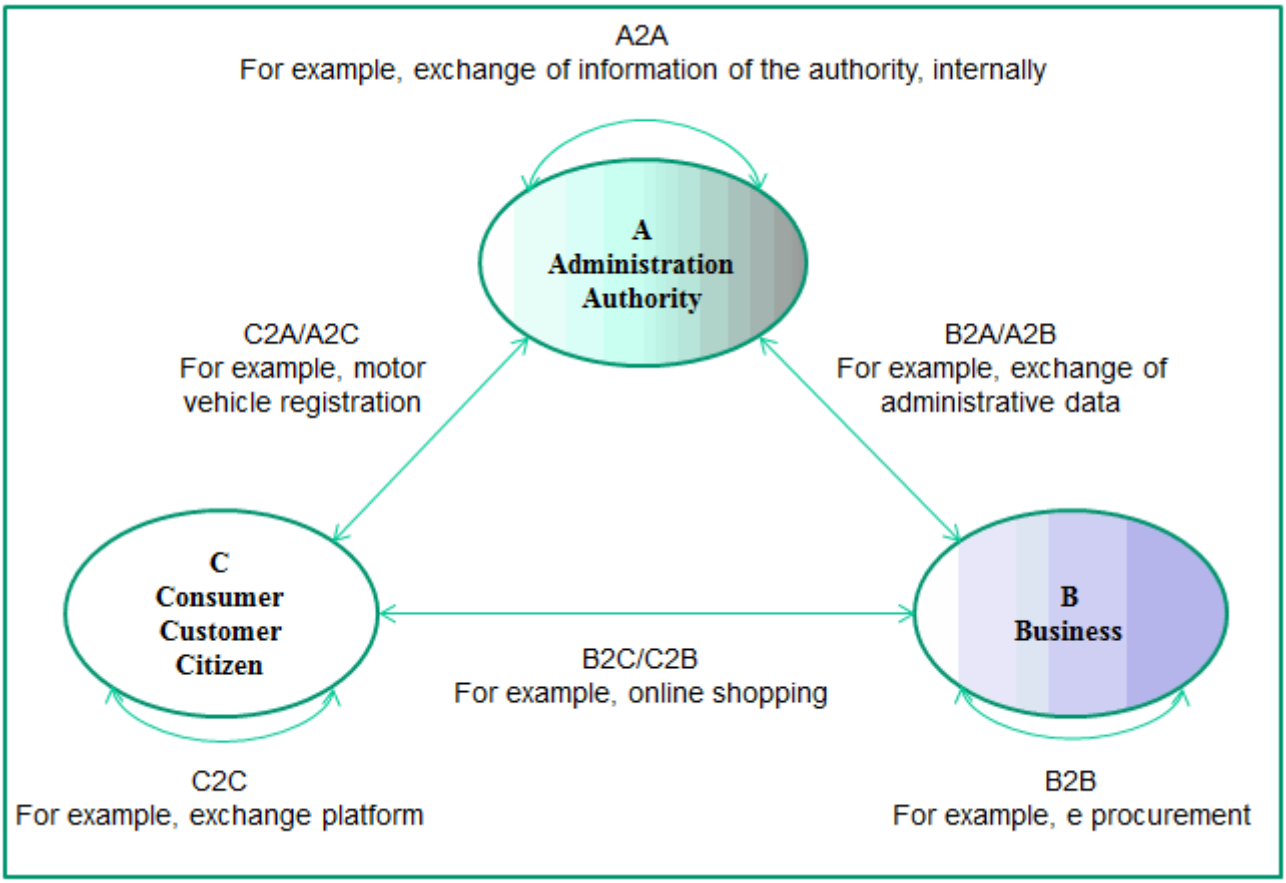
Business to Business describes the transactions taking place between companies. For example, this is enabled by linking the companies' supply chains.

B2C (Business to Consumer)

Business to Consumer describes transactions taking place between companies and their customers. For example, this includes purchases by customers in online shops.

B2A/C2A (Business/Consumer to Administration)

Business/Consumer to Administration describes all transactions between companies or individuals and public administration. Contacts between companies and administration are considered to be an area with great cost-cutting potential.



Transaction options in e-business

In addition to differentiating between various partners, a distinction can also be made regarding the scope of relationships between the partners: one to one, one to many, many to many. Especially the marketplace scenario is of major significance.

Marketplaces

Electronic marketplaces are virtual places where any number of people buy and sell products and services (openly) and exchange information.

To support these scenarios, the E-Business scenario diagram was developed. In conjunction with other methods and various components supplied by ARIS, it enables optimal support of the implementation of e-business projects. This chapter on e-business scenario diagrams first describes the method with all objects and evaluation options and then goes on to discuss interrelationships with other methods. At the end of the chapter, a use case demonstrates the complex possibilities.

7.2. E-Business scenario diagram method

7.2.1. The idea

The possibility of viewing a value-added chain in its entirety, that is, from the end user to each of the companies involved in a procedure, provides a basis for developing optimization potential. The aim is, for

example, the improvement of the supply chain, the reduction of procurement and distribution costs, or the optimization of the information system architecture. The e-business scenario diagram representation allows visualization of the content that is to be examined to achieve the designated objectives. Due to the column representation, the interfaces between very different process partners are depicted in an abstracted form by the column borders. Various reports supplement the models and offer important analysis capabilities.

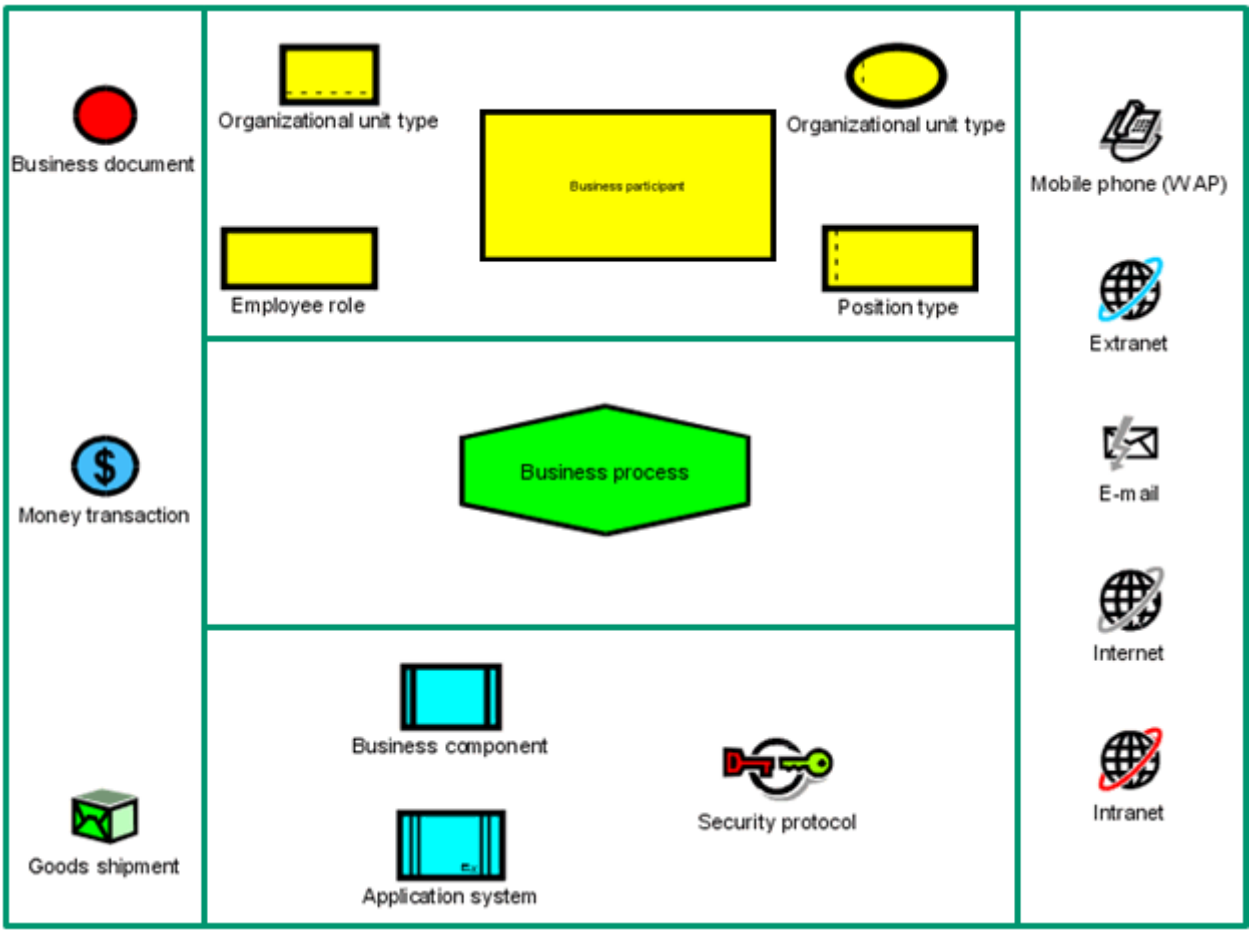
7.2.2. The model and its objects

The economic agents considered in the model are placed in the header and identified as **business participants**. They originate from the organization view and can be assigned organizational charts that may serve to describe the company structure or the relationships between the objects of the individual columns in more detail.

An economic agent's individual processes participating in the overall process and the interfaces between them are the central and structurally relevant objects of the model. An individual process is a **business process** that plays an important role in inter-company cooperation. A more precise representation and analysis of such an individual process can be obtained by assigning a process model. All processes running in a company are modeled in the row below the business participant, but in the same column. Inter-company coordination also requires precise analysis of **application systems** and hardware in use by the various economic agents in support of their individual processes (for example, ERP systems). These elements are represented by **Business components**. To coordinate the various components, responsibilities for the systems must be specified exactly. For this purpose, **Organizational unit type** objects are available. Even the roles of the employees involved in the process can be defined. These are referred to in the model as **Employee role**. The integration of interfaces is a particular requirement of e-business modeling. In this context the column borders are of crucial importance since they symbolize the interfaces between the process participants. They can be viewed from several perspectives.

One focus can be the transfer of process-specific information. That is the purpose of **Business documents**, which can assume the form of XML or HTML documents. The business document can be assigned a model of the data view, for example, a document type definition. Alternatively, the flow of money or goods can be displayed using the **Money transaction** or **Goods shipment** objects.

Another important aspect: data security must be ensured, especially the security of electronic payments sent via Internet. Different encoding techniques can be used for this purpose, for example, SET (Secure Electronic Transaction) or SSL (Secure Socket Layer). The security aspect can be taken account of by using the **Security protocol** object. It is also possible to define persons responsible for the security of transactions, which can be represented by an **Organizational unit type**. Furthermore, the focus may be put on analyzing a more technical aspect, namely the technical design of the data transfer at the interfaces. For this purpose, the model uses various information carriers. Individual processes can be linked via **intranet**, **extranet**, or **Internet**. Data transfer can take place by **e-mail**. The mobile phone has also gained ground as a transmission medium.



E-business scenario diagram (objects)

7.2.3. 'Transmission type' attribute group

Modeled objects can be further specified by their attributes. The relevant attributes are especially tailored to meet the e-business requirements.

Attention should be given to the **Transmission type** attribute group of the **Business document**, **Money transaction**, and **Goods shipment** objects. Specifying transmission type attributes not only identifies the transmission path, but also expresses the need for securing the transaction. In the case of an online transmission, for example, it is important not to omit the above-mentioned securing of confidential information and data.

7.3. Evaluations using reports

Various evaluation options provide support for modeling e-business scenarios. These evaluations are performed in the form of reports. ARIS provides several predefined evaluation reports, but the creation of user-defined ones is possible as well. The following reports for e-business scenarios are provided.

7.3.1. Data security check

The security of data transferred online is one of the most important issues influencing e-business acceptance. Protecting personal data or payments from access by unauthorized persons is an issue that

must be resolved in order to avoid loss of confidence of customers and partners. A report allows all products/services exchanged (money transaction and goods shipment) and all data (business documents) to be verified in this regard. The **Transmission type** attribute group already mentioned is evaluated and, in the event that it is an online transmission, checked to determine if data encoding takes place. Thus, potential security gaps and obsolete encoding methods can be identified and eliminated.

7.3.2. System support

A second important aspect in e-business projects is the harmonization of application systems. A company needs to consider many questions in this regard. Which processes must be supported by which systems? Who will be responsible for operating which systems? Where might training expenses be incurred? What adjustments of existing systems are necessary? Here too, the answers can be found in a report. Individual processes are listed along with the corresponding systems and the persons responsible for them.

7.3.3. Information flow

In contrast to other process models, e-business scenarios focus on transactions. Special attention is given the data and products/services being exchanged. Therefore, evaluations are provided to monitor data and service exchange. Important questions are: What data and products/services are generated, where are they generated, and where are they used? The required information may be obtained by running the report that outputs the modeled data and products/services as well as the processes in which they are involved as input or output.

7.3.4. Collaborative business maps

The collaborative business maps used by SAP constitute a special type of model. They emphasize the view of various partners. There are two views that focus on different target groups. The 'aggregated view' is designed for senior management and contains only business participants and processes, while the 'detailed view' is designed for operating departments and includes documents and roles of the process managers in charge. The reports enable a transfer of the information shown in the e-business scenario diagram to both views at any time.

7.4. Connection to other methods and components

Using the various modeling methods information can be viewed from different perspectives and made available to various target groups with a special focus. The e-business scenario is the starting point of these views. Details for specific target groups can be specified in its objects. In this way, an e-business project can be represented in its entirety. In addition, the various components of ARIS can be used to perform evaluations for the models created to ensure optimal support of projects in the e-business environment.

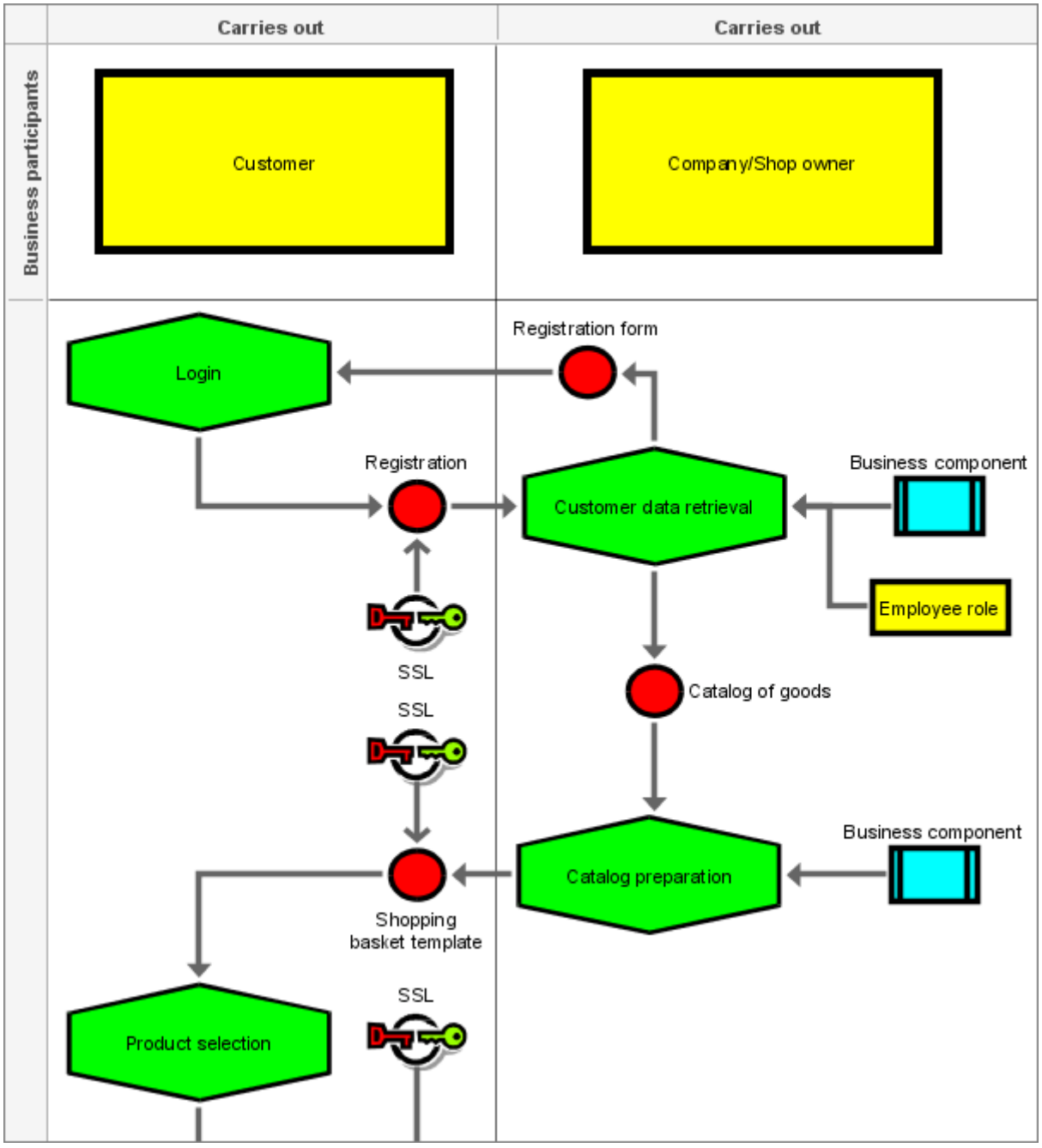
Example: Creation of an online shop

The first step is defining the objectives that are to be achieved by the planned e-business activities. This step can be performed with the help of the ARIS component Balanced Scorecard (see the chapter **Balanced Scorecard method [107]**). This makes it possible to identify the processes that have to be

optimized to achieve the objectives. In our example, the objective is identified as the development of a new distribution channel, namely the Internet. In order to pursue this new path optimally, precise documentation and planning are indispensable.

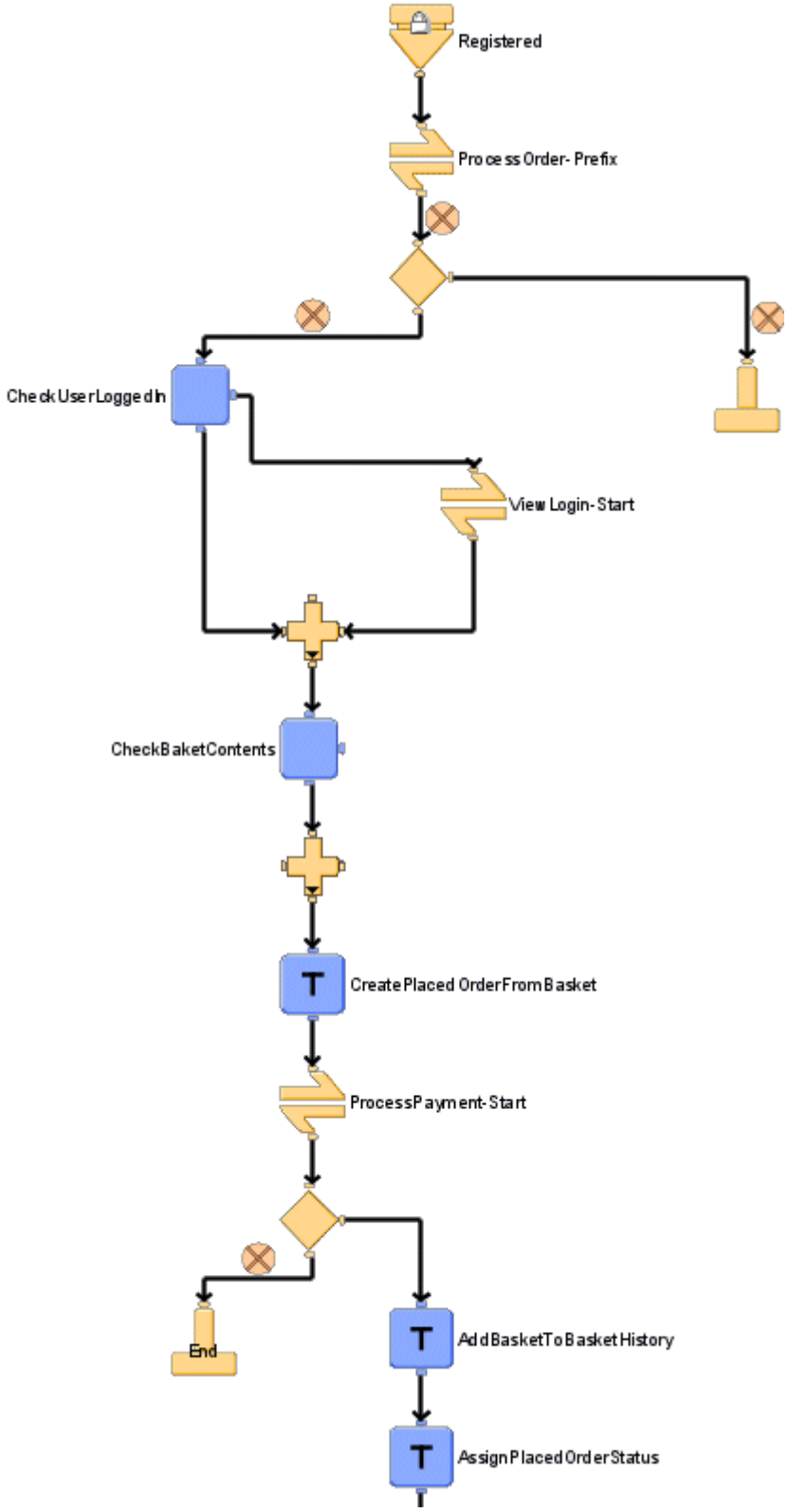
Not only must the process flow itself be taken into account, but also the organization of responsibilities, interfaces between various systems, and data security.

Starting point is the e-business scenario diagram. The business participants in our example are the company that implements the shop system and the customer who will use this offer. The entire process from 'entering the shop' to 'leaving' is broken down into subprocesses. The representation shows the view of the customer and that of the company. The e-business scenario diagram serves as an entry point to the implementation project. The following figure shows how the overall process is divided up.



E-Business scenario diagram

The various process steps can now be refined by EPCs: for example, they can be verified with the Simulation component, displayed after optimization by means of pipeline diagrams, and converted into a finished shop system through Intershop Enfinity to be further improved.



Pipeline diagram

If a shop is to be connected to an external ERP system (Enterprise Resource Planning system), the data to be transferred needs to be in the correct format. For this purpose, there are various ways of standardizing

documents and data. One such standardization option is offered by the use of Extensible Markup Language (XML). Since XML is a language that is being developed along completely different paths, a uniform foundation is needed here. Different organizations, consisting of companies and scientific institutions, are involved in standardizing XML for various sectors.

The use of standardized XML documents facilitates the integration of ERP systems.

The problems arising from the need to harmonize various components have already been mentioned. The application system type diagram can serve to visualize the systems and can be assigned as a model to application systems or business components to explain the systems' interrelationships in more detail.

The new e-business activities will also affect the organizational structure. It may be necessary to define new responsibilities or make new allocations. The e-business scenario diagram can describe roles and organizational units for individual process steps. What position they take in the company's organization or in the process can be analyzed further by means of organizational charts.

Implementation begins with the realization of modeled content. If the Intershop method is used, the content modeled is converted into an operational system by means of Intershop Enfinity.

8. IT City Planning

8.1. Enterprise Architecture and IT City Planning

IT City Planning is an architectural approach that was developed by the Frenchman Jacques Sassoon in the 1980s. The aim of IT City Planning is to bring harmony to a heterogeneous system landscape by thoroughly analyzing its interactions, that is, the exchange of information among the applications in that system.

Based on the approach used in town planning, the procedure for drawing up an IT city plan is driven by the idea of enabling long-term, strategic IT management that considers not only the present but also aspects of the past (legacy systems) and those of the future.

However, there is no need to redesign the entire system. Instead, a project-by-project, incremental approach is adopted.

As with Model Driven Architecture (MDA),which is supported by ARIS products, the subject of IT City Planning can be approached through models that describe the information system without reference to technology-related information. However, IT City Planning dispenses with UML, which simplifies familiarization with the subject for those with a less technical background, and increases its acceptance.

8.2. Which companies may benefit from IT City Planning?

IT City Planning addresses the following companies:

- Companies with a large portfolio of applications.
- Companies that have a long history of using information technologies.
- Companies for which information technology is of strategic importance.
- Companies that are involved in a merger.

Aims of IT City Planning:

- Reusing software resources to avoid the creation of further redundancies.
- Reducing maintenance costs by 'block-by-block' overhaul and by defining new software resources that can replace existing resources and cover the various use cases.
- Consolidating information systems.
- Preparing the deployment of EAI software at a higher level.

Creating an IT city plan is the responsibility of the Integration Competence Center. The plan itself must address both the design pattern and the information and technology architecture.

EAI = Enterprise Application Integration. Company-wide integration of applications. EAI provides the e-business infrastructure. EAI software is the technical middleware required as a prerequisite for implementing an e-business strategy.

8.3. IT City Planning with ARIS

ARIS provides the following views of an information system:

- Data view
- Function view
- Organization view
- Product/Service view
- Process view

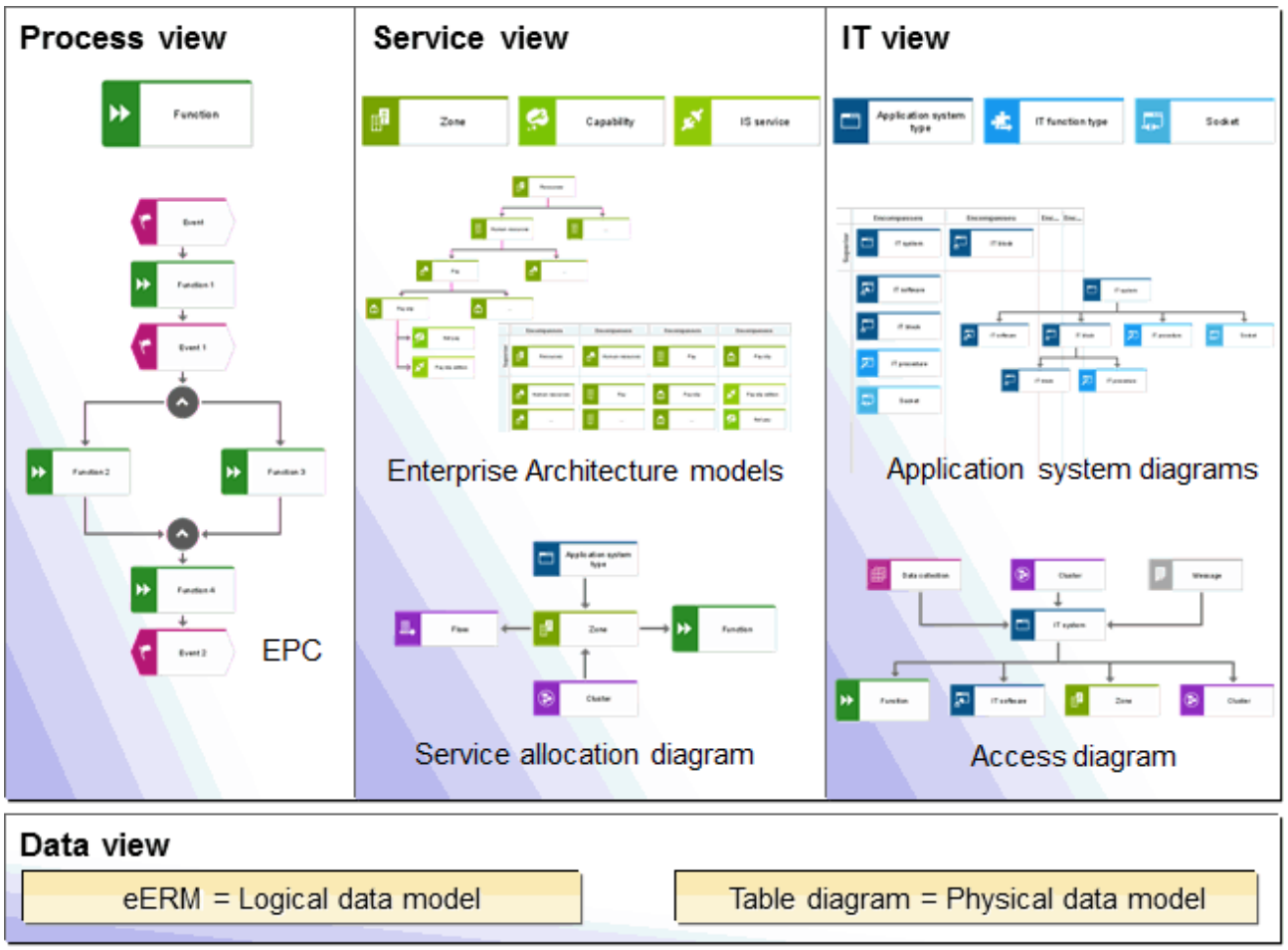
Each of these views is subdivided into the **Requirements definition**, **Design specification**, and **Implementation** descriptive levels. These are based on the lifecycle of an information system and their proximity to information technology.

The design specification and implementation levels essentially describe the software system. The concepts and terms of these levels are closely interrelated and their 'translation' is unproblematic.

This is not the case with the transition between the requirements definition and the design specification. When creating the design specification, the business management view must be aligned with the standard software. This requires both business management expertise and data processing knowledge (see Scheer, A.-W., ARIS - Business Process Frameworks, 1998, 3rd edition, p. 7).

The information system view (IS view) in IT City Planning can assist as a mediator between levels. In ARIS, the object types of the IS view are located at a level between functions and application systems, thus extending the function view in ARIS. Like functions, IS elements are associated with the various constructs from the familiar views of the ARIS House. These extensions mainly relate to process view and data view. In the following, the term IS view refers to the model types from the function and process views of the ARIS House, which describe relationships between IS elements or give a detailed description of IS elements seen in the context of the other ARIS views.

Application system types, IT function types, and sockets are considered as elements of the IT (information technology) view. In analogy to the IS view, the IT view includes all model types in which relationships between application system types, IT function types, and the new **Socket** object type are described, or which are used to describe one of these elements in detail.



IT City Planning (1): Process view, Service view, IT view

8.4. Service view

ARIS provides the following model types for describing the Service view:

- Service architecture diagram
- Service allocation diagram
- Service collaboration diagram

The two service architecture models arrange the structural elements of the information system in a hierarchy.

An IS hierarchy can include the following levels:

- Zone
- District
- Building cluster
- Functional block
- Capability
- IS service

- Business service

Zone, District, Building cluster, Functional block, and Business service are **Service type** object types. Service types are used to organize an information system in independent units/blocks by function.

Each service type is characterized in that it is the 'owner' of the data it uses and of the associated processing methods. Other service types can access these data and processing methods only if they call a service of the 'owner service type'.

Within a service type, similar data is used and identical activities and business functions are carried out.

At the top level, the information system is divided into zones. A zone can correspond to an operational and development area, for example.

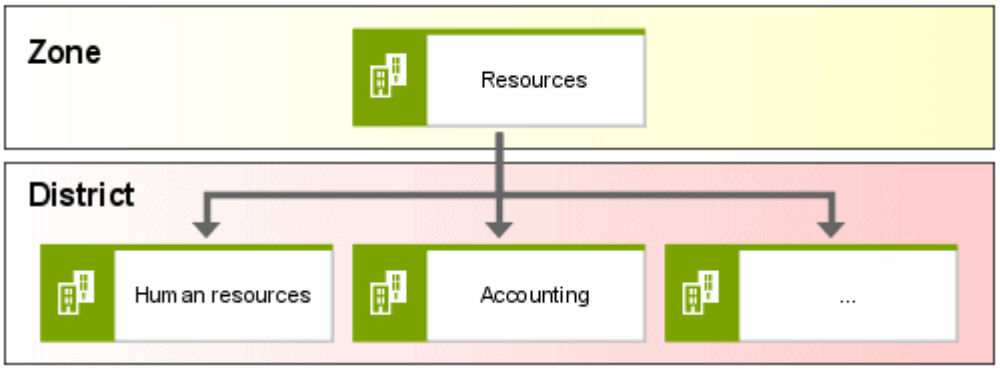
The following figure shows the zones into which a company's information system can be typically broken down.

