INTERNATIONAL STANDARD

ISO 20022-3

First edition
2013-05-01

Financial services — Universal financial industry message scheme —

Part 3: **Modelling**

Services financiers — Schéma universel de messages pour l'industrie financière —

Partie 3: Modélisation

Reference number
ISO 20022-3:2013(E)

COPYRIGHT PROTECTED DOCUMENT

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of technical committees is to prepare International Standards. Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

ISO 20022-3 was prepared by Technical Committee ISO/TC 68, *Financial services*.

This first edition cancels and replaces ISO/TS 20022-3:2004.

ISO 20022 consists of the following parts, under the general title *Financial services — Universal financial industry message scheme*:

— *Part 1: Metamodel*

— *Part 2: UML profile*

— *Part 3: Modelling*

— *Part 4: XML Schema generation*

— *Part 5: Reverse engineering*

— *Part 6: Message transport characteristics*

— *Part 7: Registration*

— *Part 8: ASN.1 generation*

ISO 20022-1:2013, ISO 20022-2:2013, ISO 20022-3:2013, ISO 20022-4:2013, ISO 20022-5:2013, ISO 20022-6:2013, ISO 20022-7:2013 and ISO 20022-8:2013 will be implemented by the Registration Authority by no later than the end of May 2013, at which time support for the concepts set out within them will be effective. Users and potential users of the ISO 20022 series are encouraged to familiarize themselves with the 2013 editions as soon as possible, in order to understand their impact and take advantage of their content as soon as they are implemented by the Registration Authority. For further guidance, please contact the Registration Authority.

For the purposes of research on financial industry message standards, users are encouraged to share their views on ISO 20022:2013 and their priorities for changes to future editions of the document. Click on the link below to take part in the online survey:

http://www.surveymonkey.com/s/20022_2013

Introduction

This International Standard defines a scalable, methodical process to ensure consistent descriptions of messages throughout the financial services industry.

The purpose of this International Standard is to describe precisely and completely the externally observable aspects of financial services messaging in a way that can be verified independently against operational messaging.

The trigger for the creation of this International Standard was the rapid growth in the scale and sophistication of messaging within financial services during the 1990s using ISO 15022. The financial services industry (from here on referred to as "the industry") created the first version of this International Standard as the successor to ISO 15022 in response to that trigger. Since ISO 15022, the industry has broadened the scope from securities to the entire industry for this International Standard.

This International Standard is based on open technology standards, which historically have evolved more rapidly than the industry itself. Consequently, this International Standard adopted a model-driven approach where the model of the industry's messaging can evolve separately from the evolution of the messaging technology standards. The period during which this International Standard has emerged followed the widespread adoption of the World Wide Web (the Web) for business. XML (eXtensible Mark-up Language) emerged as the *de facto* standard for document representation on the Web and it became the first syntax for ISO 20022.

The modelling process is further refined into three levels which, in addition to the messaging technology standard, is why this International Standard is based on four levels: the Scope level, the Conceptual level, the Logical level and the Physical level.

This four-level approach is based on the first four levels of the Zachman Framework. The remaining two levels of the Zachman Framework are equivalent to the implementations and the operational levels, respectively.

In ISO 20022-1, the first, second and third levels are described in UML (Unified Modelling Language) because it is widely supported and supports multiple levels of abstraction. The models created in accordance with this International Standard are technology independent in that they do not require any particular physical expression or implementation. Such models aim to describe all parts of the message exchange. The models form the definition of the protocol between participants exchanging messages. This International Standard defines a method that describes a process by which these models can be created and maintained by the modellers.

The models and the Physical level artefacts are stored in a central repository, serviced by a Registration Authority. This International Standard's repository is available on the World Wide Web and offers public access for browsing.

The Repository is organized into two areas:

— A DataDictionary containing the industry model elements likely to have further or repeated use.

— A BusinessProcessCatalogue that contains models describing specific message definitions and business processes, and physical syntax implementations.

This International Standard is organized into the following parts.

— ISO 20022-1 describes in MOF (Meta-Object Facility) the metamodel of all the models and the Repository.

— ISO 20022-2 covers the UML profile, a grounding of general UML into a specific subset defined for this International Standard (to be used when UML is selected to define the models).

— This part of ISO 20022 describes a modelling method to produce models for this International Standard.

— ISO 20022-4 covers XML schema generation rules to transform a Logical level model into a Physical level description in the syntaxes.

— ISO 20022-5 covers logical model alignment and reverse engineering of existing message syntaxes.

— ISO 20022-6 covers message transport characteristics that define the quality of service required by the business process definitions so that they can operate successfully.

— ISO 20022-7 describes the process of managing the registration of models and physical syntax implementations.

— ISO 20022-8 gives ASN.1 syntax generation rules to transform a Logical level model into a Physical level description in ASN.1.

Financial services — Universal financial industry message scheme —

Part 3: Modelling

1 Scope

This part of ISO 20022 describes the modelling workflow, complementing ISO 20022-1 and ISO 20022-2. The modelling workflow describes the required steps a modeller follows in order to develop and maintain standardized BusinessTransactions and MessageSets.

This part of ISO 20022 is not intended to describe what will be the permissible artefacts and/or documents to be submitted to the Registration Authority (this information is contained in ISO 20022-7).

Examples are provided only to illustrate the modelling methodology and are not normative.

2 Normative references

ISO 20022-1, *Financial services — Universal financial industry message scheme — Part 1: Metamodel*

ISO 20022-2, *Financial services — Universal financial industry message scheme — Part 2: UML profile*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO 20022-1 and ISO 20022-2 apply.

4 Workflow activities overview

The objective of a standardized BusinessTransaction is to define a commonly agreed solution for communication problems existing among different organizations within the context of a given BusinessProcess.

For a given communication problem in a given business context, several solutions can be developed. The purpose of this part of ISO 20022 is to explain the different steps a modeller should follow to ensure that all ISO 20022 items such as BusinessComponents/BusinessElements, MessageComponentTypes/MessageElements, BusinessTransactions and MessageDefinitions are defined in a consistent way.

The ISO 20022 methodology is composed of a set of activities. These activities are grouped into the following levels:

— Scope level;

— Conceptual level;

— Logical level;

— Physical level.

For each of these activities, this part of ISO 20022 describes:

— the artefacts needed to start this activity (required input);

— the artefacts that should be the result of this activity (expected output);

— an example (where useful);

— any constraints and rules of modelling that should be followed or taken into account.

