I n t e r n a t i o n a l   T e l e c o m m u n i c a t i o n   U n i o n

ITU-T Technical Report

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

(dd MMM YYYY)

MSTR-ODEF
Order Definition

Draft, neither submitted, approved or published

01.05.2017 Arve Meisingset

International
Telecommunication
Union

Summary

This technical report is a tutorial for how to define orders.

Keywords

[if any provided]

Change Log

This document contains Version 1 of the ITU-T Technical Report on "*Order Definition*" approved at the ITU-T Study Group 2 meeting held in Geneva, dd-dd MMMM YYYY.

Editor:	Arve Meisingset Movinpics AS Norway	Tel: +47 9139 2863 Fax: Email: arve.meisingset@movinpics.com

CONTENTS

Page

Contents

List of Tables

Page

List of Figures

Page

Technical Report ITU-T MSTP-ODEF

Technical Report ITU-T
Order Definition

Summary

This Technical Report is a tutorial for how to define orders.

1 Scope

This Technical Report explains how Orders can be created. We do not explain the tasks to create the Orders; we explain the data needed to create the Orders.

Orders are about the same data as of the main register of an application, ie. if the main register is about Customer and Product Management, then the Orders are about Customers and Products, as well. If the main register is about Logical Network Inventory Management, then the Orders will contain the same data classes.

ITU-T Recommendation M.1402 define data of Customer and Product Management; M.1405 define Orders of the same. M.1401 define data of Logical Network Inventory Management. M.1401 define Orders of the same. M.1403 is a common introduction to M.1404-M.1405. This Technical Report is a tutorial on M.1403-M.1405.

An application may contain both main register data and Order data. We recommend that they are implemented together as one integrated application, and we discourage separate implementations, as so much can be achieved by inheritance and references between the parts.

The application may contain instance data in its population and class data in its schema. We need to define and manage both. The schema data may appear in a Data dictionary. These data are instances relative to the meta schema. This Technical Report starts out with defining the meta schema, which is instantiated into schemata, which are instantiated into populations.

This Technical Report will show entity classes only, and references between these entity classes. This Technical Report will not discuss attribute groups, attributes or values, as of M.1401 and M.1402.

This Technical Report will address the contents of the External terminology layer only, ie. we discuss the application data in a normal form only. We are not discussing the contents of the six other layers of the Data transformation architecture according to ITU-T Z.601.

A last chapter of this Technical Report introduces Work flows as an extension of M.1403-M.1405.

This Technical Report may be used as reference material

- for writing requirements in Requests for Quotations (RfQs)
- design of efficient implementations
- coordinating interworking between several system domains

The primary audiences of this Technical Report are IT architects and IT developers. The contents affects end users and managers. However, the contents is not presented for their use.

2 References

[1] ITU. ITU-T M.1400-M.1999. *Designations and Information Exchange*. http://www.itu.int/ITU-T/recommendations/index.aspx?ser=M.

[2] ITU. ITU-T M.1401. *Formalization of interconnection designations among operators' telecommunication networks*. http://www.itu.int/ITU-T/recommendations/rec.aspx?rec=8854. Geneva 07/2006.

[3] ITU. ITU-T M.1402. *Formalization of data for service management*. http://www.itu.int/ITU-T/recommendations/rec.aspx?rec=11620. Geneva 05/2012.

[4] ITU. ITU-T M.1403. *Formalization of generic orders.* http://www.itu.int/itu-t/recommendations/rec.aspx?rec=9092. Geneva 08/2007.

[5] ITU. ITU-T M.1404. *Formalization of orders for interconnections among operators' networks*. http://www.itu.int/ITU-T/recommendations/rec.aspx?rec=9093. Geneva 08/2007.

[6] ITU. ITU-T M.1405. *Formalization of orders for service management among operators*. http://www.itu.int/ITU-T/recommendations/rec.aspx?rec=9093. Geneva 08/2007.

[7] ITU. ITU-T M.3190. Shared information and data model (SID). http://www.itu.int/itu-t/recommendations/rec.aspx?rec=9541. Geneva 07/2008.

[8] ITU. ITU-T Z.351. *Data oriented human-machine interface specification technique – Introduction.* http://www.itu.int/ITU-T/recommendations/rec.aspx?rec=3187. Geneva 3/1993.

[9] ITU. ITU-T Z.352. *Data oriented human-machine interface specification technique – Scope, approach and reference model*. http://www.itu.int/ITU-T/recommendations/rec.aspx?rec=3188. Geneva 3/1993.

[10] ITU. ITU-T Z.601. *Data architecture of one software system*. http://www.itu.int/ITU-T/recommendations/rec.aspx?rec=9061. Geneva 2/2007.

[11] ITU. List of ITU Carrier Codes (ICCs). https://www.itu.int/oth/T0201.

[12] NECA. Company Codes (OCNs). https://www.neca.org/Code_Administration.aspx.

3 Terms and definitions

3.1 Terms defined elsewhere

This technical report uses the following terms defined elsewhere:

None.

3.2 Terms defined here

This technical report defines the following terms:

Each subsection of sections 8 - 12 defines a term.

4 Abbreviations

BSS	Business Support Systems
CC	Country Code
CRM	Customer Relationship Management
ICC	ITU Carrier Code

ISO	International Organization for Standardization
NECA	National Exchange Carrier Association
OSS	Operation Support Systems

5 Background

Recommendations M.1403-M1405 provide a modern and general definition of Orders for both OSS and BSS. However, these Recommendations do not explain the thinking behind the data definitions. This is the purpose of the current Technical Report.

This Technical Report aims at

- help understanding
- guide implementation
- advice on scope

of Order implementations.

This Technical Report will not teach the various management tasks and not teach about software tools to manage the data. Rather, we teach the data structures, which will be common for both human users and software applications.

The Order data are defined as they will appear at all user interfaces.

6 Terminology

ITU-T Recommendation M.1401 uses a particular formalism for defining the data. This Technical Report will add some more explanation on this formalism.

The particular data structure that defines the language used at the human-computer interface is defined in an External terminology schema. This is one component of the Data transformation architecture for IT systems, defined in ITU-T Recommendation Z.601 Data architecture of one software system [8].

An External terminology schema defines classes that act as general prototypes/templates for the data instances about a particular Universe of Discourse. M.1402 defines the External terminology schema for service management among operators. A collection of instances is called a population. This Technical Report will define the classes and show example instances.

Both schemata and populations may be documented in a graphical or alphanumerical notation. In this technical report we will use only the graphical notation. We use the same notation for classes and instances. Hence, the data are not classes or instances in an absolute way; they are only so relative to each other.