IT City Planning (2)

Each zone, in turn, can contain one or more districts.

Districts of a zone are characterized by a high degree of similarity in terms of processes and temporal features (for example, similar lifecycles and information processing cycles). Districts can be terms of payment or of price, human resources administration, travel guidelines etc.

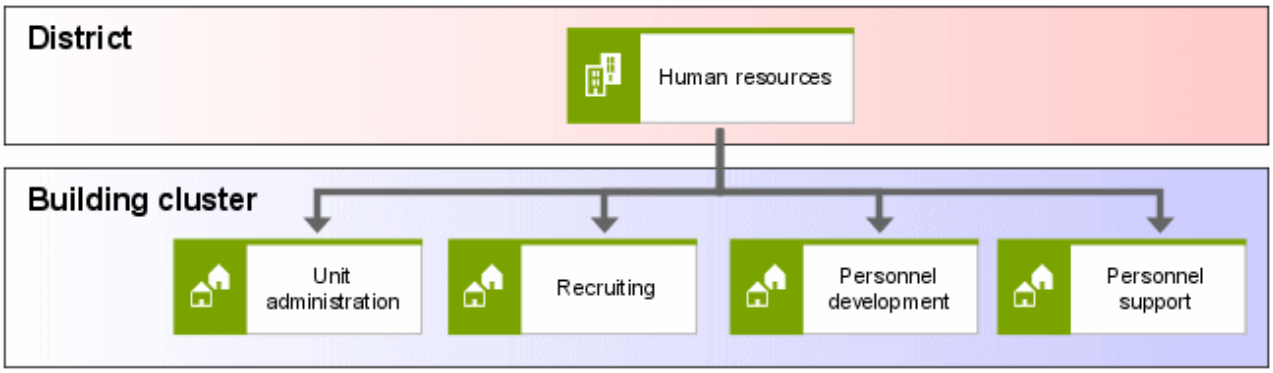
The **Resources** zone may include the **Human resources** and **Accounting** districts:



IT City Planning (3)

A district may contain one or more building clusters that serve a functional purpose, for example, salary payments, invoicing, etc.

The Human resources district includes the Business unit administration, Recruiting, Human resources development, and Human resources support building clusters.



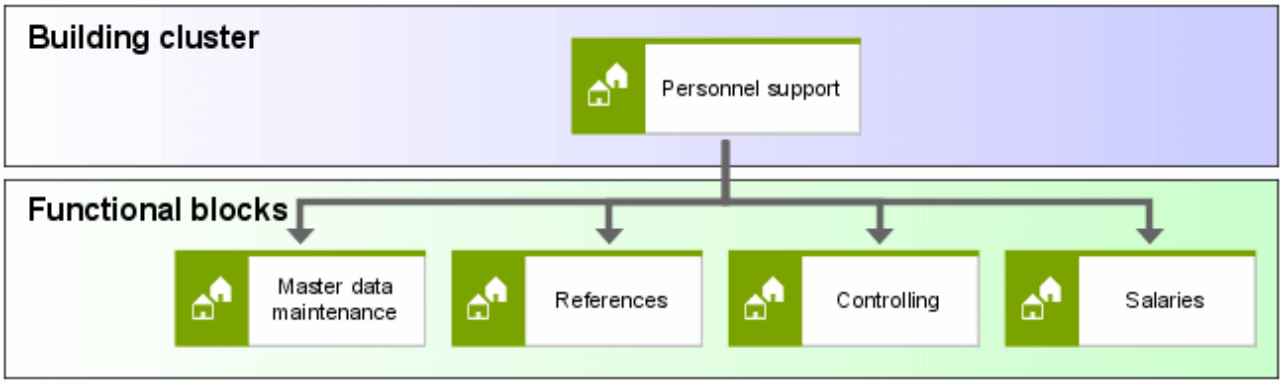
IT City Planning (4)

Each building cluster can encompass one or more functional blocks. Functional blocks are characterized by substantial similarity regarding the business objects and events they manage.

A functional block is an independent, reusable functional component. Capabilities and IS services are combined to form a functional block according to the following rules:

- There is a close interrelationship between the objects they manage and the functions they support.
- There is only minimal exchange with other functional blocks.

The building cluster **Human resources support** of the given example includes the **Master data maintenance**, **References**, **Controlling**, and **Salaries** functional blocks.



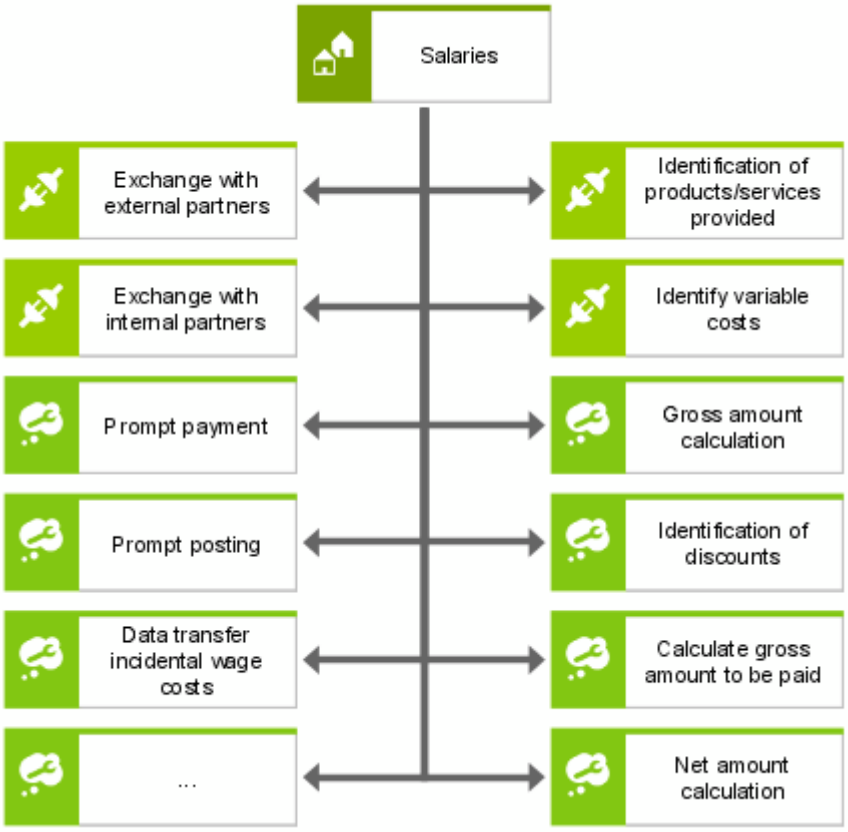
IT City Planning (5)

A capability is an elementary functionality within a system. It supports the realization of an activity within a process.

An IS service is an interface of a service type or a capability. IS services allow other IS elements controlled access to data and processing methods of the IS element that provides a service.

These interfaces can be used to exchange messages with other elements of the IS view.

The following figure shows the capabilities and IS services of the **Salaries** functional block.

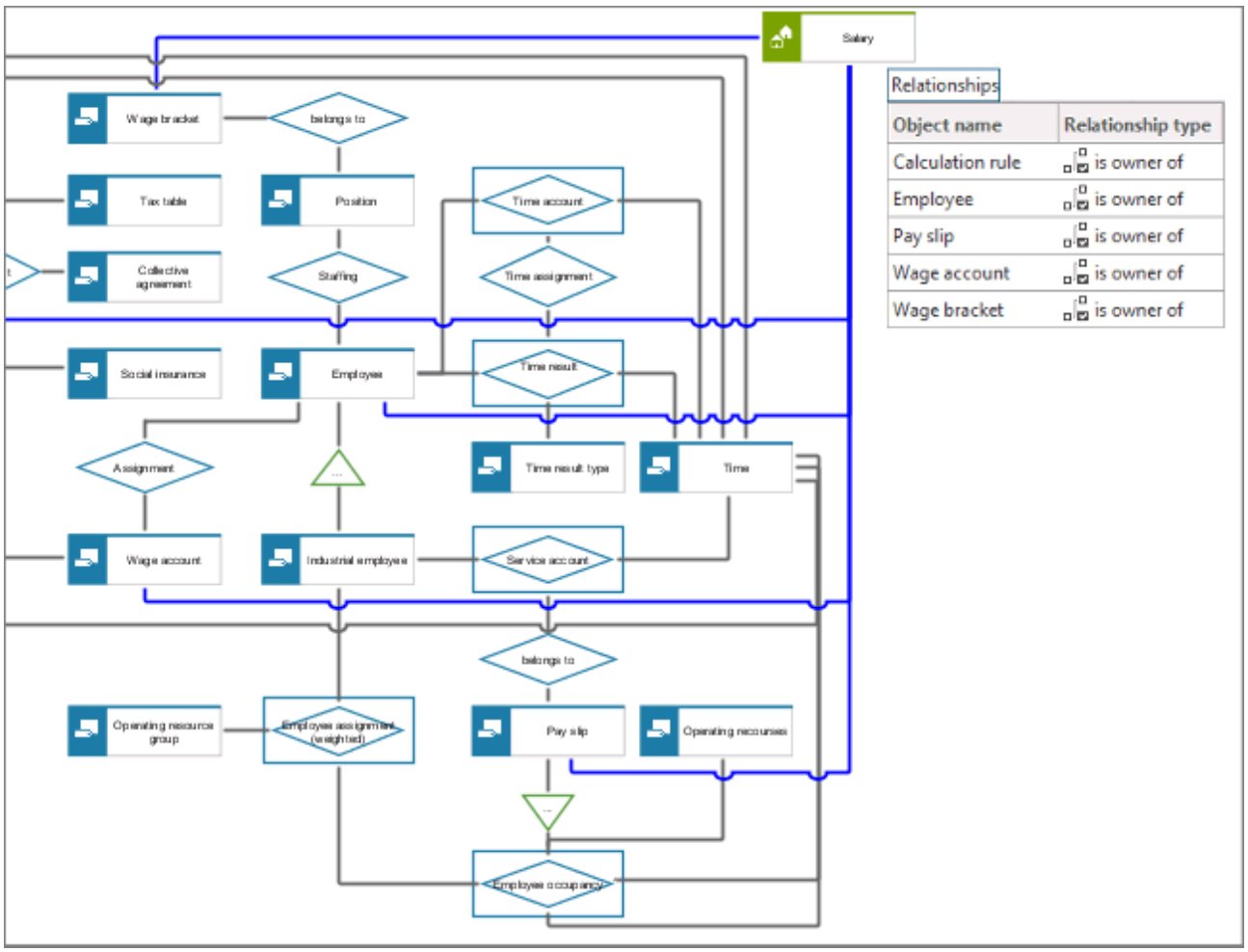


Capabilities and IS services of the 'Salaries' functional block

For describing the IS hierarchy it is not necessary to entirely model all levels described here. The **Capability** and **IS service** IS elements are not regarded as elements of the city plan in IT City Planning. The city planner's tasks end at the building block level. Capabilities and IS services are the responsibility of the architect (see Longépé, Christoph: Le projet d'urbanisation du système d'information, p. 18).

8.5. Service types and their data

The eERM is used to describe which data belong to a service type or capability. In the context of the city planning approach, the eERM provides the IS view symbols. A connection of the **is owner of** type can be used to link these objects with relationship and entity types.



IT City Planning (7): Service types and their data

8.6. Detail description of service types

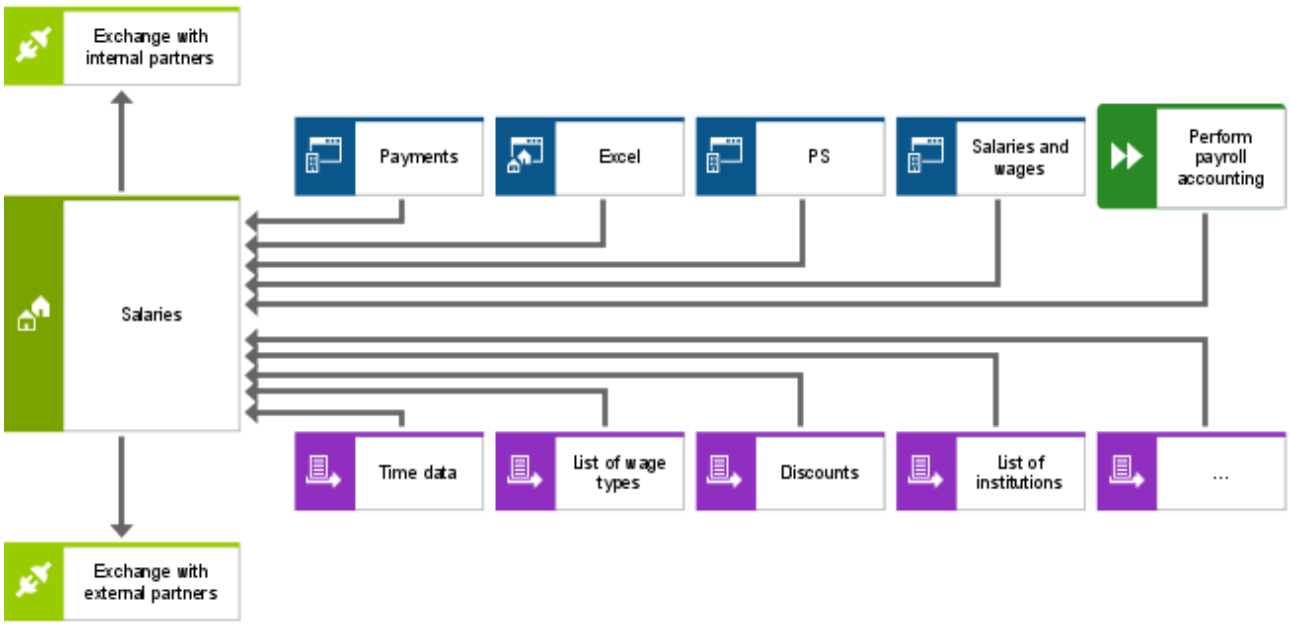
A detailed description of service types and capabilities of an information system can be given in the service allocation diagram. This includes

- the interfaces of a block,
- the interactions between the blocks,
- the application systems supporting the block, and
- business management functions that are supported by the block.

Zones, districts, building clusters, functional blocks, and capabilities can be connected to an IS service using the **provides** connection.

Input/Output connections can be drawn between IS elements and data elements to describe the information flows between the service types.

The various application system and IT function type objects can be assigned to the objects of the IT view via a connection of the **supports** type. If the city plan is interpreted as a development plan of a city, this connection provides information about which information system areas are 'populated' by which application systems. The **supports** connection is also available for use between IS elements and the function.



IT City Planning (8): Detail description of clusters

8.7. Chronological-logical operational sequences of IS elements

In the service allocation diagram, the relationships to the object types of the organization, data, and process views can be created for the service types, capabilities, and IS services specified in the service architecture diagram. Possible chronological-logical operational sequences of IS elements cannot be represented.

In IT City Planning, the service collaboration diagram is used to illustrate chronological-logical operational sequences of IS elements, that is, to describe the dynamic aspects within the information system. As a model type it is equivalent to the program flow chart of the IT view (see chapter Program flow chart [86]). It provides events for displaying operational sequences. As with the allocation of IT elements and events in the program flow chart, events in the service collaboration diagram can serve to define the sequence of functional modules. In this context, the event is seen as a trigger that activates service types, capabilities, or IS services. Branches can be represented using the rules known from the EPC or the program flow chart. Operational sequences can also be defined without the use of events.

8.8. IT view

The IT view contains the following model types:

- Application system type diagram
- Access diagram
- Program flow chart

Application system hierarchy

In the context of city planning, the current application system hierarchy of a company is depicted using the application system type diagram.

The following levels of an application system type hierarchy can be depicted:

- IT system
- Subsystem
- IT software
- IT block
- IT procedure
- Socket

IT system, Subsystem, IT software, and IT block are symbols of the **Application system type** object type. The hierarchy is built using the **encompasses** relationship type.

IT systems are at the top level of the application system type hierarchy. An IT system is made up of a structured quantity of IT elements, usually subsystems. Administration and operation of an IT system are the task of a specified organizational unit.

A subsystem is a component of an IT system. The components of a subsystem are called IT software.

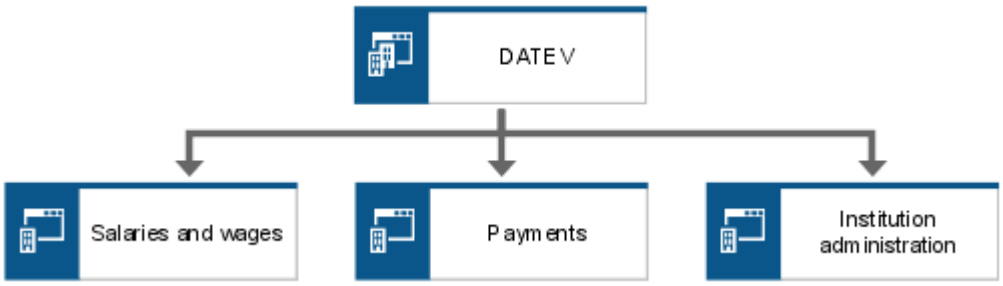
IT software supports a homogeneous set of functions. It is user-oriented and supports one or more business processes. IT blocks are components of IT software.

Generally, an IT block includes IT procedures that access the same data (databases, tables, files, etc.).

IT procedures are objects of the **IT function type** type. Each IT procedure supports a specific function.

A socket corresponds to the IS service, that is, it is an interface that an IT element provides for other IT elements so that these can access the IT element's data and processing methods.

The following figure shows an example of the subsystem structure of the DATEV system:



IT City Planning (9)

8.9. IT elements and their data

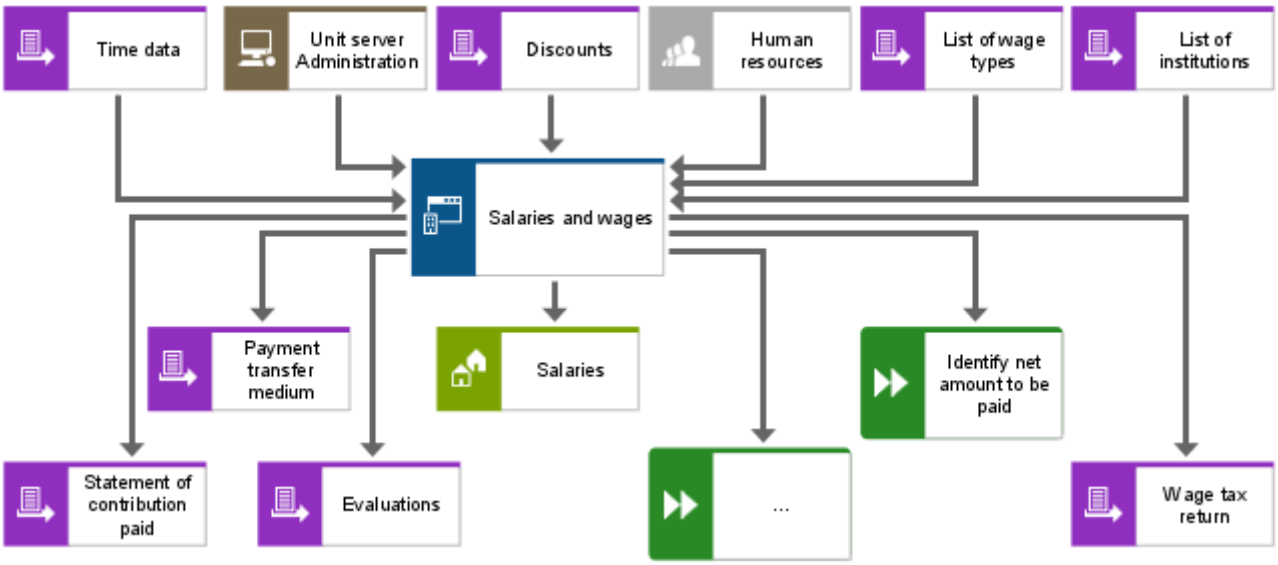
As with IS elements in the eERM, the **is owner of** connection is available, which can be drawn from application system types, IT function types, or sockets to entity types or relationship types to describe the data that belong to an IT element.

8.10. Detail description of IT elements

IT elements of the IT city plan are described in detail in the access diagram. It corresponds to the service allocation diagram of the IS view.

The description may refer to:

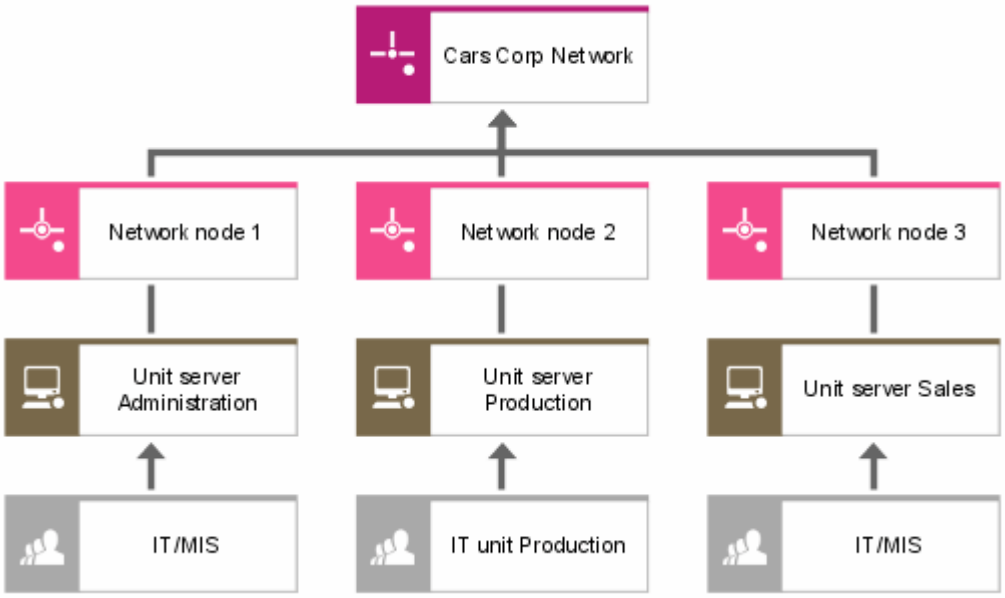
- input and output relationships of the IT element in question
- supported business functions
- supported IS elements
- activation of other IT elements by the element under consideration
- platform on which the IT element runs
- users of the IT element



IT City Planning (10): Detail description of IT elements

8.11. Organizational aspects

The detail description of an IT element also incorporates information from the organization view. This includes information about which organizational elements can be users of an IT element, and more. A network diagram can be used to show the influences and effects of the IT infrastructure.



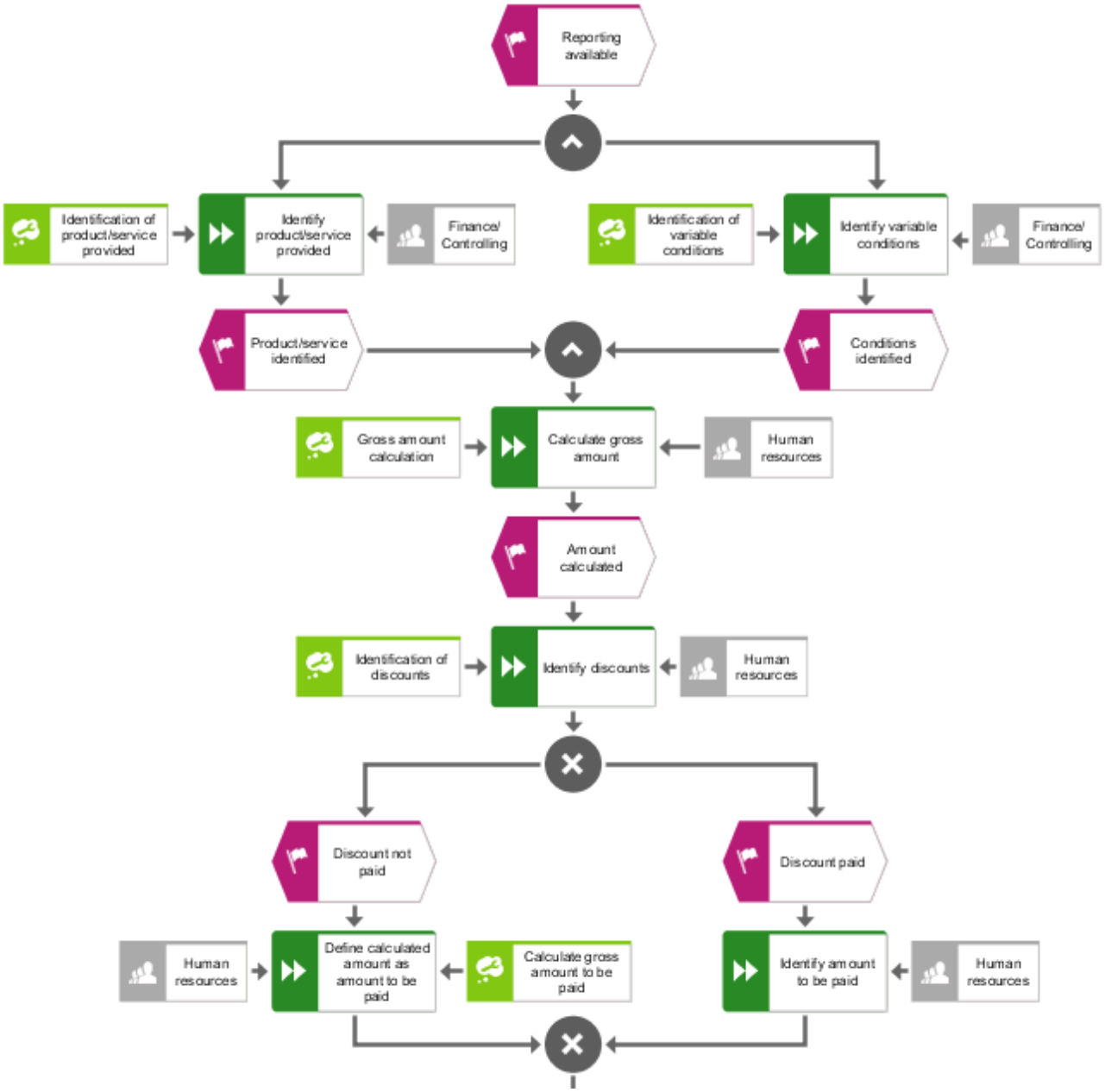
IT City Planning (11): Organizational aspects

8.12. Chronological-logical operational sequences of IT elements

In analogy to the Service collaboration diagram, the Program flow chart (PF) is used to describe the chronological-logical operational sequence of the **Application system type**, **IT function type**, and **Socket** IT elements.

8.13. Chronological-logical operational sequences within the architecture

Various process models (all variations of the EPC) and the program flow chart provide appropriate objects for illustrating how IS elements and IT elements are integrated into a chronological-logical operational sequence.



IT City Planning (12): Chronological-logical operational sequences

8.14. Possible evaluations

Based on the modeling options described in the sections above, evaluations are available to answer the following questions and help set up the information system:

- What data is managed by a given IS element?
- Which application systems support an IS element?
- Which functions are supported by a given IS element?
- What data is used by the IT elements of a given IS element?
- What data is generated by the IT elements of a given IS element?

- Which IS services are provided by an IS element and in which processes are they used?
- On which hardware components do application systems of a particular IS element run?

The following evaluations are provided for selected application system types, IT function types, and sockets:

- Data used by an IT element
- IS elements supported by an IT element
- Functions supported by an IT element
- Data used by IS elements that are supported by an IT element
- Data created by IS elements that are supported by an IT element
- Hardware components on which an IT system is running

9. Business process modeling

The interactions and transactions between companies and their partners, suppliers, and customers are becoming ever more complex, mostly due to new information and communication technologies. It is becoming more and more evident that the further development and performance of business processes depend on close cooperation between the various business associates.

On the one hand, it is important for a company to be able to better understand its own actions and those of its business associates; on the other hand, organizations should be given the ability to adapt faster to internal and market-driven changes. A standardized process modeling language can help companies to describe their internal and external business processes clearly and flexibly. Companies must also be in a position to communicate the modeled processes to their business associates appropriately, clearly, and comprehensibly. All parties involved should speak the same 'process language'.

To reach these goals, the Business Process Management Initiative (BPMI.org) offers a standardized modeling language: "Business Process Model and Notation (BPMN)". BPMN is a graphical notation for describing business processes.

The notation is required to be easily understood by all users. This makes it suitable not only for business process analysts and those who monitor and manage processes, but also for developers who implement the process execution technologies.

Furthermore, it is important to ensure that XML-based languages can be visualized with this notation for business process automation, for example, Business Process Execution Language for Web Services (BPEL4WS).

9.1. Process classes and the business process diagram

Business Process Model and Notation (BPMN) uses the **Business process diagram (BPD)** model type for describing processes. This model depicts three classes of business processes and the relationships between them:

- **Private business processes** (internal business processes)
- **Abstract business processes** (public business processes)
- **Collaboration processes** (global business processes)

Private business processes are business processes that are performed exclusively within an organization. They are generally known as workflow or BPM processes.

Various internal business processes are modeled as a sequence flow within individual pools (see chapter Implementation of BPMN in ARIS [144]) whose interactions are represented using message flows.

BPMN uses the terms Sequence flow and Message flow instead of Control flow because the process is controlled not only by events, but also by the messages exchanged.

Abstract business processes describe interactions between private processes of different pools, between objects of different pools, or combinations of both. Besides the sequence flow within the private process, the message flow between the individual processes is particularly important. Interactions are modeled using message flows.

Abstract business processes are integrated in individual pools and can be modeled separately or as part of an entire BPMN diagram. If an abstract business process occurs in the same model as a corresponding private business process, they can be associated with each other.

Collaboration processes describe only interactions between two or more business entities (business associates). A sequence of activities is modeled to reflect the pattern of message exchanges between the various business associates. The sequence flow has no part in this.

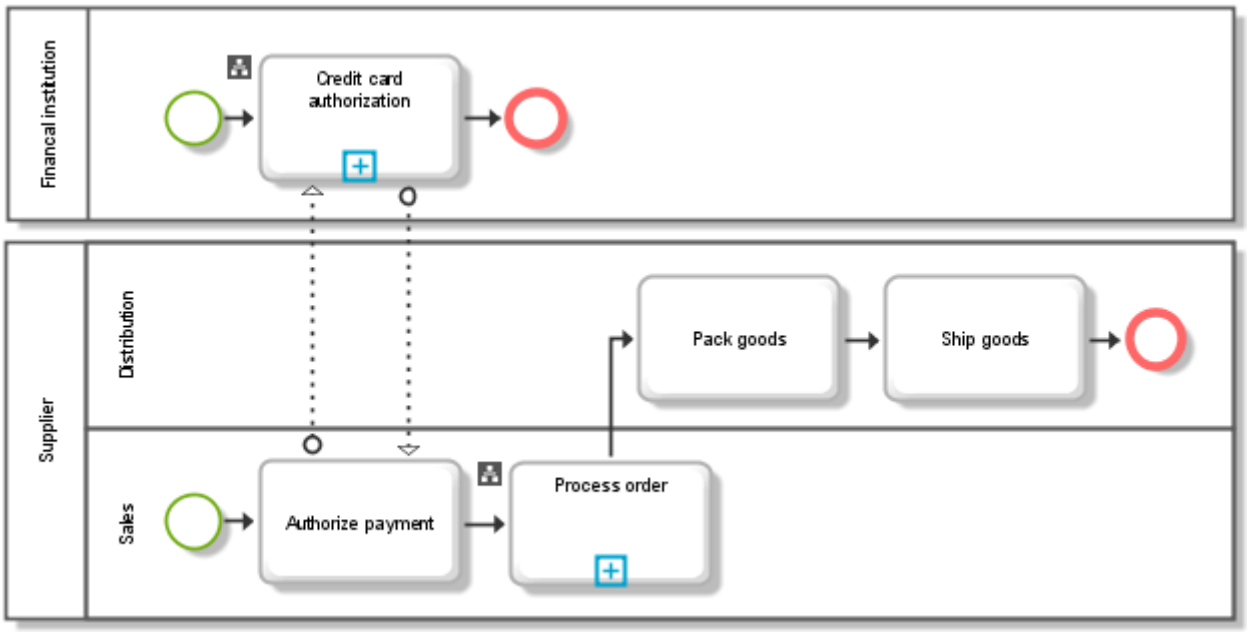
Relevant languages for collaborations include bXML BPSS, RosettaNet, or W3C Choreography Working Group. The mapping specification is planned for later versions of the BPMN specification.