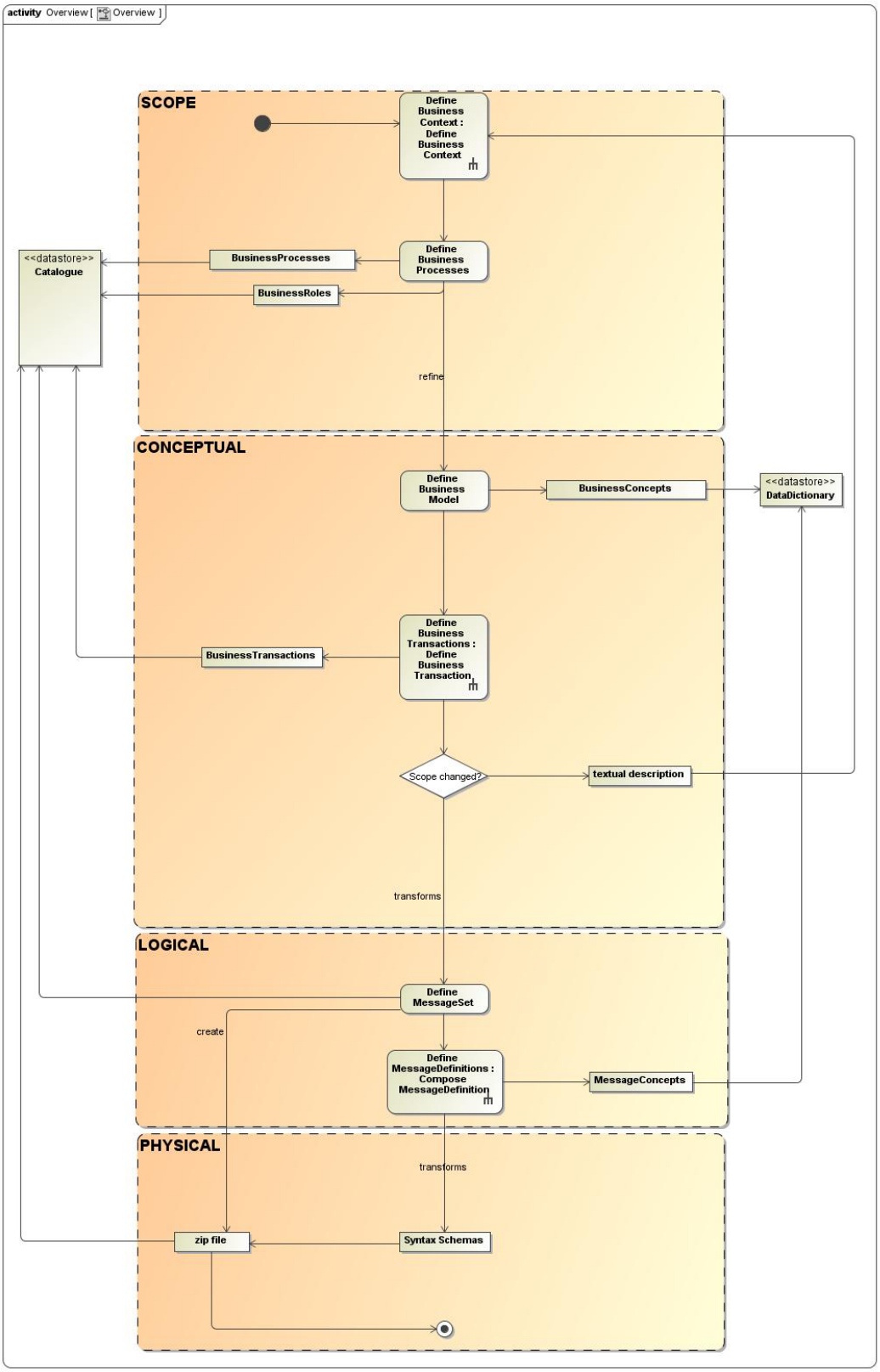

Figure 1 — High level workflow activity diagram

5 Scope level

5.1 General

5.1.1 Purpose

The purpose of the Scope level is to achieve a better understanding of the BusinessArea and BusinessProcesses for which ISO 20022 compliant BusinessTransactions and MessageSets will be developed.

a) Describing the BusinessProcesses, including the BusinessRoles and their need for business information, helps in the identification of the communication problems that exist among the organizations that take part in these processes. Those communications problems are the main drivers for the next phase (Conceptual level).

b) Identifying business information manipulated in a BusinessArea is also important because the MessageDefinitions, which will be designed later, will contain data elements that are related to the BusinessConcepts. An explicit link between BusinessConcepts and MessageConcepts will be helpful for interoperability for later maintenance and for change management; if something changes in a BusinessArea, it will be possible to identify the impact on previously defined MessageDefinitions.

5.1.2 Key topics

The key topics of the Scope level are:

— identification of the business context of the communication problem to be solved;

— understanding the daily business in the BusinessArea and BusinessProcesses with no special focus on the BusinessTransactions and MessageSets to be developed;

— capturing the Business Concepts manipulated within the BusinessProcesses;

— ensuring that all users, such as business experts and standards developers, have a common understanding of the BusinessArea and the BusinessProcesses.

5.1.3 Main activities

The main activities of the Scope level are:

— specification of the boundaries of the project by selecting the BusinessArea(s), defining the Business Goal and identifying key BusinessComponents;

— specification of the BusinessProcesses.

5.1.4 Deliverables

The deliverables of the Scope level are:

— a textual description of the Business Context (objectives, scope and boundaries);

— models describing the BusinessProcesses, and the BusinessContext and BusinessRoles involved in these BusinessProcesses. (all model elements are enriched with textual descriptions, including a glossary of business terms).

5.2 Activities

5.2.1 Define business context

The following steps are followed to define the business context (see Figure 2).

— Define the BusinessArea(s): identification and any assumptions related to the BusinessArea(s) covered by the scope.

— Specify the business goal: global objectives and purposes for the considered BusinessArea, expected functionality and constraints (e.g. market infrastructures to be part of the solution, performance requirements such as settlement having to occur one business day after the Trade, interoperability requirements, market practices to be considered).

— Identify key BusinessComponents and Elements that are handled and/or used within the business context. They can be classified into input information (i.e. information that influences the BusinessProcesses but that is not controlled within the scope of the business context) and output information (i.e. information that is controlled within the scope of the business context and impacts other BusinessProcesses).

