BS ISO 29481-2:2012

BSI Standards Publication

Building information models — Information delivery manual

Part 2: Interaction framework

bsi.

...making excellence a habit.™

National foreword

This British Standard is the UK implementation of ISO 29481-2:2012.

The UK participation in its preparation was entrusted to Technical Committee B/555, Construction design, modelling and data exchange.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

ISBN 978 0 580 75359 6

ICS 35.240.99; 91.010.01

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 31 March 2013.

Amendments issued since publication

Date	Text affected

BS ISO 29481-2:2012

INTERNATIONAL STANDARD

ISO 29481-2

First edition
2012-12-15

Building information models — Information delivery manual —

Part 2: Interaction framework

Modèles des informations de la construction — Contrat d'interchange —

Partie 2: Cadre d'interaction

Reference number
ISO 29481-2:2012(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 part of ISO 29481 may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

ISO 29481-2 was prepared by Technical Committee ISO/TC 59, *Buildings and civil engineering works*, Subcommittee SC 13, *Organization of information about construction works*.

ISO 29481 consists of the following parts, under the general title *Building information models — Information delivery manual*:

— *Part 1: Methodology and format*

— *Part 2: Interaction framework*

The following parts are under preparation:

— *Part 3: Model view definitions*

Introduction

Building information modelling provides a concept for describing and displaying information required in the design, construction, and operation of constructed facilities. It can bring together the diverse sets of information used in construction into a common information environment — reducing, and often eliminating, the need for the many types of paper documentation currently in use.

An information delivery manual (IDM) provides significant help in getting the full benefit from a building construction information model (BIM). If the information required is available when it is needed and the quality of information is satisfactory, the construction process itself will be greatly improved. For this to happen, there should be a common understanding of the building processes and of the information that is needed for and results from their execution.

This part of ISO 29481 focuses on aspects of the construction process that refer to management and coordination of the involved parties. Coordination is dependent on communication, which should be well structured, unambiguous, explicit, and prompt. Due to a sharp focus on coordination and interaction, this part of ISO 29481 provides a natural complement to standards that focus on building modelling like ISO 10303-239 and ISO 16739.

This part of ISO 29481 sets out a methodology and format for describing coordination acts between actors in a construction project. It describes how to identify and define the coordination processes undertaken and the information required for their execution. The resulting interaction frameworks enable standardization of interaction in building processes on national, local, and project level. It also gives a format to support solutions provided by ICT-solution providers. Support of this part of ISO 29481 in different ICT-solutions means that this joins together different process management systems. In doing so, it provides a basis for reliable information exchange/sharing for users, so that they can be confident that the information they are sending or receiving is accurate and sufficient for the coordination activities they need to perform.

The development of this part of ISO 29481 has been driven by the need of users for reliability in information exchange. It is mainly based on the Dutch VISI standard developed in 2003.

INTERNATIONAL STANDARD **ISO 29481-2:2012(E)**

Building information models — Information delivery manual —

Part 2: Interaction framework

1 Scope

This part of ISO 29481 specifies a methodology and format for describing 'coordination acts' between actors in a building construction project during all life cycle stages.

It therefore specifies

— a methodology that describes an interaction framework,

— an appropriate way to map responsibilities and interactions that provides a process context for information flow,

— a format in which the interaction framework should be specified.

This part of ISO 29481 is intended to facilitate interoperability between software applications used in the construction process, to promote digital collaboration between actors in the building construction process, and to provide a basis for accurate, reliable, repeatable, and high-quality information exchange.

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 29481-1, *Building information modelling — Information delivery manual — Part 1: Methodology and format*

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

3.1
IDM
Information Delivery Manual
documentation which captures the business process and gives detailed specifications of the information that a user fulfilling a particular role would need to provide at a particular point within a project

3.2
interaction framework
formal description of the elements of interaction, including definition of roles, transactions, messages in transaction, and data elements in messages

3.3
interaction framework schema
formal description of the rules with which an interaction framework must comply

3.4
interaction schema
formal description of the rules with which sent and received messages must comply

3.5
promotor
algorithm that generates an interaction schema from an interaction framework, interaction framework schema, and templates file as input

3.6
templates file
file containing a number of templates, independent of the interaction framework, for generating an interaction schema

3.7
VISI
acronym for Dutch standard for communication between partners in construction projects

Note 1 to entry: VISI stands for "Voorwaarden scheppen voor Invoeren Standaardisatie ICT in de Infrastructuur-sector" which translates as "Creating conditions for the implementation of ICT standardization for the construction industry")

4 Standard principles

4.1 General

This clause is included to highlight and help explain essential concepts on which this part of ISO 29481 is based.

4.2 BIM and IDM

Building information modelling brings together the diverse sets of information used in construction into a common information environment. For this to happen, there should be a common understanding of the building processes and the information that is needed for and from their execution.

ISO 29481 is a standard that sets out a method for the development of an Information Delivery Manual.

The IDM methodology given in ISO 29481-1 shall be used for all references to development and use of IDM.

4.3 Components of IDM

The methodology and components of IDM are described in ISO 29481-1. In that part, an illustration is given that diagrammatically shows what the different components of IDM are and how they are related.

Within IDM, there are two perspectives. These are seen as user requirements and technical solutions. Within the two perspectives, there are a number of zones that characterize the various components of IDM (see Figure 1).

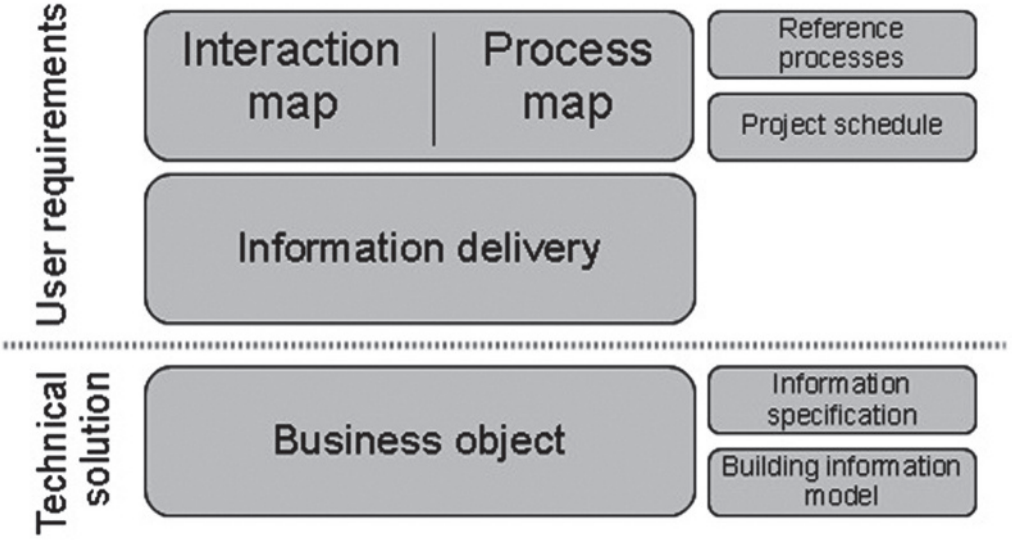

Figure 1 — IDM zones

Within the user-requirements perspective, these zones are

— interaction maps, describing the roles and interactions between them,

— process maps, describing the overall process in which information exchange occurs,

— information delivery, describing the information exchange needs,

— reference processes (stored exchange descriptions),

— the project schedule (occurrences of processes in the context of a project).