Collaboration processes can be integrated in pools. Interactions of the partners involved are described in individual lanes. This allows the processes to be modeled separately or as part of a comprehensive BPMN diagram. If a collaboration occurs in the same diagram as one of its internal processes, activities common in both can be associated with each other.

In turn, various types of business processes can be derived from the three process classes:

- Private business processes at a higher level
- Private business processes at a detail level (target or actual processes)
- Processes running between detail processes and external entities
- Processes running between detail processes
- Processes running between detail processes and abstract processes
- Processes running between detail processes and collaboration processes
- Processes running between abstract processes
- Processes running between abstract processes and collaborations
- Processes running between collaborations
- Processes running between multiple detail processes interacting through their abstract processes
- Processes running between multiple detail processes interacting through a collaboration
- Processes running between multiple detail processes interacting through their abstract processes and a collaboration

The following figure shows an example of a BPMN collaboration diagram with two business associates to which a separate process has been assigned. Both detail processes comprise a start event, activities, sequence flow connections, and an end event. Message flow connections are modeled between the activities of the two detail processes.



Two pools with sequence and message flow

As processes of multiple business associates can be shown in one BPMN collaboration diagram and each business associate has a different view of the same process, it is useful to specify a 'point of view'. The BPMN does not dictate how the point of view is to be emphasized in a BPMN collaboration diagram.
The easiest way is to specify the names of the assigned business entities (business associates) in the **Description/Definition** attribute (see the figure).

9.2. Implementation of BPMN in ARIS

Although BPMN supplies only the **Business process diagram (BPD)** model type, two model types can be used in ARIS: the EPC and the new **Business process diagram (BPD)** model type. In this way, processes existing in ARIS can be reused as private processes. The EPC has all the model attributes that are specified by BPMN for the business process diagram. By using the Business process diagram (BPD) model type, you keep existing models of the EPC type free of B2B-specific aspects. As a result, the complexity of EPC models does not rise due to additional relationship types.

The new business process diagram in turn inherits all BPMN-relevant model attributes from the EPC and all sequence flow-relevant objects, connections, and symbols. The new Business process diagram (BPD) model type allows sequence flow-relevant EPC concepts to be reused. It is also possible to depict pools, lanes, and message flows.

9.3. Elements of the business process diagram

9.3.1. Pools and lanes

Pools give business process diagrams a clear structure.

A pool is a graphical container in which a set of activities of a business entity are combined.

A business entity can be a function, an application system, an organizational element (including Organizational unit, Organizational unit type, Group, Role, Position, Person, Location, System organizational unit, and System organizational unit type), **or a data element** (including Technical term, Cluster/Data model, Entity type, Relationship type, ERM attribute, Business object, Complex object type, COT attribute, Class, and Information carrier).

In BPMN, two pools represent two different business entities. The technique of structuring a model into pools is typically used in a B2B context.

A pool combines a process partner's various activities that are structured and organized using lanes. In this way, a differentiation is made to the activities of other process partners (see the previous figure Two pools with sequence and message flow [142]).

In a BPD, pools need not necessarily contain process elements. It is also possible to insert an empty pool ('black box') into a model, for example, for the purpose of integrating the interrelationships of a subprocess (for example, of a business associate) that is involved, but whose details are unknown into an overall process. Avoiding overcomplexity may be a reason for not modeling details of a subprocess (see the figure E-mail voting process [152]).

Pools include at least one lane. A lane can in turn contain additional lanes that are nested or defined as a matrix. If a pool has only one lane, the pool and the lane will have the same name. If a pool includes more than one lane, lane names and a special pool name must be specified.

Pools and lanes

In ARIS, pools and lanes are individual object types that are initially placed in the model. Within the pool, the process can be modeled in a way similar to an EPC. All functions, events, and rules of the process are placed on the pool object. Use the **belongs to** connection to describe the association of these objects with a pool. You are recommended to create it as an implicit relationship. A connection of the **depicts** type links the pool object to an organizational element, application system type object, data element, or function. It should be noted that each pool may have only one connection of this type throughout the given database. These relationships should also be implicit.

According to BPMN specifications, a pool does not need to be represented in the model by a symbol. Furthermore, the borders of **a single** pool may be hidden, especially if the diagram contains only one pool (see the figure **E-mail voting process [152]**). However, these options are not recommended for models containing multiple pools because this would affect the model's transparency.

9.3.2. Modeling guidelines for pools and lanes

- Only one pool may exist with invisible borders in a diagram.
- If the **Pool type** attribute was set to **Collaboration**, no owner (**Person responsible** attribute) should be specified.
- Each lane may have only one superior pool.

9.3.3. Sequence flow

A process in the form of a sequence flow describes the sequence in which activities of a process are performed. The sequence flow combines the **Event**, **Activity**, and **Gateway** object types. Sequence flows are permitted only within pools and may not cross their borders (see the following figure).

The sequence flow is represented by a solid line with a black arrow head:

Appropriate connection types, such as activates, is evaluated by, creates, links, or leads to are specified depending on the connection's source and target object type.

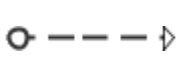
9.3.4. Modeling guidelines for sequence flow connections

- For sequence flows that follow an XOR (data-based) gateway or an inclusive gateway, a value must be set for the **Condition** attribute.
- If the value **Expression** is set in the **Condition** attribute, the diamond symbol is to be placed at the beginning of the connection.
- If the **Condition** attribute is set to **Default** and the source object is a function, the **** (backslash) character is to be placed as a symbol at the beginning of the connection.
- The **** (backslash) symbol must not be placed if the source object is a gateway.
- No condition should be set if the source object is one of the following symbols:
  - Event-based gateway
  - Complex gateway
  - Parallel gateway
  - Start event
  - Intermediate event
- If the **Default** value is enabled in the **Condition** attribute for a sequence flow connection, no condition must be specified.
- The **Condition** attribute may be set to **Default** if the source object is a function or an XOR (data-based) gateway.
- If the value **Expression** is set in the **Condition** attribute, the **Expression** attribute must also be specified.

9.3.5. Message flow

A message flow describes the exchange of information between two pools. The message flow can be placed either directly between two pool objects, or between objects in the sequence flow of the processes in the corresponding pool. Only message flows are allowed to cross pool borders, and a message flow connection must not be placed between two objects of the same pool (see the figure Two pools with sequence and message flow [142]).

The connection is represented by a dashed line. The beginning of the line is marked by a circle, and the end is a white arrow head.



Each message flow comprises a sender object, a connection of the **sends** type, a connection of the **is received by** type, and the recipient. No message flow connections may begin at a start event or intermediate event. However, an end event does not receive message flows, but can be a sender itself. Lanes, gateways, data objects, and text annotations do not have message flows.

9.3.6. Modeling guidelines for message flow connections

Source and target objects must belong to different pools.

9.3.7. Association

An association is used to provide the sequence or message flow components with information. This information can be of a textual or graphical nature. If multiple processes are part of the same diagram, their individual process elements can be associated with each other via connections.

An association is represented by a dotted line, to which open arrow heads may be added, if required. This applies in particular when assignments of artifacts of the **Data object** type are involved.

Appropriate connection types, such as **has as output**, **is input for**, **provides input for**, or **creates output to** are specified depending on the connection's source and target object type.

In BPMN, the assignment of artifacts of the 'Data object' type to activities is of particular importance.

This assignment is directed and describes how information is used and changed within a process. It is implemented in the BPD (BPMN) using the following relationships:

Function - creates output to - data elements (especially information carriers)

Data element (especially information carrier) - provides input for - function

9.3.8. Events

An event is a state that occurs in the course of the business process. Events influence the process flow. Normally, they represent triggers or effects within the processes. Depending on when an event occurs, it is either a start event, an intermediate event, or an end event. The three event categories are represented by their own symbols in BPMN:

Event categories

These categories are broken down into specialized subcategories. Additional symbols are added to the symbols of the three event categories when the **Event type** attribute is specified, as shown in the following three examples:

Event types

All attributes relevant for the **Event** object type are combined in the **BPMN** attribute type group.

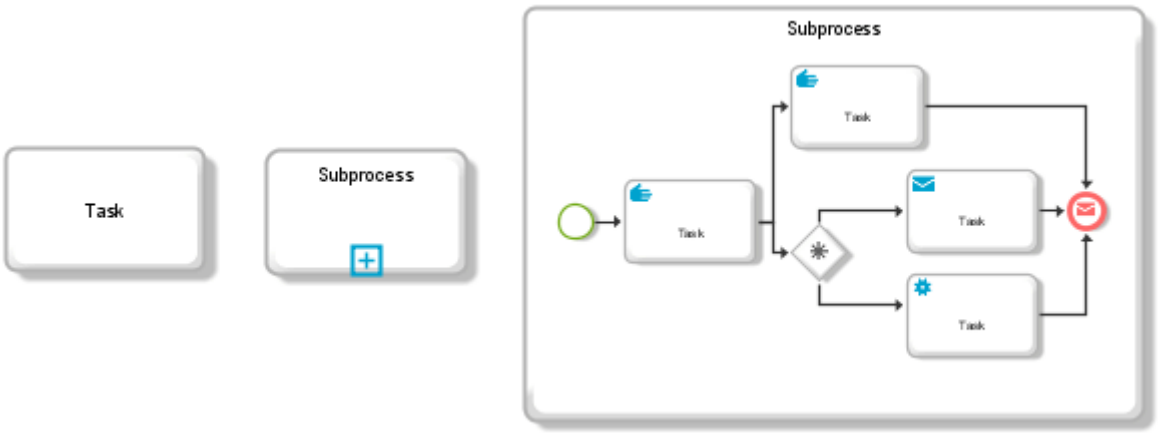
9.3.9. Modeling guidelines for events

- For start events, the **Event type** attribute type may have only one of the following values: **Message**, **Timer**, **Rule**, **Link**, or **Multiple**.
- For end events, the **Event type** attribute type may have only one of the following values: **Message**, **Exception**, **Cancel**, **Compensation**, **Rule**, **Link**, **Multiple**, or **Terminate**.
- For intermediate events, the **Event type** attribute type may have only one of the following values: **Message**, **Timer**, **Exception**, **Cancel**, **Compensation**, **Rule**, **Link**, and **Multiple**.
- Depending on the event type set, additional information must be specified in appropriate attributes.
- A start event may have multiple outgoing sequence flow connections. No value must be set for the **Condition** attribute of these connections.
- Intermediate events that indicate an exception or a compensation should be placed at the border of the function.
- If an intermediate event is placed at the border of a function, a value other than **Link** must be specified.
- The **Multiple**, **Rule**, and **Cancel** values must not be set for intermediate events that are located within a normal sequence flow of a process.
- The value **Cancel** must not be set if
  - the intermediate event is placed at the border of a function and the **Transaction** attribute of the function is not enabled, or
  - the event is not part of a process that describes a transaction.
- If an intermediate event is placed at the border of a function, it must not be the target object of a sequence flow connection.
- If an intermediate event is located within the normal sequence flow of a process (that is, it is not placed at the border of a function), it may have exactly one incoming sequence flow connection. For the **Event type** attribute of the event, it is possible to specify no value or one of the following values: **Message**, **Timer**, **Exception**, **Link**, **Compensation**.
- The value **Link** may be set for intermediate events in a normal sequence flow only if the source object is a gateway whose **Gateway type** attribute has the value **XOR (event-based)**.
- Each intermediate event must have exactly one outgoing sequence flow connection.
- An intermediate event whose **Event type** attribute has the value **Message** may have an incoming message flow (incoming connection of the **is received by** type).
- An intermediate event must not have an outgoing message flow (outgoing connection of the **sends** type).

9.3.10. Activities

An activity is performed as part of a process. It can be atomic or non-atomic (compound). BPMN uses three categories of activities: Process, Subprocess, and Task.

BPMN provides the following symbols for activities:



Activities according to BPMN

In ARIS, activities are modeled as functions by default:

The function is provided with all attributes that BPMN defines for processes, subprocesses, and tasks. As with events, the **BPMN** attribute type group is used, which contains additional subgroups for the activity types.

In terms of BPMN, a process describes an activity that is performed within a company or organization. A process is described by a graph with flow objects that represent a set of different activities and control objects. Processes are hierarchically structured and can be defined at all levels of detail. In contrast to a process, a business process in BPMN describes a set of activities that are performed across corporate/organizational boundaries.

In terms of BPMN, a subprocess is a combined activity with a detailed description. A subprocess occurs as an object within a process flow.

Usually, a subprocess is assigned a detailed process. Unlike in BPMN, ARIS does not identify an assigned activity by a plus sign, but by an assignment icon.

Besides identifying an assigned function, BPMN also provides the ability to show the detail process at the next higher process level. This is done by clicking the plus sign.

9.3.11. Modeling guidelines for activities

Process

- If the **Ad hoc** attribute is **= True**, the **Completion condition** attribute must be specified.
- If an ad hoc process is refined, no sequence flows must be modeled within the assigned model.

Subprocess

- If the value **Independent** is set for the **Subprocess type** attribute, the **Process reference** attribute must also be specified.
- If the **Transaction** attribute is enabled for a subprocess, the **Transaction ID** attribute must also be available.
- If the **Loop type** attribute is specified, the **Loop condition** attribute is also required.
- If models are to be transferred to BPEL4WS, a check is recommended to determine whether the **Loop type** attribute is set to **Maximum** for processes with the value **Standard**.
- If the value **Standard** is specified for the **Loop type** attribute, the **Test before** attribute is also required. The **Test before** attribute should be disabled by default.
- If the value **Multi-instance** is specified for the **Loop type** attribute, the **Parallel instance generation** attribute is also required. The **Parallel instance generation** attribute should be disabled by default.
- If the **Loop type** attribute of a subprocess has the value **Multi-instance** and, at the same time, the **Parallel instance generation** attribute is enabled, the **Loop flow condition** attribute must be specified as well.
- If the value **Complex** is set for the **Loop flow condition** attribute in a process, an expression that determines when and how many process markers are passed on after the subprocess is complete must be specified for the **Complex** attribute.

Task

- If the value **Receive** is specified for the **Task type** attribute, the function should not have any outgoing message flow connections.
- If the value **Send** is specified for the **Task type** attribute, the function should not have any incoming message flow connections.
- If the **Task type** attribute is not specified or the values **Script** or **Manual** are set, the function should not have any incoming or outgoing message flow connections.
- For functions whose **Task type** attribute is set to **Abstract**, the **Abstract type** attribute must also be specified. In addition, these functions may be used only in pools of the **Abstract** type or in **Collaborations**.

9.3.12. Gateway

Gateways describe how sequence flows split or join within a process. They determine the behavior of incoming and outgoing connections. In ARIS they are represented as objects of the **Rule** type.

Similar to events, various types of gateways can be specified. Depending on the type, further symbols are shown in the center of the Gateway symbol.

A few differentiating gateway symbols:

Gateway types

The BPMN specification stipulates that a number of gates must be defined for each gateway. In ARIS the number of gates is determined by the number of incoming and outgoing connections. Therefore, gate-dependent attributes are specified for the incoming and outgoing sequence flow connections of the rule.

A special case is the complex gateway for which the **Incoming condition** and **Outgoing condition** attributes are defined. These attributes are required if there are multiple incoming or outgoing sequence flow connections for a given gateway. The attribute content of the incoming condition can contain sequence flow names and process properties (data). The outgoing condition contains references to sequence flow IDs and process characteristics (data).

9.3.13. Modeling guidelines for gateways

- Gateways of the XOR (data-based) type: For all outgoing connections of an XOR (data-based) gateway, the value **Expression** must be set for the **Condition** attribute, and a valid expression must be used for the **Condition expression** attribute.

Sequence flow, especially after gateways

- For every XOR gateway of the **XOR (data-based)** type, the **Default gateway** attribute should be specified for exactly one outgoing sequence flow connection (**activates** connection type). Under no circumstances must multiple outgoing connections be marked with this attribute.
- For each XOR gateway of the **XOR (event-based)** type there must be at least two outgoing sequence flow connections (**activates** or **leads to** type).
- For all outgoing connections of an event-based XOR gateway, no value must be specified for the **Condition** attribute. The **Condition expression** attribute should not be specified.
- The following target objects are permitted for outgoing sequence flow connections of an event-based XOR gateway:
  - Functions for which the **Receive** task type was set.
  - Intermediate events whose **Event type** attribute type has a value other than **Compensation** or **Multiple**.
- If there is a function in the set of target objects, this set must not contain an event of the **Message** type.
- If a gateway of the **OR** type has no or exactly one incoming sequence flow connection, there must be at least two outgoing sequence flow connections.
- For all outgoing sequence flow connections of an OR gateway, the value **Expression** must be set for the **Condition** attribute, and a valid expression must be used for the **Condition expression** attribute. The expression must unambiguously relate to the current gateway.
- If an OR gateway has exactly one outgoing sequence flow connection, no value must be specified for the **Condition** attribute of this connection.
- If a gateway of the **Complex** type has no or exactly one incoming sequence flow connection, there must be at least two outgoing sequence flow connections.
- For all outgoing connections of a complex gateway, the value **None** must be specified for the **Condition** attribute, especially if there is only one outgoing connection.
- If a complex gateway has multiple incoming sequence flow connections, a condition that references the sequence flow names and process properties (data) must be specified for the **Incoming condition** attribute.
- If a complex gateway has multiple outgoing sequence flow connections, a condition that references the sequence flow names and process properties (data) must be specified for the **Outgoing condition** attribute.
- If an AND gateway has no or exactly one incoming sequence flow connection, there must be at least two outgoing sequence flow connections.

- For all outgoing sequence flow connections of an AND gateway, no value must be specified for the **Condition** attribute.

9.3.14. Artifact

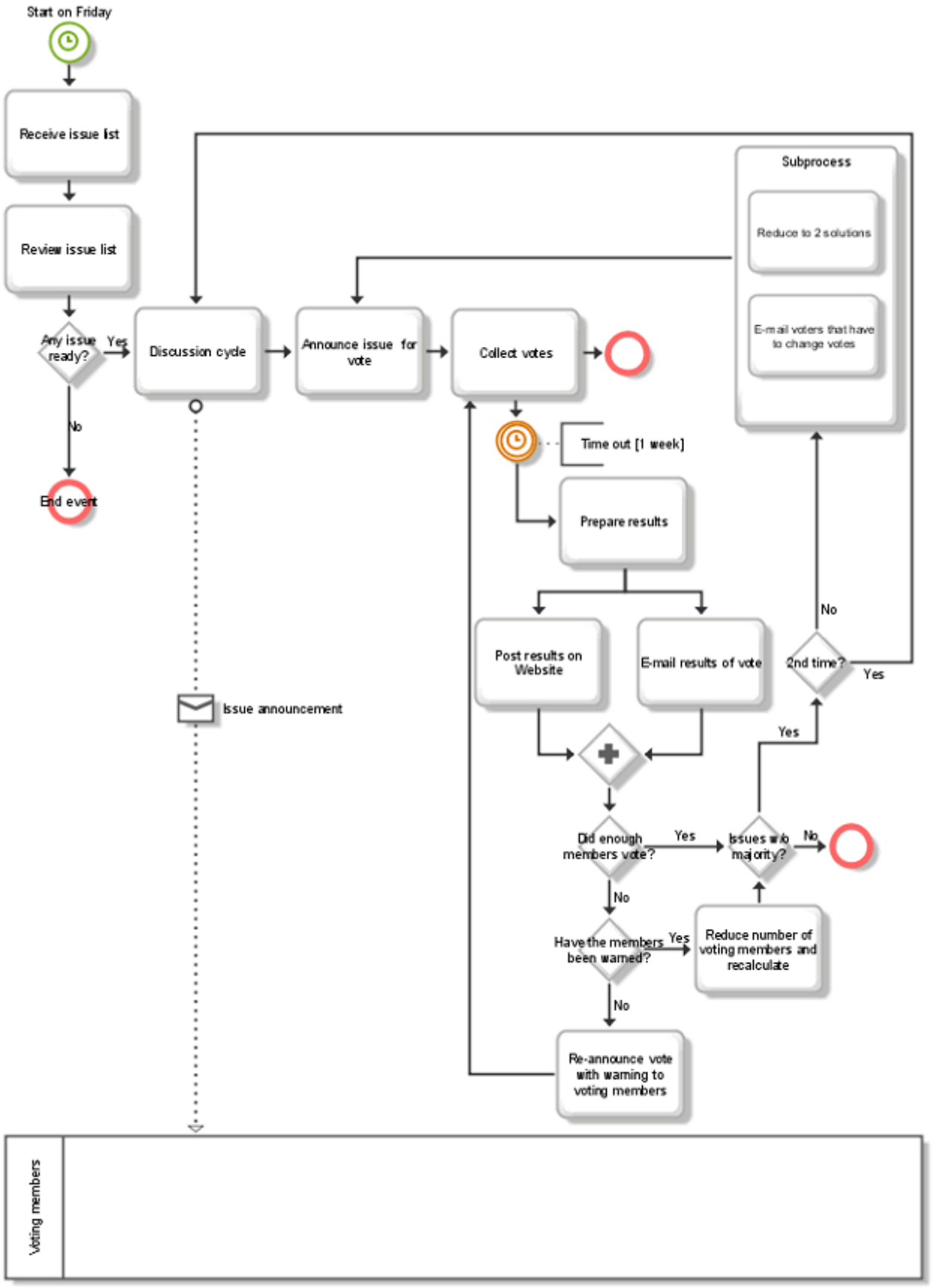
Artifacts provide information about the process. This information does not belong to the sequence flow or message flow. A total of three artifact types are differentiated: **Data objects**, **Groups**, and **Annotations** (the type list can be extended as required).

Data objects are comparable to information carriers or data elements in ARIS. However, in the broadest sense they could encompass all assignments. Data objects influence neither the sequence flow nor the message flow, instead they supply information about what happens during the process. They show how documents, data, and other objects change during the process.

A **Group** is a graphical emphasis of associated process elements. In ARIS, graphic objects, such as rectangles or polygons are useful for this.

Alternatively, groupings may also serve this purpose. However, this only makes sense if the grouping includes a graphic.

Annotations correspond to remarks about objects or connections, as in the following example **Time out [1week]**. In ARIS they are often realized with the help of the **Remark/Example** attribute. It is important that this attribute be placed in the model, as shown in the following example with **Yes** and **No**.



E-mail voting process

This figure shows an example of how a business collaboration diagram is implemented in ARIS according to BPMN 2.0. The diagram contains two pools, with the boundaries of the upper pool hidden. The individual elements of the lower pool are not shown.

9.3.15. Sources of figures

Figure **Two pools with sequence and message flow [142]** :

Business Process Modeling Notation, Working Draft (1.0); BPMI.org; August 25, 2003; page 85.

Figure **Pool with two lanes according to BPMN [144]** :

Business Process Modeling Notation, Working Draft (1.0); BPMI.org; August 25, 2003; page 87.

Figure **Event categories [147]** and figure **Examples of event types [147]** :

Business Process Modeling Notation, Working Draft (1.0); BPMI.org; August 25, 2003; page 27.

Figure **Activities according to BPMN [148]** :

Business Process Modeling Notation, Working Draft (1.0); BPMI.org; August 25, 2003; page 28.

Figure **Gateway types [150]** :

Business Process Modeling Notation, Working Draft (1.0); BPMI.org; August 25, 2003; page 28.

10. Modeling BPMN 2.0

10.1. Introduction

10.1.1. Initial situation and objective

BPMN (Business Process Modeling and Notation) has emerged as a widely adopted standard for process modeling. Its popularity is based on the fact that it is has been developed by the Object Management Group (OMG), a consortium of organizations that also released other important modeling standards like UML.

The primary goal of BPMN is to provide a notation that is understandable by all users: business analysts designing and documenting business processes, developers implementing these business processes, and business end users executing, managing and monitoring their business processes. The most recent version released by the OMG is BPMN 2.0. This is the standard supported by ARIS. In a first step, the objective is to focus on process modeling conformance, one of four conformance types defined by the OMG.

The four conformance types are described in detail in the BPMN specification: Business Process Model and Notation (BPMN), Version 2.0.

10.1.2. Purpose of this chapter

The BPMN specification has considerably increased in technical complexity and fails to distinguish between the elements required for business process modeling and those required for business process execution.

The purpose of this chapter is to describe how the BPMN 2.0 elements that are a part of business process modeling and document the process flow are implemented in ARIS. Those parts that are needed for the executable design are ignored. The elements relevant for business process modeling are essentially those displayed in a diagram.

The mapping described in the chapters of this document is based on the BPMN specification **Business Process Model and Notation (BPMN)**, **Version 2.0** (https://www.bpmn.org).

The attribute and model association tables are also taken from the BPMN 2.0 specification and were extended to describe the implementation in ARIS.

10.2. BPMN Core elements and their implementation in ARIS

The BPMN Core consists of four packages:

- Foundation,
- Infrastructure,
- Common Elements,
- Service.

It provides the basis for modeling the core elements Process, Collaboration, Choreograpy and Conversation. These packages are described in detail in chapter 8 of the BPMN specification.

In the following sections, the core constructs including their attributes and associations are mapped to ARIS constructs.

10.2.1. Infrastructure

The Infrastructure package consists of two elements that are particularly relevant for import and export. Thus, their attributes and model associations are not included in the current version of the BPMN 2.0 implementation in ARIS.

See: Business Process Model and Notation (BPMN), Version 2.0.

10.2.2. Foundation

The Foundation package contains classes that are shared amongst other packages in the BPMN Core. The Foundation package consists of eight classes: BaseElement, Documentation, RootElement, Extension, ExtensionDefinition, ExtensionAttributeDefinition, ExtensionAttributeValue, and Relationship. See: Business Process Model and Notation (BPMN), Version 2.0.

Mapping the attributes and model associations to ARIS

Class	BPMN attribute name	Implementation in ARIS
BaseElement	id: string	The ARIS GUID of the corresponding modeling construct represents the BPMN ID. For imported BPMN elements, an attribute type from the attribute type group **Attributes of external system** is used.
	documentation: Documentation [0..*]	see below: Documentation
	extensionDefinitions: ExtensionDefinition [0..*]	ARIS Method can be enhanced, for example, by user-defined attributes.
	extensionValues: ExtensionAttributeValue [0..*]	ARIS Method can be enhanced, for example, by user-defined attributes.
Documentation	inherits from BaseElement	
	text: string	All ARIS attribute types assigned to model types, object types, and connection types can be used for documentation purposes. The attribute types Description/Definition (AT_DEC) and Remark/Example (AT_REM) should be used to provide general information. Specific attribute types should be used to store specific information.
Extension	mustUnderstand: boolean [0..1] = False	Currently not implemented.
	definition: ExtensionDefinition	
ExtensionDefinition	name: string	Currently not implemented.
	extensionAttributeDefinitions: ExtensionAttributeDefinition [0..*]	
ExtensionAttribute Definition	name: string	Currently not implemented.
	type: string	

Class	BPMN attribute name	Implementation in ARIS
	isReference: boolean [0..1] = False	
ExtensionAttribute Value	value: [Element [0..1]	Currently not implemented.
	valueRef: [Element [0..1]	
	extensionAttributeDefinition: ExtensionAttributeDefinition	
Relationship	inherits from BaseElement	Currently not implemented.
	type: string	
	direction: RelationshipDirection {none | forward | backward | both}	
	sources: [Element [1..*]	
	targets: [Element [1..*]	
RootElement	inherits from BaseElement	RootElement is an abstract class and therefore has no direct representation in ARIS. For example, ARIS object types are root elements, ARIS attribute types are not.

10.2.3. Common Elements

Common Elements are basic elements that can be used in more than one diagram type, for example, in the Process, Collaboration, Conversation, and Choreography diagrams. The Common Elements are categorized into seventeen different groups.

10.2.3.1. Artifacts

Artifacts are used in the ARIS model types BPMN process diagram (BPMN2.0) or BPMN collaboration diagram (BPMN2.0) to depict additional information that is not directly related to a sequence flow or message flow. BPMN 2.0 provides three standard artifacts:

- Association,
- Group,
- Text Annotation.

Data Objects are no longer artifacts, they are concepts of their own (see chapter Items and Data [183]).

See: Business Process Model and Notation (BPMN), Version 2.0.

Mapping the attributes and model associations to ARIS

Class	BPMN attribute name	Implementation in ARIS
Association	inherits from BaseElement	By various connection types.
	associationDirection: AssociationDirection = None {None | One | Both}	This attribute is represented by the direction and the style of the corresponding ARIS connection type.