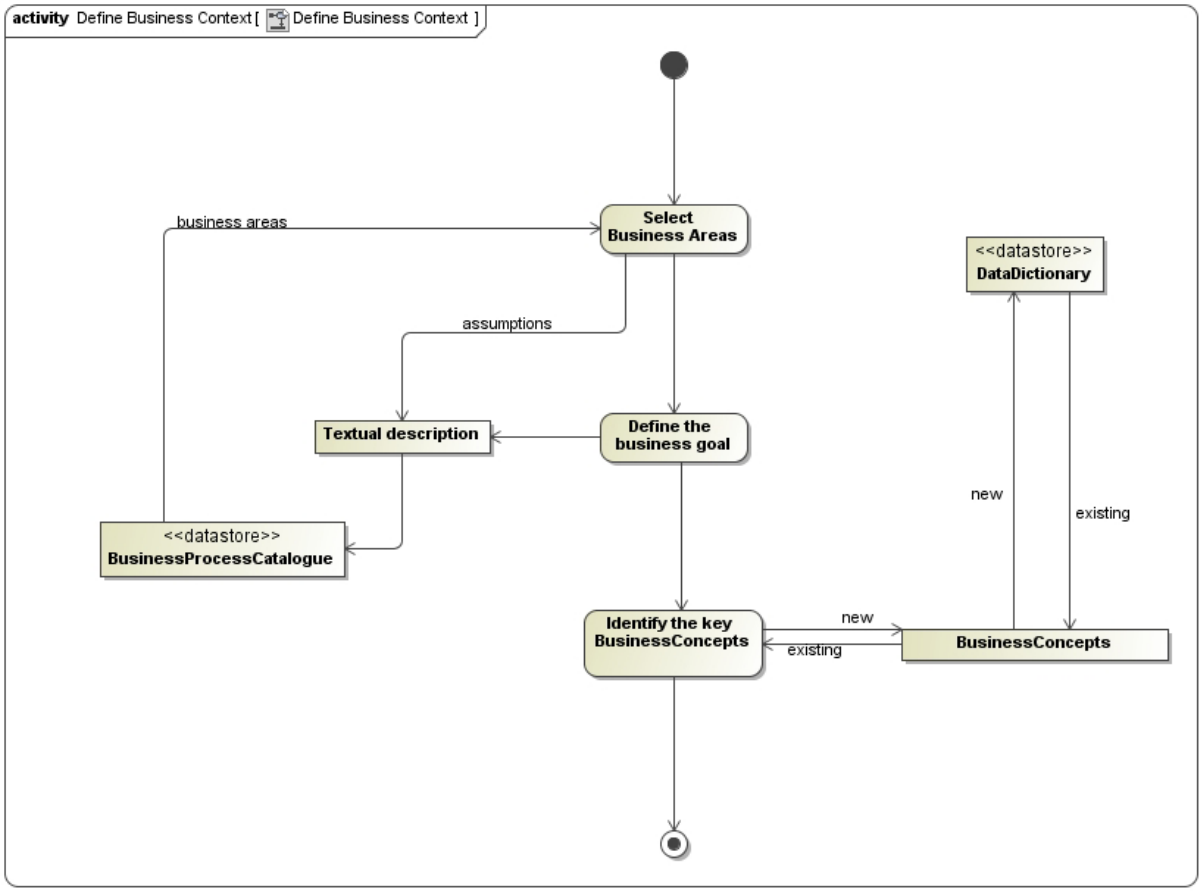

Figure 2 — Define business context

NOTE See Figure 1.

5.2.2 Define BusinessProcesses

This activity defines the following elements.

1) The organization of the BusinessProcesses, starting from a top-level node (representing the overall business goal). Refinements can be made using "include" and "extend" relations between BusinessProcesses.

2) For each BusinessProcess and each activity within a BusinessProcess:

   i) the definition, i.e. a description of the activity;

   ii) the trigger, i.e. an event that causes the start of the activity;

   iii) the pre-conditions, i.e. conditions that shall be fulfilled in order to start the activity;

   iv) the post-conditions, i.e. conditions that shall be fulfilled when the activity is completed;

   v) the arguments, i.e. information that is required, created or changed for the execution of the activity.

3) The BusinessRoles that are relevant for the defined BusinessProcess(es). A BusinessRole also applies to any BusinessProcesses that are extended or included from the owning BusinessProcess.

NOTE See Figure 1.

5.3 Guidelines

Apply a "bird's eye view". At the Scope level, it is desirable to concentrate on the business and avoid discussing the solution or even the communication problem. This means that it is assumed that there is no communication problem and each of the BusinessRoles has a perfect knowledge of and access to all information manipulated within its BusinessProcess. It should be remembered that a "good" BusinessProcess adds tangible business value.

Focus mainly on BusinessProcesses that provide a lot of added value, by eliciting BusinessComponents or BusinessRoles. It is not advisable to become immersed in details, e.g. "cancel", "modify", "create", error handling, at this stage. For example, the description of an "Interbank Transfer" BusinessProcess will elicit concepts like "Account", "Credit", etc.; the description of the "Cancel Interbank Transfer" will have no specific added value and should not be considered at this stage. These details will be elaborated during the Conceptual level and the Logical level, when BusinessTransactions and MessageSets are defined.

Concentrate on functional roles, rather than on real-life actors. For instance, it is important at this stage to identify the role "Buyer", but it is not yet important to identify whether the buyer is an individual, a corporate or a financial institution.

Depending on the useful level of detail, one may decide to decompose a BusinessProcess into more detailed BusinessProcesses.

Roles should only be associated to the most detailed BusinessProcesses, i.e. the lowest level.

Business terms will either be accompanied by a short and clear description to remove possible ambiguities or, preferably, refer to an existing glossary of business terms.

6 Conceptual level

6.1 General

6.1.1 Purpose

The purpose of the Conceptual level is to define and understand the business semantics and to define the communication requirements caused by the physical separation of the various BusinessRoles in the BusinessProcesses.

The Conceptual level identifies and specifies all communication requirements that exist within the agreed scope of BusinessProcesses and activities. It identifies who needs what kind of information, from whom, and at what moment. As such, the Conceptual level will provide the specifications for the solution (i.e. the BusinessTransactions) that will be developed, without going into the actual definition of messages.

6.1.2 Key topics

The key topics of the Conceptual level are:

— analysis of the business model in order to discover the communication requirements that arise;

— precise definition of the expected properties of the BusinessTransactions and MessageSets to be developed;

— establishing the overall architecture of the solution, i.e. whether to pursue a user-to-user solution or a centrally co-ordinated solution;

— establishing what the BusinessTransactions are (the different MessageInstances can be exchanged according to a number of message flows that need to be identified).

6.1.3 Main activities

The main activities of the Conceptual level are:

— specification of functional (behavioural) requirements of the BusinessTransactions and MessageSets to be developed;

— specification of Constraints (imposed restrictions) of the BusinessTransactions and MessageSets to be developed;

— refinement of the BusinessProcesses (expressed as use cases) into concrete BusinessTransactions (expressed as Sequence Diagrams);

— identification of the overall architecture of the solution.

6.1.4 Deliverables

The deliverables of the Conceptual level are:

— textual descriptions refining the scope and boundaries of the final solution;

— a BusinessComponent diagram depicting the information used by each of the Participants (possibly complemented with textual descriptions of some business related Rules);

— a BusinessTransaction diagram containing a description of the BusinessTransactions.

6.2 Activities

6.2.1 Define business model

Having established a physical separation among the different BusinessRoles, refine the BusinessProcesses and activities that need to be supported by the BusinessTransactions and MessageSets to be developed. This is done based on the BusinessProcess diagrams defined during Scope level.

Produce a diagram of all BusinessComponents[1)] derived from the "arguments" in the use case. It can contain inheritance, association and aggregation relations. Go through all existing BusinessComponents in the DataDictionary and add the required ones to the BusinessComponent diagram. Complete the diagram with those BusinessConcepts that have been identified and that do not yet exist in the DataDictionary. All these new artefacts will have to be submitted to the Registration Authority.

The BusinessComponent diagram can contain information regarding multiplicity and shall indicate the type (represented by a DataType or a BusinessComponent) of each Business Element.