The technical-solution perspective includes

— the business objects comprising the exchange requirement model,

— the information specification, describing the schema on which the information exchange is based,

— the building information model.

This part of ISO 29481 focuses on the interaction map and is based on general principles of business communication.

4.4 Basic principles of business communication

Once a client or customer has asked to deliver a product or provide a service, there will be a chain of activities in operation, whose combined effect is to provide the product or service. Such a chain of activities is called a business process. More specifically, we speak here of a primary business process because it is initiated externally.

Part of the business process is the communication between the involved parties. This part of ISO 29481 concentrates on the communication that relates to the delivery of an outcome (performative communication). The initiation and execution of a request is through communicative actions. In a communicative action, two parties are always involved: the person who performed the action and the person to whom the action is directed. The handling of a request appears to occur in a particular pattern called the transaction.

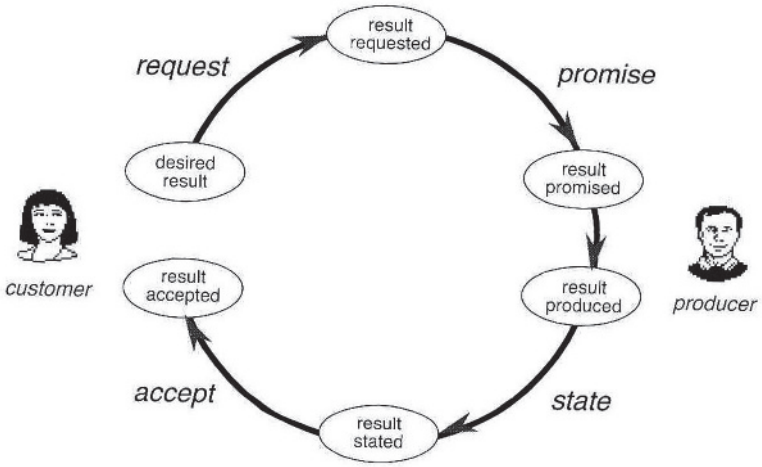

Figure 2 — A transaction pattern (Dietz, 2006)

In Figure 2, the simplest form of this transaction pattern is presented. It shows that bringing about of a new production result (for example, the 'desired result' is the delivery of a document) starts with requesting of this result by someone in the role of customer from someone in the role of producer. This brings the process to the state "result requested". The producer responds to the request by promising to produce the desired result, which brings the process to the state "result promised". This presents a to-do item for the producer: he has to comply with the promise by actually preparing the document and deciding to deliver the document. In the act of handing over the document to the customer, he states that he has complied with his promise. The customer responds to this state by accepting the result as produced. This act completes the transaction.

In the execution of the business process, often many actors are involved. Their behaviour is dependent on their role in the process. Roles/actors do business with other roles/actors by executing transactions. A useful representation of the interaction between roles/actors is called the interaction map.

4.5 Interaction map

An interaction map shall identify the relevant role types and transaction types for a certain process. IDM draws a distinction between a role that makes a request, the initiator, and the role that gives effect to that request, the executor. A transaction shall have only one initiating role and only one executing role. Figure 3 shows the components of the interaction map.

NOTE The notation of the interaction map is based on the construction model as described in the publication of Prof. Jan L.G. Dietz. This notation differs from BPMN and is used to prepare maps that are as simple as possible. Also, it provides the concept of 'transaction', which is not available in BPMN.

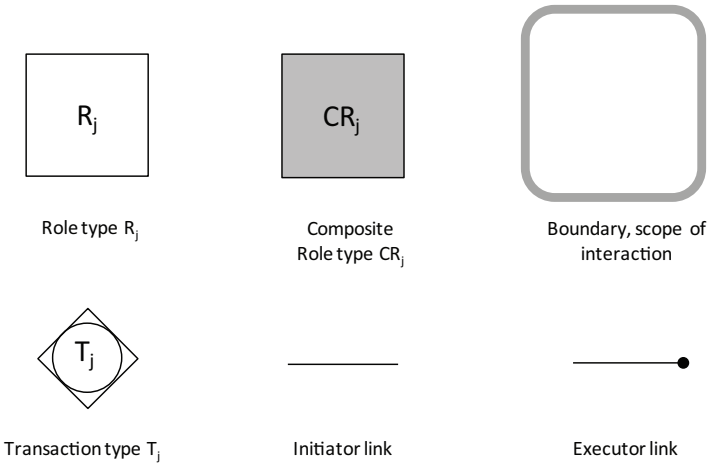

Figure 3 — Components of the interaction map

The advantage of the interaction map is that it focuses attention on the interfaces between roles while hiding the complexity of the process within the domain of the roles and hiding the details of the interaction between the roles. The use of abstract roles makes the interaction map valid for many different situations. The interaction map is a valuable tool for analysing and defining essential elements of a business process. Figure 4 shows a simplified example of an interaction map of a design office.

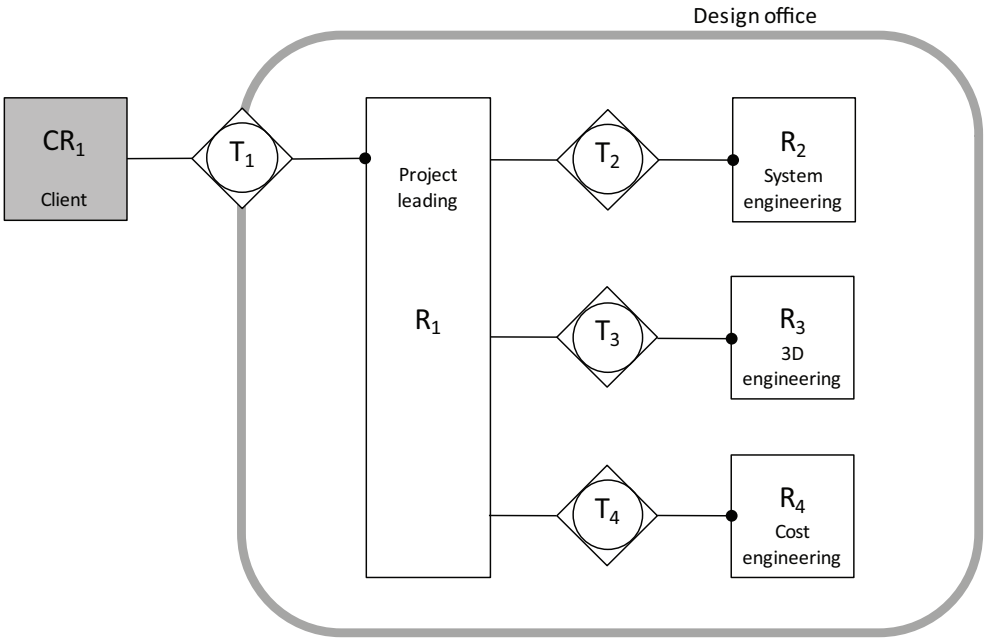

Figure 4 — Example of an interaction map

In an interaction map, all transactions needed for the handling of required contributions of relevant roles to the BIM shall be included. All roles and transactions within the interaction map shall have a unique identity and name. The numbering is arbitrary. The name of the role is derived from the main activity undertaken by the role; this brings focus on the contribution of the role to the BIM. A composite role is a role which may consist of multiple roles but whose composition is unknown or not relevant.