Class	BPMN attribute name	Implementation in ARIS
	sourceRef: BaseElement	Corresponds to the source object type associated with the connection type representing the association.
	targetRef: BaseElement	Corresponds to the target object type associated with the connection type representing the association.
Group	inherits from BaseElement	Object type: Structural element (OT_STRCT_ELMT) Symbol: 'Structural element' in model type Structuring model (MT_STRCT_DGM) Symbol: Group (ST_BPMN_GROUPING_1)
	categoryValueRef: CategoryValue [0..1]	Attribute type Name (AT_NAME) associated with object type Structural element (OT_STRCT_ELMT)
Category	inherits from BaseElement	Object type Structural element (OT_STRCT_ELMT) in model type Structuring model (MT_STRCT_DGM)
	categoryValue: CategoryValue [0..*]	Connection type in model type 'Structuring model': * Structural element (representing the category) 'contains' Structural element (representing the category value).
CategoryValue	inherits from BaseElement	Object type: Structural element (OT_STRCT_ELMT) Symbol: 'Structural element' in model type Structuring model (MT_STRCT_DGM) Symbol: Group (ST_BPMN_GROUPING_1) in BPMN 2.0 diagrams
	value: string	Attribute type 'Name' associated with object type 'Structural element'
	category: Category [0..1]	Connection type in model type 'Structuring model': * Structural element (representing the category) 'contains' Structural element (representing the category value).
	categorizedFlowElements: FlowElement [0..*]	Connection type 'belongs to' [CT_BELONGS_TO_1] in the model types BPMN process diagram (BPMN 2.0) and BPMN collaboration diagram (BPMN 2.0). Target object type: Structural element (OT_STRCT_ELMT; ST_BPMN_GROUPING_1) Source object types: * Function (OT_FUNC) representing activities * Event (OT_EVT) * Rule (OT_RULE) representing gateways * Cluster/Data model (OT_CLST) representing data objects * Information carrier (OT_INFO_CARR) representing data stores
Text Annotation	inherits from BaseElement	
	text: string	For object types in the model types BPMN process diagram (BPMN 2.0), BPMN collaboration diagram (BPMN 2.0), and BPMN conversation diagram (BPMN 2.0): * Text annotation (OT_BPMN_ANNOTATION) with symbol Text annotation (ST_BPMN_ANNOTATION_1) 'is associated with' <target object type>. Target object types are all object types available in the corresponding model type. For connection types in the model types BPMN process diagram (BPMN 2.0), BPMN collaboration diagram (BPMN 2.0), and BPMN conversation diagram (BPMN 2.0), three attribute types from the attribute type group **BPMN 2.0 attributes > BPMN text annotations** are available: * Text annotation 1 (AT_BPMN_TEXT_ANNOTATION_1) * Text annotation 2 (AT_BPMN_TEXT_ANNOTATION_2) * Text annotation 3 (AT_BPMN_TEXT_ANNOTATION_3)

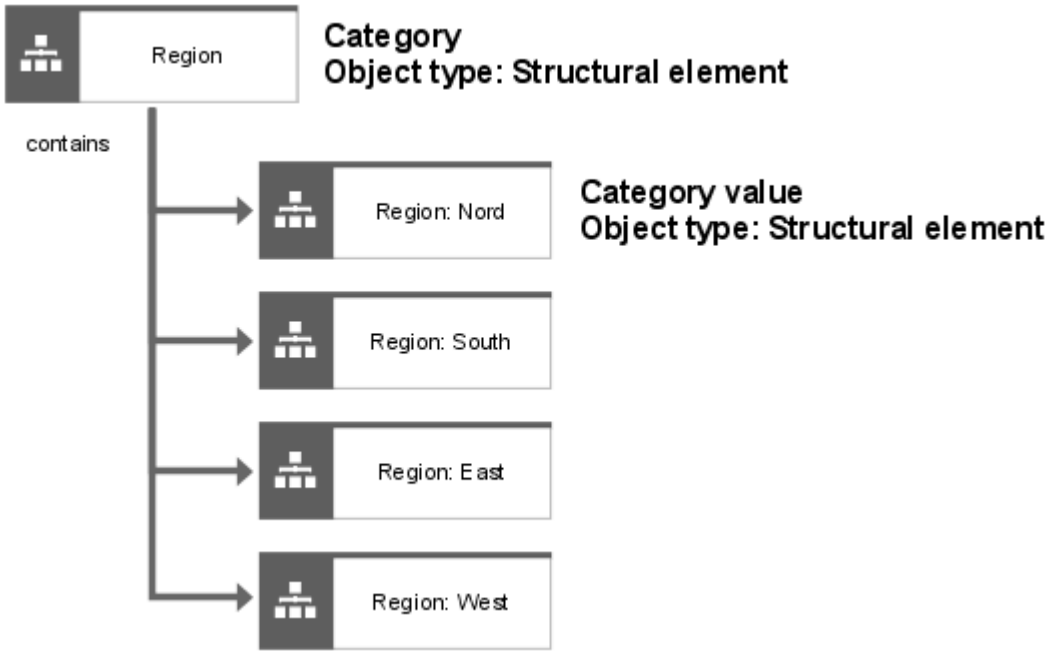
10.2.3.2. Associations

Associations are used to associate information and artifacts with other BPMN elements. In ARIS, associations are therefore (usually) represented by connection types. The relevant connection types are described in the context of the object types being associated.

10.2.3.3. Groups

BPMN 2.0 uses three different classes to represent groupings, but there is only one symbol: **Group**. Thus, a Group is the graphical representation of a CategoryValue element.

Categories and their values are modeled in an auxiliary ARIS model of type **Structuring model**.



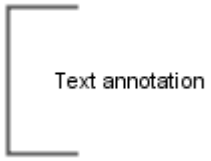
Structuring model

In ARIS, the graphical element **Group** is an occurrence copy of a category value object and is depicted by a special symbol in the BPMN 2.0 models. The symbol name is **Group**.

BPMN object symbol Group

10.2.3.4. Text Annotations

Text Annotations are used to provide a reader of a BPMN model with additional textual information. In ARIS, they can be associated with graphical elements in a model, with ARIS objects, or with ARIS connections.



BPMN object symbol Text annotation

Text Annotation elements are implemented in ARIS in two different ways:

Text annotations associated with ARIS objects

The object type **Text annotation** and the connection type **is associated with** are used to annotate objects (occurrences) in a model.

Text annotations associated with ARIS connections

Objects (here: 'Text annotation') cannot be assigned to connections. Thus, the program provides a new functionality: Modelers select the **Text annotation** symbol in the **Symbols** bar, place it on or near the connection they want to annotate, and enter the text. The program draws a line that looks like an Association and stores the text in a text annotation attribute of the corresponding connection. In the first step, three text annotation attributes are provided in the attribute type group **BPMN 2.0 attributes > BPMN text annotations**:

- Text annotation 1 (AT_BPMN_TEXT_ANNOTATION_1)
- Text annotation 2 (AT_BPMN_TEXT_ANNOTATION_2)
- Text annotation 3 (AT_BPMN_TEXT_ANNOTATION_3)

10.2.3.5. Callable Elements

CallableElement is an abstract class with four specialized classes: Process, Global Task, Choreography, and Choreography Task. Only the elements Process and Global Task are relevant for business process modeling compliance. In ARIS, these elements are represented by the object type **Function**.

See: Business Process Model and Notation (BPMN), Version 2.0.

Mapping the attributes and model associations to ARIS

Class	BPMN attribute name	Implementation in ARIS
CallableElement	inherits from BaseElement	Object type: Function (OT_FUNC) Symbol: Call activity (ST_BPMN_CALL_ACTIVITY)
	name: string [0..1]	Attribute type Name (AT_NAME) of object type Function (OT_FUNC)
	supportedInterfacesRefs: Interface [0..*]	Currently not implemented.
	ioSpecification: InputOutputSpecification [0..1]	Currently not implemented.
	ioBinding: InputOutputBinding [0..*]	Currently not implemented.
InputOutputBinding	inputData: DataInput	Currently not implemented.
	outputData: DataOutput	Currently not implemented.
	operationRef: Operation	Currently not implemented.

10.2.3.6. Events

Events are described in detail in the context of the BPMN **Process diagram** (see chapter Events [186]).

10.2.3.7. Expressions

FormalExpressions belong to the execution design level and are not included in the current version of the BPMN 2.0 implementation in ARIS.

However, natural-language expressions are used to allow the modeler to specify conditions. Such expressions are described in the context of the corresponding BPMN elements (object types and connection types).

10.2.3.8. Flow Elements

Flow Elements are described in detail in the context of the BPMN **Process diagram** (see chapter Processes [170]).

See: Business Process Model and Notation (BPMN), Version 2.0.

Mapping the attributes and model associations to ARIS

Class	BPMN attribute name	Implementation in ARIS
FlowElement	inherits from BaseElement	No direct representation in ARIS -> abstract class
	name: string [0..1]	Attribute type Name (AT_NAME) of the object types representing flow nodes.
	auditing: Auditing [0..1]	Currently not implemented.

Class	BPMN attribute name	Implementation in ARIS
	monitoring: Monitoring [0..1]	Currently not implemented.

10.2.3.9. Flow Elements Container

FlowElementsContainer is an abstract super class for BPMN diagrams (or views). The four FlowElementsContainer types include the elements Process, Sub-Process, Choreography, and Choreography Sub-Process.

The specific attributes and model associations associated with the Process and Sub-Process elements are described in detail in the context of the BPMN **Process diagram**.

See: Business Process Model and Notation (BPMN), Version 2.0.

Mapping the attributes and model associations to ARIS

Class	BPMN attribute name	Implementation in ARIS
FlowElementsContainer	inherits from BaseElement	Model type BPMN process diagram (BPMN 2.0) (MT_BPMN_PROCESS_DIAGRAM)
	flowElements: FlowElement [0..*]	Occurrences of the object and connection types allowed in model type BPMN process diagram (BPMN 2.0).
	artifacts: Artifact [0..*]	Occurrences of the object and attribute types that represent groups and text annotations and are allowed in model type BPMN process diagram (BPMN 2.0), as well as occurrences of the related connection types allowed in this model type.

10.2.3.10. Gateways

Gateways are described in detail in the context of the BPMN **Process diagram** (see chapter Gateways [193]).

See: Business Process Model and Notation (BPMN), Version 2.0.

Mapping the attributes and model associations to ARIS

Class	BPMN attribute name	Implementation in ARIS
Gateway	inherits from FlowElement	Object type: Rule (OT_RULE)
	gatewayDirection: GatewayDirection = unspecified { unspecified | converging | diverging | mixed }	The number of incoming and outgoing sequence flows depends on the modeling context, that is, the position of the gateway in the process. Thus, there is no ARIS attribute type representing the gateway direction. Gateways whose direction is unspecified or mixed should be avoided.

10.2.3.11. Messages

Messages usually represent information exchanged between two Participants in a BPMN **Collaboration diagram**.

A message is represented in ARIS by the symbol **Message** used for the ARIS object type **Message**.

BPMN object symbol Message

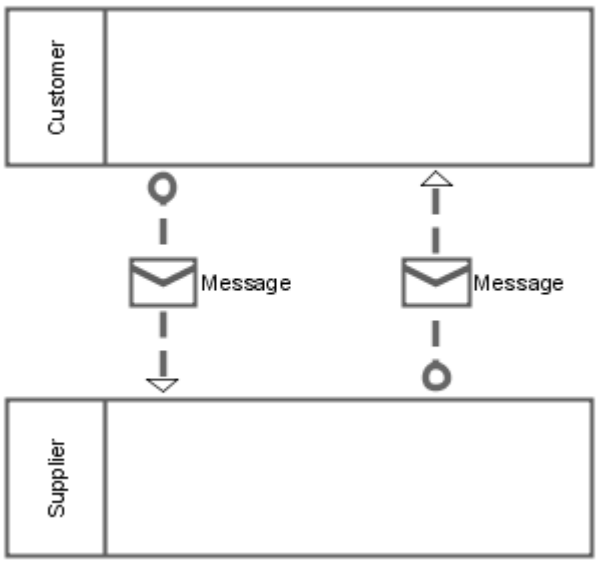
See: Business Process Model and Notation (BPMN), version 2.0, page 93 and the following.

Mapping the attributes and model associations to ARIS

Class	BPMN attribute name	Implementation in ARIS
Message	inherits from BaseElement	Object type: Message (OT_MSG_FLW)
		Symbol: Message (ST_BPMN_MESSAGE_2)
	name: string	Attribute type Name (AT_NAME) of object type Message (OT_MSG_FLW)
	structureRef : ItemDefinition [0..1]	Currently not implemented.

10.2.3.12. Message Flows

The message exchange between Participants is shown by a Message Flow element that is used to connect two Pools or the objects within the Pools.



BPMN message flow

A Message Flow is represented in ARIS by the connection type **message flow**. If the message sent from one participant to another is to be displayed in a BPMN diagram, the connection type **message flow** is replaced by the object type **Message** (symbol **Message**) and two connection types:

- <Source object type> 'sends' Message
- Message 'is received from' <Target object type>

More details can be found in the chapter Message Flows [199].

MessageFlowAssociation elements are used to map Message Flows modeled in two different diagrams, for example, in a Conversation and a Collaboration diagram. These associations are realized in ARIS by occurrence copies of the message flow connections.

The Message Flow element is also described in the context of the BPMN **Collaboration diagram** (see chapter Message Flows [199]) and the BPMN **Conversation diagram** (see chapter Message Flows in a Conversation [202]).

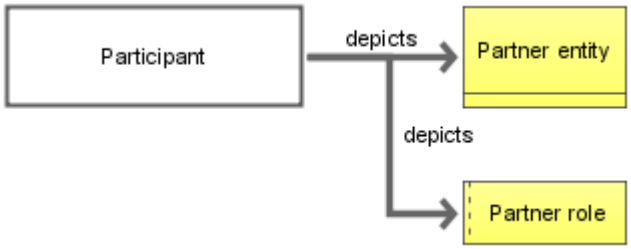
Mapping the attributes and model associations to ARIS

Class	BPMN attribute name	Implementation in ARIS
Message Flow	inherits from BaseElement	Connection type: 'message flow' (CT_BPMN_MESSAGE_FLOW)
	name: string	Attribute type 'Connection role' associated with connection type 'message flow' (CT_BPMN_MESSAGE_FLOW)
	sourceRef: MessageFlowNode	Source object type (Participant, Function, Event) accociated with connection type 'message flow' (CT_BPMN_MESSAGE_FLOW)
	targetRef: MessageFlowNode	Target object type (Participant, Function, Event) accociated with connection type 'message flow' (CT_BPMN_MESSAGE_FLOW)
	messageRef: Message [0..1]	Object type: Message (OT_MSG_FLW) Symbol: Message (ST_BPMN_MESSAGE_2) Connection types in model type BPMN collaboration diagram (BPMN 2.0): * Participant 'sends' (CT_SENDS_2) Message. * Event 'sends' (CT_SENDS_2) Message. * Function 'sends' (CT_SENDS_2) Message. * Message 'is received from' (CT_IS_RECEIVED) Participant. * Message 'is received from' (CT_IS_RECEIVED) Function. * Message 'is received from' (CT_IS_RECEIVED) Event.
FlowNode		Object types that can be the source or target for the 'message flow' connection type (CT_BPMN_MESSAGE_FLOW): Participant (OT_BPMN_POOL), Function (OT_FUNC), Event (OT_EVT)
MessageFlow Association	inherits from BaseElement	This association is used to map message flows modeled in two different diagrams (for example, in a BPMN collaboration diagram (BPMN 2.0) and a BPMN conversation diagram (BPMN 2.0)).
	innerMessageFlowRef: Message Flow	Occurrence copy of a 'message flow' connection in a BPMN collaboration diagram (BPMN 2.0) or a BPMN conversation diagram (BPMN 2.0).
	outerMessageFlowRef: Message Flow	Occurrence copy of a 'message flow' connection in a BPMN collaboration diagram (BPMN 2.0) or a BPMN conversation diagram (BPMN 2.0).

10.2.3.13. Participants

A Participant represents a PartnerEntity and/or PartnerRole that participates in a Collaboration. Participants can be modeled in a BPMN **Collaboration diagram** or a BPMN **Conversation diagram**.

In ARIS, the assignment of a partner entity and/or a partner role to a participant is transferred to the model of type BPMN allocation diagram (BPMN 2.0) assigned to an object of type Participant (OT_BPMN_POOL).



BPMN object symbol Participant

The usage of Participants is described in the context of the BPMN **Collaboration diagram** (see chapter Pools and Participants [199]) and the BPMN **Conversation diagram** (see chapter Participants [201]).

Participant, PartnerEntity, and PartnerRole inherit from BaseElement.

See: Business Process Model and Notation (BPMN), Version 2.0.

Mapping the attributes and model associations to ARIS

Class	BPMN attribute name	Implementation in ARIS
Participant	inherits from BaseElement	Object type: Participant (OT_BPMN_POOL) Symbol: Pool (ST_BPMN_POOL_1)
	name: string [0..1]	Attribute type Name (AT_NAME) of object type Participant (OT_BPMN_POOL)
	processRef: Process [0..1]	Model type BPMN process diagram (BPMN 2.0) assigned to object of type Participant (OT_BPMN_POOL) Process displayed within in the pool
	partnerRoleRef: PartnerRole [0..1]	Model type: BPMN allocation diagram (BPMN 2.0) Object type: Role (OT_PERS_TYPE) Symbol: Partner role (ST_BPMN_PARTNER_ROLE) Connection type: 'depicts' (CT_DEPICTS_1) Role
	partnerEntityRef: PartnerEntity [0..1]	Model type: BPMN allocation diagram (BPMN 2.0) Object type: Organizational unit (OT_ORG_UNIT) Symbol: Partner entity (ST_BPMN_PARTNER_ENTITY) Connection type: 'depicts' (CT_DEPICTS_1) Organizational unit
	interfaceRef: Interface [0..*]	Currently not implemented.

Class	BPMN attribute name	Implementation in ARIS
	participantMultiplicity: participantMultiplicity [0..1]	Attribute type in the attribute type group **BPMN 2.0 attributes > Participant multiplicity attributes** associated with object type Participant (OT_BPMN_POOL): * Multi-instance participant (AT_BPMN_MI_PARTICIPANT) The mini-symbol (three vertical lines) is displayed by the program if the value of the attribute type 'Multi-instance participant' is set to 'true'.
	endpointRefs: EndPoint [0..*]	Currently not implemented.
PartnerEntity	inherits from BaseElement	Object type: Organizational unit (OT_ORG_UNIT) Symbol: Partner entity (ST_BPMN_PARTNER_ENTITY)
	name: string	Attribute type Name (AT_NAME) associated with object type Organizational unit (OT_ORG_UNIT)
PartnerRole	inherits from BaseElement	Object type: Role (OT_PERS_TYPE) Symbol: Partner role (ST_BPMN_PARTNER_ROLE)
	name: string	Attribute type 'Name' associated with object type Role (OT_PERS_TYPE)
Participant Multiplicity	minimum: integer [0..1] = 2	Attribute type in the attribute type group **BPMN 2.0 attributes > Participant multiplicity attributes** associated with object type Participant (OT_BPMN_POOL): * Minimum participant multiplicity (AT_BPMN_MINIMUM_MI_PARTICIPANT)
	maximum: integer [0..1] = 2	Attribute type in the attribute type group **BPMN 2.0 attributes > Participant multiplicity attributes** associated with object type Participant (OT_BPMN_POOL): * Maximum participant multiplicity (AT_BPMN_MAXIMUM_MI_PARTICIPANT)
Participant Association	inherits from BaseElement	
	innerParticipantRef: Participant	Occurrence copy of the relevant participant.
	outerParticipantRef: Participant	Occurrence copy of the relevant participant.

10.2.3.14. Resources

Resources can be human resources as well as any other resource assigned to activities during process execution. Due to the use of semantically differing object types for representing resources in ARIS, a direct mapping of BPMN resources to ARIS constructs is not possible. Besides these differing object types, ARIS also uses differing connection types.

BPMN 2.0 knows only one object type called **Resource**. The **Resource** class and its individual occurrences correspond to specific ARIS connection types in combination with the relevant object types. Therefore, resources are not included in the current version of the BPMN 2.0 implementation in ARIS.

See: Business Process Model and Notation (BPMN), Version 2.0.

10.2.3.15. Sequence Flows

The BPMN Sequence Flow is mapped to nine ARIS connection types that are used to depict the control flow in traditional ARIS process models.

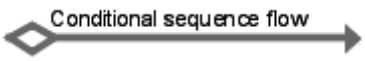
See: Business Process Model and Notation (BPMN), Version 2.0.

Source object type	Connection type	Target object type
Event	occurs before	Event
Event	activates	Function
Event	is evaluated by	Rule
Function	creates	Event
Function	is predecessor of	Function
Function	leads to	Rule
Rule	leads to	Event
Rule	activates	Function
Rule	links	Rule

BPMN distinguishes three types of connecting objects:

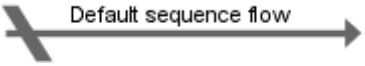
- **Sequence Flow**
  The Sequence Flow is the **normal** flow to which no specific conditions apply. In other words, its condition has always the value **true** . This type of Sequence Flow is depicted by a solid line with a solid arrowhead.

- **Conditional Sequence Flow**
  The Conditional Sequence Flow starting from an activity is drawn with a little diamond at the beginning of the connecting object, which indicates a data condition. A Conditional Sequence Flow starting from a gateway has the same shape as a normal Sequence Flow.
  Conditional Sequence Flow starting from an activity:



  BPMN conditional sequence flow

- **Default Sequence Flow**
  The Default Sequence Flow is denoted by a slash marker at the beginning of the connecting object, which means **otherwise** . It is enabled if no other Sequence Flow condition evaluates to **true** .



  BPMN default sequence flow

All ARIS connection types used in BPMN diagrams must hold attributes for recording text annotations [160]. Connection types used as outgoing connections associated with activities and gateways need additional attributes for recording sequence flow conditions.

Mapping the attributes and model associations to ARIS

Class	BPMN attribute name	Implementation in ARIS
Sequence Flow	inherits from FlowElement	A sequence flow is depicted by nine different connection types in the model types BPMN process diagram (BPMN 2.0) (MT_BPMN_PROCESS_DIAGRAM) and BPMN collaboration diagram (BPMN 2.0) (MT_BPMN_COLLABORATION_DIAGRAM): * Event 'occurs before' (CT_SUCCEED) Event * Event 'activates' (CT_ACTIV_1) Function * Event 'is evaluated by' (CT_IS_EVAL_BY_1) Rule * Function 'creates' (CT_CRT_1) Event * Function 'is predecessor of' (CT_IS_PREDEC_OF_1) Function * Function 'leads to' (CT_LEADS_TO_1) Rule * Rule 'leads to' (CT_LEADS_TO_2) Event * Rule 'activates' (CT_ACTIV_1) Function * Rule 'links' (CT_LNK_2) Rule
	name: string	Attribute type 'Connection role' associated with connection type 'message flow' (CT_BPMN_MESSAGE_FLOW)
	sourceRef: FlowNode	Source object associated with a sequence flow connection. Relevant object types: * Function * Event * Rule
	targetRef: FlowNode	Target object associated with a sequence flow connection. Relevant object types: * Function * Event * Rule
	conditionExpression: Expression [0..1]	Attribute type **Condition expression** (AT_BPMN_CONDITION_EXPRESSION) in the attribute type group **BPMN 2.0 attributes** associated with the following connection types: * activates (CT_ACTIV_1) * creates (CT_CRT_1) * links (CT_LNK_2) * leads to (CT_LEADS_TO_1) * leads to (CT_LEADS_TO_2) * is predecessor of (CT_IS_PREDEC_OF_1) The value of the attribute type 'Sequence flow condition' in the attribute type group **BPMN 2.0 attributes** must be set to 'Conditional sequence flow'.
	isImmediate: boolean	Currently not implemented.
FlowNode	incoming: Sequence Flow [0..*]	Incoming connections representing the sequence flow of the flow node object (object types: Function, Event, Rule)
	outgoing: Sequence Flow [0..*]	Outgoing connections representing the sequence flow of the flow node object (object types: Function, Event, Rule)

10.2.3.16. Elements not included in the current implementation

The following elements belong to the execution design level and are not included in the current version of the BPMN 2.0 implementation in ARIS:

- Correlation (See Business Process Model and Notation (BPMN), version 2.0, page 136 and the following.)
- Conversation Association (See Business Process Model and Notation (BPMN), version 2.0, page 135 and the following.)
- Error (See Business Process Model and Notation (BPMN), version 2.0, page 81 and the following.)
- Interaction Node (See Business Process Model and Notation (BPMN), version 2.0, page 123.)
- Item Definition (See Business Process Model and Notation (BPMN), version 2.0, page 91 and the following.)
- Service (See Business Process Model and Notation (BPMN), version 2.0, page 104 and the following.)

10.3. BPMN diagrams and ARIS model types: An overview

According to the BPMN 2.0 specification, three diagram types are required for process modeling conformance: **Process diagram**, **Collaboration diagram** and **Conversation diagram** (see Business Process Model and Notation (BPMN), version 2.0, page 2).

A careful consideration of these BPMN diagrams shows that the modeling constructs of the Process diagram are a subset of the modeling constructs used in the Collaboration diagram. There are also overlapping constructs in the Collaboration diagram and the Conversation diagram.

The model types listed in the following table are available in ARIS:

BPMN diagram	ARIS model type
Process diagram	BPMN process diagram (BPMN 2.0)
Process diagram	Enterprise BPMN process diagram
Collaboration diagram	BPMN collaboration diagram (BPMN 2.0)
Collaboration diagram	Enterprise BPMN collaboration diagram
Conversation diagram	BPMN conversation diagram (BPMN 2.0)
	BPMN allocation diagram (BPMN 2.0)

The model type **BPMN allocation diagram (BPMN 2.0)** enables the mapping of BPMN attributes and associations to the semantically richer ARIS Method where graphical elements are often used to represent these attributes and associations.

In the models of type **Enterprise BPMN process diagram** and **Enterprise BPMN collaboration diagram**, the following object types of ARIS Method are available as lane symbols in addition to what is defined in the BPMN 2.0 specification:

- Application system type
- Organizational unit
- Position
- Role
- Group

The model types **Enterprise BPMN process diagram** and **Enterprise BPMN collaboration diagram** furthermore provide the following connections to objects of type 'Task' in addition to what is defined in the BPMN 2.0 specification:

- Application system type **supports** Task
- Organizational unit **supports** Task
- Position **carries out** Task
- Role **carries out** Task
- Group **carries out** Task

10.4. Processes

See: Business Process Model and Notation (BPMN), Version 2.0.

The BPMN **Process diagram** depicts a BPMN Process. A Process is a specialization of a FlowElementsContainer. It contains the following elements:

- Flow Node (Event, Activity, and Gateway)
- Sequence Flow
- Artifact (see chapter Artifacts [157])

Mapping the attributes and model associations to ARIS

Class	BPMN attribute name	Implementation in ARIS
Process	inherits from CallableElement inherits from lowElementsContainer	Model type: BPMN process diagram (BPMN 2.0)
	processType: ProcessType = none { none | executable | non-executable | public }	Attribute type in the attribute type group **BPMN 2.0 attributes** associated with model type BPMN process diagram (BPMN 2.0): * Process type (AT_BPMN_PROCESS_TYPE) Attribute values: * Undefined (= none), * Executable process (AVT_BPMN_EXECUTABLE), * Non-executable process (AVT_BPMN_NON_EXECUTABLE) * Public process (AVT_BPMN_PUBLIC)
	auditing: Auditing [0..1]	Currently not implemented.
	monitoring: Monitoring [0..1]	Currently not implemented.
	laneSets: LaneSet [0..*]	Object type Lane (OT_BPMN_LANE) Symbol: Lane (ST_BPMN_LANE_1)
	IsClosed: boolean = false	Attribute type **Is closed** (AT_BPMN_IS_CLOSED) in the attribute type group **BPMN 2.0 attributes** associated with model type BPMN process diagram (BPMN 2.0)
	supports: Process [0..*]	Currently not implemented.
	properties: Property [0..*]	Currently not implemented.

Class	BPMN attribute name	Implementation in ARIS
	definitionalCollaborationRef: Collaboration [0..1]	The model of type BPMN collaboration diagram (BPMN 2.0) that contains the process

A Process is a particular construct: On the one hand, it is a model. On the other hand, a Process can be visualized within a Pool in a Collaboration. However, a Pool is not identical with a Process, and vice versa. A Pool represents a Participant in a Collaboration (see chapter Collaborations [198]). A Pool can contain the Process the Participant uses in a specific Collaboration.

The core elements for modeling a BPMN Process are those constructs that can be connected to each other using a Sequence Flow connection. They are called FlowNode. The corresponding ARIS object types and the symbols provided for them in the **Symbols** bar are listed in the table below.

BPMN element	ARIS object type	ARIS symbol	API name
Event	Event (OT_EVT)	Start event	ST_BPMN_START_EVENT
		Intermediate event	ST_BPMN_INTERMEDIATE_EVENT
		End event	ST_BPMN_END_EVENT
Activity	Function (OT_FUNC)	Task	ST_BPMN_TASK
		Subprocess	ST_BPMN_SUBPROCESS
		Call activity	ST_BPMN_CALL_ACTIVITY
Gateway	Rule (OT_RULE)	Gateway	ST_BPMN_RULE_1

These constructs are described in detail in the separate chapters below.

10.4.1. Activities

A BPMN Activity is represented by the ARIS object type **Function**.

BPMN 2.0 differentiates three basic Activity types: Task (atomic Activity), Sub-Process (non-atomic Activity), and Call Activity. In ARIS, the symbols depicting these types are provided in the **Symbols** bar.

BPMN object symbol Activity

When a modeler places an Activity symbol in an ARIS model, the software sets the corresponding value of the ARIS attribute type **Activity type** (AT_BPMN_ACTIVITY_TYPE). This activity type controls the correct behavior of the symbol. For example, a subprocess may have **embedded** flow elements while a task must not; a call activity may reference another task or process while tasks and subprocesses must not.

Mapping the attributes and model associations to ARIS

Class	BPMN attribute name	Implementation in ARIS
Activity	inherits from FlowElement	Object type: Function (OT_FUNC) Attribute type **Activity type** (AT_BPMN_ACTIVITY_TYPE) in the attribute type group **BPMN 2.0 attributes** associated with object type 'Function'. Attribute values: * Task (AVT_BPMN_TASK) * Subprocess (AVT_BPMN_SUB_PROCESS) * Call activity (AVT_BPMN_CALL_ACTIVITY)
	Compensation activity:: boolean = false	Attribute type **Compensation activity** (AT_BPMN_COMPENSATION_ACTIVITY) in the attribute type group **BPMN 2.0 attributes** associated with object type 'Function'
	loopCharacteristics: LoopCharacteristics [0..1]	See below: Loop characteristics
	resources: ActivityResource [0..*]	Currently not implemented.
	default: SequenceFlow [0..1]	Attribute type **Sequence flow condition** (AT_BPMN_SEQ_FLOW_CONDITION) in the attribute type group **BPMN 2.0 attributes** associated with the following connection types: * Activity 'creates' ... * Activity 'is predecessor of' ... * Activity 'leads to' ... The attribute value must be set to 'Default sequence flow'.
	ioSpecification: InputOutputSpecification [0..1]	Currently not implemented.
	properties: Property [0..*]	Currently not implemented.
	boundaryEventRefs: BoundaryEvent [0..*]	Connection type: Function 'can trigger' (CT_BPMN_CAN_TRIGGER) Event
	dataInputAssociations: DataInputAssociation [0..*]	Currently not implemented.
	dataOutputAssociations: DataOutputAssociation [0..*]	Currently not implemented.
	startQuantity: integer = 1	Currently not implemented.
Class	**completionQuantity**: integer = 1	Currently not implemented.

10.4.1.1. Resource Assignments

Resource assignments are not included in the current version of the BPMN 2.0 implementation in ARIS. They will be dealt with in detail when the execution design level is being implemented in ARIS.

See: Business Process Model and Notation (BPMN), Version 2.0.

10.4.1.2. Performers

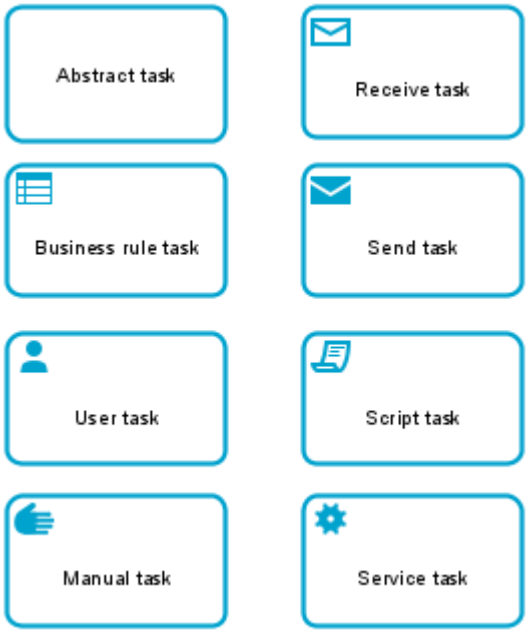
Performers are not included in the current version of the BPMN 2.0 implementation in ARIS. They will be dealt with in detail when the execution design level is being implemented in ARIS.

See: Business Process Model and Notation (BPMN), Version 2.0.

10.4.1.3. Activity type: Task

See: Business Process Model and Notation (BPMN).

BPMN 2.0 distinguishes eight Task types which are represented by different symbols. In ARIS, only the type **Abstract task** is available in the **Symbols** bar. The symbols used for the remaining seven special Task types are not available in the **Symbols** bar but are handled by the program.



Activity type: Task

When a modeler selects a specific task symbol in ARIS, the software sets the corresponding value of the ARIS attribute type **Task type**. This attribute type is read-only. It provides the following values: Abstract task, Business rule task, Manual task, Script task, Send task, Service task, Receive task, and User task.