- BusinessComponents have meaning and a context and are composed of BusinessElements and Constraints.
- DataTypes have no, or insignificant, context or semantics; they have no identity. BusinessAttributes shall be typed by DataTypes or other BusinessComponents.
- In IdentifierSets, each individual value has very little context and has an insignificant impact on the semantics of the BusinessAttribute.
- CodeSets contain considerably more semantics than IdentifierSets; each individual value (i.e. each CodeSetLiteral) qualifies the BusinessAttribute and gives it specific meaning.
- There are different relations possible between two BusinessComponents.
    - Where a BusinessComponent is entirely part of another BusinessComponent, the latter's BusinessAssociationEnd (i.e. its associationDomain) has its Property aggregation set to "COMPOSITE". Real business containment means that in real life the contained element (the part) can never exist without the container (the whole). So be careful in the BusinessComponent diagram to use aggregation only to represent real business containment.

EXAMPLE An account balance cannot exist without an account.

    - Where a BusinessComponent is part of several BusinessComponents, the latter's BusinessAssociationEnd (i.e. its associationDomain) has its Property aggregation set to "SHARED".

EXAMPLE A person has an address, but so has a financial institution.

    - Where the BusinessComponents are related and independent, both BusinessAssociationEnds have their Property aggregation set to "NONE".

EXAMPLE A party is not contained in an account and there should be therefore no aggregation relation between account and party.

NOTE See Figure 1.

1) BusinessComponents are defined as classes in the Business Model. A BusinessComponent is the exhaustive definition of a business notion. Note that BusinessComponents will never be used directly in MessageDefinitions because they are generic and do not take into account specific needs of the message context.

6.2.2 Define BusinessTransactions

6.2.2.1 Overview

Describe the way(s) in which the use cases will be realized by the Participants and what information flows are required.

Taking into account the boundaries and Participants that have just been defined, refine the way(s) in which the BusinessProcesses will be realized and what information flows are required.

The BusinessTransactions will be at least described by text and by a sequence diagram, showing the information flow between the Participants. The description(s) of the BusinessTransactions will provide the following information about the information flows between them:

— the sequence in which these flows will take place;

— exception handling;

— trigger;

— pre- and post-conditions related to each information flow.

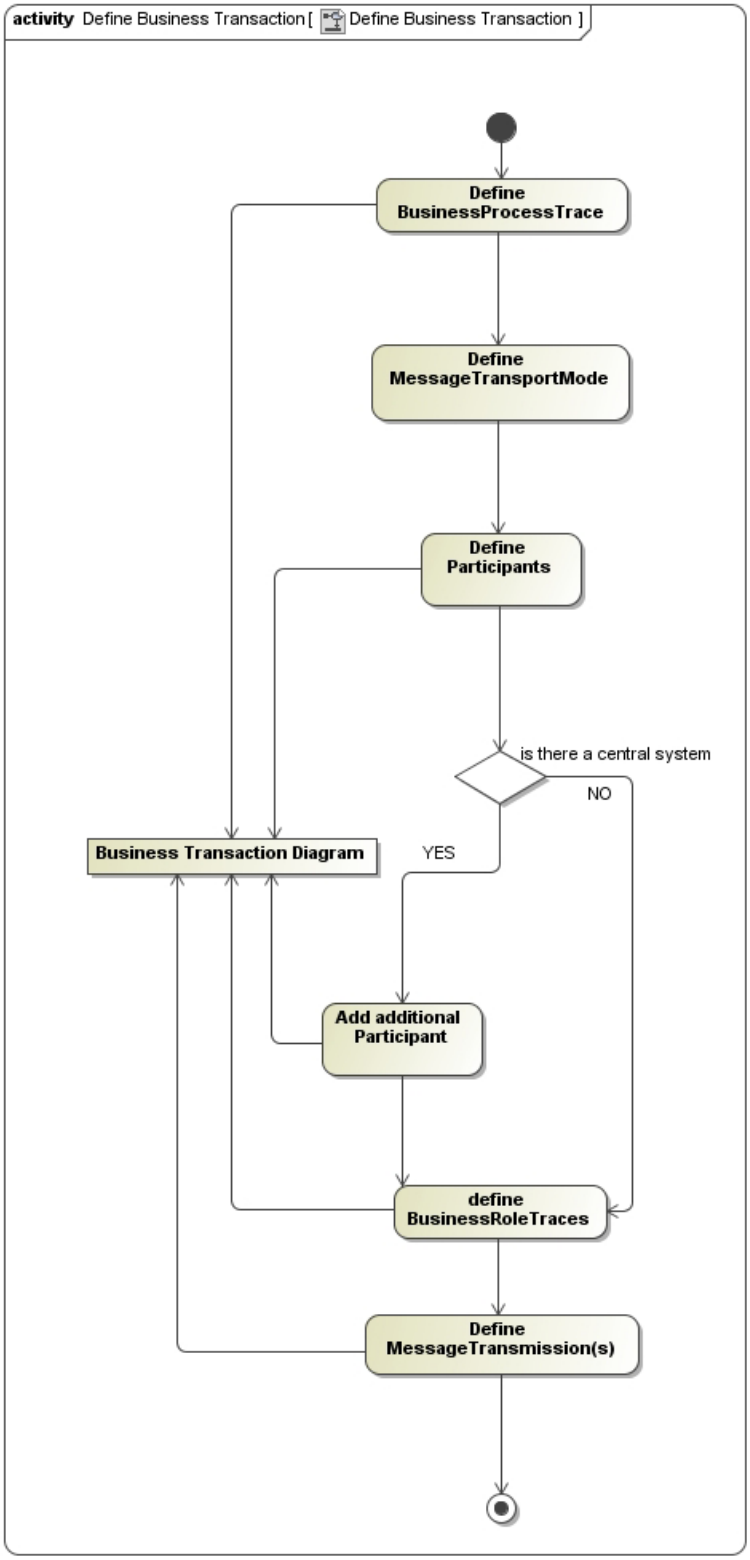

Figure 3 — Define BusinessTransaction

6.2.2.2 Define BusinessProcessTrace

Define the trace between each BusinessTransaction (which is modelled as a sequence diagram) and its corresponding BusinessProcess.

6.2.2.3 Define MessageTransportMode

For each BusinessTransaction define, where relevant, the parameters that constrain the MessageTransportMode (see ISO 20022-6):

- BoundedCommunicationDelay;
- DeliveryAssuranceDurability;
- MaximumClockVariation;
- MaximumMessageSize;
- MessageCasting;
- MessageDeliveryOrder;
- MessageDeliveryWindow;
- MessageSendingWindow;
- MessageValidationLevel;
- MessageValidationOnOrOff;
- MessageValidationresults;
- ReceiverAsynchronicity;
- SenderAsynchronicity.

6.2.2.4 Define Participants

BusinessRoles (Actors) of the BusinessProcesses (Use Cases) are refined into Participants in the derived BusinessTransactions (Activity diagrams). Each Participant is represented in the BusinessTransaction diagram as a swimlane.

6.2.2.4.1 Establish whether there is a central system

Decide whether a central system (e.g. a central Transaction Flow Monitor or market infrastructure) will be involved. If this is the case, there will be a Participant associated to each central system. Add an additional swimlane to act as the central system in the BusinessTransactionDiagram.