7 Notation

This section presents a graphic notation for defining data classes and show example data instances.

We will use the following notation:

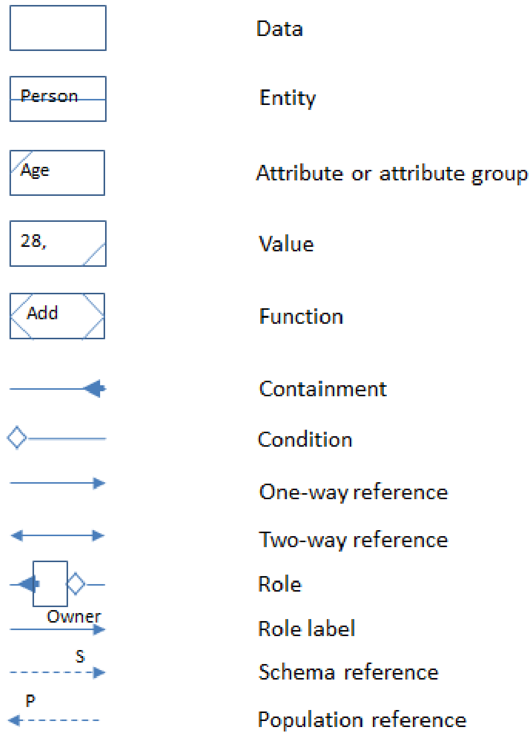

Figure 7-1 Graphic notation

The relationships are stated as

- Containments, or
- One-way references
- Two-way references

Containments tell that the contained class has a class label local to the superior class. An entity class can only be local to one superior entity class.

Instances are created by taking copies of the class and its superior containment. If class B is contained in class A, then an instance of a class B must be contained in an instance of class A.

Moreover, if class B is contained in class A, then the value of the identifying attribute (group) of an instance of a class B shall be local to the value of the identifying attribute (group) in an instance of class A. Hence, the containment shows how entities are identified locally to each other.

In this Technical Report, we will use both one-way and two-way references. Reference classes between entity classes are copied into reference instances between entity instances.

The copying described in the previous paragraphs is called an instantiation mechanism, and it explains how the contents of populations are created. Data instances and data classes look alike, and they cannot be distinguished when viewed in isolation.

We will additionally use the one-way schema reference in this Technical Report. This reference will have the role label S.

The reader may find more explanation of the notation in Appendix III of [2].

8 The External Terminology Layer

Orders are data, and they may contain data, within the External Terminology Layer (eTL) of an Application System (A). The Order data are about the same Universe of Discourse as of the main register of the Application. Therefore, we have to understand the Application data before we can understand the Order data.

The root data nodes within the eTL may be External Terminology Populations (eTP), External Terminology Schemata (eTS) or External Terminology Meta-Schema (eTMS). The External Terminology Meta-Population (eTMP) may be an eTS. A schema for As and their eTLs is shown in Figure 8-1.

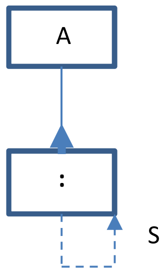

Figure 8-1 Schema for eTL nodes

That an eTL is contained in an A is indicated by a containment arrow in the schema, ie. with a reversed arrow head. The schema in Figure 8-1 allows an instance of A to have many contained eTL instances.

We will allow the eTL nodes to have similar capabilities as of any data node in our applications. Therefore, we indicate this node by a colon (:) without any name label in this meta schema. Instances of this node may be eTPs, eTSes or eTMSes. However, other data nodes are permissible, as well.

We have added a recursive schema reference (S) from and to this node in the above schema. This allows schema references to be stated between its instances, eg. from an eTP to an eTS, and from an eTS to an eTMS. This is illustrated in Figure 8-2.

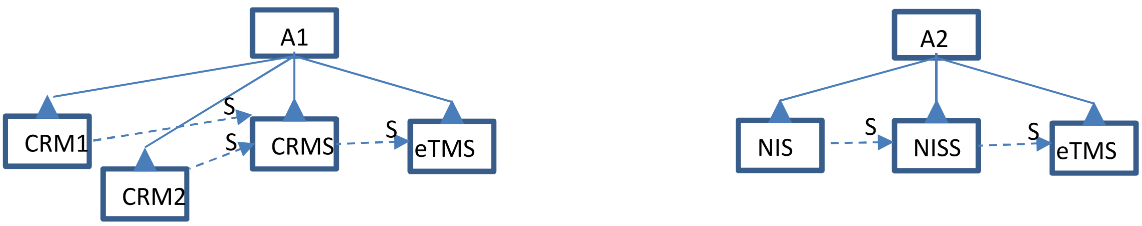

Figure 8-2 eTL instances

In Figure 8-2 have defined two applications, A1 and A2.

Note in that the name labels of the nodes in Figure 8-2 are insignificant. It is the schema references that give them their roles, eg. that CRM2 is an eTP relative to CRMS, which is an eTS. The role of an eTS, eg. CRMS, is to contain data that can be copied into its eTPs. The contents of an eTS are called classes relative to their corresponding instances in an eTP.

Anything written in an eTS may be copied into its eTP multiple times, and anything written in the eTMS may be copied into its eTS. What is written in a schema prescribes what may be written in its populations. The classes are prototypes of their instances.

Any data node may contain other data nodes, recursively. This is illustrated in Figure 8-3.

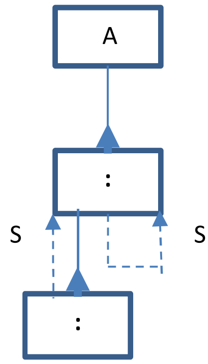

Figure 8-3 Schema of any data node in eTL

The subordinate data node in Figure 8-3 contains a schema reference (S) to its superior node. This allows the subordinate node to inherit any property of its superior node in this schema, eg. that it may contain subordinate to subordinate nodes recursively. Figure 8-3 prescribes how we can create trees of data in the eTLs, in the eTMS itself and both in the eTSes and in the eTPs.

In Figure 8-4, we add a Condition (<>) from the subordinate to the superior data node. Hence, the previously introduced schema references allow conditions to be stated both among the instances of eTL and among its subordinate nodes and their instances.

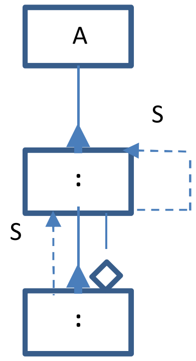

Figure 8-4 Conditions on any data node in eTL

Conditions are used to state references between nodes. Node B may contain node C that has a condition that states a reference to node D. Node B is an instance of the node or subordinate node in Figure 8-4. This is illustrated in Figure 8-5.