The interactions can be summarized in a transaction table.

Table 1 — Transaction table of a simplified design office

Transaction result	Transaction type
Design is delivered	T_1, Deliver design
System specification is delivered	T_2, Deliver system specification
3D model is delivered	T_3, Deliver 3D model
Cost calculation is delivered	T_4, Deliver cost calculation

4.6 Messages in transaction

A transaction shall contain a set of messages that are exchanged for a particular purpose. The transaction also stipulates the participating roles, point in the life cycle, and the sequence in which messages should be delivered (if appropriate).

An example of a transaction is the handling of a request for a 3D model. See Figure 5, which shows the messages in a transaction as a sequence diagram in UML notation. The transaction can only be initiated by R1 Project leading with the message 'Request for 3D model'. The 3D engineer (role R3) can respond with a message 'Work done and request for approval'. After a message 'Work approved' or 'Work not approved', the transaction is completed.

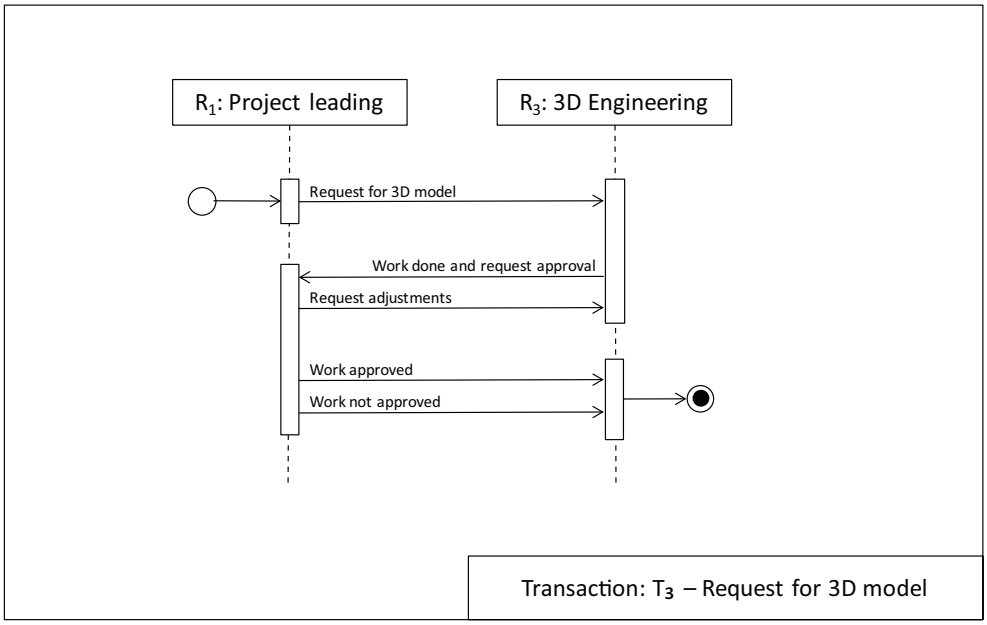

Figure 5 — Example of messages in a transaction

A message is a populated information model and contains data. Attachments may be linked to messages. As an attachment, an exchange requirement can be transferred to the executing role, and the result (contribution to the BIM) is delivered to the initiating role. By using transactions, the information transfer is brought in a process context.

4.7 Interaction framework

In order to give guidance to a process and information transfer, the elements of interaction need to be described in a coherent manner. This coherent description is called an interaction framework. An interaction framework shall include

— definition of relevant roles,

— transactions,

— messages in transaction,

— the order of messages in transaction,

— data elements in messages.

An interaction framework can be prepared for a defined application area and used as a standard on (inter-)national level, organization level, or project level. For example, in the Netherlands, an interaction framework is developed at the national level for the completion of all contractual procedures during the execution of a construction project. This part of ISO 29481 is used as a template by organizations and projects and adjusted to specific needs.

EXAMPLE An interaction framework may include the attribute CostEstimation as an instance of SimpleElementType to be used as a mandatory element for a certain message. It also may include a restriction on the format of the attribute CostEstimation (e.g. only euros with two decimals).

4.8 Supporting the software solutions

4.8.1 Overview

The next step is to support the interaction framework with software solutions. The aim is

— to support the editing of an interaction framework,

— to guarantee the completeness and validity of an interaction framework,

— to support the portability of an interaction framework,

— to support the operation of information systems,

— to support the interoperability of communication.

In the support of software solutions, two levels can be identified. The first level concerns the interaction framework. The second level concerns the actual communication which is based on the interaction framework. This part of ISO 29481 applies to both levels.

An overview on how the software solutions are supported is given in Figure 6. The following sections provide an explanation.

4.8.2 Supporting the interaction framework

In order to support the portability of an interaction framework, it should be clear with which rules an interaction framework must comply. These rules shall be included in an interaction framework schema, which is recorded as an XSD schema file. An interaction framework comprises instances of classes defined in the schema and shall be recorded as an XML file.

EXAMPLE The interaction framework schema defines that you may include the definition of attributes (SimpleElementType) and restrictions to attributes (UserdefinedType) in an interaction framework.

Chapter 5 describes the interaction framework schema and the available classes.

Every interaction framework should comply with the interaction framework schema.

An interaction framework editor should use the interaction framework schema to validate the produced frameworks.

4.8.3 Promotor

Once a valid interaction framework is available, it can be interpreted by a suitable information system. Then this system can support the communications in accordance with the options set out in the interaction framework. Finally, it is desirable to be able to validate the received and sent messages; this is done with the interaction schema.

The interaction schema is generated with a generic algorithm called Promotor. The Promotor 'promotes' XML instances tot XSD classes. The input is

— interaction framework (XML),

— interaction framework schema (XSD),

— templates file (XSD), containing a number of templates not described in an interaction framework but are valid for every interaction schema, for example the message header.

The output is an interaction schema recorded as an XSD file.

EXAMPLE The 'Promotor' takes information from the interaction framework to include the attribute CostEstimation to be used as a mandatory element for a certain message and creates an interaction schema which defines the message with the CostEstimation attribute.

Annex B describes the templates XSD file.

Annex D gives the principles of the Promotor.