Mapping the attributes and model associations to ARIS

Class	BPMN attribute name	Implementation in ARIS
Task	inherits from Activity	The value of attribute type Activity type (AT_BPMN_ACTIVITY_TYPE) is set to **Task** in the attribute type group **BPMN 2.0 attributes** associated with object type 'Function'. Object type: Function (OT_FUNC) Symbol: Task (ST_BPMN_TASK) or a special task symbol (see below)
Service Task	inherits from Activity	The value of attribute type Task type (AT_BPMN_TASK_TYPE) is set to **Service task** in the attribute type group **BPMN 2.0 attributes > Task attributes** associated with object type 'Function'. Object type: Function (OT_FUNC) Symbol: Service task (ST_BPMN_SERVICE_TASK)
	implementation: Implementation = Web Service {Web Service | Other | Unspecified}	Currently not implemented.
	operationRef: Operation [0..1]	Currently not implemented.
SendTask	inherits from Activity	The value of attribute type Task type (AT_BPMN_TASK_TYPE) is set to **Send task** in the attribute type group **BPMN 2.0 attributes > Task attributes** associated with object type 'Function'. Object type: Function (OT_FUNC) Symbol: Send task (ST_SEND_TASK)
	messageRef: Message [0..1]	Connection type in model type BPMN collaboration diagram (BPMN 2.0): * Function 'sends' Message
	operationRef: Operation [0..1]	Currently not implemented.
	implementation: Implementation = Web Service {Web Service| Other | Unspecified}	Currently not implemented.
Receive Task	inherits from Activity	The value of attribute type Task type (AT_BPMN_TASK_TYPE) is set to **Receive task** in the attribute type group **BPMN 2.0 attributes > Task attributes** associated with object type 'Function'. Object type: Function (OT_FUNC) Symbol: Receive task (ST_RECEIVE_TASK)
	messageRef: Message [0..1	Connection type in model type BPMN collaboration diagram (BPMN 2.0): * Message 'is received from' Function
	Instantiate: boolean = False	Currently not implemented.
	operationRef: Operation [0..1]	Currently not implemented.
	implementation: Implementation = Web Service {Web Service | Other | Unspecified}	Currently not implemented.
UserTask	inherits from Activity	The value of attribute type Task type (AT_BPMN_TASK_TYPE) is set to **User task** in the attribute type group **BPMN 2.0 attributes > Task attributes** associated with object type 'Function'. Object type: Function (OT_FUNC) Symbol: User task (ST_USER_TASK)

Class	BPMN attribute name	Implementation in ARIS
	Implementation: UserTaskImplementation = Other {HumanTaskWebService | WebService | Other | Unspecified}	Currently not implemented.
	renderings: Rendering [0..*]	Currently not implemented.
Manual Task	inherits from Activity	The value of attribute type Task type (AT_BPMN_TASK_TYPE) is set to **Manual task** in the attribute type group **BPMN 2.0 attributes > Task attributes** associated with object type 'Function'. Object type: Function (OT_FUNC) Symbol: Manual task (ST_MANUAL_TASK)
Business RuleTask	inherits from Activity	The value of attribute type Task type (AT_BPMN_TASK_TYPE) is set to **Business rule task** in the attribute type group **BPMN 2.0 attributes > Task attributes** associated with object type 'Function'. Object type: Function (OT_FUNC) Symbol: Business rule task (ST_BUSINESS_RULE_TASK)
	Implementation: BusinesRuleTaskImplementation = Other {BusinessRuleWebService | WebService | Other | Unspecified}	Currently not implemented.
ScriptTask	inherits from Activity	The value of attribute type Task type (AT_BPMN_TASK_TYPE) is set to **Script task** in the attribute type group **BPMN 2.0 attributes > Task attributes** associated with object type 'Function'. Object type: Function (OT_FUNC) Symbol: Script task (ST_SCRIPT_TASK)
	scriptLanguage: string [0..1]	Currently not implemented.
	script: string [0..1]	Currently not implemented.

10.4.1.4. Human Interactions

The elements User Task and Manual Task are relevant for modeling human interactions. Their attributes and model associations can be found in the chapter Activity type: Task [173]. They will be dealt with in detail when the execution design level is being implemented in ARIS.

See: Business Process Model and Notation (BPMN), Version 2.0.

10.4.1.5. Activity type: Sub-Process

See: Business Process Model and Notation (BPMN), Version 2.0.

BPMN 2.0 knows four Sub-Process types:

- Sub-Process (standard)
  (A standard Sub-Process corresponds to the Embedded Sub-Process used in BPMN 1.x.)
- Event Sub-Process
- Transaction
- Ad Hoc Sub-Process

Every Sub-Process type can be displayed as follows:

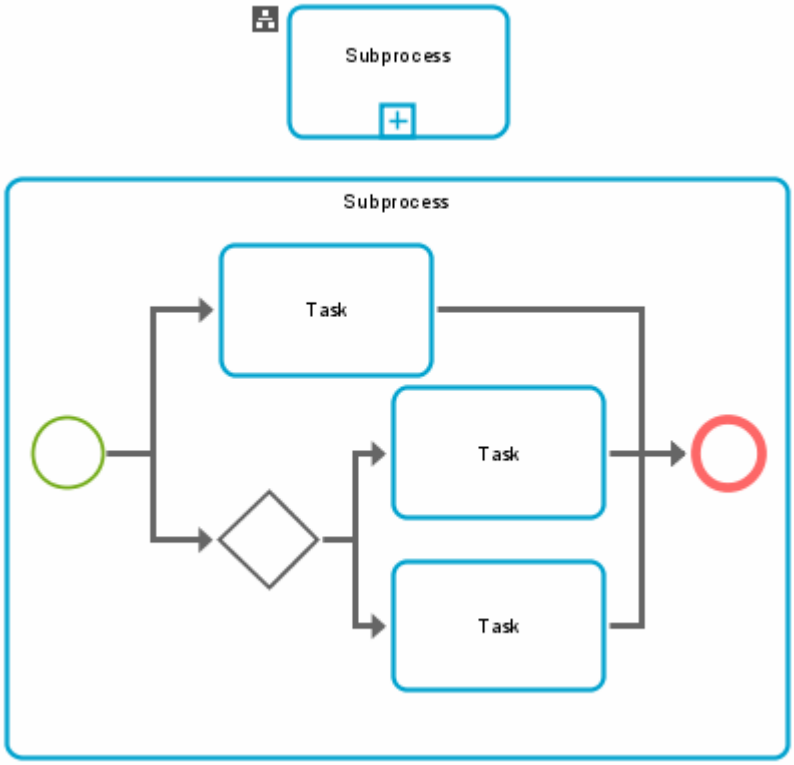
- Collapsed Sub-Process
- Expanded Sub-Process

Collapsed Sub-Processes have a special marker displayed at the bottom of the corresponding Sub-Process symbol:

10.4.1.6. Sub-Process type: Sub-Process

A standard Sub-Process has the same shape as a Task. In the collapsed form, the Sub-Process object is marked with a **+** to distinguish it from a Task. The element Expanded Sub-Process has no marker given that these elements show their **embedded** objects.

The ARIS symbol representing an Expanded Sub-Process is available in the **Symbols** bar, while the symbol representing a Collapsed Sub-Process is handled by the software.



Subprocess type: Subprocess

The attributes and model associations associated with a Sub-Process and their mapping to ARIS constructs are listed in the table below.

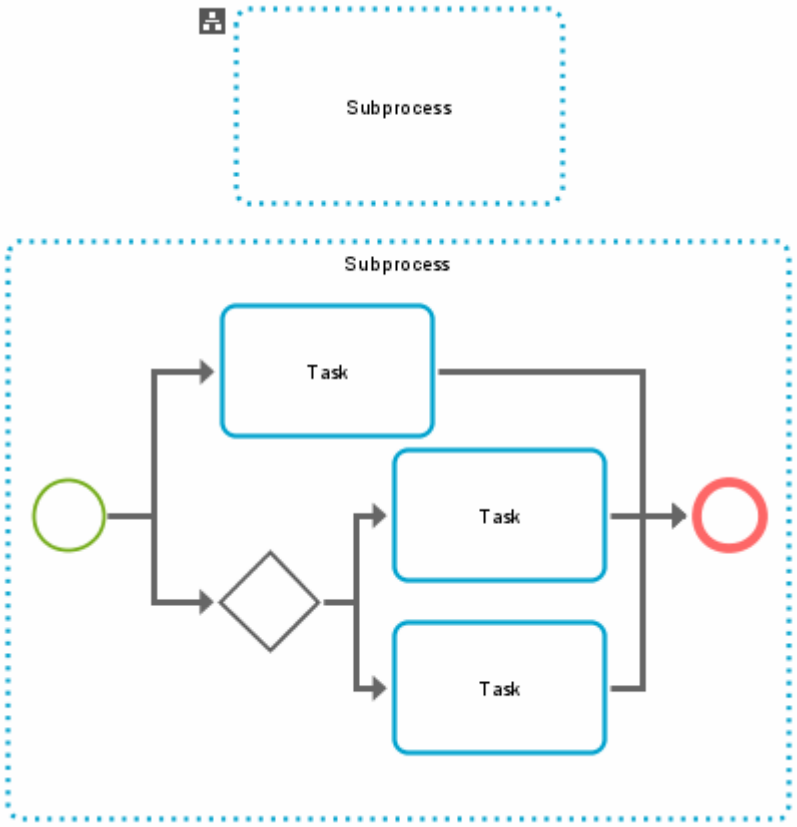
Mapping the attributes and model associations to ARIS

Class	BPMN attribute name	Implementation in ARIS
Sub-Process	inherits from Activity inherits from FlowElementsContainer	The value of attribute type Activity type (AT_BPMN_ACTIVITY_TYPE) is set to **Subprocess** in the attribute type group **BPMN 2.0 attributes** associated with object type Function (OT_FUNC). Object type: Function (OT_FUNC) Symbols: * Subprocess (ST_BPMN_SUBPROCESS) * Subprocess (collapsed) (ST_BPMN_SUB_PROCESS_COLLAPSED) * or a special subprocess symbol (see below)
	triggeredByEvent: boolean = false	Attribute type Event subprocess (AT_BPMN_EVENT_SUBPROCESS) in the attribute type group **BPMN 2.0 attributes > Subprocess attributes** associated with object type 'Function'. Object type: Function (OT_FUNC) Symbols: * Event subprocess (ST_BPMN_EVENT_SUBPROCESS) * Event subprocess (collapsed) (ST_BPMN_EVENT_SUBPROCESS_COLLAPSED) The symbols are rendered by the program.

10.4.1.7. Sub-Process type: Event Sub-Process

An Event Sub-Process is a specialized Sub-Process that is used within a Process or Sub-Process. Unlike a standard Sub-Process which uses the flow of the parent Process as a trigger, an Event Sub-Process is not a part of the normal flow of the parent Process; there is no incoming and no outgoing Sequence Flow. An Event Sub-Process has a Start Event with a trigger. Each time the Start Event is triggered while the parent Process is active, the Event Sub-Process will start.

The symbols used for an Event Sub-Process are shown below. If the Event Sub-Process is collapsed, the associated Start Event is used as a marker at the upper left corner of the symbol. The software renders this marker.



Subprocess type: Event subprocess

The Boolean ARIS attribute type **Event subprocess** is used in ARIS to represent the BPMN attribute **triggeredByEvent** (see table under Sub-Process type: Sub-Process [176]). This attribute type is read-only and used by the software.

10.4.1.8. Sub-Process type: Transaction

A Transaction Sub-Process (denoted with a double-lined boundary) is a specialized type of Sub-Process. In a Transaction Sub-Process, all activities must either complete successfully, or the Sub-Process must be rolled back to its original consistent state. A Transaction Sub-Process has a special behavior: It is associated with a transaction protocol used to verify that all activities were successfully completed. The corresponding symbol is not available in the **Symbols** bar in ARIS but is handled by the software. The program furthermore sets the value of the ARIS attribute type **Subprocess type** to **Transaction**.

Subprocess type: Transaction

A Transaction inherits from the element **Activity**. The attributes and model associations associated with a Transaction Sub-Process and their mapping to ARIS constructs are shown in the table below.

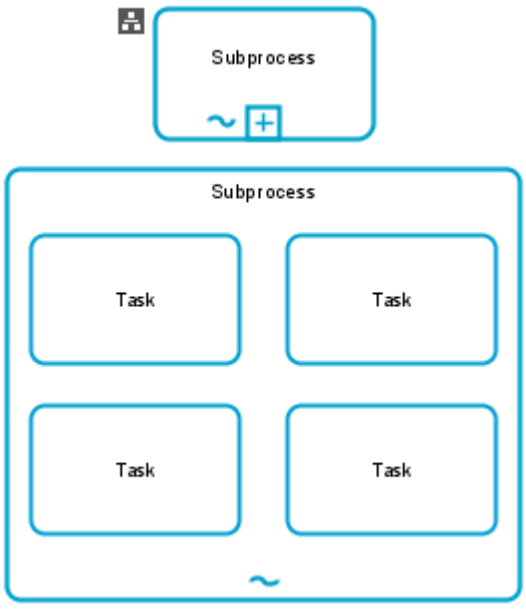
Mapping the attributes and model associations to ARIS

Class	BPMN attribute name	Implementation in ARIS
Transaction	inherits from Activity	The value of attribute type Subprocess type (AT_BPMN_SUBPROCESS_TYPE) is set to **Transaction** in the attribute type group **BPMN 2.0 attributes > Subprocess attributes** associated with object type 'Function'. Object type: Function (OT_FUNC) Symbols: * Transaction (ST_BPMN_TRANSACTION) * Transaction (collapsed) (ST_BPMN_TRANSACTION_COLLAPSED_1) The symbols are rendered by the software.
	protocol: string [0..1]	Currently not implemented.
	method: TransactionMethod = compensate { compensate | store | image }	Currently not implemented.

10.4.1.9. Sub-Process type: Ad Hoc Sub-Process

An Ad Hoc Sub-Process (denoted with a tilde marker) is a specialized type of Sub-Process. It contains a set of activities that can be performed. The use of a Sequence Flow between activities is optional in an Ad Hoc Sub-Process. The activities to be performed as well as the sequence and number of executions are determined by the performers of the activities. During the execution of the parent Process, one or more of these activities are allowed to be active.

ARIS Method provides the tilde marker as a mini-symbol for the value **Ad hoc subprocess** of the **Subprocess type** attribute. The program renders the symbols for the Ad hoc subprocess.



Subprocess type: Ad hoc subprocess

Mapping the attributes and model associations to ARIS

Class	BPMN attribute name	Implementation in ARIS
Ad Hoc Sub-Process	inherits from Activity	The value of attribute type Subprocess type (AT_BPMN_SUBPROCESS_TYPE) is set to **Ad hoc subprocess** in the attribute type group **BPMN 2.0 attributes > Subprocess attributes** associated with object type 'Function'. Object type: Function (OT_FUNC) Symbols: The mini-symbol **tilde** is rendered by the program.
	completionCondition: Expression	Attribute type **Ad hoc completion condition** (AT_BPMN_COMPLETION_CONDI) in the attribute type group **BPMN 2.0 attributes > Subprocess attributes > Ad hoc subprocess attributes** associated with object type 'Function'.
	ordering: AdHocOrdering = parallel { parallel | sequential }	Currently not implemented.
	cancelRemainingInstances: Boolean = True	Currently not implemented.

10.4.1.10. Sub-Process type: Call Activity

A Call Activity represents the invocation of either a reusable Global Task or a Process. The Call Activity represents the calling element, while the Global Task or Process represents the called element.

In ARIS, the symbol **Call activity** is available in the **Symbols** bar. If the modeler places this symbol, the value of the attribute **Activity type** is set to **Call activity** and the software provides a dialog where the modeler selects the task or the process being called. Depending on this selection, the value of the attribute type **Called element** is set to **Global task** or **Global process**. The program renders the symbol for the call activity. It corresponds to the symbol used for the called task or process but is drawn with a thick border.

If a task is selected, the program automatically creates a connection (Call activity **invokes** Task) at the definition level. If a process is selected, the related process diagram is assigned to the call activity.

Mapping the attributes and model associations to ARIS

Class	BPMN attribute name	Implementation in ARIS
CallActivity	inherits from Activity	The value of attribute type Activity type (AT_BPMN_ACTIVITY_TYPE) is set to **Call activity** in the attribute type group **BPMN 2.0 attributes** associated with object type 'Function'. Object type: Function (OT_FUNC) Symbol: The symbol depends on the activity being called and is rendered by the program.

Class	BPMN attribute name	Implementation in ARIS
	calledElement: CallableElement [0..1]	**For tasks**: The value of attribute type **Called element** (AT_BPMN_CALLED_ELEMENT) is set to **Global task**. The program creates the connection type **invokes** [CT_INVOKES] at the definition level: Function 'invokes' Function **For processes**: The value of attribute type **Called element** (AT_BPMN_CALLED_ELEMENT) is set to **Global process**. The BPMN process diagram of the called process is assigned to the call activity. In both cases, the software provides a corresponding dialog.

10.4.1.11. Global Tasks

The Global Task element is described in the chapter Callable Elements [160].

GlobalTask has no specific attributes or model associations; it inherits from CallableElement.

10.4.1.12. LoopCharacteristics

BPMN 2.0 provides two Activity types for modeling repeated Activities (including both Tasks and Sub-Processes):

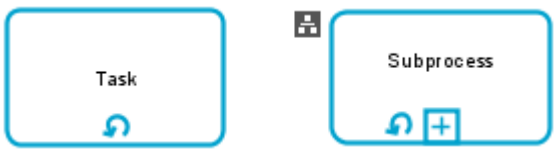
- Loop Activity (= standard loop)
- Multi-Instance Activity

10.4.1.13. LoopCharacteristics: Representations

An Activity can be specified to repeat based on a condition. Such an Activity is called **standard loop** in BPMN. A standard loop is equivalent to the **do while** and **do until** structure in programming. The number of iterations is unknown.

The Multi-Instance Activity is another type of repeated activity useful for performing actions from a list of items. A Multi-Instance Activity is equivalent to a **for each** structure in programming. The number of iterations is known when the activity starts as it corresponds to the number of items in the given list. Iterations of a Multi-Instance Activity can be performed concurrently or sequentially.

The marker for a standard loop is a circular arrow at the bottom center of the Activity symbol.



BPMN: Standard loop

The markers for Multi-Instance Activities are three bars at the bottom center of the Task or Sub-Process symbol.

Vertical bars are used to represent concurrent/parallel performances:

Symbols of BPMN multi-instance (parallel) activities

Horizontal bars are used to represent sequential performances:

Symbols for activities of the BPMN multi-instance (parallel)

LoopCharacteristics has no specific attributes; it inherits the attributes and associations from BaseElement.

In ARIS, the attribute type **Loop type** is used to specify whether the loop is a standard loop, a multi-instance parallel loop, or a multi-instance sequential loop. The attribute values are visualized by mini-symbols.

10.4.1.14. StandardLoopCharacteristics, MultiInstanceLoopCharacteristics, and ComplexBehaviorDefinition

The attributes and model associations associated with activities with Loop marker or Multi-Instance marker or associated with ComplexBehaviorDefinitions are summarized in the table below.

Mapping the attributes and model associations to ARIS

Class	BPMN attribute name	Implementation in ARIS
LoopCharacteristics	inherits from BaseElement	Attribute type group **Loop characteristics** (AT_BPMN_LOOP_CHARACTERISTICS) in the attribute type group **BPMN 2.0 attributes** associated with object type Function (OT_FUNC).
StandardLoop Characteristics	inherits from BaseElement	The value of attribute type Loop type (AT_BPMN_LOOP_TYPE_2) is set to **Standard loop** (AVT_BPMN_STANDARD_LOOP) in the attribute type group **BPMN 2.0 attributes > Loop characteristics** associated with object type Function (OT_FUNC).
	testBefore: boolean = False	Attribute type **Test before** (AT_BPMN_LOOP_TEST_TIME) in the attribute type group **BPMN 2.0 attributes > Loop characteristics > Standard loop attributes** associated with object type Function (OT_FUNC).
	loopMaximum: Expression [0..1]	Attribute type **Loop maximum** (AT_BPMN_MAX_LOOP) in the attribute type group **BPMN 2.0 attributes > Loop characteristics > Standard loop attributes** associated with object type Function (OT_FUNC).

Class	BPMN attribute name	Implementation in ARIS
	loopCondition: Expression [0..1]	Attribute type **Loop condition** (AT_BPMN_LOOP_CONDITION) in the attribute type group **BPMN 2.0 attributes > Loop characteristics > Standard loop attributes** associated with object type Function (OT_FUNC).
MultiInstanceLoop Characteristics	inherits from BaseElement	The value of attribute type Loop type (AT_BPMN_LOOP_TYPE_2) is set to **Multi-instance sequential loop** (AVT_BPMN_MULTI_INSTANCE_SEQUENTIAL_LOOP) or **Multi-instance parallel loop** (AVT_BPMN_MULTI_INSTANCE_PARALLEL_LOOP) in the attribute type group **BPMN 2.0 attributes > Loop characteristics** associated with object type Function (OT_FUNC).
	isSequential: boolean = False	isSequential = true corresponds to: Loop type = Multi-instance sequential loop isSequential = false corresponds to: Loop type = Multi-instance parallel loop
	loopCardinality: Expression [0..1]	Attribute type **Loop cardinality** (AT_BPMN_LOOP_CARDINALITY) in the attribute type group **BPMN 2.0 attributes > Loop characteristics > Multi-instance loop attributes** assoiated with object type Function (OT_FUNC).
	loopDataInput: DataInput [0..1]	Currently not implemented.
	loopDataOutput: DataOutput [0..1]	Currently not implemented.
	inputDataItem: Property [0..1]	Currently not implemented.
	outputDataItem: Property [0..1]	Currently not implemented.
	completionCondition: Expression [0..1]	Currently not implemented.
	behavior: MultiInstanceBehavior = all { none | one | all | complex }	Currently not implemented.
	complexBehaviorDefinition: ComplexBehaviorDefinition [0..*]	Currently not implemented.
	oneBehaviorEventRef: EventDefinition [0..1]	Currently not implemented.
	noneBehaviorEventRef: EventDefinition [0..1]	Currently not implemented.
ComplexBehavior Definition	inherits from BaseElement	Currently not implemented.
	condition: Formal Expression	
	event: ImplicitThrowEvent	

10.4.2. Items and Data

As mentioned above, the current implementation of BPMN 2.0 in ARIS focuses on the business process level. Therefore, only data objects and data stores are provided, either as input for or as output of activities. Detailed data modeling aspects (for example, data structures, data states, data associations) are omitted.

See: Business Process Model and Notation (BPMN), Version 2.0.

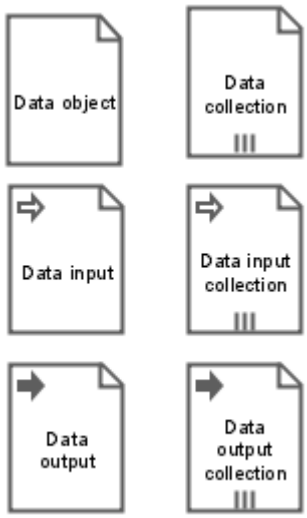
10.4.2.1. Data Objects

In BPMN 1.x, data were considered as an Artifact, whereas in BPMN 2.0, DataObject elements were upgraded to objects in the BPMN metamodel.

At the business level, where the data structures and mappings are not included, data objects are represented in ARIS by six symbols used for the object type **Cluster/Data model**:

- Data object
- Data collection
- Data input
- Data input collection
- Data output
- Data output collection

When a data object is placed, the object symbols **Data object**, **Data input**, and **Data output** are provided. Furthermore, the object symbols **Data collection**, **Data input collection**, and **Data output collection** can be selected using the **Object appearance** page of the **Object properties** dialog.



BPMN: Data object symbols

Only the **Data object** symbol is available in the **Symbols** bar in ARIS.

When combined with the following connection types, data objects can represent either the input for or output from activities:

- Cluster/Data model **is input for** Function
- Function **has as output** Cluster/Data model

The data input symbols must not be the target of a **has as output** connection, and the data output symbols must not be the source of an **is input for** connection. The software ensures this.

Mapping the attributes and model associations to ARIS

Class	BPMN attribute name	Implementation in ARIS
ItemAwareElement	inherits from BaseElement	
	itemSubjectRef: ItemDefinition [0..1]	Currently not implemented.
	dataState: DataState [0..1]	Currently not implemented.
DataObject	inherits from FlowElement and ItemAwareElement	Object type: Cluster/Data model (OT_CLST) Six symbols: * Data object (ST_BPMN_DATA_OBJECT) * Data collection (ST_BPMN_DATA_COLLECTION) * Data input (ST_BPMN_DATA_INPUT) * Data input collection (ST_BPMN_DATA_INPUT_COLLECTION) * Data output (ST_BPMN_DATA_OUTPUT) * Data output collection (ST_BPMN_DATA_OUTPUT_COLLECTION)
	isCollection: Boolean = False	Represented by special symbols used for the object type **Cluster/Data model** (OT_CLST)

10.4.2.2. Data Stores

Unlike the Data Object element, which lives only as long as the process instance is running, a Data Store represents information that persists beyond the lifetime of a particular process.

At the business level, a Data Store is represented in ARIS by the symbol **Data store** associated with the object type **Information carrier**. This symbol is available in the **Symbols** bar.

When combined with the following ARIS connection types, data stores can represent either the input for or the output of activities:

- Information carrier **provides input for** Function
- Function **creates output to** Information carrier.

BPMN: Data store symbol

DataStore inherits from FlowElement and ItemAwareElement.

Mapping the attributes and model associations to ARIS

Class	BPMN attribute name	Implementation in ARIS
DataStore	inherits from FlowElement and ItemAwareElement	Object type: Information carrier (OT_INFO_CARR) Symbol: Data store (ST_BPMN_DATA_STORE)
	name: string	Attribute type Name (AT_NAME) of object type Information carrier (OT_INFO_CARR)
	capacity: Integer [0..1]	Currently not implemented.
	isUnlimited: Boolean = False	Currently not implemented.
DataStore Reference	inherits from FlowElement and ItemAwareElement	
	dataStoreRef: DataStore	Occurrence copies of the (referenced) data store.

10.4.3. Events

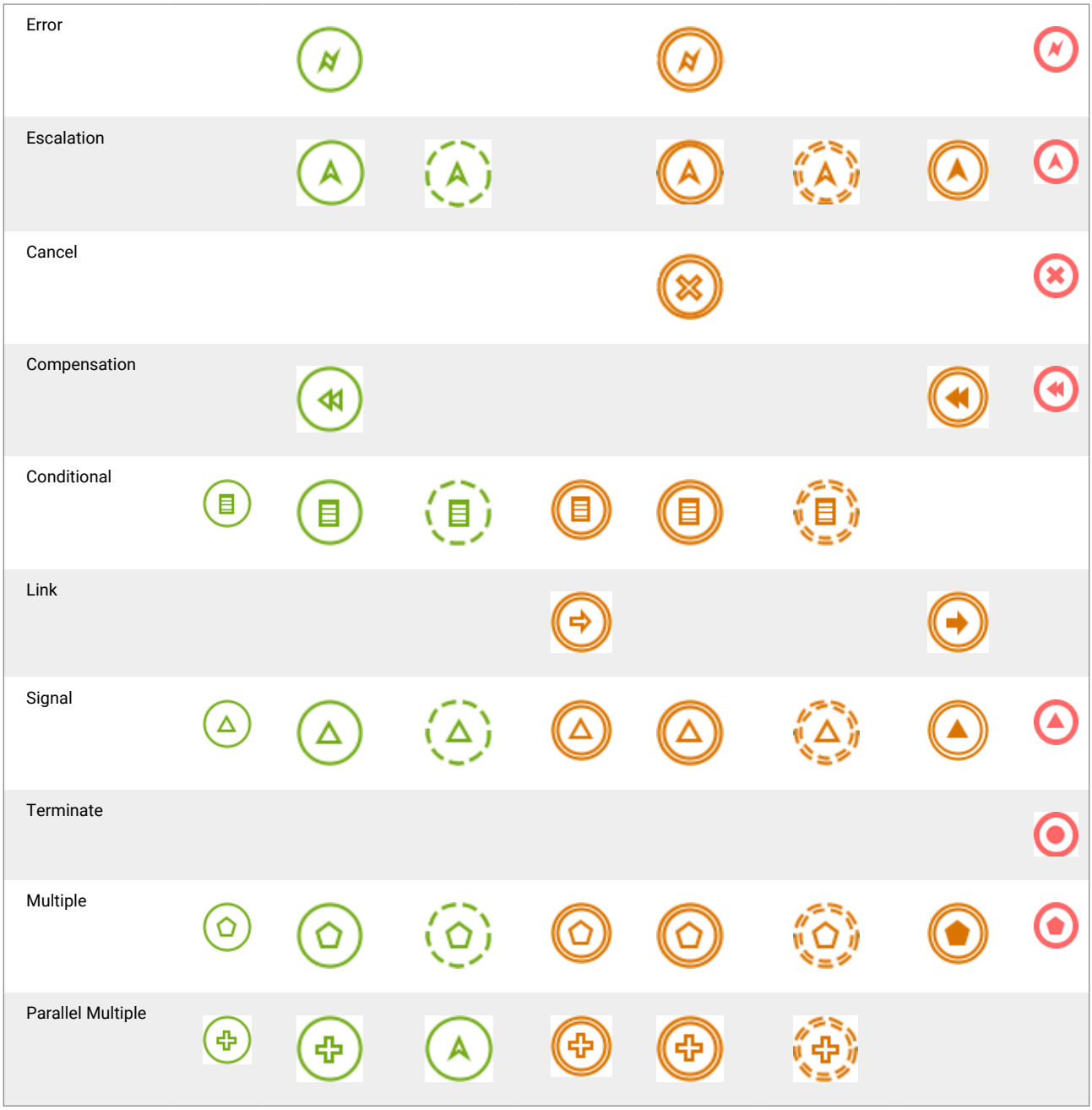
BPMN Events are represented in ARIS by the object type **Event**. Altogether, there are sixty-three symbols available in BPMN 2.0. The main event types include the following:

- Start event
- Intermediate event
- End event

In ARIS, only these three types of events are provided in the **Symbols** bar (see table: Types = None). The remaining symbols are provided by ARIS Method.

BPMN Events (see: Business Process Model and Notation (BPMN), version 2.0, page 233 and the following):

Types	Start			Intermediate				End
	Top level	Event Sub-process Inter-rupting	Event Sub-process Non-Inter-rupting	Catching	Boundary Inter-rupting	Boundary Non-Inter-rupting	Throwing	
None								
Message								
Timer								



10.4.3.1. Catch Events and Throw Events

Events can be of the following types:

- Catch Event (all Start Events and some specific Intermediate Events)
- Throw Event (all End Events and some specific Intermediate Events)

Mapping the attributes and model associations to ARIS

Class	BPMN attribute name	Implementation in ARIS
Event		Object type: Event (OT_EVT) Symbols: Sixty-three different symbols (see below)
CatchEvent	inherits from Event	Object type: Event (OT_EVT) Symbols: Different 'Start event' or 'Intermediate event' symbols
	eventDefinitionRefs: EventDefinition [0..*]	Occurrence copy of the corresponding throw event.
	eventDefinitions: EventDefinition [0..*]	Attribute type **Event definition** (AT_BPMN_EVENT_DEFINITION) in the attribute type group **BPMN 2.0 attributes** associated with object type Event (OT_EVT). The values of this attribute type include None, Message, Timer, Error, Escalation, Cancel, Compensation, Conditional, Link, Signal, Multiple, Parallel multiple (as a special case of Multiple). Each event definition has a specific marker inside the event symbol.
	dataOutputAssociations: DataOutputAssociation [0..*]	Currently not implemented.
	dataOutput: dataOutput [0..*]	Connection type 'has as output' in the model types BPMN process diagram (BPMN 2.0) and BPMN collaboration diagram (BPMN 2.0): Event (symbol: only catch events) 'has as output' Cluster/Data model
	outputSet: OutputSet [0..1]	Currently not implemented.
ThrowEvent	inherits from Event	Object type: Event (OT_EVT) Symbols: Different 'Intermediate event' or 'End event' symbols
	eventDefinitionRefs: EventDefinition [0..*]	Occurrence copy of the corresponding catch event.
	eventDefinitions: EventDefinition [0..*]	Attribute type **Event definition** (AT_BPMN_EVENT_DEFINITION) in the attribute type group **BPMN 2.0 attributes** associated with object type Event (OT_EVT). The values of this attribute type include None, Message, Error, Escalation, Cancel, Compensation, Link, Signal, Terminate, Multiple. Each event definition has a specific marker inside the event symbol.
	dataInputAssociations: DataInputAssociation [0..*]	Currently not implemented.
	dataInput: DataInput [0..*]	Connection type 'is input for' in the model types BPMN process diagram (BPMN 2.0) and BPMN collaboration diagram (BPMN 2.0): Cluster/Data model 'is input for' Event (symbol: only throw events)
	inputSet: InputSet [0..1]	Currently not implemented.
ImplicitThrow Event	inherits from ThrowEvent	Currently not implemented.