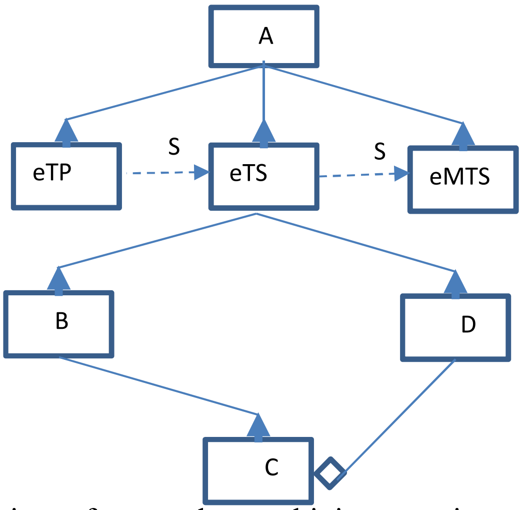

Figure 8-5 Stating references by combining containment and conditions

The shorthand notation for a reference is an arrow. The name label of the contained node (C) in Figure 8-5 may be written over the arrow, close to the arrow head. If no name label is shown, it is assumed to be identical to the name label of the referenced node. See Figure 8-6. This role label may also be omitted due to lack of space.

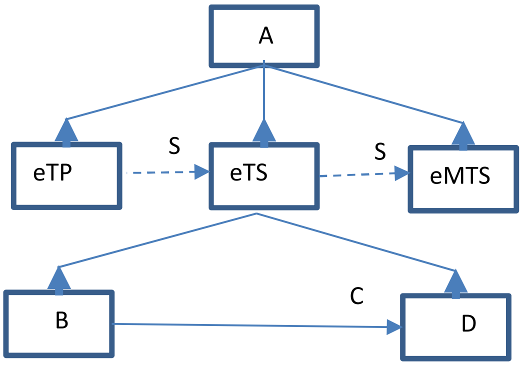

Figure 8-6 Short hand notation for the information in Figure 5

Two-way references have arrow heads in both ends. The reference to the right has the name label above the two-way arrow. The reference to the left has the name label below the arrow.

The schema references are a special case of references, where node C has the name label S.

Note that the two schema references in Figure 8-2 may be replaced by a condition and a subordinate to subordinate S, as being shown in Figure 8-7.

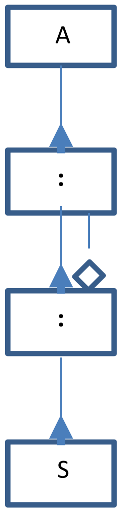

Figure 8-7 The Schema label S is a subordinate node to the referencing node

When the S is used, it is a schema reference. When S is not used, it is an ordinary reference. The short-hand notation for a schema reference is a dashed arrow with an S.

Figure 8-7 illustrates how name labels are treated. They are treated as subordinate nodes to the actual node. The name labels are attached in order to simplify search and identification. Normally, we use only local names, ie. name labels and identifiers are local to the superior node of the current node. The superior node is the root of the name space of its subordinate nodes.

If the S is wanted as a role label of a regular reference, and not as a schema reference, it may be replaced by the disjunction operator (v). This means that what can be written in the population is a disjunction of what is written in the schema.

We have now introduced the means to state an actual eTS. An example eTS is shown in Figure 8-8.

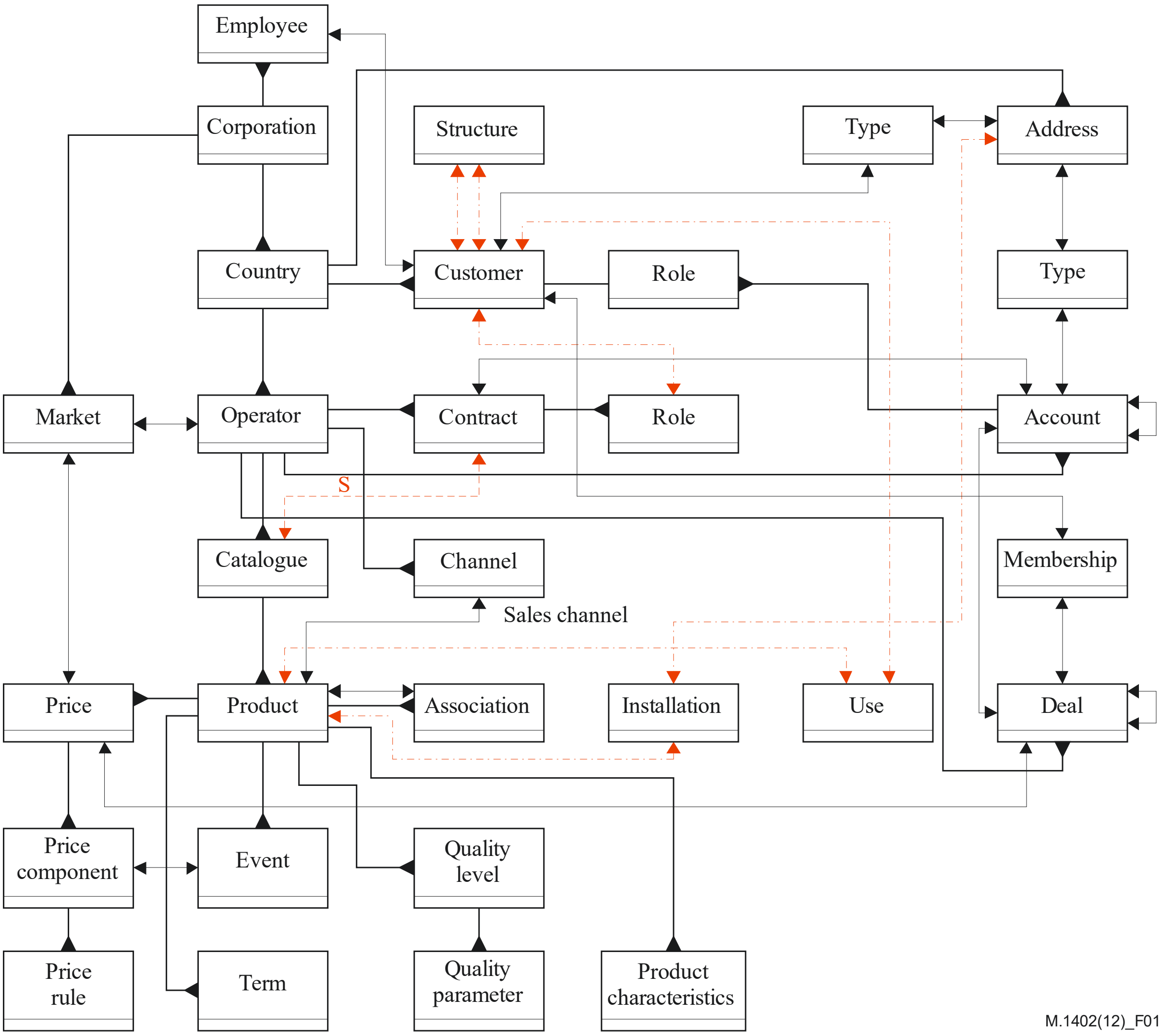

Figure 8-8 eTS for Customers and Products according to ITU-T Recommendation M.1402