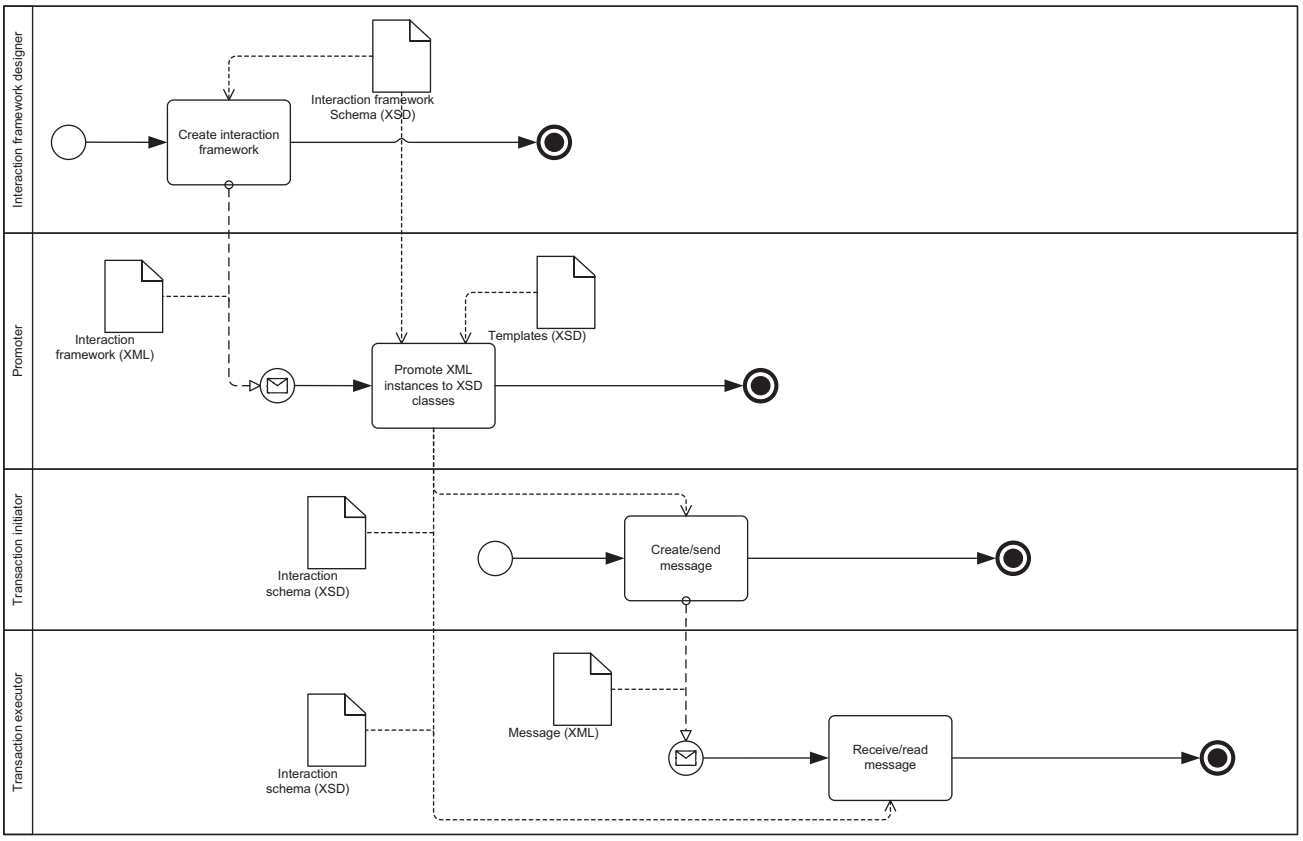

Figure 6 — How software solutions are supported

4.8.4 Supporting communication

Every information system that participates in the communication, as defined in the interaction framework, should operate on the basis of the corresponding interaction framework and interaction schema. Every message that is sent or received should be valid according to the interaction schema.

4.8.5 Technical implementation of communication

In order to ensure that messages with attachments in technical sense can be exchanged between information systems, there is a need for implementation guidelines. Topics to be covered include

— communication protocol,

— communication architecture/servers,

— encryption,

— SOAP function calls.

Implementation guidelines are beyond the scope of this part of ISO 29481.

5 Format of an interaction framework

5.1 Introduction

Clause 4 explains that, in order to support software solutions, every interaction framework must comply with the interaction framework schema. This clause is included to define the format of an interaction framework through a description of the interaction framework schema.

5.2 provides an overview of the information classes that can occur in an interaction framework and are defined in the interaction framework schema. Since an interaction framework is defined in XML, the word 'type' is used rather than class. Annex A gives the full description of the interaction framework schema. Annex C provides an example of an instance of an interaction framework.

5.2 Information types in the interaction framework schema

5.2.1 Introduction

A schema is populated by many classes or types. In this section, a short description is given of the available types in the interaction framework schema. Annex A contains a full description in XML of the interaction framework schema. An interaction framework shall be created from instances of these types and shall have a header which points to the schema with the defined available types.

5.2.2 AppendixType

The AppendixType is a definition to define the structure of elements regarding metadata. An instance of an AppendixType is used to define certain types of files or documents which can be part of the send/received messages. The structure of the elements related to an instance of an AppendixType represents the specific metadata which is required for a certain type of file or document.

5.2.3 ComplexElementType

A ComplexElementType is a collection of SimpleElementTypes. Every SimpleElementType occurs exactly the number of times it is related to.

5.2.4 ElementCondition

An instance of an ElementCondition describes the behaviour of element values in successive messages. For example, when an instance of the type ElementCondition is created with the value FIXED, it indicates that elements in successive messages must be copied when the same element is available and the value cannot be changed. An ElementCondition can refer to different levels in a framework. It can be directly related to a SimpleElement but it is also possible to relate an ElementCondition to a ComplexElement or a MessageInTransactionType. In this case, the ElementCondition is valid for all elements which are part of the element structure/collection of the related types.

5.2.5 GroupType

A GroupType makes it possible to create multiple instances of a group with their own specific content. The GroupType can be used to categorize messages within a transaction or documents related to messages within a transaction.

5.2.6 MessageType

The MessageType is used to define the content of a message. The elements that are part of a message are, in turn, grouped into one or more instances of a ComplexElementType.

5.2.7 MessageInTransactionType

The MessageInTransactionType (MITT) is a definition which is used to relate MessageType's to TransactionType's. Or simply said: the same message type may occur more than once in a given transaction type and vice versa. It is possible to relate more of the same instance of a MessageType to one instance of a TransactionType and one instance of a MessageType to multiple instances of a TransactionType. Besides, a MITT offers an option to reverse the direction from executor to initiator. Another option controls if the message flow is blocked by open secondary transactions or not.

5.2.8 OrganisationType

Definition of a certain group of organizations. In common, at least one instance is available in the framework with the particular reason to define the structure of elements of an organization.

5.2.9 PersonType

Definition of a type of person. In common, at least one instance is available in the framework with the particular reason to define the structure of elements to define a person. The PersonType can be used to categorize groups of persons who are related to a role.

5.2.10 ProjectType

Definition of a type of project. In common, at least one instance is available in the framework with the particular reason to define the structure of elements to define the project.

5.2.11 RoleType

Definition of a role. Instances of a RoleType are prerequisites to create a TransactionType in a framework.

5.2.12 SimpleElementType

The SimpleElementType is a definition which describes properties which can occur within object structures. The relation to an object is always via a ComplexElementType.

5.2.13 TransactionPhaseType

Definition which can be used to define an instance which describes the phase a transaction is in. For example, an instance of a TransactionPhaseType can be "result requested" or "on hold".

5.2.14 TransactionType

Definition of a transaction. In an instance of a Transaction, the roles of the initiator and executor are defined.

5.2.15 UserDefinedType

The UserDefinedType is used to set data types (for example, a string) and XSD restrictions. For example, with an instance of a UserDefinedType, the minimal length of a string can be defined.

Figure 7 visualizes the model of the interaction framework schema, including all its references.