6.2.2.5 Define BusinessRoleTraces

Trace each Participant to the BusinessRole identified in the BusinessProcess from which the BusinessTransaction is derived.

6.2.2.6 Define MessageTransmission(s)

Now that the different actors that participate in the BusinessTransaction have been identified, the exchange of information among the Participants in order to complete the BusinessTransaction shall be shown as follows.

— For each information flow that crosses a swimlane, define a <<MessageTransmission>>-stereotyped UML Message. <<MessageDefinition>>-stereotyped Signals, identified at the Logical level, will always have to be traced to one of these <<MessageTransmissions>>.

— Repeat this activity for each identified information need.

— Define for each identified information flow the name of the MessageDefinition-to-be, the required MessageDefinition content and, if possible, the high-level MessageDefinition structure (i.e. what information should logically be put together). This information shall be documented in the sequence diagram (linked to each information flow). The "MessageDefinition content" can be identified by taking into account the information that is required and provided by the different BusinessActivities.

— The BusinessTransaction shall have constraints that tie a MessageTransmission to a specific BusinessTransaction, so a BusinessTransaction can be identified.

6.3 Guidelines

The Business Model contains a definition of what things are, i.e. their essence and business semantics. The policies, rules of modelling and preferences on the use of the model are not part of the Business Model. Policy is not captured inside the ISO 20022 Business Models. In other words, the Business Model is a model of reality, not what things should be or are in our current experience.

Business Models have the perspective of all the data in Financial Services. Message Models, however, have the perspective of all the data in a particular Message.

7 Logical level

7.1 General

7.1.1 Purpose

The purpose of the Logical level is to specify the details of the BusinessTransactions and MessageSets to be developed.

The focus is therefore on defining the MessageDefinitions that are needed to get the required information at the right time to the right Participant. The BusinessTransactions and MessageSets are still looked at from a pure business perspective, meaning that the focus still remains on the semantics (i.e. the underlying business meaning) and not yet on the syntax (i.e. how physically to represent a MessageInstance and a set of Constraints). All decisions are driven by the requirements (functional requirements and constraints) that have been identified and specified during the Conceptual level.

The Logical level is said to be "logical" because it deals with the description of the MessageSets from an abstract point of view versus a concrete, technical point of view. This abstract approach is needed to allow interoperability between different technical representations.

7.1.2 Key topics

The following items show the key topics of the Logical level.

— Establishing what the MessageDefinitions are, in terms of their business content. The exchanged information should have a business value. MessageComponentTypes/MessageElements are defined from and traced to BusinessComponents/BusinessElements that were identified during the Conceptual level.

— Establishing which rules apply to the various MessageDefinitions and defining the scope of each rule. If the scope is the MessageDefinition only, the rule refers only to the content of the MessageDefinition. If

the scope is the BusinessTransaction, the rule checks the MessageDefinition content against the overall BusinessTransaction information, e.g. the information contained in the previously exchanged MessageInstances.

7.1.3 Main activities

The main activities of the Logical level are:

— defining a MessageSet to group the MessageDefinitions;

— identifying MessageDefinitions and Constraints related to these message flows, based on the BusinessTransactions;

— designing the MessageDefinitions, i.e. identifying the business content, overall structure and organization of the MessageInstances and the Constraints that apply to the MessageDefinitions;

— formalizing the MessageDefinition contents;

— identifying suitable MessageComponentTypes;

— consolidating all of these items into a MessageDefinition Diagram.

7.1.4 Deliverables

The deliverables of the Logical level are:

— a MessageSet;

— a MessageDefinition, or MessageDefinitions;

— constraints that are written in a formal language and that complete the formalization of the MessageDefinitions.

7.2 Activities

7.2.1 Define the MessageSet

Based on the information flows that have been identified when defining the BusinessTransactions, create a MessageSet which will contain all MessageDefinitions that are needed to perform the BusinessTransactions.

NOTE See Figure 1.

7.2.2 Compose MessageDefinition

7.2.2.1 Overview

Now that the MessageSet has been defined, create for each MessageTransmission a MessageDefinition.

Combine the selected (or new) MessageComponent Types to create the final structure of the corresponding MessageDefinition in the MessageDefinition Diagram (see ISO 20022-1).

The Dictionary is updated with the new MessageComponentTypes that were created during this activity.

NOTE See Figure 1.