All nodes in Figure 8-8 are instances of the superior node in Figure 8-4. Some of the nodes in Figure 8-8 do not show a superior containment. All these nodes are contained in an eTS. All nodes in Figure 8-8 are entities, not attributes or values. Entities are indicated by the underlining in each box.

The red dotted arrows in Figure 8-8 indicate references that may be derived from other data.

Figure 8-9 depicts some data instances in an eTP according to the schema in Figure 8-8.

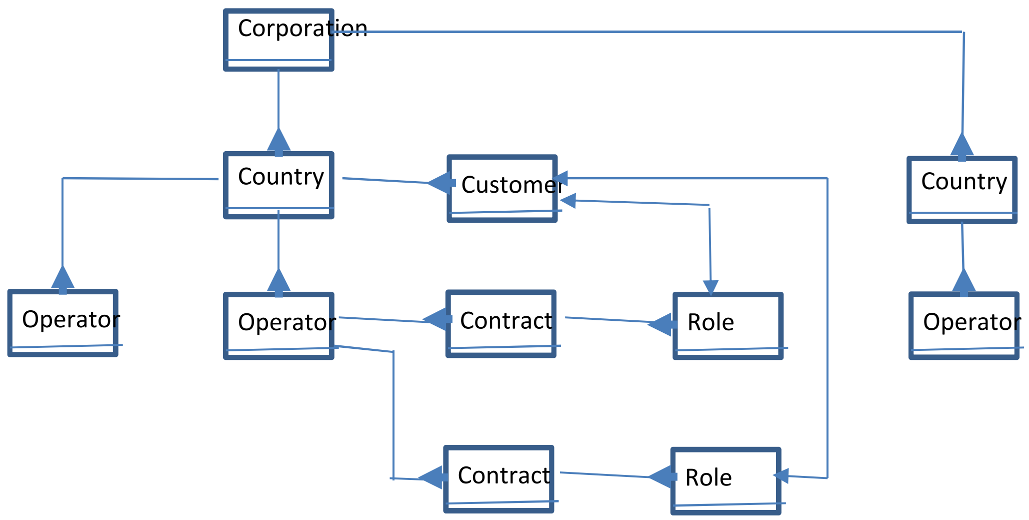

Figure 8-9 Example data instances

In Figure 8-9 we have omitted the containment arrows from the eTP to its contained Corporation. Note that we allow for use of both local names and significant duplicates within the same name space. Also, we have not shown attributes and values, which may be similar or different from each other. We do not assume uniqueness, when this is not explicitly stated.

In Figure 8-9 we have three Operators, two in one Country and one in a separate Country. They are data instances within one eTP. In one of the Countries there is one Customer who has two Contracts with one and the same Operator.

9 Time dimension

We want to record the time sequence of when data are entered into our Application System (A). This is achieved by adding a recursive reference to the superior data node, which may be inherited to its subordinate nodes, recursively. We may use this reference to record the creation sequence, eg. D, A, K, B etc., of the data instances

The recursive time reference is shown in Figure 9-1.

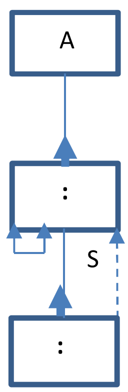

Figure 9-1 Recursive time reference between data nodes

The recursive time reference in Figure 9-1 allows for stating time references between any data node across all data classes. This is a global time reference, sorting every event in the application into one sequence.

The time reference has a sequence of past and future.

In Figure 9-2, we introduce a local time reference between data instances of the same class only.