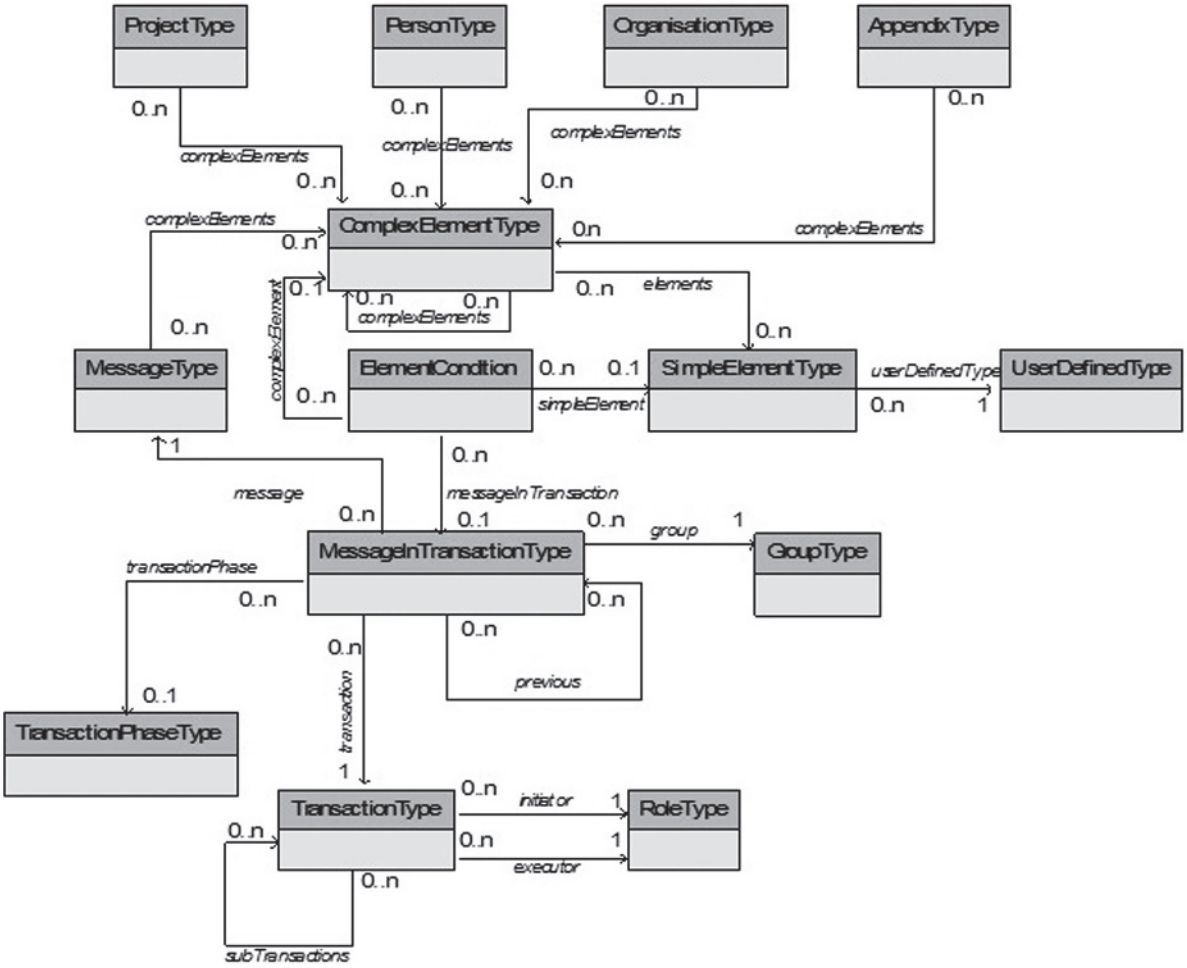

Figure 7 — Types and references of the interaction framework schema

Annex A
(normative)

Interaction framework schema definition

A.1 Introduction

An interaction framework must comply with the rules that are included in an interaction framework schema. In this annex, the interaction framework schema definition is given in EXPRESS format and in XSD format. Also, descriptions are given of Element types, attributes, elements, and references.

All objects in an interaction framework require a start date and end date. This requirement is to enable time constraints to the validity of a certain object.

A.2 Interaction framework schema definition (EXPRESS format)

SCHEMA ISO 29481_Part_2A;

ENTITY ProjectType; — The definition of a specific group of projects. Generally, only one instance will be present in an interaction framework defining the structure of elements that each instance of project should expose.

namespace: STRING;

description: STRING;

startDate: DATETIME;

endDate: DATETIME;

state: STRING;

dateLaMu: DATETIME;

userLaMu: STRING;

language: OPTIONAL STRING;

category: OPTIONAL STRING;

helpInfo: OPTIONAL STRING;

code: OPTIONAL STRING;

complexElements: OPTIONAL SET [0:?] OF ComplexElementType;

END_ENTITY;

ENTITY PersonType; — The definition of a specific group of persons. Generally, only one instance will be present in an interaction framework defining the structure of elements that each instance of person should expose.

description: STRING;

startDate: DATETIME;

endDate: DATETIME;

state: STRING;

dateLaMu: DATETIME;

userLaMu: STRING;

language: OPTIONAL STRING;

category: OPTIONAL STRING;

helpInfo: OPTIONAL STRING;

code: OPTIONAL STRING;

complexElements: OPTIONAL SET [0:?] OF ComplexElementType;

END_ENTITY;

ENTITY OrganisationType; — The definition of a specific group of organizations. Generally, only one instance will be present in an interaction framework defining the structure of elements that each instance of organization should expose.

description: STRING;

startDate: DATETIME;

endDate: DATETIME;

state: STRING;

dateLaMu: DATETIME;

userLaMu: STRING;

language: OPTIONAL STRING;

category: OPTIONAL STRING;

helpInfo: OPTIONAL STRING;

code: OPTIONAL STRING;

complexElements: OPTIONAL SET [0:?] OF ComplexElementType;

END_ENTITY;

ENTITY AppendixType; — An AppendixType contains the definition of an appendix. Which data items should be recorded with an appendix can be specified in the complex element section.

description: STRING;

startDate: DATETIME;

endDate: DATETIME;

state: STRING;

dateLaMu: DATETIME;

userLaMu: STRING;

language: OPTIONAL STRING;

category: OPTIONAL STRING;

helpInfo: OPTIONAL STRING;

code: OPTIONAL STRING;

complexElements: OPTIONAL SET [0:?] OF ComplexElementType;

END_ENTITY;

ENTITY ComplexElementType; — A ComplexElementType contains a set of SimpleElementTypes. Each stated SimpleElementType occurs exactly the number of times it is mentioned.

description: STRING;

startDate: DATETIME;

endDate: DATETIME;

state: STRING;

dateLaMu: DATETIME;

userLaMu: STRING;

language: OPTIONAL STRING;

category: OPTIONAL STRING;

helpInfo: OPTIONAL STRING;

complexElements: OPTIONAL SET [0:?] OF ComplexElementType;

simpleElements: OPTIONAL SET [0:?] OF SimpleElementType;

END_ENTITY;

ENTITY MessageType; — The definition of a type of message (MessageType), specifying the structure of the message and which set of SimpleElementType's (via ComplexElementType's) may accompany.