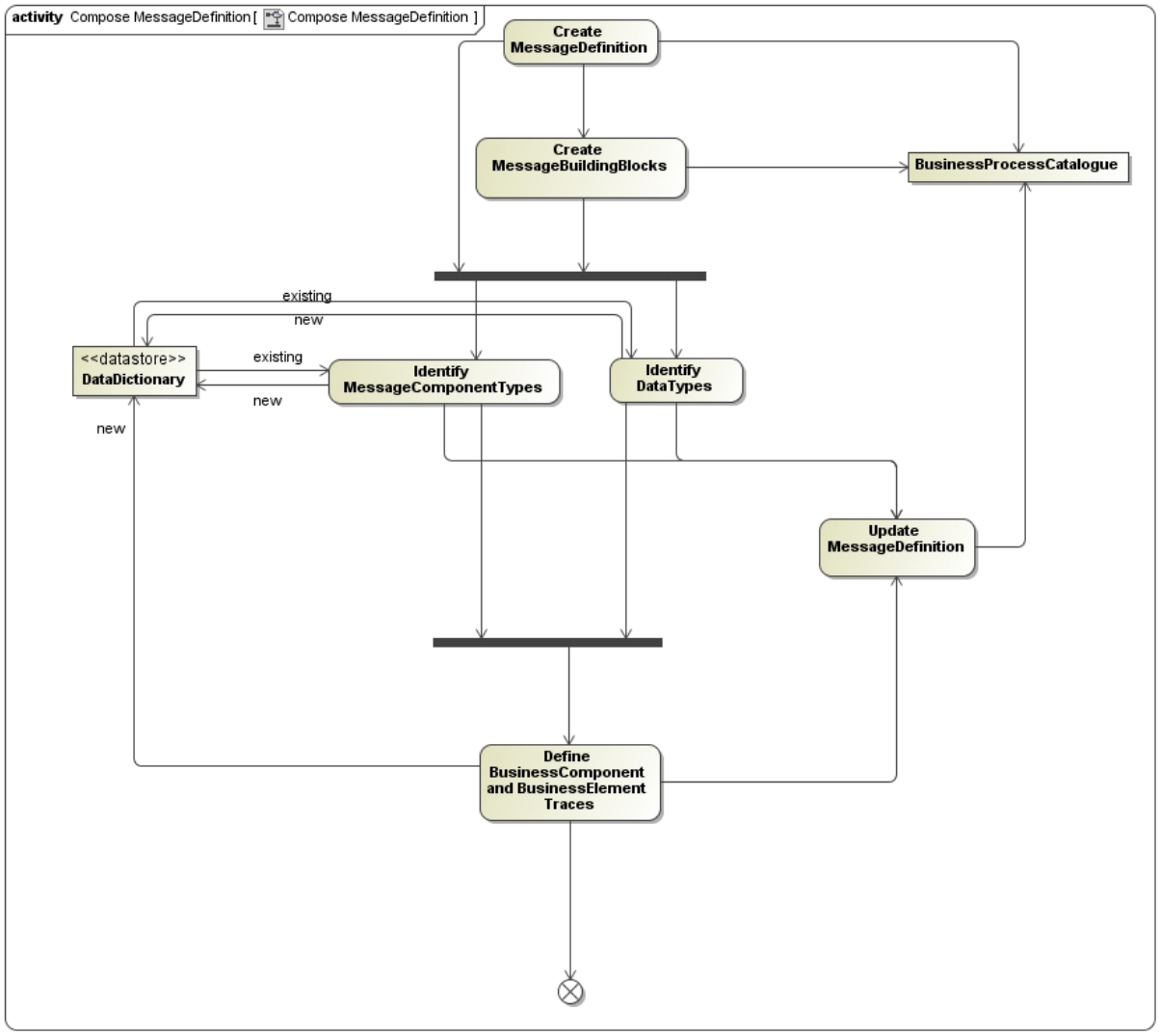

Figure 4 — Compose MessageDefinition

7.2.2.2 Create a MessageDefinition

The name of the class with stereotype MessageDefinition shall be unique within the repository.

A MessageDefinition has a unique MessageDefinitionIdentifier which has the following structure:

"xxxx.nnn.aaa.bb"

where

xxxx is an alphabetic code in four positions (fixed length) identifying the BusinessArea,

nnn is an alphanumeric code in three positions (fixed length) identifying the MessageFunctionality,

aaa is a numeric code in three positions (fixed length) identifying a particular flavour of message functionality,

bb is a numeric code in two positions (fixed length) identifying the Version [2] ,

'.' is the delimiter character between elements.

Define the MessageBuildingBlocks that are used by the MessageDefinition class.

7.2.2.3 Define MessageComponentTypes

The high level structure that was defined during the Logical level will give a first indication of what type of information should be kept together.

The first (and simplest) case is when a number of BusinessElements are needed from one single BusinessComponent. In that case, the DataDictionary can be searched for all MessageComponentTypes that are based on this BusinessComponent. If there is a MessageComponentType that contains the exact required elements with the correct multiplicity and rules, this is the obvious choice.

What if there is no MessageComponentType that exactly fits the needs?

— A MessageComponentType can be reused even if it contains too many elements (if these extra elements are acceptable).

— A MessageComponentType can be reused if it is less restrictive on multiplicity and/or rules (if a lesser validation is acceptable).

— Two or more MessageComponentTypes can be used to obtain all the required elements. Either the MessageComponentTypes can be put next to each other in the Message, or it can be proposed to the RA to create a new MessageComponentType out of the combination.

If a number of BusinessElements are needed from multiple BusinessComponents and there is no need to express relations between the different BusinessComponents, use multiple MessageComponentTypes by following the approach described above.

If a number of BusinessElements are needed from multiple BusinessComponents and there is a need to express relations between the different BusinessComponents (e.g. a link is needed between an Account with AccountOwner and AccountServicer), the DataDictionary should be searched for all MessageComponentTypes that are based on one of the identified BusinessComponents. Concentrate on those MessageComponentTypes that express already a relation between two (or more) of the BusinessComponents.

If a MessageComponentType only needs elements from one single BusinessComponent, propose a MessageComponentType that is based on this BusinessComponent and that contains exactly the MessageElements needed, with the required multiplicity and rules.

In addition to the above, a MessageComponentType might need additional elements that do not exist in any BusinessComponent and that only make sense in a specific MessageInstance, i.e. "technical" MessageElements. In this case, and if the MessageComponentType does not yet exist, propose it as described in the previous point and add the required technical MessageElements (for which no equivalent BusinessElements exist) with the required multiplicity and Constraints.

If a MessageComponentType needs elements from multiple BusinessComponents (because a relationship between the BusinessComponents shall be expressed), the following options are available.

— Use several "simple" MessageComponentTypes (i.e. a MessageComponentType based on a single BusinessComponent) and, if necessary, add rules at the MessageDefinition level to express the relationship.

2) Version numbers are incremental, starting from 1. Since the version number is exactly 2 characters, when the version number is smaller than 10 it is prefixed by a zero.

— Aggregate "simple" MessageComponentTypes into a “parent” MessageComponentType, whereby all simple MessageComponentTypes are “siblings” (e.g. the new component has three MessageComponentTypes: A, B and C). In this case, the modeller will express the fact that the different MessageComponentTypes belong together and he or she can express the necessary multiplicity information and/or rules. Submit the new MessageComponentTypes to the RA.

— Create a new MessageComponentType that contains the required MessageElements from other MessageComponents (e.g. the new MessageComponentType A1 is based on BusinessComponent A and contains an aggregation with a MessageComponentType B1 based on BusinessComponent B). In this case, it is possible either to keep an explicit aggregation or to "import" the MessageElements from the dependent MessageComponentType into the parent MessageComponentType (i.e. using attributes instead of aggregation). This last option should be used carefully as it can express less dependency (e.g. it cannot express that either all the MessageElements coming from MessageComponentType B1 or none should be present). A guideline is to use this option mainly when there's only one MessageElement coming from MessageComponentType B1 involved. The below figure shows the two described options (“aggregation” at the left and “import” at the right).

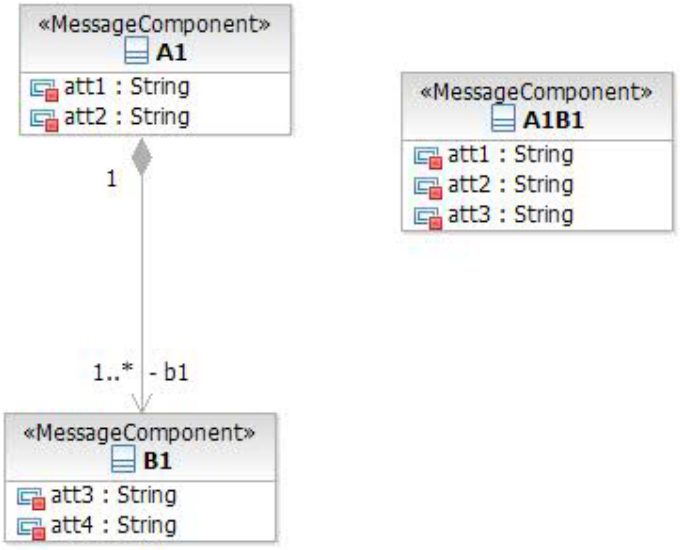

Figure 5 — Aggregation and import

7.2.2.4 Define DataTypes

ISO 20022 user-defined DataTypes are always based on a DataType representation. A DataType representation is a way to classify user-defined DataTypes and allows grouping DataTypes with a number of common characteristics. Each ISO 20022 user-defined DataType is represented as a UML Datatype and is stereotyped by one of the allowed DataType representations. A DataType representation has a number of characteristics that are passed on to (‘inherited by’) all DataTypes that are using that DataType representation. In this way, characteristics common to a number of DataTypes are grouped together.

ISO 20022 XSD DataTypes are represented as UML DataTypes as defined in the Profile::TypeLibrary.