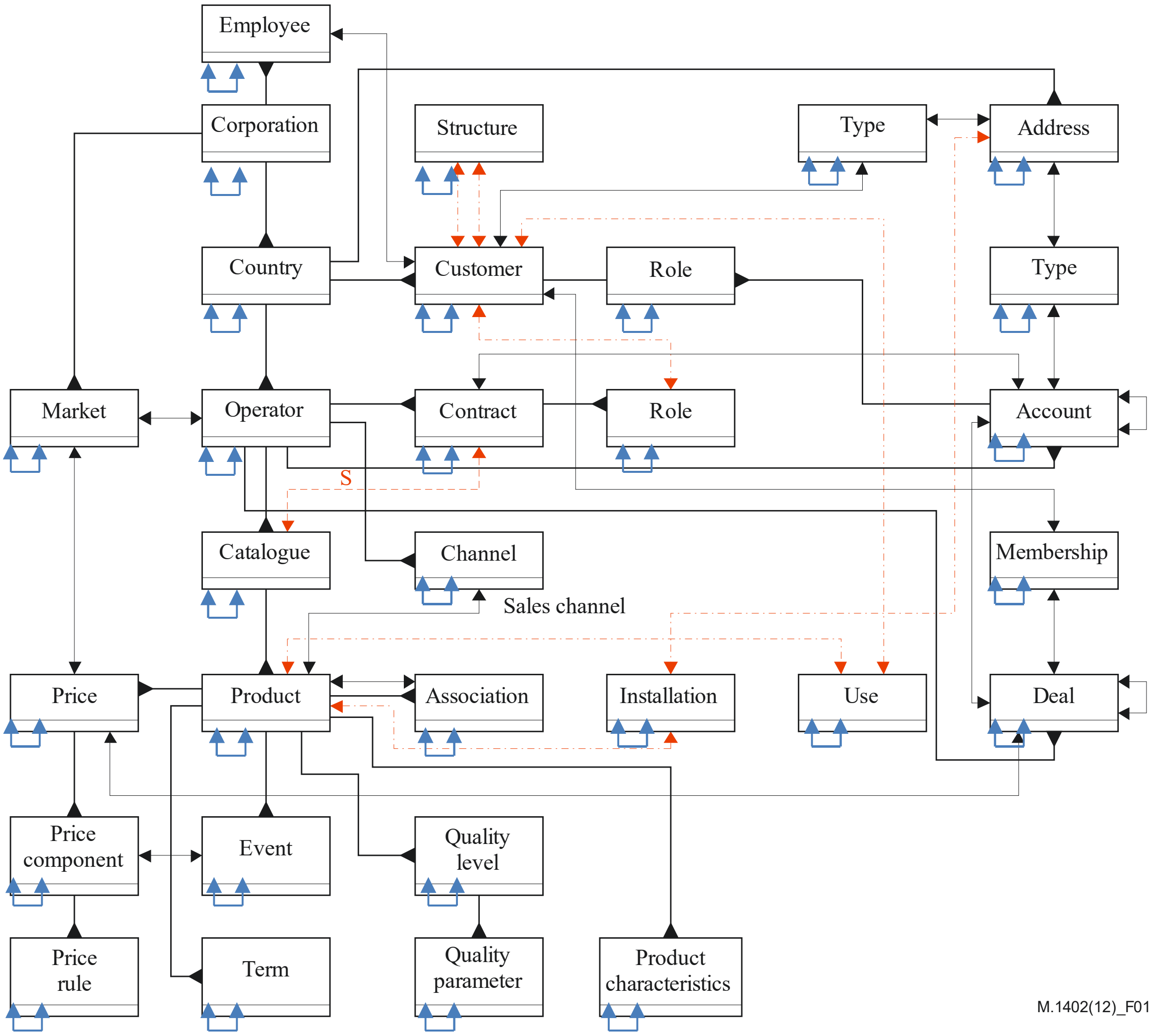

Figure 9-2 Example time references local to each class in Figure 8-8

We may further restrict the time references to be between data nodes having the same name label and identifier, ie. between duplicates of the same entity instance. Each time reference instance is then used to record any change to this entity instance. This is illustrated in Figure 9-3. We are now in an eTP.

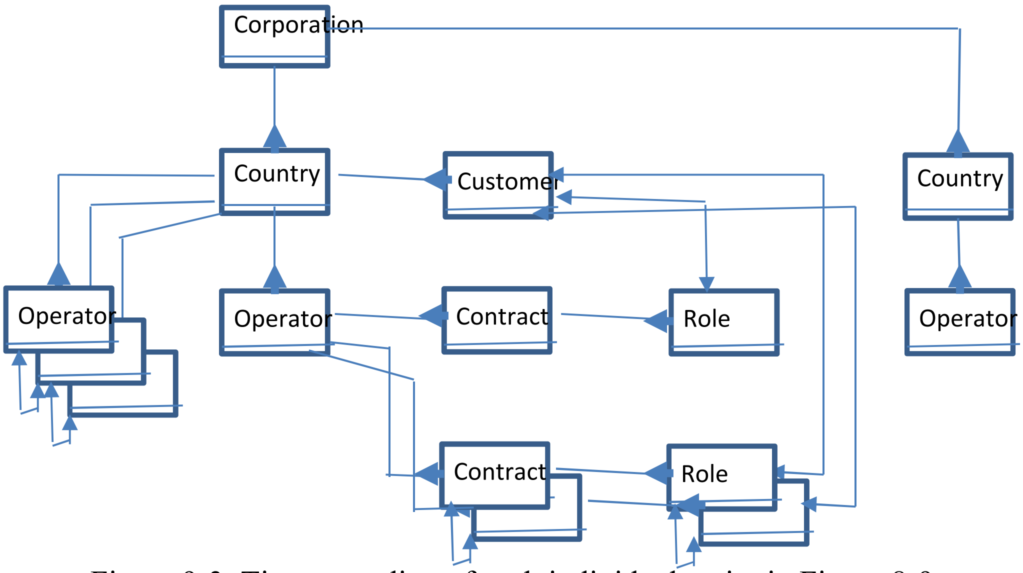

Figure 9-3 Time recording of each individual entity in Figure 8-9

Support of time recording within each entity instance, as illustrated in Figure 9-3, is called an Immutable database. This approach is used in some commercial software packages for Customer Relationship Management (CRM). Note that Figure 9.3 illustrates time sequences of the events of one Operator only, one Contract only and one Role only.

The M.1400 series Recommendations do not use Immutable databases. Rather, Orders are used to record time.

Use of Immutable databases increases the complexity of the business logic, as there may be many data nodes for each entity instance. Therefore, every cardinality of relationships become n:m, which will affect all business logic. Also, Immutable databases affect the user interface, as the user may have to distinguish the various versions of the same entity.

The notion of time dimension in Immutable databases will be used in the next chapter to define the time dimension of Orders.

Note that the time dimensions being discussed in this chapter are about the sequences of events among the data. The users of the data are often more interested in the timing within the Universe of Discourse (UoD) of the data, ig. when a fixed line is installed, when services are activated in the network, when invoices are posted to the customer, when payments are made etc. We will treat this kind of time in the UoD as time stamp attributes specific to each entity class.

10 Orders

A telecom Operator may manage Orders.

Orders may come from Customers, be created for activation, be sent to network management, sent to partners, such as other telecom Operators, etc.

Orders can be used between organizations, people and IT systems. Orders are means to handle long transactions between these actors, to record events within these transactions and to control their sequences and fulfilment.

Orders are used to convey data between actors that do not handle the data and give responses immediately. Orders are used for asynchronous processing.

Orders are additionally used as a repository of what has happened, and of what is ordered to happen in the future. This additional use of Orders may be replaced by Immutable databases, or vice versa.

Some commercial software packages for CRM do not use Orders to register and change Customer data. Rather they allow the Customers to change their data directly. Then it becomes important to keep copies of previous versions of the data in an Immutable database.

When an Order is sent to another actor, this actor may issue a new Order. Therefore, the Orders need to be linked by a recursive reference between source and sink Orders. The sink Orders are derived from the source Orders. This is shown in Figure 10-1.

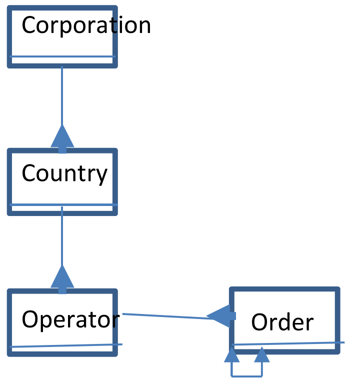

Figure 10-1 Orders

According to the M.1400 series Recommendations, Orders can be issued and managed by Operators only. However, each Operator is free to define its Vendors to be Operators, even if they are not recognized as Operators. This unification may simplify Order management, as it may allow use of the same Order management solution to all partners.