description: STRING;

startDate: DATETIME;

endDate: DATETIME;

state: STRING;

dateLaMu: DATETIME;

userLaMu: STRING;

language: OPTIONAL STRING;

category: OPTIONAL STRING;

helpInfo: OPTIONAL STRING;

code: OPTIONAL STRING;

complexElements: OPTIONAL SET [0:?] OF ComplexElementType;

appendixTypes: OPTIONAL SET [1:?] OF AppendixType;

END_ENTITY;

ENTITY ElementCondition; — The condition of a SimpleElementType as used within a specific MessageType.

description: STRING;

condition: STRING;

helpInfo: OPTIONAL STRING;

complexElement: OPTIONAL ComplexElementType;

simpleElement: OPTIONAL SimpleElementType;

messageInTransaction: OPTIONAL MessageInTransactionType;

END_ENTITY;

ENTITY SimpleElementType; — The definition of a simple element type (SimpleElementType). This element type specifies a property which may occur within various object structures, for example in MessageType (see also AppendixType, ProjectType, PersonType, and OrganisationType). A SimpleElementType is always embedded in a ComplexElementType.

description: STRING;

interfaceType: STRING;

state: STRING;

dateLaMu: DATETIME;

userLaMu: STRING;

language: OPTIONAL STRING;

category: OPTIONAL STRING;

helpInfo: OPTIONAL STRING;

valueList: OPTIONAL STRING;

userDefinedType: UserDefinedType;

END_ENTITY;

ENTITY UserDefinedType; — A specification of a data type (to be used in a SimpleElementType). This data type encapsulates fill-in areas in the final message, for example a Dutch zip code starts always with four digits and then two characters.

description: STRING;

state: STRING;

dateLaMu: DATETIME;

userLaMu: STRING;

baseType: STRING;

xsdRestriction: OPTIONAL STRING;

language: OPTIONAL STRING;

helpInfo: OPTIONAL STRING;

END_ENTITY;

ENTITY MessageInTransactionType; — The occurrence of a MessageType within a TransactionType related to a certain group type (GroupType).

requiredNotify: INTEGER;

dateLaMu: DATETIME;

userLaMu: STRING;

received: BOOLEAN;

send: BOOLEAN;

state: STRING;

initiatorToExecutor: OPTIONAL BOOLEAN;

openSecondaryTransactionsAllowed: OPTIONAL BOOLEAN;

firstMessage: OPTIONAL BOOLEAN;

message: MessageType;

previous: OPTIONAL SET [0:?] OF MessageInTransactionType;

transaction: TransactionType;

transactionPhase: OPTIONAL TransactionPhaseType;

group: GroupType;

appendixTypes: OPTIONAL SET [1:?] OF AppendixType;

END_ENTITY;

ENTITY TransactionPhaseType; — The definition of a phase related to a transaction. Examples are 'assignment accepted' or 'result part received'.

description: STRING;

startDate: DATETIME;

endDate: DATETIME;

state: STRING;

dateLaMu: DATETIME;

userLaMu: STRING;

language: OPTIONAL STRING;

category: OPTIONAL STRING;

helpInfo: OPTIONAL STRING;

code: OPTIONAL STRING;

END_ENTITY;

ENTITY TransactionType; — The definition of a type of transaction. A transaction type may reference other transaction types. A transaction will be initiated by a person belonging to an organization fulfilling a certain role. At this level, the role type of the initiator should be stated (the promoted schema will enforce this). The same holds for the executor.

description: STRING;

startDate: DATETIME;

endDate: DATETIME;

state: STRING;

dateLaMu: DATETIME;

userLaMu: STRING;

language: OPTIONAL STRING;

category: OPTIONAL STRING;

helpInfo: OPTIONAL STRING;

code: OPTIONAL STRING;

result: OPTIONAL STRING;

subTransactions: OPTIONAL SET [1:?] OF TransactionType;

initiator: RoleType;

executor: RoleType;

appendixTypes: OPTIONAL SET [1:?] OF AppendixType;

END_ENTITY;

ENTITY RoleType; — The definition of a specific role type.

description: STRING;

startDate: DATETIME;

endDate: DATETIME;

state: STRING;

dateLaMu: DATETIME;

userLaMu: STRING;

language: OPTIONAL STRING;

category: OPTIONAL STRING;

helpInfo: OPTIONAL STRING;

code: OPTIONAL STRING;

responsibilityScope: OPTIONAL STRING;

responsibilityTask: OPTIONAL STRING;

responsibilitySupportTask: OPTIONAL STRING;

responsibilityFeedback: OPTIONAL STRING;

END_ENTITY;

ENTITY GroupType; — The definition of a group to store appendices sent with a message within a transaction.

description: STRING;

startDate: DATETIME;

endDate: DATETIME;

state: STRING;

dateLaMu: DATETIME;

userLaMu: STRING;

language: OPTIONAL STRING;

category: OPTIONAL STRING;

helpInfo: OPTIONAL STRING;

END_ENTITY;

END_SCHEMA;