The allowed value space of the original primitive type (e.g. string) and of the DataType representation (e.g. Amount) is defined within the Profile; each representation has its own set of constraints.

IdentifierSets are different from CodeSets; each individual value has very little context and has an insignificant impact on the semantics of the MessageAttribute. CodeSets contain semantics; each individual value (i.e. each CodeSetLiteral) qualifies the MessageAttribute and gives it specific meaning.

7.2.2.5 Mapping between the Business Layer and the Message Layer

If a MessageElement is semantically related to a BusinessElement or a BusinessComponent, then that MessageElement shall be traced to that BusinessElement/BusinessComponent.

A traceability link will be defined in the DataDictionary between a MessageElement and its related BusinessComponent/BusinessElement. A <<Trace>>-stereotyped Dependency shall be used for this traceability link; this is implemented in UML as a Dependency (represented graphically by the UML Dependency dotted line notation).

Such a MessageElement is a sort of "copy" of a BusinessElement from a particular BusinessComponent or an entire BusinessComponent, and has the following characteristics.

— If the MessageElement is typed by a user-defined DataType, then the BusinessElement shall be typed by the same DataType or a less restrictive DataType.

— If the BusinessElement is typed by a BusinessComponent and MessageElement is typed by a MessageComponentType, this MessageComponentType shall be based on the BusinessComponent that types the BusinessElement.

— If the semantic of the MessageElement is exactly the same as the semantic of the BusinessElement/BusinessComponent, the definition and name of the BusinessElement/BusinessComponent shall also be exactly the same.

— If the semantic of the MessageElement is richer or more specific than the semantic of the BusinessElement/BusinessComponent, the definition and optionally the name of the BusinessElement/BusinessComponent shall be adapted for the MessageElement.

— There can be several MessageElements defined and traced to one BusinessElement/BusinessComponent.

— Some MessageElements might be present in the MessageDefinition for "message specific" reasons (e.g. page number, certain references, etc.); i.e. they make sense only in a MessageDefinition context. These elements are said to be "technical" and they have no traceability link to any BusinessElement. Technical MessageElements have their Property 'isTechnical' set to 'TRUE'.

EXAMPLE It might be required to reference a specific instance of a BusinessElement (the date of the "last" entry instead of the entry date) or to reference calculated data. In this case, the MessageElement definition and name express the specific semantic (e.g. LastEntryDate).

It is also possible to trace an entire MessageComponentType to a BusinessComponent:

— A traceability link will be defined in the DataDictionary between a MessageComponentType and its related BusinessComponent. A Trace shall be used for this traceability link.

— There can be several MessageComponentTypes defined and traced to one BusinessComponent.

— Some MessageComponentTypes might be present in the MessageDefinition for "message specific" reasons, i.e. they make sense only in a MessageDefinition context. These components are said to be "technical" and they have no traceability link to any BusinessComponent. Technical MessageComponentTypes have their Property 'isTechnical' set to 'TRUE'.

7.3 Constraints

7.3.1 Inheritance

Inheritance in BusinessComponents shall not be duplicated in MessageComponentTypes.

7.3.2 Normalized MessageDefinitions

When a MessageElement can be computed (or arrived at) using other MessageElements, it shall be marked as a derived element by using the standard UML notation for representing derived elements: a forward slash preceding the MessageElement declaration.

EXAMPLE In a report, if all amounts and their total are shown, then the total amount is a derived MessageElement.

7.3.3 Normalization

A MessageDefinition shall not require the inclusion of any information more than once. This is termed the "Normalization Principle".

EXAMPLE In a MessageDefinition for an Invoice, the Postal Address and Billing Address should be included as Address. The Postal Address and Billing Address refer to an Address through a Message Association as, for some Invoices, Postal Address and Billing Address might be the same.

7.3.4 Non-Constraint preference

In a choice between representing a Repository Concept using a Constraint Metamodel Concept or a non-Constraint Metamodel Concept, then the most succinct expression is preferred. In cases of equal succinctness, the non-Constraint Metamodel Concept is preferred.

EXAMPLE 1 To model that only one of a set MessageElements may be used, express this as a ChoiceComponent, in preference to an exclusive or Constraint.

EXAMPLE 2 Instead of having a code "PhysicalDelivery = yes or no", an address component and a Constraint saying "If the PhysicalDelivery = Yes then the address shall be present", it is more efficient to create a component "PhysicalDeliveryAddress".

7.3.5 Sibling constraints

Every sibling constraint between the BusinessElements or their contents of a BusinessComponent, and between MessageElements or their contents of a MessageComponentType, shall be fully specified as Invariant Constraints.

EXAMPLE A Postal Address BusinessComponent requires a Post Code Business Element if a Post City BusinessElement is not present. This is expressed as an Invariant Constraint upon a Postal Address.

7.3.6 Value derivation

A derived BusinessElement or MessageElement shall have its isDerived property set to the value True. Each derived BusinessElement and MessageElement shall be given a derivation rule as a Constraint.

7.3.7 Versions versus flavours

A new version of a MessageDefinition is created when the current version is deprecated. The current version will be removed as soon as it has been removed from the last MessageSet that is using that version.

A new flavour is created (with version "01") when the current version will remain to exist independently from the new flavour.

7.3.8 Modelling associations between MessageComponent Types.

In case the Property is not set to "COMPOSITE" then the target MessageComponentType shall contain a MessageElement with type "ID".

7.4 Guidelines

7.4.1 BusinessTransactions

The BusinessTransactions shall take into account (and thus describe) the exception handling, the error handling, the acknowledgements, etc. These can result into additional Sequence Diagrams.

NOTE A BusinessTransaction describes only one possible solution for a single set of requirements. There can be multiple solutions for the same problem. In that case there would be multiple BusinessTransactions for the same requirements use case.

7.4.2 MessageInstance style

MessageInstances shall convey semantics only and not any form of style (e.g. a report style).

7.4.3 Party Neutral MessageInstance

The MessageDefinition shall be symmetrical to the BusinessRoles participating in the MessageTransmission. This is sometimes known as 'Party Neutral MessageInstances'.

EXAMPLE A Buyer and Seller of an Asset would produce the same TradeExecuted MessageDefinition.

7.4.4 MessageInstance granularity

The initial decisions given below concerning MessageInstance granularity shall be taken.

MessageDefinitions shall be precise and fit-to-specific purposes: the MessageInstance scope shall match exactly the requirements of the MessageTransmission.

Having very specific MessageDefinitions will lead to a higher number of MessageDefinitions and will therefore require some help for selecting the right MessageDefinition to use in any particular case. This can be achieved by following the same top-down approach that has been used in the methodology, leading sequentially to the identification of the relevant BusinessArea, BusinessProcess, BusinessTransaction and MessageDefinition.

The proposed approach leads to MessageDefinitions that support limited but accurate business functionality. These MessageDefinitions will have fewer optional fields. It also results in shorter MessageInstances, since a number of elements that are in a MessageDefinition to further define the message functionality can be removed.

EXAMPLE We do not need the same information when buying a security or selling a security. The returned confirmations will also be different. This should therefore result in different MessageDefinitions.