The containment of Orders is found in Messages. Messages are comparable to sentences; they may be simple or complex. However, the contents of the Messages is normalized, and is not nested like how the information may appear at the user interface.

Any Message can be a template for other Messages. Therefore, any Message may have a schema reference to its template Message, if there is one. This is shown in Figure 10-2.

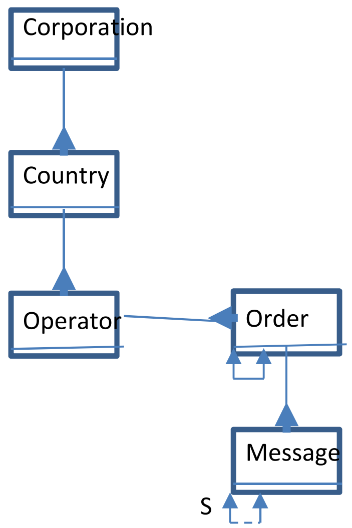

Figure 10-2 Some Messages may be schema Messages for other Messages

The contents of a Message can be anything that may be stated in a Corporation. This is stated by the schema reference from Message to Corporation. See Figure 10-3.

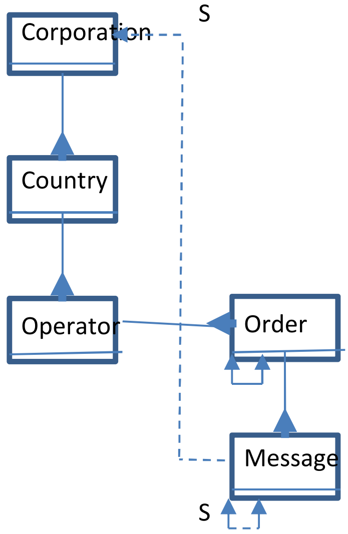

Figure 10-3 Schema reference from Message to Corporation

Message in Figure 10-3 has two schema references. It is up to the application software or end user to use which one or both.

Figure 10-3 does not allow a Corporation to be contained in a Message, but Country does. This is illustrated in Figure 10-4. If we want to address other Corporations, this would have to be defined as a special attribute either under Order or in Message.

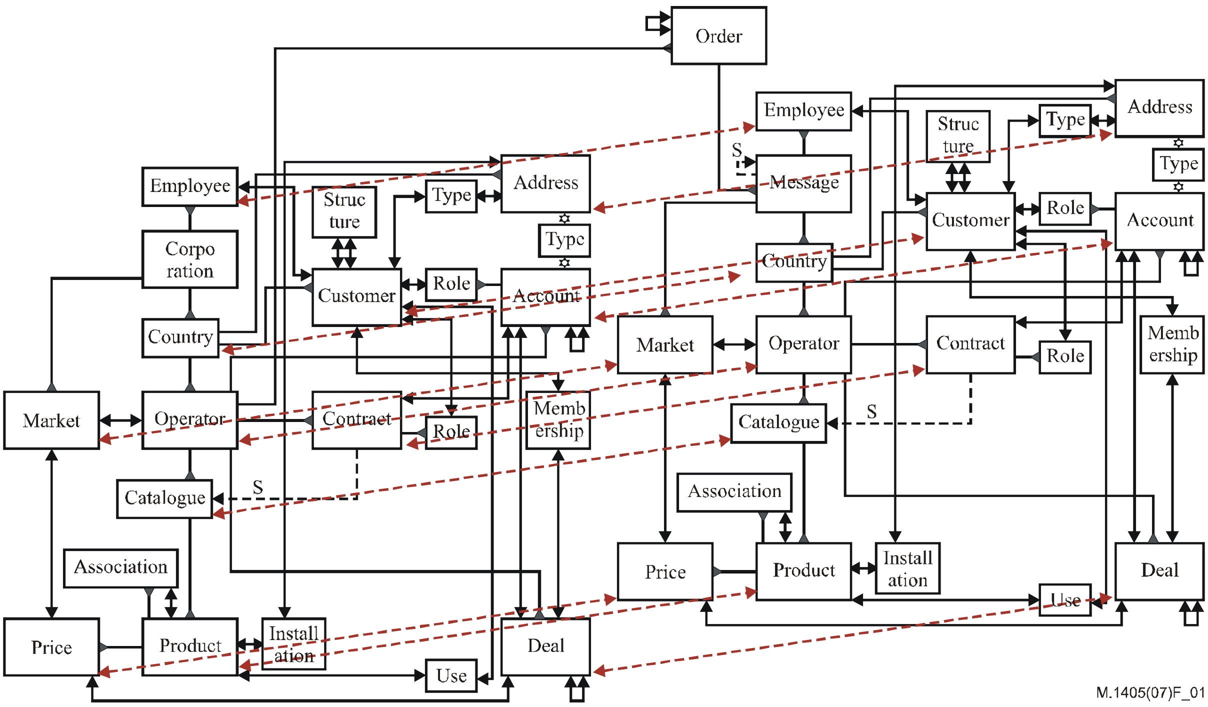

Figure 10-4 Schema of combined main and Order registers

Note that Figure 10-4 does not contain all entity classes as of 9-2. This is due to that M.1402 has been extended after issuing of M.1405, and M.1405 has not been updated.

An alternative to Figures 10-3 and 10-4 would be to define the schema reference from Message to what contains the Corporations, ie. to the eTS.

The dashed brown arrows in Figure 10-4 indicate the time dimension. The arrows refer from one entity class in one register to the same entity class in the other register. When being instantiated, the references from an entity instance in the main register refer to all mentioning of this instance in the Order register independently of which Order they appear in; these mentionings are sorted in a strict time sequence. The references the other way point to the main register entity and its current state.

The dashed brown arrows in Figure 10-4 are the same arrows as the blue recursive arrows in Figure 9-2. Hence, the references have to navigate up via a common node. How this is stated is outside the scope of this Technical Report.

Note that the entity in the main register does not always tell the current situation in the Universe of discourse. The entity may contain values of a planned state, or contain identities and values of an entity that no longer exists. The entity should contain attributes telling about its Existence status and time of past, current or planned Existence in the Universe of Discourse.

It should be noted that most commercial software packages do not support the general implementation of Orders, as described above. They may implement one or more Order types, eg. one Order type for customer requests, one Order type for provisioning, one Order type for Billing updates etc. Each Order type may use particular data types that do not correspond exactly to what is found in the main register. This Order solution may be developed by a separate team or company that does not know the details about the surrounding IT solutions. Hence, a lot of adaptions have to be made, and these Order solutions tend to become complex and very specialized.