A.3 Interaction framework schema definition (XSD format)

```
<?xml version="1.0" encoding = "UTF-8"? >
<xsd:schema targetNamespace="http://www.ISO 29481_Part_2A.com/schemas" xmlns:iso29481p2a = "http://
www.ISO 29481_Part_2A.com/schemas" xmlns:xsd = "http://www.w3.org/2001/XMLSchema" elementFormDe-
fault = "qualified" >
        <!- root element declaration (for SCHEMA definitions) →
        <xsd:element name="ISO 29481_Part_2A" >
                <xsd:complexType>
                        <xsd:choice maxOccurs="unbounded" >
                                <xsd:element ref="iso29481p2a:AppendixType"/ >
                                <xsd:element ref="iso29481p2a:ComplexElementType"/ >
                                <xsd:element ref="iso29481p2a:ElementCondition"/ >
                                <xsd:element ref="iso29481p2a:GroupType"/ >
                                <xsd:element ref="iso29481p2a:MessageInTransactionType"/ >
                                <xsd:element ref="iso29481p2a:MessageType"/ >
                                <xsd:element ref="iso29481p2a:OrganisationType"/ >
                                <xsd:element ref="iso29481p2a:PersonType"/ >
                                <xsd:element ref="iso29481p2a:ProjectType"/ >
                                <xsd:element ref="iso29481p2a:RoleType"/ >
                                <xsd:element ref="iso29481p2a:SimpleElementType"/ >
                                <xsd:element ref="iso29481p2a:TransactionPhaseType"/ >
                                <xsd:element ref="iso29481p2a:TransactionType"/ >
                                <xsd:element ref="iso29481p2a:UserDefinedType"/ >
                        </xsd:choice>
                </xsd:complexType>
        </xsd:element>
        <!- element declarations (for ENTITY definitions) →
        <xsd:element name="AppendixType" type = "iso29481p2a:AppendixTypeType" >
                <xsd:annotation>
                        <xsd:documentation>An AppendixType contains the definition of an appendix. Which
data items should be recorded with an appendix can be specified in the complex element section.</
xsd:documentation>
                </xsd:annotation>
        </xsd:element>
        <xsd:element name="ComplexElementType" type = "iso29481p2a:ComplexElementTypeType" >
                <xsd:annotation>
                        <xsd:documentation>A ComplexElementType contains a set of SimpleElementTypes.
Each stated SimpleElementType occurs exactly the number of times mentioned.</xsd:documentation>
                </xsd:annotation>
        </xsd:element>
        <xsd:element name="ElementCondition" type = "iso29481p2a:ElementConditionType" >
                <xsd:annotation>
                        <xsd:documentation>The condition of a SimpleElementType as used within a
specific MessageType.</xsd:documentation>
                </xsd:annotation>
        </xsd:element>
        <xsd:element name="GroupType" type = "iso29481p2a:GroupTypeType" >
                <xsd:annotation>
                        <xsd:documentation>The definition of a group to store appendices sent with a
message within a transaction.</xsd:documentation>
```



```
                </xsd:annotation>
        </xsd:element>
        <xsd:element name=”MessageInTransactionType” type = ”iso29481p2a:MessageInTransactionTypeT
ype” >
                <xsd:annotation>
                        <xsd:documentation>The occurrence of a MessageType within a TransactionType
related to a certain group type (GroupType).</xsd:documentation>
                </xsd:annotation>
        </xsd:element>
        <xsd:element name=”MessageType” type = ”iso29481p2a:MessageTypeType” >
                <xsd:annotation>
                        <xsd:documentation>The definition of a type of message (MessageType), specifying
the structure of the message and which set of SimpleElementType’s (via ComplexElementType’s) may
accompany.</xsd:documentation>
                </xsd:annotation>
        </xsd:element>
        <xsd:element name=”OrganisationType” type = ”iso29481p2a:OrganisationTypeType” >
                <xsd:annotation>
                        <xsd:documentation>The definition of a specific group of organisations. Generally
only one instance will be present in a interaction framework defining the structure of elements that
each instance of organisation should expose.</xsd:documentation>
                </xsd:annotation>
        </xsd:element>
        <xsd:element name=”PersonType” type = ”iso29481p2a:PersonTypeType” >
                <xsd:annotation>
                        <xsd:documentation>The definition of a specific group of persons. Generally only
one instance will be present in a interaction framework defining the structure of elements that each
instance of person should expose.</xsd:documentation>
                </xsd:annotation>
        </xsd:element>
        <xsd:element name=”ProjectType” type = ”iso29481p2a:ProjectTypeType” >
                <xsd:annotation>
                        <xsd:documentation>The definition of a specific group of projects. Generally only
one instance will be present in a interaction framework defining the structure of elements that each
instance of project should expose.</xsd:documentation>
                </xsd:annotation>
        </xsd:element>
        <xsd:element name=”RoleType” type = ”iso29481p2a:RoleTypeType” >
                <xsd:annotation>
                        <xsd:documentation>The definition of a specific role type.</xsd:documentation>
                </xsd:annotation>
        </xsd:element>
        <xsd:element name=”SimpleElementType” type = ”iso29481p2a:SimpleElementTypeType” >
                <xsd:annotation>
                        <xsd:documentation>The definition of a simple element type (SimpleElementType).
This element type specifies a property whic may occur within various object structures like for
example in MessageType (see also AppendixType, ProjectType, PersonType and OrganisationType). A
SimpleElementType is alway embedded in a ComplexElementType.</xsd:documentation>
                </xsd:annotation>
        </xsd:element>
        <xsd:element name=”TransactionPhaseType” type = ”iso29481p2a:TransactionPhaseTypeType” >
                <xsd:annotation>
                        <xsd:documentation>The definition of a phase related to a transaction. Examples
are ‘assignment accepted’ or ‘result part received’.</xsd:documentation>
                </xsd:annotation>
        </xsd:element>
        <xsd:element name=”TransactionType” type = ”iso29481p2a:TransactionTypeType” >
                <xsd:annotation>
                        <xsd:documentation>The definition of a type of transaction. A transaction type
may reference again other transaction types. A transaction will be initiated by a person belonging to
an organisation fulfilling a certain role. At this level the role type of the initiator should be
stated (the promoted schema will enforce this). The same holds for the executor.</xsd:documentation>
                </xsd:annotation>
        </xsd:element>
        <xsd:element name=”UserDefinedType” type = ”iso29481p2a:UserDefinedTypeType” >
                <xsd:annotation>
                        <xsd:documentation>A specification of a data type (to be used in a
SimpleElementType).</xsd:documentation>
                </xsd:annotation>
        </xsd:element>
        <!- element reference declarations (for ENTITY definitions) →
        <xsd:element name=”AppendixTypeRef” type = ”iso29481p2a:AppendixTypeTypeRef”/ >
        <xsd:element name=”ComplexElementTypeRef” type = ”iso29481p2a:ComplexElementTypeTypeRef”/ >
        <xsd:element name=”ElementConditionRef” type = ”iso29481p2a:ElementConditionTypeRef”/ >
        <xsd:element name=”GroupTypeRef” type = ”iso29481p2a:GroupTypeTypeRef”/ >
        <xsd:element name=”MessageInTransactionTypeRef” type = ”iso29481p2a:MessageInTransactionTypeTy
peRef”/ >
        <xsd:element name=”MessageTypeRef” type = ”iso29481p2a:MessageTypeTypeRef”/ >
        <xsd:element name=”OrganisationTypeRef” type = ”iso29481p2a:OrganisationTypeTypeRef”/ >
        <xsd:element name=”PersonTypeRef” type = ”iso29481p2a:PersonTypeTypeRef”/ >
```

```
        <xsd:element name="ProjectTypeRef" type = "iso29481p2a:ProjectTypeTypeRef"/ >
        <xsd:element name="RoleTypeRef" type = "iso29481p2a:RoleTypeTypeRef"/ >
        <xsd:element name="SimpleElementTypeRef" type = "iso29481p2a:SimpleElementTypeTypeRef"/ >
        <xsd:elementname="TransactionPhaseTypeRef"type = "iso29481p2a:TransactionPhaseTypeTypeRef"/ >
        <xsd:element name="TransactionTypeRef" type = "iso29481p2a:TransactionTypeTypeRef"/ >
        <xsd:element name="UserDefinedTypeRef" type = "iso29481p2a:UserDefinedTypeTypeRef"/ >
        <!- complex types for element declarations (for ENTITY definitions) →
        <xsd:complexType name="AppendixTypeType" >
                <xsd:complexContent>
                        <xsd:extension base="iso29481p2a:elementType" >
                                <xsd:sequence>
                                        <xsd:element name="description" type = "xsd:string"/ >
                                        <xsd:element name="startDate" type = "xsd:dateTime"/ >
                                        <xsd:element name="endDate" type = "xsd:dateTime"/ >
                                        <xsd:element name="state" type = "xsd:string"/ >
                                        <xsd:element name="dateLaMu" type = "xsd:dateTime"/ >
                                        <xsd:element name="userLaMu" type = "xsd:string"/ >
                                        <xsd:element name="language" type = "xsd:string" minOc-
curs = "0"/ >
                                        <xsd:element name="category" type = "xsd:string" minOc-
curs = "0"/ >
                                        <xsd:element name="helpInfo" type = "xsd:string" minOc-
curs = "0"/ >
                                        <xsd:element name="code" type = "xsd:string" minOccurs = "0"/ >