10.4.3.2. Start Events

The symbols used for Start Event elements are depicted in the chapter Events [186].

In ARIS, only the start event **None** is available in the **Symbols** bar. When this start event is placed in ARIS, the modeler is guided by a special functionality of the program.

Mapping the attributes and model associations to ARIS

Class	BPMN attribute name	Implementation in ARIS
StartEvent	inherits from CatchEvent	Object type: Event (OT_EVT) Symbol: Start event (ST_BPMN_START_EVENT)
	isInterrupting: boolean	Interrupting start events are represented by specific event symbols.

10.4.3.3. Intermediate Events

The symbols used for Intermediate Event elements are depicted in the chapter Events [186].

In ARIS, only the intermediate event **None** is available in the **Symbols** bar. When this start event is placed in ARIS, the modeler is guided by a special functionality of the program. Intermediate events have no specific attributes and associations.

Mapping the attributes and model associations to ARIS

Class	BPMN attribute name	Implementation in ARIS
IntermediateEvent		Object type: Event (OT_EVT) Symbol: Intermediate event (ST_BPMN_INTERMEDIATE_EVENT)

Some Intermediate Event types can be attached to the boundary of activities. They are called **Boundary Event** (see table columns **Boundary Interrupting** and **Boundary Non-Interrupting** in the chapter Events [186].

Boundary Events are always Catch Events. Their attributes and model associations are shown in the table below.

Mapping the attributes and model associations to ARIS

Class	BPMN attribute name	Implementation in ARIS
BoundaryEvent	inherits from CatchEvent	Boundary events are specific intermediate events.
	AttachedTo: Activity	Connection type 'can trigger' in the model types BPMN process diagram (BPMN 2.0) and BPMN collaboration diagram (BPMN 2.0): Function 'can trigger' (CT_BPMN_CAN_TRIGGER) Event (symbol: Intermediate event)
	CancelActivity: boolean	Non-interrupting events are represented by specific event symbols.

10.4.3.4. End Events

The symbols used for End Event elements are depicted in the chapter Events [186].

In ARIS, only the end event **None** is available in the **Symbols** bar. When this start event is placed in ARIS, the modeler is guided by a special functionality of the program.

Mapping the attributes and model associations to ARIS

Class	BPMN attribute name	Implementation in ARIS
EndEvent		Object type: Event (OT_EVT) Symbol: End event (ST_BPMN_END_EVENT)

10.4.3.5. Event Definitions

BPMN 2.0 distinguishes the following EventDefinition elements: None, Message, Timer, Error, Escalation, Cancel, Compensation, Conditional, Link, Signal, Terminate, Multiple (and Parallel Multiple as a special case of Multiple). The various definitions are visualized by specific markers placed within the **None Start Event**, **Intermediate Event**, and **End Event** symbols.

Mapping the attributes and model associations to ARIS

Class	BPMN attribute name	Implementation in ARIS
EventDefinition	inherits from BaseElement	Attribute type **Event definition** (AT_BPMN_EVENT_DEFINITION) in the attribute type group **BPMN 2.0 attributes** associated with object type Event (OT_EVT). Attribute values: None, Message, Timer, Error, Escalation, Cancel, Compensation, Conditional, Link, Signal, Terminate, Multiple. This attribute is read-only and automatically set by the software.
CancelEvent Definition	inherits from BaseElement	Object type: Event (OT_EVT) Symbols: * Cancel intermediate event (ST_BPMN_CANCEL_INTERMEDIATE_EVENT) * Cancel end event (ST_BPMN_CANCEL_END_EVENT)
CompensationEvent Definition	inherits from BaseElement	Object type: Event (OT_EVT) Symbols: * Compensation start event (ST_BPMN_COMPENSATION_START) * Compensation intermediate event (catch) (ST_BPMN_COMPENSATION_INTERMEDIATE_CATCH) * Compensation intermediate event (throw) (ST_BPMN_COMPENSATION_INTERMEDIATE_THROW) * Compensation end event (ST_BPMN_COMPENSATION_END_EVENT)
	activityRef: Activity [0..1]	Attribute type **Wait for completion** (AT_BPMN_WAIT_FOR_COMPLETION) in the attribute type group **BPMN 2.0 attributes > Compensation event attributes** associated with object type Event (OT_EVT).

Class	BPMN attribute name	Implementation in ARIS
ConditionalEvent Definition	inherits from BaseElement	Object type: Event (OT_EVT) Symbols: * Conditional start event (ST_BPMN_RULE_START_EVENT) * Conditional start event (non-interrupting) (ST_BPMN_CONDITIONAL_START_NI) * Conditional intermediate event (ST_BPMN_RULE_INTERMEDIATE_EVENT) * Conditional intermediate event (non-interrupting) (ST_BPMN_CONDITIONAL_INTERMEDIATE_NI)
	condition: Expression	Attribute type **Condition** (AT_BPMN_RULE_EXPRESSION) in the attribute type group **BPMN 2.0 attributes > Conditional event attributes** associated with object type Event (OT_EVT).
ErrorEventDefinition	inherits from BaseElement	Object type: Event (OT_EVT) Symbols: * Error start event (ST_BPMN_ERROR_START) * Error intermediate event (ST_BPMN_ERROR_INTERMEDIATE_EVENT) * Error end event (ST_BPMN_ERROR_END_EVENT)
	errorCode: string	Currently not implemented.
	error: Error [0..1]	Currently not implemented.
EscalationEvent Definition	inherits from BaseElement	Object type: Event (OT_EVT) Symbols: * Escalation start event (ST_BPMN_ESCALATION_START) * Escalation start event (non-interrupting) (ST_BPMN_ESCALATION_START_NI) * Escalation intermediate event (catch) (ST_BPMN_ESCALATION_INTERMEDIATE_CATCH) * Escalation intermediate event (non-interrupting) (ST_BPMN_ESCALATION_INTERMEDIATE_NI) * Escalation intermediate event (throw) (ST_BPMN_ESCALATION_INTERMEDIATE_THROW) * Escalation end event (ST_BPMN_ESCALATION_END)
	escalationCode: string	Currently not implemented.
	escalationRef: Escalation [0..1]	Currently not implemented.
LinkEventDefinition	inherits from BaseElement	Object type: Event (OT_EVT) Symbols: * Link intermediate event (catch) (ST_BPMN_LINK_INTERMEDIATE_CATCH) * Link intermediate event (throw) (ST_BPMN_LINK_INTERMEDIATE_THROW) These catch and throw link events are referred to each other by occurrence copies.
	name: string	Attribute type Name (AT_NAME) of object type Event (OT_EVT).

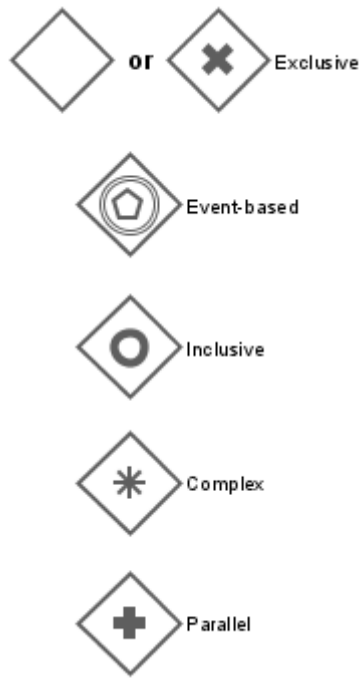
Class	BPMN attribute name	Implementation in ARIS
MessageEvent Definition	inherits from BaseElement	Object type: Event (OT_EVT) Symbols: * Message start event (ST_BPMN_MESSAGE_START_EVENT) * Message start event (non-interrupting) (ST_BPMN_MESSAGE_START_NI) * Message intermediate event (catch) (ST_BPMN_MESSAGE_INTERMEDIATE_CATCH) * Message intermediate event (non-interrupting) (ST_BPMN_MESSAGE_INTERMEDIATE_NI) * Message intermediate event (throw) (ST_BPMN_MESSAGE_INTERMEDIATE_THROW) * Message end event (ST_BPMN_MESSAGE_END_EVENT)
	MessageRef: Message [0..1]	Currently not implemented.
	operationRef: Operation [0..1]	Currently not implemented.
Multiple Event		Object type: Event (OT_EVT) Symbols: * Multiple start event (ST_BPMN_MULTIPLE_START_EVENT) * Multiple start event (non-interrupting) (ST_BPMN_MULTIPLE_START_NI) * Multiple intermediate event (catch) (ST_BPMN_MULTIPLE_INTERMEDIATE_CATCH) * Multiple intermediate event (non-interrupting) (ST_BPMN_MULTIPLE_INTERMEDIATE_NI) * Multiple intermediate event (throw) (ST_BPMN_MULTIPLE_INTERMEDIATE_THROW) * Multiple end event (ST_BPMN_MULTIPLE_END_EVENT)
None Event		Object type: Event (OT_EVT) Symbols: * Start event (ST_BPMN_START_EVENT) * Intermediate event (ST_BPMN_INTERMEDIATE_EVENT) * End event (ST_BPMN_END_EVENT) These symbols are available in the **Symbols** bar.
Parallel Multiple Event		Object type: Event (OT_EVT) Symbols: * Parallel multiple start event (ST_BPMN_PARALLEL_MULTIPLE_START) * Parallel multiple start event (non-interrupting) (ST_BPMN_PARALLEL_MULTIPLE_START_NI) * Parallel multiple intermediate event (ST_BPMN_PARALLEL_MULTIPLE_INTERMEDIATE) * Parallel multiple intermediate event (non-interrupting) (ST_BPMN_PARALLEL_MULTIPLE_INTERMEDIATE_NI)

Class	BPMN attribute name	Implementation in ARIS
SignalEventDefinition	inherits from BaseElement	Object type: Event (OT_EVT) Symbols: * Signal start event (ST_BPMN_SIGNAL_START_EVENT) * Signal start event (non-interrupting) (ST_BPMN_SIGNAL_START_NI) * Signal intermediate event (catch) (ST_BPMN_SIGNAL_INTERMEDIATE_EVENT) * Signal intermediate event (non-interrupting) (ST_BPMN_SIGNAL_INTERMEDIATE_NI) * Signal intermediate event (throw) (ST_BPMN_SIGNAL_INTERMEDIATE_THROW) * Signal end event (ST_BPMN_SIGNAL_END_EVENT)
	signalRef: Signal	Currently not implemented.
TerminateEvent Definition	inherits from BaseElement	Object type: Event (OT_EVT) Symbol: * Terminate end event (ST_BPMN_TERMINATE_END_EVENT)
TimerEventDefinition	inherits from BaseElement	Object type: Event (OT_EVT) Symbols: * Timer start event (ST_BPMN_TIMER_START_EVENT) * Timer start event (non-interrupting) (ST_BPMN_TIMER_START_NI) * Timer intermediate event (ST_BPMN_TIMER_INTERMEDIATE_EVENT) * Timer intermediate event (non-interrupting) (ST_BPMN_TIMER_INTERMEDIATE_NI)
	timeDate: Expression [0..1]	Attribute type **Time date** (AT_BPMN_TIMEDATE) in the attribute type group **BPMN 2.0 attributes > Timer event attributes** associated with object type Event (OT_EVT).
	timeCycle: Expression [0..1]	Attribute type **Time cycle** (AT_BPMN_TIMECYCLE) in the attribute type group **BPMN 2.0 attributes > Timer event attributes** associated with object type Event (OT_EVT).

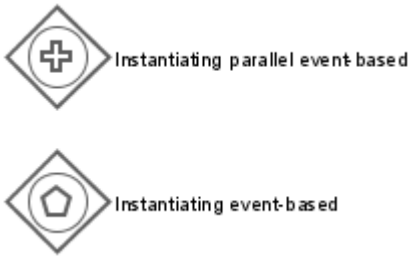
10.4.4. Gateways

The ARIS object type **Rule** depicts the BPMN element **Gateway**. Although BPMN 2.0 knows five different Gateway types, only one symbol is available in the ARIS **Symbols** bar:

The remaining Gateway symbols are handled by the program. The following figure depicts all (basic) Gateway symbols.



For the element Event-Based Gateway, there are two additional symbols which are used to start a process:



ARIS Method provides eight gateway symbols. Contrary to events, an ARIS attribute recording the gateway type is not required. It is up to the modeler to ensure that gateways are used in a semantically correct way. Modelers should not reuse gateways.

10.4.4.1. Exclusive Gateways

Mapping the attributes and model associations to ARIS

Class	BPMN attribute name	Implementation in ARIS
Exclusive Gateway	inherits from Gateway	Object type: Rule (OT_RULE) Symbols: * Gateway (ST_BPMN_RULE_1) * Exclusive gateway (ST_BPMN_RULE_XOR_3) * Event-based gateway (ST_BPMN_RULE_XOR_4)

Class	BPMN attribute name	Implementation in ARIS
	default: SequenceFlow [0..1]	Attribute type **Sequence flow condition** (AT_BPMN_SEQ_FLOW_CONDITION) in the attribute type group **BPMN 2.0 attributes** associated with the following connection types: * Rule 'leads to' (CT_LEADS_TO_2) Event * Rule 'activates' (CT_ACTIV_1) Function * Rule 'links' (CT_LNK_2) Rule The attribute value must be set to Default sequence flow. The symbol (slash) is automatically set by the software.

10.4.4.2. Inclusive Gateways

Mapping the attributes and model associations to ARIS

Class	BPMN attribute name	Implementation in ARIS
Inclusive Gateway	inherits from Gateway	Object type: Rule (OT_RULE) Symbol: * Inclusive gateway (ST_BPMN_RULE_OR_1)
	default: SequenceFlow [0..1]	See: Exclusive Gateways

10.4.4.3. Parallel Gateways

Parallel Gateway elements have no specific attributes.

Mapping the attributes and model associations to ARIS

Class	BPMN attribute name	Implementation in ARIS
Parallel Gateway	inherits from Gateway	Object type: Rule (OT_RULE) Symbol: * Parallel gateway (ST_BPMN_RULE_AND_1)

10.4.4.4. Complex Gateways

Mapping the attributes and model associations to ARIS

Class	BPMN attribute name	Implementation in ARIS
Complex Gateway	inherits from Gateway	Object type: Rule (OT_RULE) Symbol: * Complex gateway (ST_BPMN_RULE_COMPLEX_1)
	activationCondition: Expression [0..1]	Attribute type **Activation condition** (AT_ACTIVATION_CONDITION) in the attribute type group **BPMN 2.0 attributes** associated with object type Rule (OT_RULE)

10.4.4.5. Event-Based Gateways

All attributes are represented by specific Gateway symbols.

Mapping the attributes and model associations to ARIS

Class	BPMN attribute name	Implementation in ARIS
Event-Based Gateway	inherits from Gateway	Object type: Rule (OT_RULE) Symbols: * Event-based gateway (ST_BPMN_RULE_XOR_4) * Instantiating event-based gateway (ST_BPMN_RULE_XOR_START) * Instantiating parallel event-based gateway (ST_BPMN_RULE_XOR_PARALLEL)
	instantiate: boolean = False	Represented by symbols. **True** if: * Instantiating event-based gateway (ST_BPMN_RULE_XOR_START) * Instantiating parallel event-based gateway (ST_BPMN_RULE_XOR_PARALLEL) **False** if: * Event-based gateway (ST_BPMN_RULE_XOR_4)
	eventGatewayType: EventGatewayType = Exclusive { Exclusive | Parallel }	Represented by symbols: **Exclusive** if: * Event-based gateway (ST_BPMN_RULE_XOR_4) * Instantiating event-based gateway (ST_BPMN_RULE_XOR_START) **Parallel** if: * Instantiating parallel event-based gateway (ST_BPMN_RULE_XOR_PARALLEL)

10.4.5. Lanes

A Lane is a subdivision of a Process or a Pool. Lanes have no semantics in BPMN. BPMN 2.0 uses Lanes as a means to categorize Flow Elements. Most often, Lanes represent organizational elements, but in principle,

any categorization may be used for them. Lanes can contain nested sub-Lanes. A LaneSet specifies the categorization represented by the Lanes.

See: Business Process Model and Notation (BPMN), Version 2.0.

Like a Pool, a Lane is drawn as a rectangular box. Its label is not boxed off.

Nested lanes

Mapping the attributes and model associations to ARIS

Class	BPMN attribute name	Implementation in ARIS
LaneSet	inherits from BaseElement	Object type: Lane (OT_BPMN_LANE) Symbol: Lane (ST_BPMN_LANE_1)
	process: Process	Model of type BPMN process diagram (BPMN 2.0) that contains the lane(s).
	lanes: Lane [0..*]	Object type: Lane (OT_BPMN_LANE) Symbol: Lane (ST_BPMN_LANE_1) Source object used with the following connection type: Lane 'belongs to' (CT_BELONGS_TO_1) Lane
	parentLane: Lane [0..1]	Target object used with the following connection type: Lane 'belongs to' (CT_BELONGS_TO_1) Lane
Lane	inherits from BaseElement	Object type: Lane (OT_BPMN_LANE) Symbol: Lane (ST_BPMN_LANE_1)
	name: string	Attribute type Name (AT_NAME) of object type Lane (OT_BPMN_LANE)
	partitionElement: BaseElement [0..1]	Currently not implemented.
	partitionElementRef: BaseElement [0..1]	Currently not implemented.
	childLaneSet: LaneSet [0..1]	Source object used with the following connection type: Lane 'belongs to' (CT_BELONGS_TO_1) Lane

Class	BPMN attribute name	Implementation in ARIS
	flowElementRefs: FlowElement [0..*]	Source objects used with the 'belongs to' (CT_BELONGS_TO_1) connection type: * Function 'belongs to' Lane * Event 'belongs to' Lane * Rule 'belongs to' Lane * Cluster/Data model 'belongs to' Lane * Information carrier 'belongs to' Lane

10.5. Collaborations

See: Business Process Model and Notation (BPMN), Version 2.0.

A Collaboration shows message exchanges between Participants. A Collaboration contains at least two Pools representing the Participants. A Pool can contain a Process (in which case it is represented as a white box with all details exposed), or a Pool can be empty (in which case it is represented as a black box with all details hidden). The message exchanges between the Participants are represented by Message Flows that connect two Pools (or the objects within the Pools). Only one Pool may be represented without a boundary.

In ARIS, the model type **BPMN collaboration diagram (BPMN 2.0)** is available to model collaborations.

Mapping the attributes and model associations to ARIS

Class	BPMN attribute name	Implementation in ARIS
Collaboration	inherits from BaseElement and InteractionSpecification	Model type: BPMN collaboration diagram (BPMN 2.0) (MT_BPMN_COLLABORATION_DIAGRAM)
	name: string	Attribute type Name (AT_NAME) associated with model type BPMN collaboration diagram (BPMN 2.0).
	choreographyRef: Choreography [0..1]	Currently not implemented.
	conversationAssociations: ConversationAssociation [0..*]	The relationships to conversations are represented by occurrence copies of pools (OT_BPMN_POOL; ST_BPMN_POOL_1), by occurrence copies of message flow connections, and by the assignment of a model of type **BPMN collaboration diagram (BPMN 2.0)** to the object type Conversation (OT_BPMN_CONVERSATION).
	conversations: Conversation [0..*]	Model of type BPMN collaboration diagram (BPMN 2.0) assigned to object type Conversation (OT_BPMN_CONVERSATION).
	artifacts: Artifact [0..*]	Artifacts [157]
	participantAssociations: ParticipantAssociations [0..*]	The relationships to participants are represented by occurrence copies of pools (OT_BPMN_POOL; ST_BPMN_POOL_1).
	messageFlowAssociations: Message flow association [0..*]	The relationships to message flows are represented by occurrence copies of 'message flow' connections (and of the involved participants).
	IsClosed: boolean = false	Attribute type **Is closed** (AT_BPMN_IS_CLOSED) in the attribute type group **BPMN 2.0 attributes** associated with model type BPMN collaboration diagram (BPMN 2.0).

The object types and connection types associated with the model type **BPMN collaboration diagram (BPMN 2.0)** are detailed in the following chapters.

10.5.1. Pools and Participants

The elements Pool and Participant play a central role in Collaborations. They are described in detail in the chapter Participants [165].

10.5.2. Object types and connection types reused from a process

Given that a pool can reference a process (in which case it is represented as a white box), all object and connection types that are allowed in the ARIS model type BPMN process diagram (BPMN 2.0) are also available in the ARIS model of type BPMN collaboration diagram (BPMN 2.0).

The object and connection types taken over from the 'BPMN process diagram (BPMN 2.0)' model type are described in detail in the chapter Processes [170].

The connection type **belongs to** is used to embed the object types of a visible process into a pool.

Source object type	Connection type	Target object type
Event (OT_EVT)	belongs to (CT_BELONGS_TO_1)	Participant (OT_BPMN_POOL)
Function (OT_FUNC)	belongs to (CT_BELONGS_TO_1)	Participant (OT_BPMN_POOL)
Rule (OT_RULE)	belongs to (CT_BELONGS_TO_1)	Participant (OT_BPMN_POOL)
Lane (OT_BPMN_LANE)	belongs to (CT_BELONGS_TO_1)	Participant (OT_BPMN_POOL)
Cluster/Data model (OT_CLST)	belongs to (CT_BELONGS_TO_1)	Participant (OT_BPMN_POOL)
Information carrier (OT_INFO_CARR)	belongs to (CT_BELONGS_TO_1)	Participant (OT_BPMN_POOL)

10.5.3. Message Flows

A Message Flow between Participants is represented in ARIS by the connection type **message flow**. This connection type is used to connect two pools or the objects within the pools. The attributes and model associations associated with the Message Flow element are described in the chapter Message Flows [163].

To show the messages being exchanged in message flows, the ARIS object type **Message** (represented by the symbol type **Message**) is used, and the **message flow** connection type is replaced by the connection types **sends** and **is received from**. This workaround is required due to the fact that it is not possible in ARIS to assign object types to connection types.

The program displays the **sends** and **is received from** connection types like a regular message flow.

10.6. Conversations

See: Business Process Model and Notation (BPMN), version 2.0, page 124 and the following.

The BPMN **Conversation diagram** was introduced with BPMN 2.0 to provide a big picture of the interactions (in terms of related message exchanges) between collaborating Participants.

The BPMN **Conversation diagram** is similar to the BPMN **Collaboration diagram**, but its Pools are not allowed to contain a Process, and a Choreography is not allowed between the Pools.

The BPMN **Conversation diagram** uses three basic elements:

- Conversation Node (Communication, Sub-Conversation)
- Participant (Pool)
- Conversation Link (ARIS connection types **message flow** and **participates in**)

They are described in the next chapters.

10.6.1. Conversation Nodes

The attributes and model associations associated with Conversation and Sub-Conversation elements are summarized in the table below.

Mapping the attributes and model associations to ARIS

Class	BPMN attribute name	Implementation in ARIS
Conversation	inherits from CallableElement, InteractionNode, ConversationNode	
	correlationKeys: CorrelationKey [0..*]	Currently not implemented.
	messageFlowRefs: MessageFlow [0..*]	Occurrence copies of 'message flow' connections (and of the involved participants)
ConversationNode	inherits from BaseElement	Model type: BPMN conversation diagram (BPMN 2.0) (MT_BPMN_CONVERSATION_DIAGRAM)
	conversationNodes: ConversationNode [0..*]	See below.
	artifacts: Artifact [0..*]	See chapter Artifacts [157].

10.6.2. Conversation Symbols

BPMN 2.0 distinguishes three Conversation Node sub-types:

- Communication
- Sub-Conversation

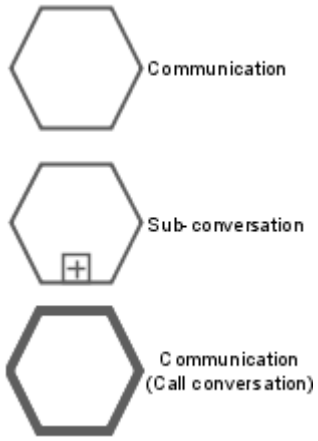
- Call Conversation

  (The concept of Call Conversation elements is not clear, which is why the type Call Conversation is ignored in the current implementation of BPMN 2.0 in ARIS. However, call conversations can be particularly distinguished in ARIS. See description below.)

Conversation nodes are represented in ARIS by the object type **Conversation**.

A **Communication** is an atomic Conversation element in a BPMN **Conversation diagram** and represents a set of Message Flows grouped together based on a single Correlation Key. A Communication involves at least two Participants.

The Communication symbol is available in the **Symbols** bar of the ARIS model type **BPMN conversation diagram (BPMN 2.0)**.



Symbols of Conversation nodes

A **Sub-Conversation** is a Conversation that is modeled with a higher level of detail in a separate BPMN **Conversation diagram**. This diagram is assigned to the corresponding Sub-Conversation object. The Conversation Nodes contained in the assigned BPMN model share the Participants of the Sub-Conversation.

ARIS Method provides a corresponding **Sub conversation** symbol which is also shown in the ARIS **Symbols** bar.

A **Call Conversation** identifies a place in a Conversation (Collaboration) where either a **global Conversation** element is used or a **GlobalConversation** (restricted Collaboration type in the form of an empty Collaboration). A GlobalConversation is a reusable definition of an atomic Communication that can be called from within any Conversation by a Call Conversation.

The concepts for the elements CallConversation and GlobalCommunication are very vague. Thus, ARIS Method does not provide any specific symbols. However, the Boolean ARIS attribute type **Call conversation** exists and allows the modeler to flag Call Conversations. If the value is set to **true**, the **Call conversation** symbol is automatically rendered by the software.

10.6.3. Participants

Participants are represented by the ARIS object type **Participant**. The **Pool** symbol is available in the ARIS **Symbols** bar. If the ARIS attribute type **Multi-instance participant** is set to **true**, the program renders the symbol with three vertical lines displayed at the bottom of the Pool symbol.

Participants/Pools are described in detail in the chapter Participants [165].

10.6.4. Artifacts

According to the metamodel, the element Artifact is allowed in a BPMN **Conversation diagram**. However, the relevance of groupings in a Conversation diagram is not evident. For that reason, only Text Annotation elements are implemented in the current version of the ARIS model type **BPMN conversation diagram (BPMN 2.0)**.

In ARIS, the symbol **Text annotation** is available in the **Symbols** bar.

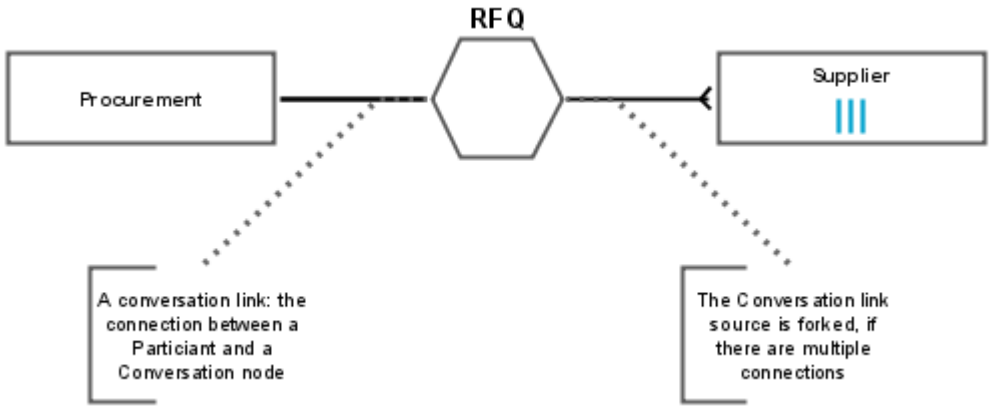
Artifacts and their usage are described in detail in the chapter Artifacts [157].

10.6.5. Conversation Links

A Conversation Link is used to link Participants with Conversation Node elements. A Conversation Node involves at least two Participants.

N-ary (n > 2) Conversation elements are allowed.

In ARIS, the connection type **participates in** (Participant **participates in** Conversation) is available. The passive name of the connection type is **has conversation link to** (Conversation **has conversation link to** Participant). Specific attributes are not required.

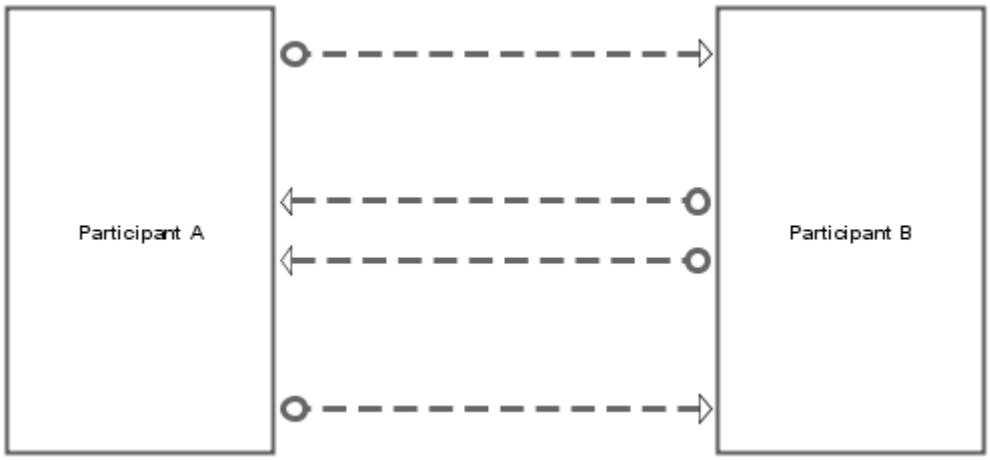


Conversation link with Participant multiplicity

The fork shown at the source of a Conversation Link must be manually set by the modeler using the properties dialog of the relevant connection type.

10.6.6. Message Flows in a Conversation

According to the specification, it is allowed in the BPMN **Conversation diagram** to model message exchanges between Participants by using the element Message Flow. (See: Business Process Model and Notation (BPMN), Version 2.0.)



Message flow between Participants in a BPMN conversation diagram (BPMN 2.0)

Thus, the ARIS connection type **message flow** (Participant **message flow** Participant) is available in the ARIS model type **BPMN conversation diagram (BPMN 2.0)**.

10.6.7. Model assignments

In ARIS, the following model types can be assigned to the object type **Conversation**:

- BPMN conversation diagram (BPMN 2.0)
- BPMN collaboration diagram (BPMN 2.0)

Only one model of each type can be assigned to an object of type **Conversation**.

10.7. Enterprise BPMN collaboration diagram

In ARIS, the model type **Enterprise BPMN collaboration diagram** is based on the model type **BPMN collaboration diagram (BPMN 2.0)** and extends this type of BPMN diagram by ARIS constructs that are already available in the EPC but out of scope in the BPMN specification. Thus, for example, the following object types can be (re-)used as lanes:

- Application system type
- Role
- Organizational unit
- Position
- Group

The **supports** connection is used to nest tasks in a lane object of the **Application system type** object type. The **carries out** connection is used to nest tasks in the lane objects of object type **Organizational unit**.