The minimal guideline is to have separate MessageDefinitions for different functionality (e.g. separate MessageDefinitions for "create", "modify", "cancel"; separate MessageDefinitions for instructions and reporting).

— With regard to "market practices", the guideline is to concentrate first on market practice independent MessageDefinitions (i.e. a MessageDefinition containing all the commonalties). Next we can focus on a specific market practice and identify the specific information and the need for more specific multiplicity, data typing and rules. Depending on the required level of differentiation, market practices might therefore result in different “flavours” of the same MessageDefinition and/or in MessageDefinitions covering the needs of multiple market practices.

— When a Participant is derived from a 'generic' BusinessRole instead of being derived of its subtype, the guideline is similar to the guideline concerning the market practices. Concentrate first on the commonalties identified in the 'generic' BusinessRole and identify afterwards the differences for each subtype in information needs, multiplicity, data typing and rules. Depending on the required level of

differentiation, this might result in different “flavours” of the same MessageDefinition and/or in MessageDefinitions covering the needs of multiple subtypes.

A useful check to verify whether it makes sense to create multiple (flavours of) MessageDefinitions versus a single one is to consider the impact of removing a particular component or element from a MessageDefinition in order to create multiple explicit MessageDefinitions. For example, if the removal implies the creation of a huge number of Messages (e.g. removing the currency from a payment MessageDefinition to create separate MessageDefinitions for payments in EUR, in GBP, in USD, etc.), it is not advisable. On the other hand, if the removal of the component or element leads to a removal or simplification of rules within the MessageDefinition, it is probably advisable. In those cases when the element that is considered to be removed contains a large number of possible values, it is not advisable to remove it since each of its values might result in a new (flavour of) MessageDefinition.

7.4.5 How to handle bi-directional relations

If a relationship between two BusinessComponents is “bi-directional”, multiple MessageComponentTypes are introduced, one for each direction of this association.

7.4.6 Factorize

Externalizing a value from a repetitive sequence is allowed if there is a business rule requiring that each occurrence has the same value.

EXAMPLE Instead of having a currency code inside a repeating sequence and a rule saying "All currency codes are identical", one can put a single currency code outside of the repeating sequence.

7.4.7 How to define an ExternalSchema

ExternalSchema is used in case the content model (i.e. the specification of the content) is defined elsewhere. Properties for the allowed namespace(s) and the level of validation shall be set.

Setting the value of processContent to "skip" is only allowed in case the sender does not want or expect the Receiver of the MessageInstance to validate/process the content at all. This value shall be used with caution since it increases the risk for low STP.

Setting the value of processContent to "strict" forces the sender to send MessageInstances that can be validated by the receiver, i.e. the receiver's validator will reject the MessageInstance if it does not have a correct Schema that describes the content.

Setting the value of processContent to "lax" is useful if some receivers do not need to process the content.

7.4.8 How to enforce canonical form for date or time related user-defined DataTypes

It is recommended to model dates (i.e. user-defined DataTypes with representation Date, Time, DateTime, Year, Month, Day, YearMonth, MonthDay) in canonical form. This means for this user-defined DataType's Property Pattern ".*Z" shall be used to enforce this.

EXAMPLE Valid values:
2000-12-20T20:00:00.000Z
2000-12-20T20:00:00Z

EXAMPLE Invalid values:
2000-12-20T20:00:00z >>> invalid dateTime because lowercase 'z' is an invalid time zone
2000-12-20T20:00:00.000-05:00 >>> valid dateTime but invalid against the pattern since it does not end with 'Z'.

8 Physical level

8.1 General

8.1.1 Purpose

The Physical level consists of a set of transformation specifications from the Logical level to a target abstract syntax (such as an ISO 20022 XML Schema, or an ISO 20022 ASN.1 structure). During this activity, the descriptions coming out of the analysis at the Logical level (of Messages, Collaborations, etc.) are mapped to the target technology.

The transformation is based on a set of mapping specifications used to automatically generate a predictable representation of the Messages and MessageSets.

This activity is said to be physical because it deals with the physical representation of the Message within the targeted technical environment.

The XML design rules can be found in ISO 20022-4. The ASN.1 transformation rules can be found in ISO 20022-8.

8.1.2 Key topics

The key topics of the Physical level are:

— Definition and packaging of the XML Schema representation or ASN.1 Modules of the MessageDefinitions.

8.1.3 Main activities

The main activities of the Physical level are:

— Generation of the specifications that contain the syntactical constraints of valid XML documents or ASN.1 instances.

8.1.4 Deliverables

The deliverables of the Physical level are:

— XML Schemas or ASN.1 Modules (One for each MessageDefinition).

— Zipped file containing the XML Schemas or ASN.1 Modules.

— XMI version of the model.

8.2 Activities

8.2.1 Create syntax scheme

The syntax schemes are created from their respective MessageDefinitions, based on the XML transformation rules specified in ISO 20022-4, or the ASN.1 transformation rules specified in ISO 20022-8.

9 Conventions

9.1 Constraints

Constraints shall be expressible as invariant constraints upon the models. Guidelines and usage policy are examples of items that are not invariant constraints.

9.2 Naming

9.2.1 General

The details of the naming conventions are described in the ISO 20022 submission templates and their related Rules of Modelling.

9.2.2 Generic rules applicable to all Repository Concepts

a) Use their natural names in the British English language.

b) Observe the (character) restrictions that are typical for most syntaxes when naming concepts:

- all names shall start with an alphabetic character;
- all characters following the first characters shall be alphabetic characters or numeric characters.

c) Apply UpperCamelCase convention:

- a name for elements and attributes may be made up of multiple words, each consisting of alphanumeric characters;
- each word starts with a capital letter;
- all white spaces between words are removed.

d) Abbreviations:

- names shall not be abbreviated.

e) Acronyms:

- acronyms that have become recognized (i.e. identified in the Oxford English Dictionary as an acronym) name terms may be used.

Annex A
(normative)

Applied UML diagrams

A.1 Correct UML Diagram to use in different cases

To model BusinessProcesses, BusinessRoles, and the relationships among them, use UseCase Diagrams.

To model BusinessTransactions and the associated Participants and MessageTransmissions, use Sequence Diagrams.

To model BusinessProcessTraces, link Sequence Diagrams to <<BusinessProcess>>-stereotyped UseCases.

To model BusinessRoleTraces, bind <<BusinessRole>>-stereotyped Actors to LifeLines in Sequence Diagrams.

To model BusinessComponents, use Class Diagrams.

To model MessageComponentTypes, use Class Diagrams.

To model BusinessComponentTraces and BusinessElementTraces, use Class Diagrams.

To model MessageDefinitions, use Class Diagrams.

To model MessageSets and BusinessAreas, use Class Diagrams

Bibliography

[1] ISO 15022 (all parts), *Securities — Scheme for messages (Data Field Dictionary)*

[2] UML (Unified Modeling Language), available on the Object Management Group website at: http://www.omg.org/technology/documents/modeling_spec_catalog.htm#UML

ICS 03.060

Price based on 24 pages