From Figure 10-4 we see by counting entities and relations that the proposed Order solution may become two times as complex as the main register. Hence, the scope of the implementation become three fold if the extension is carried out manually. By using the schema notion in Figure 10-3, the Order solution may become a much smaller extension.

If the implementation of Orders is done separately, without understanding the general relations to the main register, as being explained in this Technical Report, the total work on Order implementation may become more that three fold the scope of the main register.

Note that the general solution described in this chapter does not only describe simple orders; it additionally describe Orders for all kinds of plans, such as redefinition of the Account hierarchy of a business Customer, change of the entire Price plan of Products within a Market, etc.

Note that this Technical Report describes the contents of the External Terminology Schema (eTS) only, and gives a few examples of the External Terminology Population (eTP). This Technical Report does not describe the contents of the Contents Layer and the Layout Layer of the implementation.

11 Operation Support Systems

Figures 9-2 through 10-4 depict examples of Customer and Product Management from the Business Support Systems domain. In this chapter, we will show examples from Logical Network Inventory Management within the Operation Support systems domain.

The schema of the main register is shown in Figure 11-1.

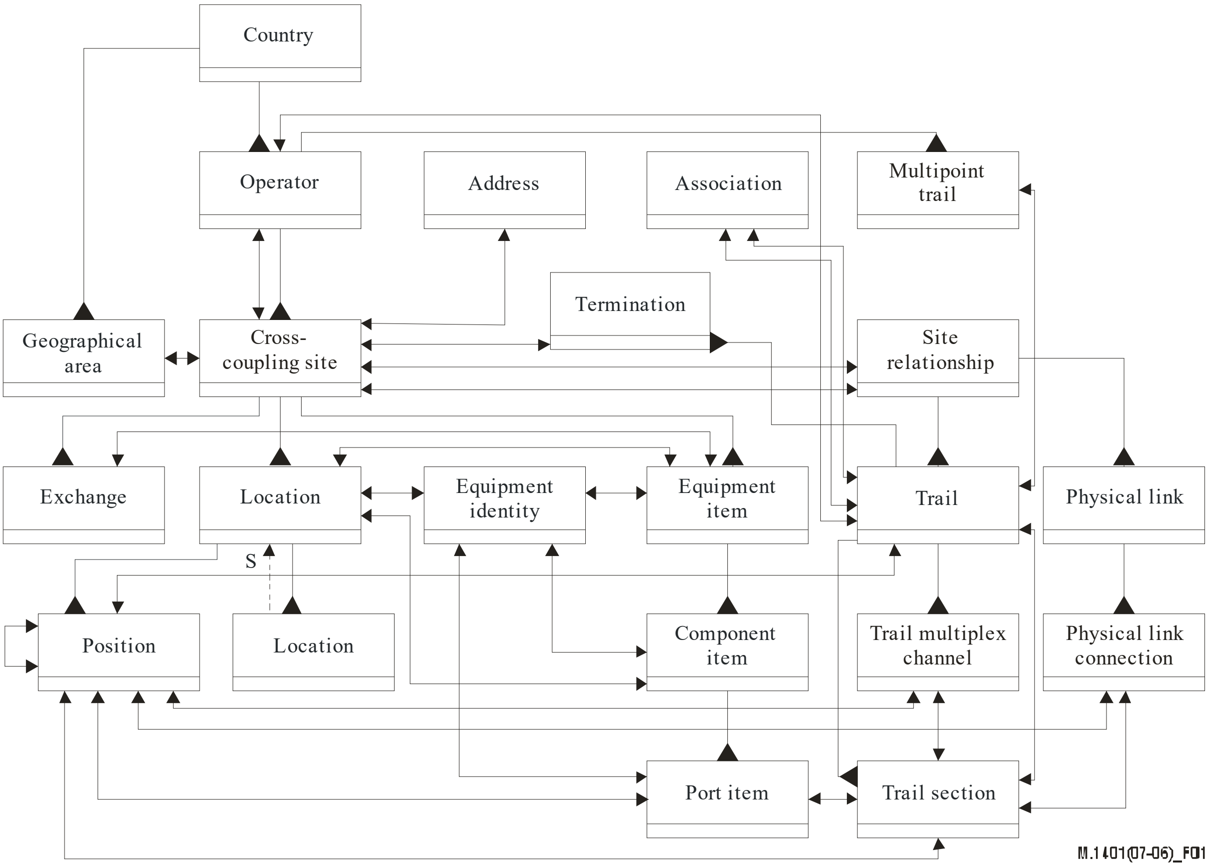

Figure 11-1 Schema of Logical Network Inventory Management

Figure 11-2 shows Orders of the same.

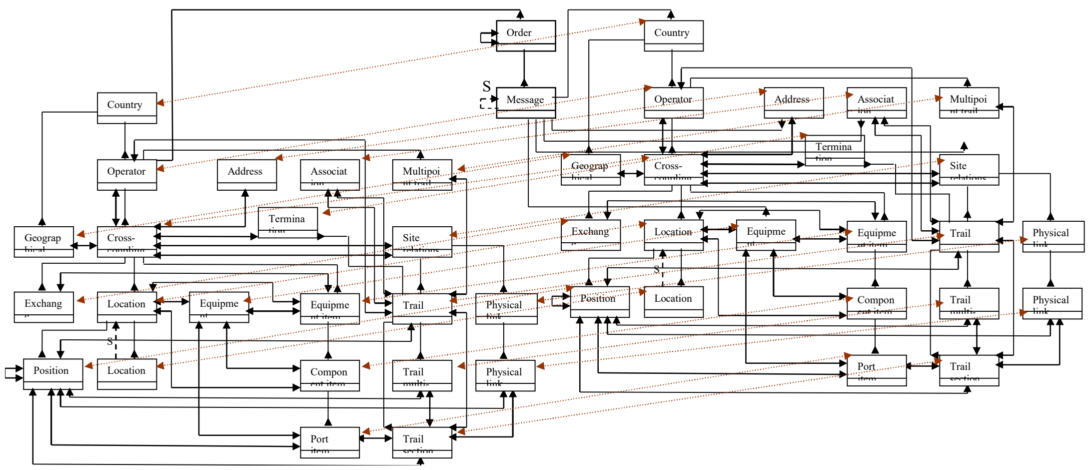

Figure 11-2 Schema of combined main and Order registers

In Figures 11-1 and 11-2 we see that Corporations are not included. However, the implementer is free to add Corporations. According to Figures 9-2 and 10-4, a Country exists only within a Corporation. Hence, Sweden in Telia and Sweden in Telenor are treated as two separate Country instances. According to Figures 11-1 and 11-2, there is only one Sweden Country instance. We realize that the differences between the two alternatives are profound.