For all other nestings known from the BPMN specification, the **belongs to** connection is used.

Similar to the EPC model type, the connection type **is process-oriented superior** is used to represent an assignment relationship between a function and the tasks within the Enterprise BPMN collaboration diagram assigned to the given function.

11. Customer Experience Management (CXM)

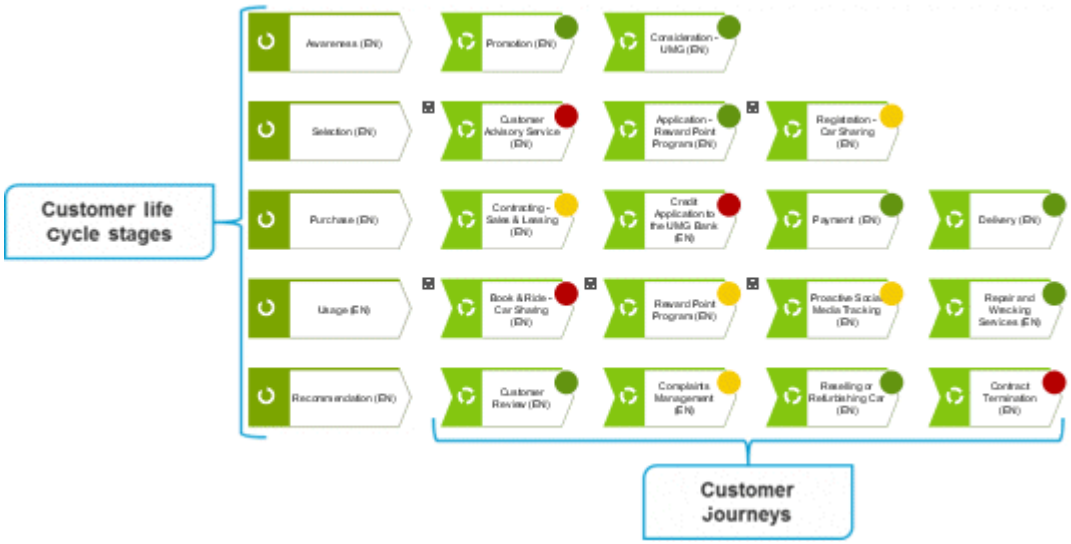
As a result of digitization, consumers are spending ever-increasing amounts of time online, and customer processes are also increasingly incorporating digital interactions. Due to these developments, Customer Experience Management (CXM) is developing into one of the most important drivers of innovation and customer loyalty.

CXM has the objectives of positively influencing the consumer behavior of a company's customers and of providing all customers with the information they need at any time using the right channel.

ARIS enables the implementation of CXM projects using two different approaches. Firstly, a CXM project can be implemented using the top-down approach, whereby the CXM project starts with the definition of the customer journey landscape and the subsequent creation of the customer journey maps. Secondly, internal processes can be enhanced with customer touchpoints and these can then be specified in more detail in customer touchpoint allocation diagrams in an alternative, bottom-up approach to CXM project implementation.

11.1. Customer journey landscape

The **Customer journey landscape** model enables the description of the customer lifecycle including the individual customer lifecycle stages and customer journeys. The model can be used either to model the customer lifecycle stages only and assign customer journeys to these, or to depict both the customer lifecycle stages and the corresponding customer journeys in one model.



Customer journey landscape

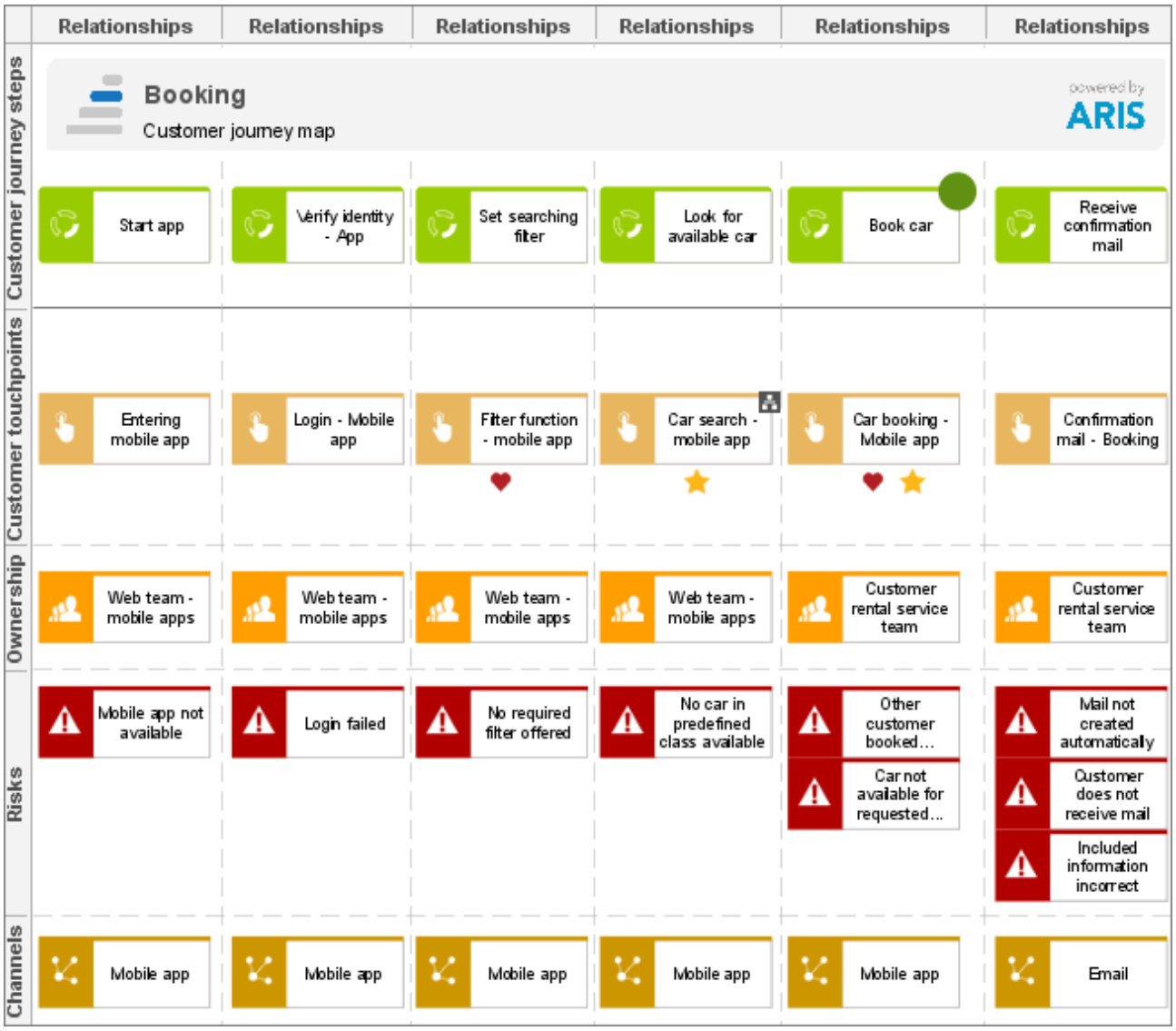
If using the top-down approach, you would start with the customer journey landscape, which provides a clear overview of all customer lifecycle stages and their associated customer journeys.

Furthermore, the **Customer journey** object type enables the customer journey owner, the business driver and business driver impact on transformation, as well as the overall customer experience to be represented using attributes.

If the corresponding CXM templates are activated, a traffic-light display can be used to visualize the relevant initiatives depending on the attribute values defined, so that the person in charge can immediately see which initiatives are required for which customer journeys.

11.2. Customer journey map

The **Customer journey map** model is a column diagram, which symbolizes one of the customer journeys from the **Customer journey landscape** model. The customer journey map provides the **Customer journey step** and **Customer touchpoint** objects to enable the illustration of the "journey" that the customer takes with the organization and that characterizes the customer's interactions with the company. The model can thus incorporate a detailed description of customer touchpoints, including the relevant KPIs, organizational responsibilities, initiatives, and risks.



Customer journey map

However, the main object in this model type is the **Customer touchpoint**, which can be described using many different attributes, including:

- customer objectives
- customer expectations and
- customer feeling

It is also possible to specify whether the customer touchpoint is a moment of truth, a pain point, or a best practice. Given that this information is very important, it is displayed in the model using special symbols.

CXM symbols

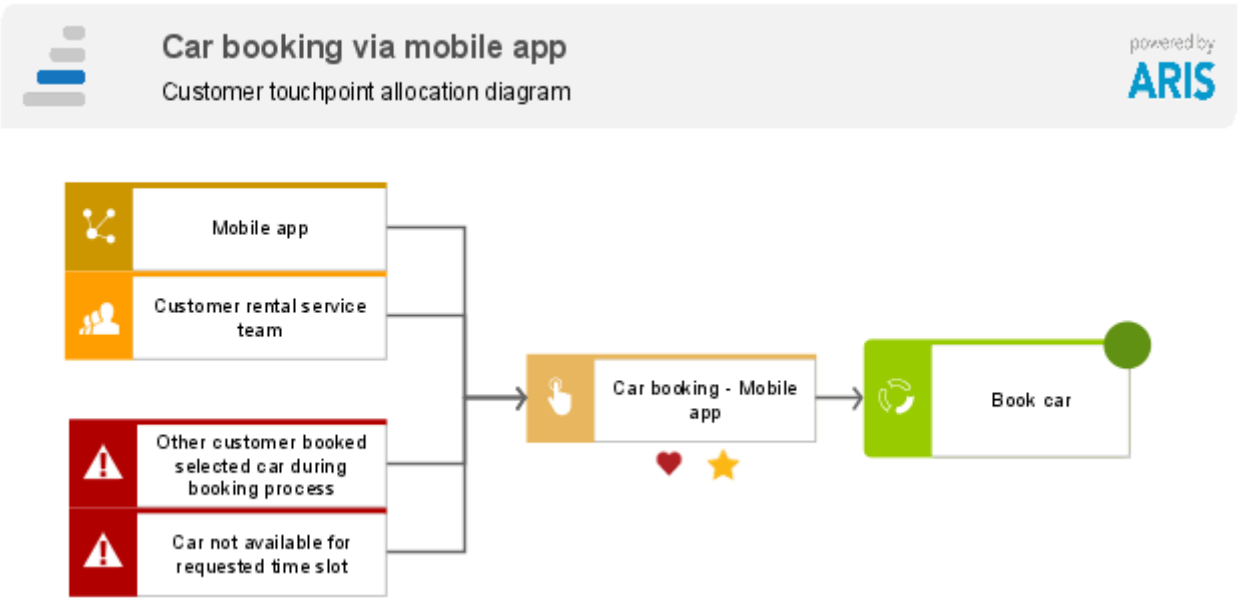
Moment of truth (**MoT**) indicates that the touchpoint is a crucial and decisive touchpoint for the company and the process. An MoT can dictate whether the relationship between the customer and the company is either reinforced or broken. Identifying moments of truth should therefore be assigned a high priority because such touchpoints have a direct effect on business.

To make the modeling of the customer journey map as simple as possible, ARIS enables the placing of objects using drag and drop. In addition, it is not necessary to draw connections between objects because this model type automatically creates the required object relationships, with the exception of objects of the **Customer journey step** type. All objects in a column belong to the **Customer touchpoint** object, which, in turn, is related to the **Customer journey step** object.

If a customer journey step has more than one customer touchpoint because there are multiple channels, the individual touchpoints can be described in more detail by assigning a customer touchpoint allocation diagram. If you do not assign an allocation diagram to a given touchpoint, all objects below that customer touchpoint will be used for all touchpoints in the relevant step. In this way, it is only a general touchpoint specification.

11.3. Customer touchpoint allocation diagram

The **Customer touchpoint allocation diagram** model type is used to describe a customer touchpoint in more detail including the relevant KPIs, organizational units, initiatives, risks, etc.



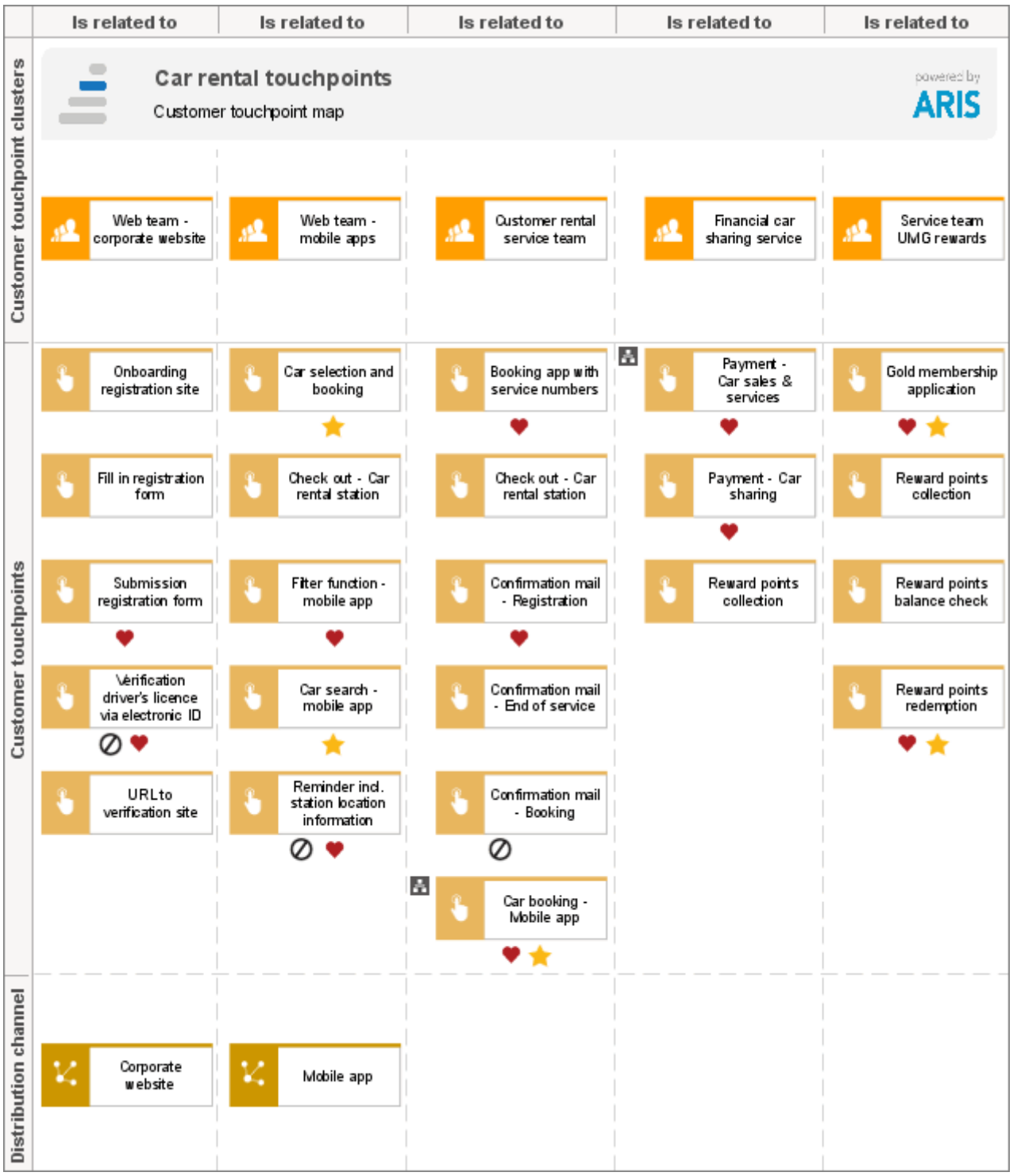
Customer touchpoint allocation diagram

Besides the option of assigning it to a customer touchpoint object in a customer journey map, this model type is particularly useful if companies already use ARIS and want to map customer touchpoints in their

internal processes without mapping them additionally in customer journey map models. This model type offers the same object types as the **Customer journey map** model type for describing the customer touchpoint in more detail.

11.4. Customer touchpoint map

The **Customer touchpoint map** model lists all customer touchpoints and can be used, for example, as the starting point in a customer experience project, in order to identify the interaction points with the customer. The customer touchpoints are grouped on the basis of a certain criterion that may be important for the analysis, for example, by organizational unit, channel, or risk.



CXM symbols

The customer touchpoint map can also be used to gain an overview of all touchpoints in an existing customer experience project and to manage these. Thus, it is possible to identify, at a glance, which customer touchpoint is a moment of truth, a pain point, and/or a best practice.

11.5. Linking CXM and BPM

ARIS enables the mapping of both the internal and external processes and perspectives and the creation of links between these using the **Customer touchpoint** object. If, for example, a company has defined

a customer touchpoint in a customer journey map, the company can then create an occurrence of the touchpoint in an internal process and reuse it, for example, in an EPC that is important for the customer journey. With this option, the two models are linked with each other and you can, for example, run an analysis to very quickly obtain an overview of which internal processes are influenced by changes to the touchpoint.

Linking CXM and BPM

11.5.1. Analysis capabilities

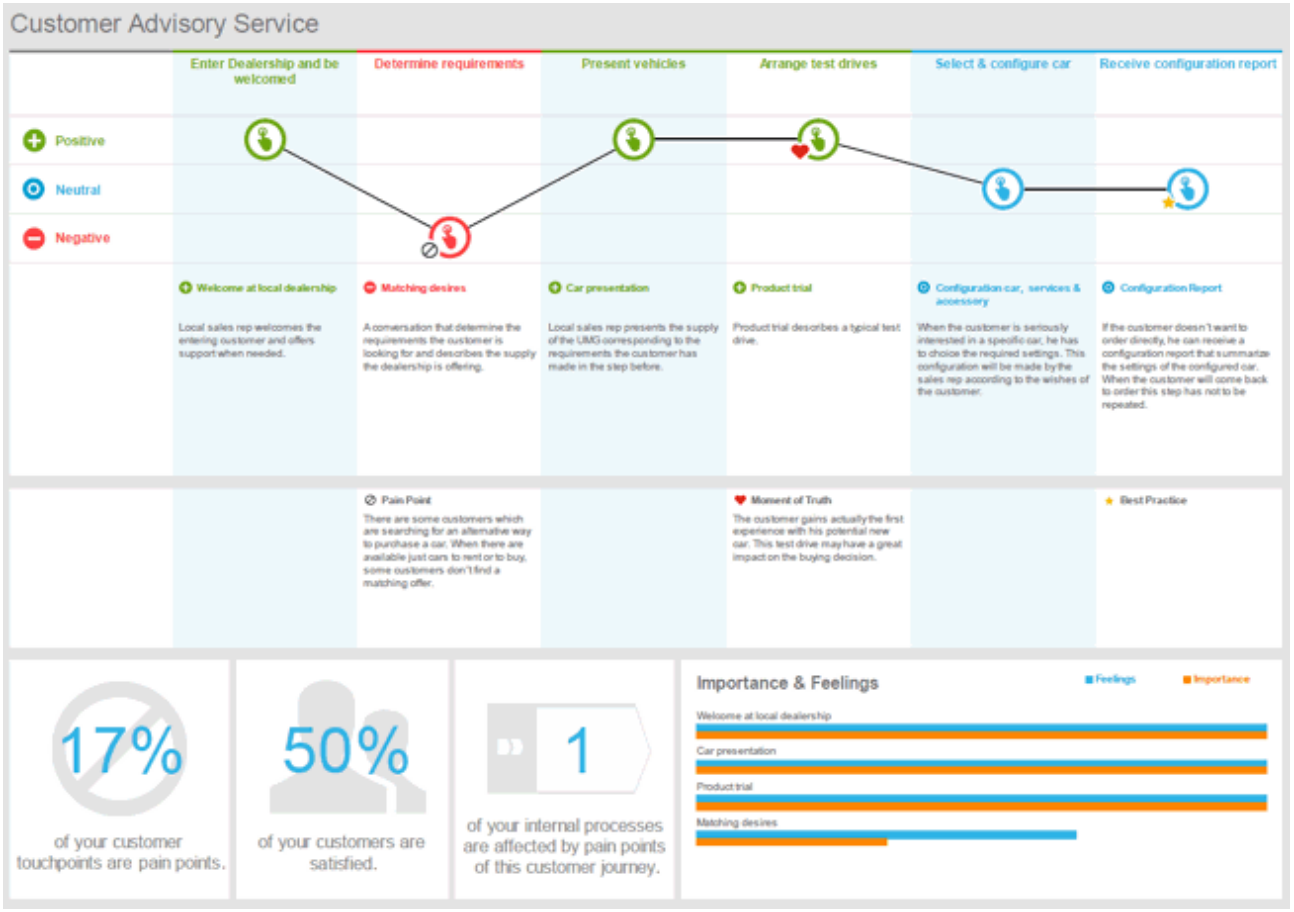
Besides the option of visualizing CXM data in the corresponding model types, it is possible to analyze these data in ARIS using reports or queries.

11.5.1.1. Report

The **Analyze customer experience** report uses an infographic to visualize how customers experience their interaction with the company during a customer journey and is intended to help identify customer satisfaction as well as customer issues.

The following information is evaluated and displayed:

- **Customer journey steps** with **customer touchpoints**
- **Moment of truth** and **pain points** with description
- **Best practice**
- **Importance to customer** & **customer feeling** (satisfaction)
- Percentage **proportion of pain points** in the **customer journey map**
- Number of **internal processes** impacted
- Percentage **proportion of satisfied customers**



CXM infographic

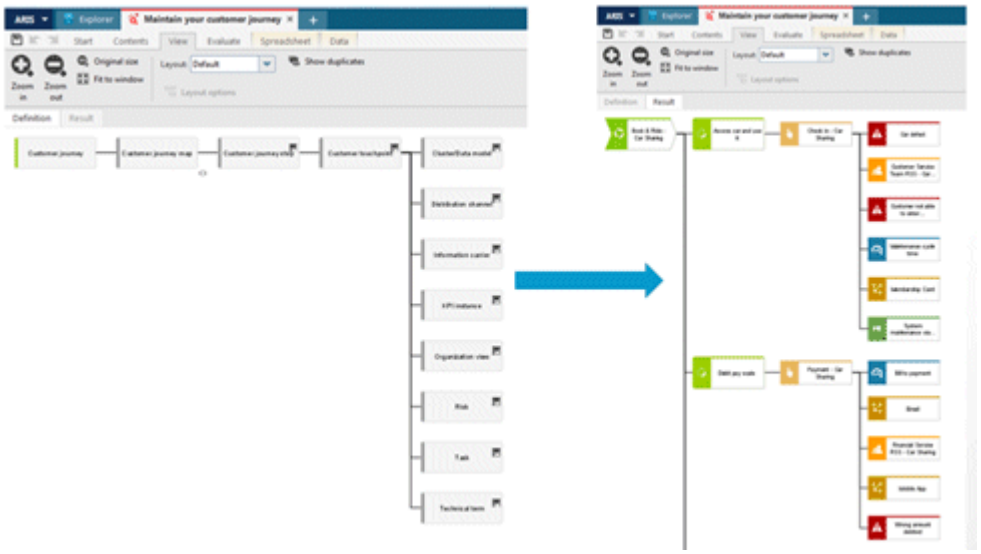
11.5.1.2. Queries

Queries enable you to visualize, with just a few clicks of your mouse, the complex interrelationships within an ARIS database. The data and interrelationships are visualized graphically, but a table format can be provided also.

11.5.1.3. Get full customer journey overview

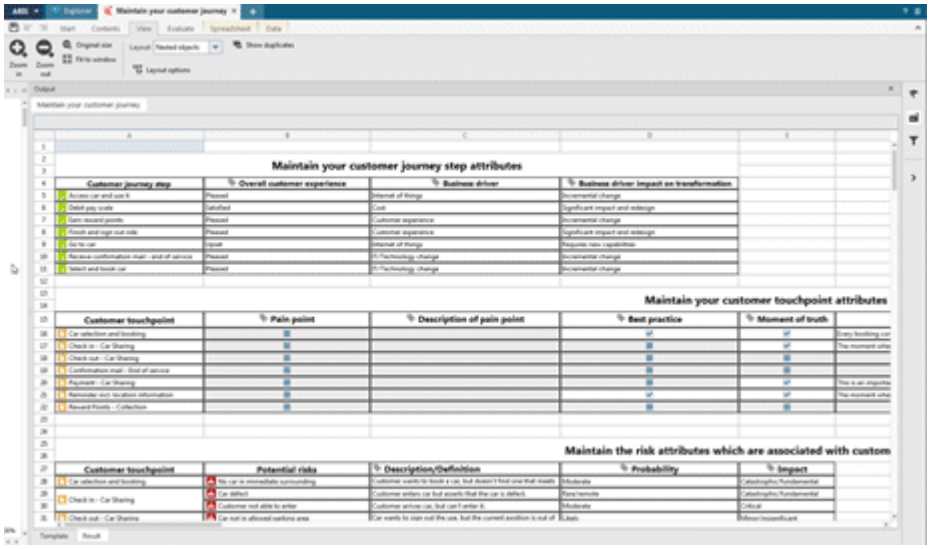
The **Get full customer journey overview** query is started for objects of the **Customer journey** object type.

The query collects all customer journey maps assigned to the selected customer journey objects and returns all information available about the customer journey steps and the related customer touchpoints. The information belonging to a customer touchpoint describes the customer touchpoint in detail, for example, the risks associated with it.



Customer journey overview

In addition to the graphical display, this query also provides a table showing a complete overview of the customer journey, which can be used, for example, for data management.



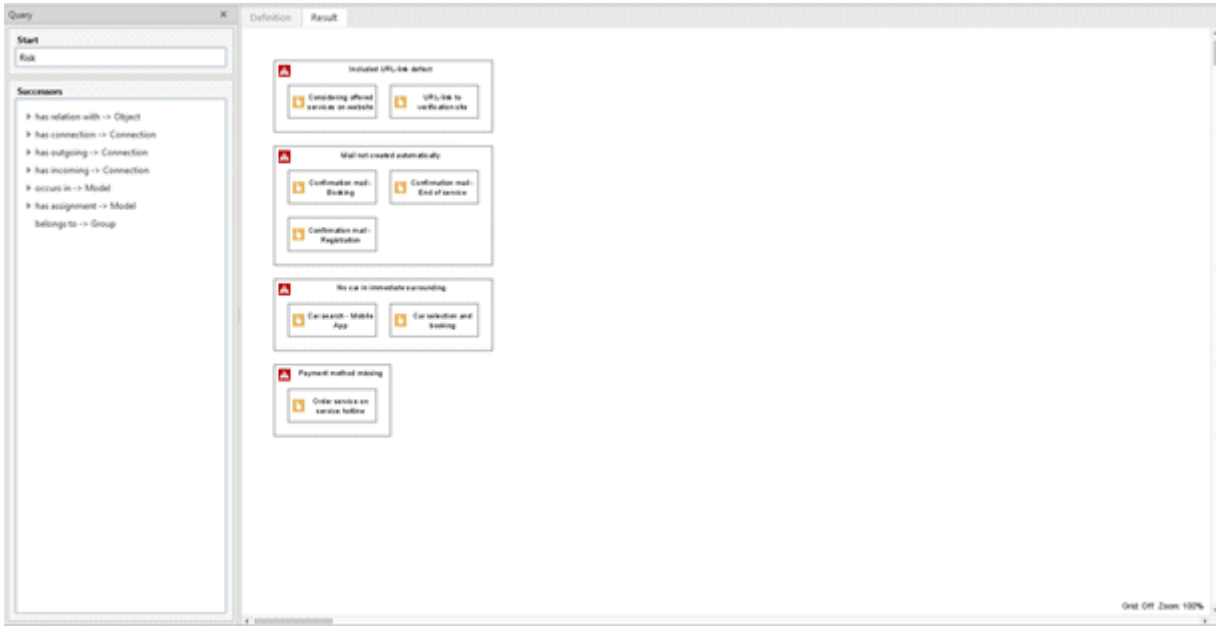
Customer journey table

The table contains information about the individual customer journey steps, for example, the overall customer experience, as well as all important details about the customer touchpoints of the selected customer journey, for example, which touchpoints represent a pain point for the customer and which do not.

In addition to providing an overview, this table also enables you to specify attributes.

11.5.1.4. Find customer touchpoints clustered by associated risk

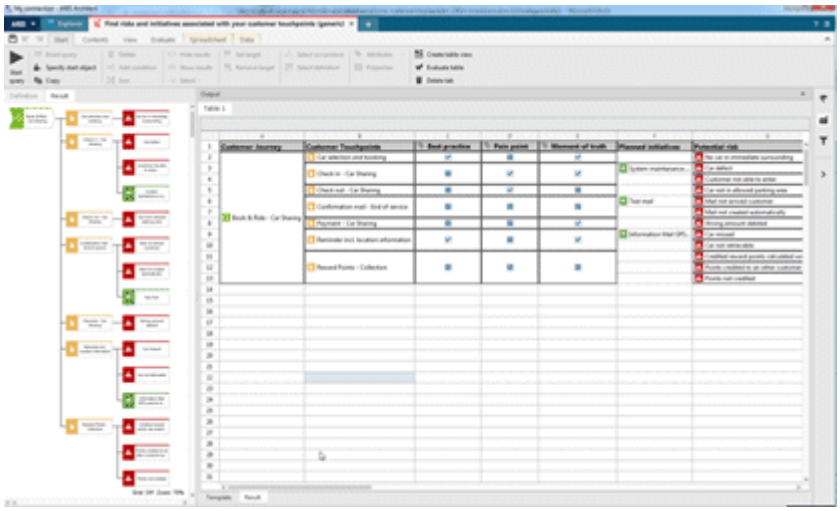
This query is started for objects of the **Risk** object type. It shows all customer touchpoints belonging to a risk to provide you with a quick overview.



Customer touchpoints by risk

11.5.1.5. Find customer touchpoints clustered by associated ownership

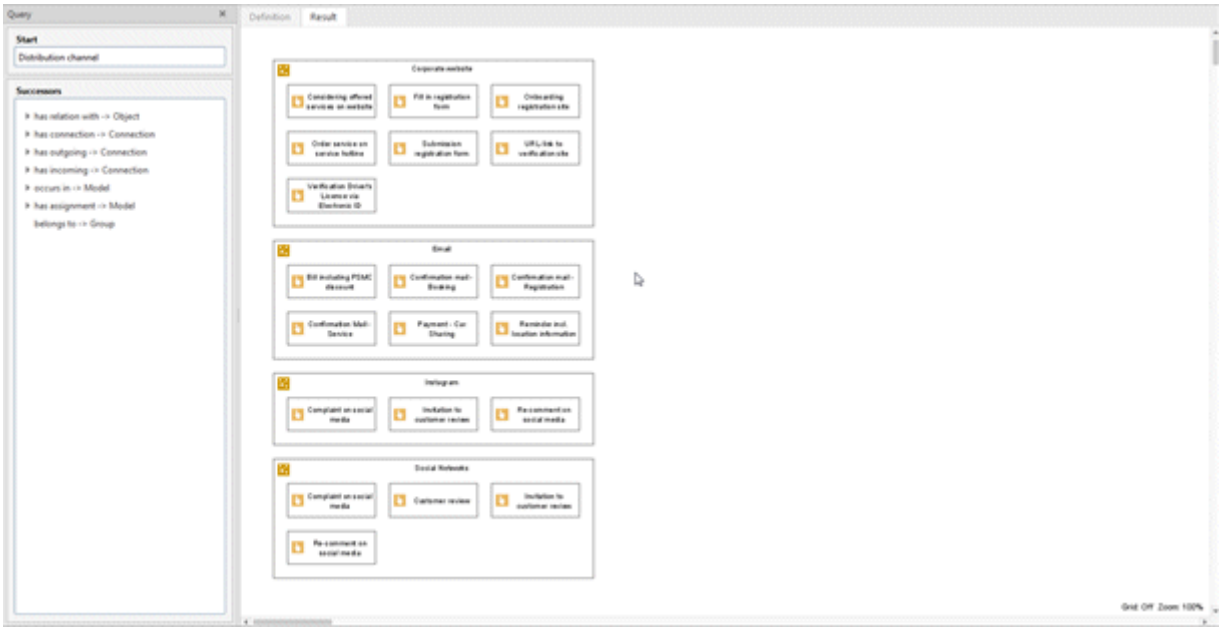
This query is started for objects of the **Organizational unit** object type and shows all customer touchpoints belonging to an area of responsibility in order to provide you with a quick overview.



Risks and initiatives for all customer touchpoints

11.5.1.6. Find customer touchpoints clustered by associated channel

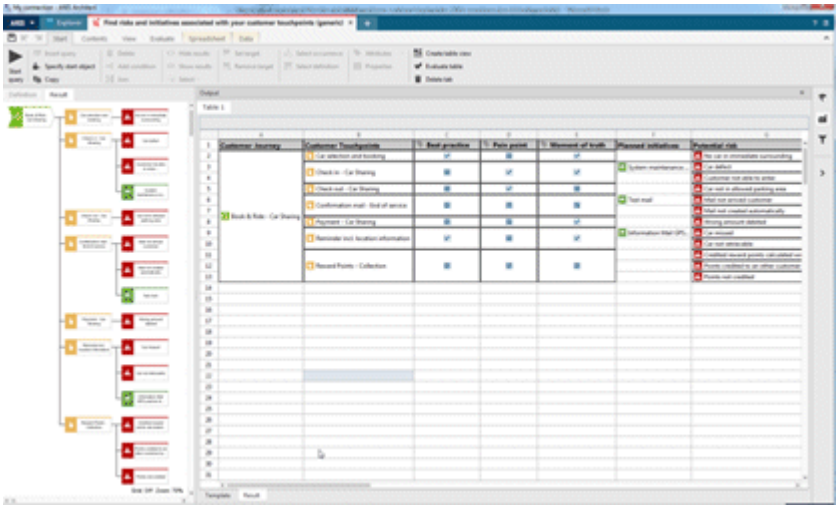
This query is started for objects of the **Distribution channel** object type and shows all customer touchpoints belonging to a distribution channel in order to provide you with a quick overview.



Find customer touchpoints clustered by associated channel

11.5.1.7. Find risks and initiatives for all customer touchpoints

The **Find risks and initiatives for all customer touchpoints** query is started for customer journey objects. It collects all customer journey maps assigned to the selected customer journey objects and retrieves the corresponding customer touchpoints, as well as the associated risks and planned initiatives for these touchpoints.

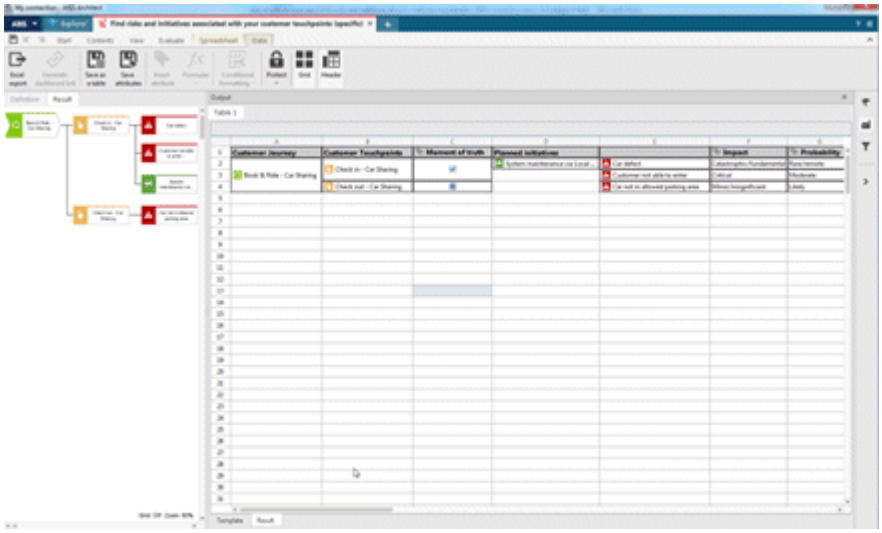


Risks and initiatives for all customer touchpoints

The information about the customer journey objects and the corresponding customer touchpoints is shown in a table returned by this query. The values for the Pain point, Moment of truth, and Best practice attributes are included as additional details on the touchpoints. Furthermore, the table contains all risks associated with the individual touchpoints, as well as the planned initiatives.

11.5.1.8. Find risks and initiatives for bad customer touchpoints only

The **Find risks and initiatives for bad customer touchpoints only** query is started for customer journey objects. It contains information similar to the information provided by the **Find risks and initiatives for all customer touchpoints** query, except that not all customer touchpoints are evaluated. Instead, it evaluates only those for which the **Pain point** attribute is specified as applicable, making this touchpoint a negative one for the customer. This query also focuses on the risks associated with the individual touchpoints, and therefore provides information about these, for example, the probability of their occurrence.



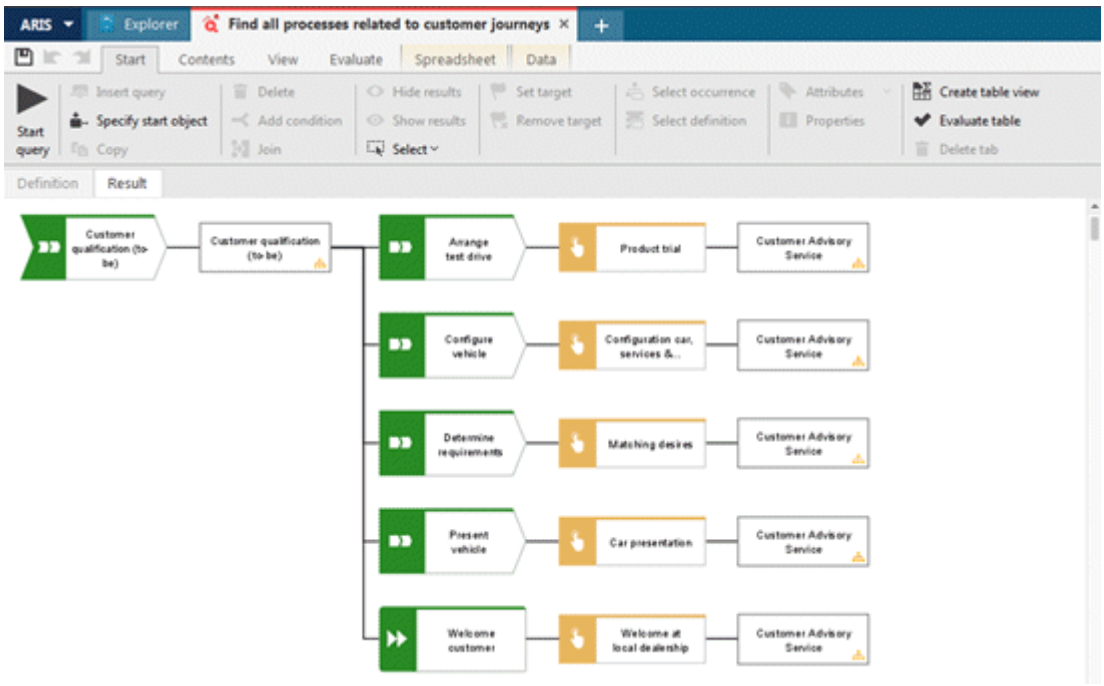
Risks and initiatives for bad customer touchpoints

11.5.1.9. Find all processes related to customer journeys

This query is started for functions of a value-added chain diagram (VACD).

You can use it to detect customer touchpoints in internal processes and, in this way, determine which customer journey map is connected with which internal process.

The query firstly shows all EPCs that are assigned to the selected functions. Then it shows all functions of the EPC that are related to customer touchpoints.



Find all processes related to customer journeys

Besides the graphical display of interrelationships, this query provides a table with the following information:

- The relevant EPC
- The process that is related to a customer touchpoint
- The customer touchpoint
- The specification whether the customer touchpoint is a pain point or a moment of truth
- The name of the customer journey in which the customer touchpoint occurs

12. Use cases

The purpose of this chapter is to assist you in finding the right ARIS support for specific business management problems. Therefore, the chapter is divided into use case scenarios (subchapters).

For each use case scenario the meaning of each scenario and the activities that are normally performed in the corresponding scenario are briefly described. Subsequently, typical tasks occurring in the scenario are described. For each task it is shown how ARIS can be used to perform the task.

The following table gives an overview of the use cases described along with the model types used:

Scenario	Scenario tasks	Model types
General company documentation [218]	Documentation of business objectives Documentation of company value-added Documentation of organizational structures Documentation of company functions Process documentation Process warehousing	Objective diagram Value-added chain diagram Organizational chart Function tree EPC
Database management/Data warehousing [219]	Data structuring/Database design Database management/Access management	ERM IEF data model Table diagram Class diagram
PC hardware and network management [220]	Identification of IT infrastructure requirements Documentation of IT infrastructure Access privileges	Network topology Network diagram
Process cost management [221]	Description of process and organizational structures Cost center analysis Process calculation	EPC Organizational chart
Quality management [221]	Creation of QM documentation Certification procedures Certification documents	Product tree Product selection matrix EPC Value-added chain diagram Structuring model Organizational chart
Reorganization measures [222]	Project documentation Performing a reorganization	Value-added chain diagram EPC Organizational chart Product model Product/Service model Objective diagram

Scenario	Scenario tasks	Model types
SAP R/3 implementation [223]	Analysis phase, specification phase (project preparation) Design phase (Business Blueprint); 3 use cases available	EPC Organizational chart
Software development and implementation [224]	Project documentation Specification of application systems and modules Description of IT processes System interface development	Value-added chain diagram Organizational chart EPC Use case diagram Application system type diagram Program flow chart Screen diagram
Knowledge management [225]	Knowledge map or yellow page Categorization of knowledge Knowledge processing in business processes	Knowledge map Knowledge structure diagram EPC Function allocation diagram
Workflow management [226]	Process customizing of workflow management systems	EPC Function allocation diagram Application system diagram Application system type diagram

12.1. General company documentation

Company characteristics, such as processes, structures, and data can be documented in suitable form for training, presentation, and evaluation purposes of any kind. The most important tasks of company documentation are briefly described below.

Task: Documentation of business objectives

ARIS support: Objective diagrams can be used for hierarchical alignment of business objectives and the corresponding success factors.

Task: Documentation of the company value added

Identification of the functions involved in value-adding activities of a company is the basis for many corporate decisions.

ARIS support: The company functions involved in the value added can be displayed using the value-added chain diagram. This model illustrates both the sequence of consecutive functions, and superior and subordinate functions.

Task: Documentation of the organizational structure

ARIS support: The structure of a company can be documented in organizational charts illustrating the hierarchy and relationships of organizational units.

Task: Documentation of company functions

ARIS support: A function tree can display an overview of a company's individual functions. The functions are divided into object-oriented, process-oriented, or execution-oriented functions.

Task: Process documentation

ARIS support: If utilization of process models by SAP® applications, simulations, workflows, etc. is required, you are recommended to use EPCs for modeling purposes.

Task: Process warehousing

Process warehousing is the systematic recording, storage, and maintenance of business process knowledge in a repository.

ARIS support: EPCs are useful for modeling process knowledge, supported by documents, images, and videos, so that the models can be used for demanding evaluations, such as simulation or process cost management.

12.2. Database management/Data warehousing

By storing company data in databases, redundant data storage is reduced and program-independent access to data used across the company is enabled. Data warehousing ensures quality, integrity, and consistency of the underlying data. The term 'Data Warehouse' generally designates a database that is isolated from operational IT systems and serves as a company-wide data basis for all forms of management support systems. It is characterized by strict separation from operational and decision-supporting data and systems. The focus of the Data Warehouse concept is on efficient provision and processing of large amounts of data to perform evaluations and analyses in decision-relevant processes.

Task: Data structuring/Database design

The structure of databases is determined by the underlying data models.

ARIS support: The most widely used method of data modeling is the entity relationship model (ERM), which serves as the basis for the implementation of a relational database.

The tables and fields of a database system are described by the table diagram.

Object-oriented database systems can be designed using the Unified Modeling Language (UML). In UML, the class diagram can be used to show the static data relationships.

Task: Database management/Access management

Assignment of users and system administrators to database systems.

ARIS support: The access diagram can be used in conjunction with relations and system components to determine the access privileges that organizational units, positions, and persons have for the database system.

12.3. PC hardware and network management

Network management is the control, monitoring, and coordination of all (distributed) resources (data networks, processors, data, and applications) that enable communication in a computer network.

Task: Identification of IT infrastructure requirements

Based on an existing organizational structure, communication and information system infrastructures suitable for supporting it efficiently are to be derived.

ARIS support: The requirements of the information systems' structure can be depicted using the network topology model type. The representation of application systems, network types, and hardware components does not show individual, identifiable specimens (for example, PC with inventory number 3423), but typifications based on the same technology.

Task: Documentation of IT infrastructure

The task is to depict an existing or planned installation of an IT infrastructure with specific hardware components, networks, and application systems.

ARIS support: An IT infrastructure can be illustrated using a network diagram as a concrete implementation of a network topology.

Task: Access privileges

The task is to demonstrate which applications and users have access to which data and in which way.

ARIS support: The access diagram can be used to describe which applications or application modules are allowed what kind of access (write/read/change) to databases and information carriers, and whether the data acts as input or output. Furthermore, it can depict which user privileges and views specific users or user groups have for the applications or application modules.

12.4. Process cost management

By recording and allocating the costs arising from the commercial provision of products and services, cost accounting provides a scheduling basis and a control instrument. Due to changes in the cost structures, in particular the increase in overhead costs, traditional cost accounting methods are replaced by process cost management. Process cost management determines the costs of processes across cost centers. Budgeting, cost transparency in the indirect performance areas, pricing, and support in make-or-buy decisions are the main advantages of process cost management.

Task: Description of process and organizational structures

The task is to determine the processes to which process cost management applies and to describe cost centers.

ARIS support: Processes are illustrated using standard model types (for example, EPC). Specifying time attributes and assigning organizational units are important steps in process cost management.

The company organization is described in an organizational chart, in which the organizational units correspond to cost centers (with the **Cost rate** and **Products/Services** attributes).

Task: Process calculation

ARIS support: As a prerequisite, a complete cost center analysis must have been performed, including the determination of process cost rates. No additional models are required to perform process calculation. The results are shown in a calculation table.

12.5. Quality management

The term 'Quality Management' (QM) applies to all activities that define a company's quality policy, objectives, and responsibilities. The means for implementing these activities include quality planning, quality control (process management), quality assurance, and quality improvement (quality promotion).

Task: Creation of QM documentation

To ensure the quality of products and processes within a company, adequate documentation has to be produced that enables the company to evaluate, compare, and improve products and processes.

ARIS support: Product trees can be used for product documentation as they allow for efficient classification of products. This type of representation is increasingly used in the service industry, particularly in public administration. Furthermore, the product selection matrix enables a company to illustrate which of its functions are required for the creation of which products, and which organizational units are responsible for production.

Another main objective of QM documentation is to document processes that can be recorded by means of EPCs, be evaluated by reports, or refer to documents and applications within the company.

Task: Certification procedures

Use of procedure models to support project management in the certification process according to nationally and internationally recognized standards, such as ISO or VDA.

ARIS support: A procedure model describing certification (for example, the ARIS procedure model) can be represented using a value-added chain diagram. Individual steps can be described in more detail by assigning additional process models.

Task: Certification documents

Creation of quality documents required for certification.

ARIS support: The structuring model subdivides individual certification standards into their components. Individual items of a structuring model can be assigned company models for quality control. For example, these models can be process models in the form of EPCs, organizational charts, or value-added chain diagrams.

Using ARIS Architect, reports can be generated that are approved as a QM manual for certification purposes.

12.6. Reorganization measures

Reorganization measures aimed at reducing costs or time or at improving the quality of results or work involve modification of business processes (process redesign) or their complete redevelopment (process reengineering).

Task: Project documentation

Documentation of reorganization measure planning, implementation, and results.

ARIS support: The main project phases of the reorganization process can be described as a procedure model by a value-added chain diagram.

Individual project activities of the reorganization project and their operational sequence can be documented by means of EPCs.

The organizational assignment of people and units involved in the project can be represented in organizational charts.

Task: Performing the reorganization

A reorganization project involves project preparation and strategic planning, followed by an analysis of the actual situation, development of the target concept, and finally implementation of the solutions.

ARIS support: Product/Service models as well as objective diagrams serve to document general strategic conditions, so that the company's main business segments can be recorded along with their products,

services, and customer groups, and the critical success factors and the objective hierarchy of the company can be depicted.

During the analysis of the actual situation, a framework containing the main business processes is developed using value-added chain diagrams. Based on employee interviews, these business processes are recorded in detail in the form of an EPC.

Following a weak point analysis that takes into account throughput times, process costs, organizational breaks, system and media breaks, data redundancies, etc., alternative target processes are defined. As with actual data, these processes are modeled using EPCs.

Once the target concept is complete, the system, organizational, and data components are described in more detail, which facilitates implementation. For example, the application system construct **Word processing** can now be specified as Microsoft® Word.

Note: The weak point analysis phase can be supported by evaluations using Simulation.

12.7. SAP R/3 implementation

The support capabilities of ARIS in the implementation of the R/3 standard software by SAP AG focus on the lifecycle of the ASAP implementation approach. However, in addition to ASAP, other approaches geared more towards business process optimization (in the broadest sense) are supported, as well. Parts of the ARIS support described in the following can be provided with Extension pack SAP® only.

Task: Analysis phase, Specification (project preparation)

The task is identifying the degree of coverage of the company-specific processes via the SAP® system, as well as timely identification of possible weak points.

ARIS support: If the analysis is not performed directly by the SAP R/3 reference model, the 'optimum business processes' to be supported can be modeled in ARIS (see general company documentation). By mapping the company model with the SAP R/3 reference model, an initial estimate of the degree of coverage can be obtained through reports.

Task: Design phase (Business Blueprint)

Process and/or function deficits were identified in the SAP R/3 reference model.

ARIS support: Based on existing R/3 reference model components, new process and scenario variants can be developed in ARIS. Furthermore, new functions, events, and rules can be added to process and scenario components (that is, through the development of new SAP® ABAP functions, if required).

Task: Design phase (Business Blueprint)

Design of interfaces between SAP® R/3 and non-SAP® applications.

ARIS support: The documentation of the attributes to be exchanged can be described in detail in process models in the ARIS Repository by assigning data or objects to functions and data models or object models.

This information can also be output in the form of reports and establishes the basis for the development of interfaces.

Task: Design phase (Business Blueprint)

Development of the organizational design of business processes and the SAP® system.

ARIS support: The company's social structure and the SAP® organizational structure can be described and juxtaposed using organizational charts.

12.8. Software development and implementation

Task: Project documentation

Documentation of the planning, process steps, and results of software development and implementation.

ARIS support: The main project phases can be described as a procedure model by a value-added chain diagram.

Individual project activities during development and implementation and their operational sequence can be documented by means of EPCs.

The organizational assignment of people and units involved in the project can be represented in organizational charts.

Task: Specification of application systems and modules

The task is to show the structure of an information system based on system requirements.

ARIS support: The use cases of the software system to be developed can be identified by means of the use case diagram. Furthermore, the system users can be defined and then assigned to individual use cases. Often, the use case diagram is the starting point for detailed process modeling. Process models can be assigned to individual use cases.

The application system type diagram may serve to describe the hierarchical structure of application systems at type level using module types and IT function types.

Concrete occurrences describing specific types in more detail can be represented in the application system diagram.

Task: Description of IT processes

The task is to describe the chronological-logical operational sequence of processes within and across modules.

ARIS support: IT processes can be modeled in program flow charts.

Task: Development of the system interface

The task is to develop and document a user interface.

ARIS support: The structural and functional organization of a screen (window) can be described with a screen diagram. The screen diagram is the basis for deriving the program code.

12.9. Knowledge management

The starting point for designing comprehensive knowledge management is the assumption that knowledge has become or is becoming the dominant production factor in companies. This results in the need to understand knowledge as a controllable element, just like the classic operational production factors.

Therefore, knowledge management focuses on acquisition, representation, and distribution of knowledge. Knowledge management is the sum of all methods, measures, and systems used by an organization to develop knowledge, render it transparent, and provide it regardless of time, people, and location. The aim of knowledge management is to increase knowledge and to apply existing knowledge in the company in an optimal way.

Task: Knowledge map or yellow page

The objective is to show what knowledge is available in the company and where.

ARIS support: The **Knowledge map** model type can be used to display the organizational distribution of different knowledge categories. It shows which organizational unit, position, or employee has expertise in certain knowledge categories, and at what level of competence.

Task: Categorization of knowledge

The task is to analytically classify the intellectual capital of an organization, that is, to describe the various types and groups of knowledge in order to design a knowledge storage structure, for example.

ARIS support: The knowledge structure diagram can be used to show how the knowledge base of an organization is divided into different knowledge categories and how these are further broken down into knowledge categories and documented knowledge. For documented knowledge, it is also possible to depict the information carriers storing the knowledge.

Task: Knowledge processing in business processes

The task is to show where knowledge is generated, modified, and required in business processes so that the most efficient use of the knowledge resource can be determined.

ARIS support: The **EPC**, **Process chain diagram**, and **Function allocation diagram** model types provide the **Knowledge category** and **Documented knowledge** objects. The knowledge structure and the organizational distribution of knowledge can be described separately by means of the knowledge structure diagram and the knowledge map.

12.10. Workflow management

In the broadest sense, a workflow can be interpreted as a business process. The term 'workflow' describes processes that are based on the division of labor and initiated to carry out business transactions. This can include both very simple business processes and complex, cross-organizational processes. The focus of the analysis is on the dynamic process flow from start to completion. Workflow management is the sum of methods, measures, and systems used in order to develop, control, and optimize workflows.

A workflow management system is actively operating, flexibly designable software that works according to an organizational framework of rules and controls a process spanning several workstations and integrating existing basic technical components. Process control systems can be used to support complex tasks involving a large number of employees and positions.

Task: Process customizing of workflow management systems

A special focus of ARIS is to support the transfer of general business process models into workflow models that can be used to configure various workflow management systems (semi-)automatically.

ARIS support: As with process modeling, the activity flow is depicted in an EPC. Modeling has to be done in strict adherence to the method! A function allocation diagram has to be created for each function, where a user as well as input and output data are allocated to the function - unless this has already been represented in the EPC.

In order for data-related applications to be launched automatically at runtime, files must be assigned to applications in an application system diagram or an application system type diagram.

13. Bibliography

13.1. General literature list

- Brombacher, R.; Bungert, W.: 'Praxis der Unternehmensmodellierung' [Enterprise modeling practise], 1992.

  Enterprise modeling practise, a seminar by IDS Prof. Scheer GmbH, Bad Soden/Taunus (Germany), November 12-13, 1992.
- Chen, P. P.: Entity-Relationship Model, 1976

  The entity-relationship model - toward a unified view of data, in: ACM Transactions on Database Systems, Volume 1 (1976), Issue 1, pages 9 - 36.
- Hoffmann, W.; Kirsch, J.; Scheer, A.-W.: 'Modellierung mit Ereignisgesteuerten Prozeßketten' [Modeling with event-driven process chains], 1993

  Modeling with event-driven process chains (Methods Manual, as of December 1992), in: Scheer, A.-W. (editor), Publication of the 'Institut für Wirtschaftsinformatik' [Institute for Information Systems], paper 101, Saarbrücken, January 1993.
- Keller, G.; Hechler, H.-J.: 'Informationsmodell' [Information Model], 1991

  'Konzeption eines integrierten Informationsmodells für die Kostenrechnung des SAP®-Systems' [Design of an integrated information model for cost accounting in the SAP® system], in: Scheer, A.-W. (editor): 'Rechnungswesen und EDV - 12. Saarbrücker Arbeitstagung 1991. Kritische Erfolgsfaktoren im Rechnungswesen und Controlling' [12th Saarbrücken conference for accounting and IT 1991. Critical success factors in accounting and controlling], Heidelberg 1991, pages 67 - 106.
- Scheer, A.-W.: Architecture of Integrated Information Systems, 1992

  Architecture of Integrated Information Systems - Foundations of Enterprise Modelling, 2nd edition, Berlin et al. 1992.
- Scheer, A.-W.: 'EDV-orientierte Betriebswirtschaftslehre' [EDP-oriented business management], 1990

  'EDV-orientierte Betriebswirtschaftslehre - Grundlagen für ein effizientes Informationsmanagement' [EDP-oriented business management - Foundations of efficient information management], 4th edition, Berlin et al. 1990.
- Scheer, A.-W.: Business Process Engineering, 1994

  Business Process Engineering - Reference Models for Industrial Enterprises, 5th edition, Berlin et al. 1994.
- Schlageter, G.; Stucky, W.: 'Datenbanksysteme' [Database systems], 1983

  'Datenbanksysteme - Konzepte und Modelle' [Database systems: Concepts and models], 2nd edition, Stuttgart 1983.
- Seubert, M.: 'SAP®-Datenmodell' [SAP® data model], 1991

  'Entwicklungsstand und Konzeption des SAP®-Datenmodells' [State of development and design of the SAP® data model], in: Scheer, A.-W. (editor): 'Datenbanken 1991 - Praxis relationaler Datenbanken: Vom Datenmodell zur Implementierung' [Databases 1991 - Relational database practise: From data model to implementation (symposium 06/04-06/05 1991 in Saarbrücken/Germany)], Saarbrücken 1991, pages 87 - 109.
- Sinz, E. J.: 'Entity-Relationship-Modell' [Entity Relationship Model], 1990

  'Das Entity-Relationship-Modell (ERM) und seine Erweiterungen' [The Entity Relationship Model (ERM) and its extensions], in: 'HMD Theorie und Praxis der Wirtschaftsinformatik (1990)' [HMD magazine on the theory and practice of business process engineering (1990)], paper 152, pages 17 - 29.
- Scheer, A.-W.: ARIS - Business Process Frameworks. 3. edition Berlin et al. 1998.
- Scheer, A.-W.: ARIS - Business Process Modeling. 3. edition Berlin et al. 1998.

- Scheer, A.-W., Jost, W.: 'ARIS in der Praxis' [ARIS in Practice], 2002

  'Gestaltung, Implementierung und Optimierung von Geschäftsprozessen' [Design, Implementation and Optimization of Business Processes], Berlin, Heidelberg, New York 2002.
- Scheer, A.-W., Abolhassan, F., Jost, W., Kirschmer, M.: Business Process Excellence, 2002

  ARIS in Practice , Berlin, Heidelberg, New York 2002.

13.2. Topic-related bibliography

13.2.1. Unified Modeling Language in ARIS

13.2.1.1. UML specification

UML specification: https://www.uml.org.

13.2.1.2. Using UML

- Burkhardt, R.: 'UML - Unified Modeling Language, Objektorientierte Modellierung für die Praxis' [Object-Oriented Modeling in Practice], Bonn 1997.
- Fowler, M.; Scott, K.: UML Distilled - Applying the Standard Object Modeling Language, Reading et al. 1997.
- Oesterreich, B.: Developing Software with UML: Object-oriented analysis and design in practice, 3rd edition, Munich/Vienna 1997.

13.2.1.3. UML and business process modeling

- Ambler, S. W.: What's Missing from the UML? Techniques that can help model effective business applications, Object Magazine 7(1997)8
- Loos, P.; Allweyer, Th.: Process Orientation and Object-Orientation - An Approach for Integrating UML and Event-Driven Process Chains (EPC), Publication of the 'Institut für Wirtschaftsinformatik' [Institute for Information Systems], Paper 144, Saarbrücken 1998.

13.2.2. Methods for knowledge management

13.2.2.1. General knowledge management

- Probst, G.; Raub, S.; Romhardt, K.: Managing Knowledge - Building Blocks for Success. Frankfurt/ Wiesbaden 1998.
- Bürgel, H. D. (editor): 'Wissensmanagement - Schritte zum intelligenten Unternehmen' [Knowledge management - Steps towards an intelligent company]. Berlin et al. 1998.

13.2.2.2. Using ARIS for knowledge management

- Allweyer, Th.: 'Modellbasiertes Wissensmanagement' [Model-based knowledge management]. In: IM Information Management & Consulting 13 (1998) 1, pp. 37-45.
- Allweyer, Th.: Using ARIS Models for Knowledge Management. In: Scheer, A.-W.: ARIS - Business Process Frameworks. 3. edition Berlin et al. 1998, p. 162-168.

13.2.3. Balanced Scorecard method

Kaplan, Robert/Norton, David: The Balanced Scorecard - Measures that Drive Performance, Harvard Business Review January/February 1992.

13.2.4. IT City Planning

- Schulman, Jeff: A New View of Architectures Needed for New Business Drivers, Gartner Briefing Presentations.
- Longépé, Christoph: Le projet d'urbanisation du système d'information, Dunod, Paris, 2001

13.2.5. Business process modeling

Business Process Modeling Notation, Working Draft (1.0); BPMI.org; August 25, 2003.

14. Support and legal information

This section provides you with some general information regarding product support and legal aspects.

14.1. Documentation scope

The information provided describes the settings and features as they were at the time of publishing. Since documentation and software are subject to different production cycles, the description of settings and features may differ from actual settings and features. Information about discrepancies is provided in the Release Notes that accompany the product. Please read the Release Notes and take the information into account when installing, setting up, and using the product.

If you want to install technical and/or business system functions without using the consulting services provided by SAG ARIS GmbH, you require extensive knowledge of the system to be installed, its intended purpose, the target systems, and their various dependencies. Due to the number of platforms and interdependent hardware and software configurations, we can describe only specific installations. It is not possible to document all settings and dependencies.

When you combine various technologies, please observe the manufacturers' instructions, particularly announcements concerning releases on their Internet pages. We cannot guarantee proper functioning and installation of approved third-party systems and do not support them. Always follow the instructions provided in the installation manuals of the relevant manufacturers. If you experience difficulties, please contact the relevant manufacturer.

If you need help installing third-party systems, contact your local ARIS sales organization. Please note that this type of manufacturer-specific or customer-specific customization is not covered by the standard ARIS software maintenance agreement and can be performed only on special request and agreement.

14.2. Support

Product support

We provide support for ARIS products to all customers with a valid support contract.

Information on product version availability and support, as well as sustained support dates, is available at ARIS Community.

Contact our Global Support services at ARIS Community to raise and update support incidents, or post ideas and feature requests.

Community

Register with ARIS Community to download products, updates and fixes, find expert information, and interact with other users.

Documentation

For information on how to use and activate products and for other technical support questions, visit the ARIS Documentation website.

If you want to give feedback on any documentation topic, write to documentation@aris.com.

14.3. Legal notices

This document applies to ARIS Version 10 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Copyright © 2010-2026 SAG Aris GmbH, Saarbrücken, Germany and/or its subsidiaries and/or its affiliates and/or their licensors.

The name ARIS and all SAG Aris GmbH product names are either trademarks or registered trademarks of SAG Aris GmbH and/or its subsidiaries and/or its affiliates and/or their licensors. Other company and product names mentioned herein may be trademarks of their respective owners.

The Software may contain or be distributed with certain commercial third party components (“Third Party Components”), which may be subject to special license terms and conditions (“Third Party Terms”) located at https://aris.com/terms.

Third Party Terms include important licensing and warranty information and disclaimers of third party licensors. For the avoidance of doubt, if Customer uses the Software in accordance with the Documentation, the Third Party Terms do not restrict the rights granted under this Agreement. In the event that a third party manufacturer no longer provides active support for any Third Party Component, Supplier will use commercially reasonable efforts to respond to any issues with the Software related to the Third Party Components.

Use, reproduction, transfer, publication or disclosure is prohibited except as specifically provided for in your License Agreement with SAG Aris GmbH and/or its subsidiaries and/or its affiliates and/or their licensors.