Figure 11-3 shows the contents of an example routing of a Trail.

Trail
Identifier
9876543100 .
Trail section

Identifier	Type	A-end CC	A-end ICC	A-end Identifier	B-end CC	B-end ICC	B-end Identifier	Section	Item
0i	O	FRA	ORANPARIS1					RO4/RA8/BL4	4
1	T	FRA	ORANPARIS1		NOR	TELN OSLO1		315	4
1i	O	NOR	TELN OSLO1					RO2/RA1/BL1	14
1ii	O	NOR	TELN OSLO1					RO2/RA2/BL4	14
1a	R	NOR	TELN OSLO1					AMPL3/CO2 P1	
1b	R	NOR	TELN OSLO1					AMPL3/CO2 P2	
1iii	O	NOR	TELN OSLO1					RO2/RA1/BL8	6
1ii	O	NOR	TELN OSLO1					RO2/RA3/BL4	2
2	T	NOR	TELN BERG2		NOR	TELN OSLO1		2	412
2i	O	NOR	TELN BERG2					RO1/RA1/BL7	2
2i	O	NOR	TELN BERG2					RO1/RA4/BL3	1
3	L	NOR	TELN BERG2		NOR	TELN BERG2-5		BERG2 C31-40	31
2i	T	NOR	TELN BERG2-5					RA1/BL1	1

Figure 11-3 Example routing of a Trail

Figure 11-4 shows how this routing may be split on several Messages in an Order.

Order
Identifier
123

Message
Identifier Action

1 ESTABLISH

Trail
Identifier
9876543100 .
Trail section

Identifier	Type	A-end CC	A-end ICC	A-end Identifier	B-end CC	B-end ICC	B-end Identifier	Section	Item
0i	O	FRA	ORANPARIS1					RO4/RA8/BL4	4
1	T	FRA	ORANPARIS1		NOR	TELN OSLO1		315	4
1i	O	NOR	TELN OSLO1					RO2/RA1/BL1	14
1ii	O	NOR	TELN OSLO1					RO2/RA2/BL4	14
1a	R	NOR	TELN OSLO1					AMPL3/CO2	P1

Message

Identifier Action

2 ESTABLISH

Trail
Identifier
9876543100 .
Trail section

Identifier	Type	A-end CC	A-end ICC	A-end Identifier	B-end CC	B-end ICC	B-end Identifier	Section	Item
1b	R	NOR	TELN OSLO1					AMPL3/CO2	P2
1iii	O	NOR	TELN OSLO1					RO2/RA1/BL8	6
1ii	O	NOR	TELN OSLO1					RO2/RA3/BL4	2
2	T	NOR	TELN BERG2		NOR	TELN OSLO1		2	412
2i	O	NOR	TELN BERG2					RO1/RA1/BL7	2

Message

Identifier Action

3 ESTABLISH

Trail

Identifier

9876543100 .

Trail section

Identifier	Type	A-end			B-end			Section	Item
		CC	ICC	Identifier	CC	ICC	Identifier		
2i	O	NOR	TELN	BERG2				RO1/RA1/BL7	2
2i	O	NOR	TELN	BERG2				RO1/RA4/BL3	1
3	L	NOR	TELN	BERG2	NOR	|TELN	BERG2-5	BERG2 C31-40	31
2i	T	NOR	TELN	BERG2-5				RA1/BL1	1

Figure 11-4 Example routing Messages

12 Work flows

Messages may be sent in Queues between Actors.

The Actors may be IT systems, organization units, and partners, including other Operators. The Queues define one-way pipes between two Actors. Hence, a Queue needs to have a To and From Actor. This is illustrated in Figure 12-1.

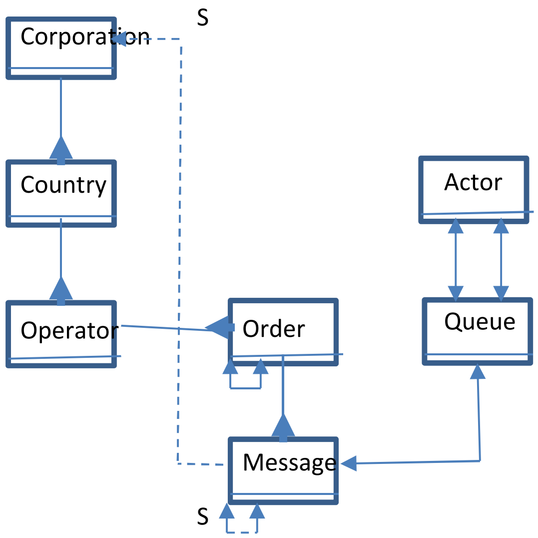

Figure 12-1 Schema of Queues between Actors

The Queues define directed connections between Actor instances. There may be more than one Queue between two Actors, eg. one Queue for each Message type or Order type. Each Actor may contain processes for how to process Messages of each Queue.

A Queue may contain a sequence of Messages from different Orders. These Messages may contain a Message Status telling where this Message is in the processing. Also, the Status may tell what is to be expected. Hence, the Status may go from To-be-activated to Is-activated.

Whenever something is written in the Message, a new version of the Message may be created, like in Immutable databases. This would generate the most complete history of what has happened in the processing. However, a simpler solution would be to allow for over-writing within the same Message instance, as this may create enough information on the history.

When allowing for over-writing within one Message instance, the question appears if the Message should be allowed to exist within several Queues. If so, the Queues may grow, and you have no cleaning of the contents of Queues. If we allow for deletion of references from Queues to Messages, the processing become much cleaner. Through the processing, each Message is linked from one incoming Queue to an outgoing Queue. We see that the implementer has many design options.

The sequence of Queues to be used by a Message may depend on the Message Type and the Message Status. The logic of this processing sequence may be defined in a long Transaction/Routine. Transactions/Routines are not defined in this Technical Report.

The Transactions/Routines may involve Brokering. Suppose the next process to be activated is Installation. Several Installation companies are registered as Actors to perform this task. Which Actor to choose depends on what manual resources are available, at which location, with which competence, at which security level, at which time and at which price. A Brokering Actor may have access to all this information and choose the optimal Installation Actor. This Technical Report does not define the Brokering mechanism.

Note that the Queues in Figure 12-1 refer to Messages. However, the Messages are not identified within Queues; they are identified within Orders. Hence, when Messages are communicated within an Operator they have to be identified by their global distinguished name, ie. by the Order identifier plus the Message identifier within that Order. Whenever the Order is communicated outside the Operator, also the Operator identifier has to be provided.