                                        <xsd:element name="complexElements" minOccurs = "0" >
                                                <xsd:complexType>
                                                        <xsd:choice maxOccurs="unbounded" >
                                                                <xsd:element ref="iso29481p2a:ComplexElem
entType"/ >
                                                                <xsd:element ref="iso29481p2a:ComplexElem
entTypeRef"/ >
                                                        </xsd:choice>
                                                </xsd:complexType>
                                        </xsd:element>
                                </xsd:sequence>
                        </xsd:extension>
                </xsd:complexContent>
        </xsd:complexType>
        <xsd:complexType name="ComplexElementTypeType" >
                <xsd:complexContent>
                        <xsd:extension base="iso29481p2a:elementType" >
                                <xsd:sequence>
                                        <xsd:element name="description" type = "xsd:string"/ >
                                        <xsd:element name="startDate" type = "xsd:dateTime"/ >
                                        <xsd:element name="endDate" type = "xsd:dateTime"/ >
                                        <xsd:element name="state" type = "xsd:string"/ >
                                        <xsd:element name="dateLaMu" type = "xsd:dateTime"/ >
                                        <xsd:element name="userLaMu" type = "xsd:string"/ >
                                        <xsd:element name="language" type = "xsd:string" minOc-
curs = "0"/ >
                                        <xsd:element name="category" type = "xsd:string" minOc-
curs = "0"/ >
                                        <xsd:element name="helpInfo" type = "xsd:string" minOc-
curs = "0"/ >
                                        <xsd:element name="complexElements" minOccurs = "0" >
                                                <xsd:complexType>
                                                        <xsd:choice maxOccurs="unbounded" >
                                                                <xsd:element ref="iso29481p2a:ComplexElem
entType"/ >
                                                                <xsd:element ref="iso29481p2a:ComplexElem
entTypeRef"/ >
                                                        </xsd:choice>
                                                </xsd:complexType>
                                        </xsd:element>
                                        <xsd:element name="simpleElements" minOccurs = "0" >
                                                <xsd:complexType>
                                                        <xsd:choice maxOccurs="unbounded" >
                                                                <xsd:element ref="iso29481p2a:SimpleEleme
ntType"/ >
                                                                <xsd:element ref="iso29481p2a:SimpleEleme
ntTypeRef"/ >
                                                        </xsd:choice>
                                                </xsd:complexType>
                                        </xsd:element>
                                </xsd:sequence>
                        </xsd:extension>
                </xsd:complexContent>
        </xsd:complexType>
        <xsd:complexType name="ElementConditionType" >
                <xsd:complexContent>
```

```
                        <xsd:extension base="iso29481p2a:elementType" >
                                <xsd:sequence>
                                        <xsd:element name="description" type = "xsd:string"/ >
                                        <xsd:element name="condition" type = "xsd:string"/ >
                                        <xsd:element name="helpInfo" type = "xsd:string" minOc-
curs = "0"/ >
                                        <xsd:element name="complexElement" minOccurs = "0" >
                                                <xsd:complexType>
                                                        <xsd:choice>
                                                                <xsd:element ref="iso29481p2a:ComplexElem
entType"/ >
                                                                <xsd:element ref="iso29481p2a:ComplexElem
entTypeRef"/ >
                                                        </xsd:choice>
                                                </xsd:complexType>
                                        </xsd:element>
                                        <xsd:element name="simpleElement" minOccurs = "0" >
                                                <xsd:complexType>
                                                        <xsd:choice>
                                                                <xsd:element ref="iso29481p2a:SimpleEleme
ntType"/ >
                                                                <xsd:element ref="iso29481p2a:SimpleEleme
ntTypeRef"/ >
                                                        </xsd:choice>
                                                </xsd:complexType>
                                        </xsd:element>
                                        <xsd:element name="messageInTransaction" minOccurs = "0" >
                                                <xsd:complexType>
                                                        <xsd:choice>
                                                                <xsd:element ref="iso29481p2a:MessageInTr
ansactionType"/ >
                                                                <xsd:element ref="iso29481p2a:MessageInTr
ansactionTypeRef"/ >
                                                        </xsd:choice>
                                                </xsd:complexType>
                                        </xsd:element>
                                </xsd:sequence>
                        </xsd:extension>
                </xsd:complexContent>
        </xsd:complexType>
        <xsd:complexType name="GroupTypeType" >
                <xsd:complexContent>
                        <xsd:extension base="iso29481p2a:elementType" >
                                <xsd:sequence>
                                        <xsd:element name="description" type = "xsd:string"/ >
                                        <xsd:element name="startDate" type = "xsd:dateTime"/ >
                                        <xsd:element name="endDate" type = "xsd:dateTime"/ >
                                        <xsd:element name="state" type = "xsd:string"/ >
                                        <xsd:element name="dateLaMu" type = "xsd:dateTime"/ >
                                        <xsd:element name="userLaMu" type = "xsd:string"/ >
                                        <xsd:element name="language" type = "xsd:string" minOc-
curs = "0"/ >
                                        <xsd:element name="category" type = "xsd:string" minOc-
curs = "0"/ >
                                        <xsd:element name="helpInfo" type = "xsd:string" minOc-
curs = "0"/ >
                                </xsd:sequence>
                        </xsd:extension>
                </xsd:complexContent>
        </xsd:complexType>
        <xsd:complexType name="MessageInTransactionTypeType" >
                <xsd:complexContent>
                        <xsd:extension base="iso29481p2a:elementType" >
                                <xsd:sequence>
                                        <xsd:element name="requiredNotify" type = "xsd:integer"/ >
                                        <xsd:element name="dateLaMu" type = "xsd:dateTime"/ >
                                        <xsd:element name="userLaMu" type = "xsd:string"/ >
                                        <xsd:element name="received" type = "xsd:boolean"/ >
                                        <xsd:element name="send" type = "xsd:boolean"/ >
                                        <xsd:element name="state" type = "xsd:string"/ >
                                        <xsd:element name="initiatorToExecutor" type = "xsd:boolean"
minOccurs = "0"/ >
                                        <xsd:element name="openSecondaryTransactionsAllowed"
type = "xsd:boolean" minOccurs = "0"/ >
                                        <xsd:element name="firstMessage" type = "xsd:boolean"
minOccurs = "0"/ >
                                        <xsd:element name="message" >
                                                <xsd:complexType>
                                                        <xsd:choice>
                                                                <xsd:element ref="iso29481p2a:MessageType
```

```
"/ >
                                                        <xsd:element ref="iso29481p2a:MessageType
Ref"/ >
                                                </xsd:choice>
                                        </xsd:complexType>
                                </xsd:element>
                                <xsd:element name="previous" minOccurs = "0" >
                                        <xsd:complexType>
                                                <xsd:choice maxOccurs="unbounded" >
                                                        <xsd:element ref="iso29481p2a:MessageInTr
ansactionType"/ >
                                                        <xsd:element ref="iso29481p2a:MessageInTr
ansactionTypeRef"/ >
                                                </xsd:choice>
                                        </xsd:complexType>
                                </xsd:element>
                                <xsd:element name="transaction" >
                                        <xsd:complexType>
                                                <xsd:choice>
                                                        <xsd:element ref="iso29481p2a:Transaction
Type"/ >
                                                        <xsd:element ref="iso29481p2a:Transaction
TypeRef"/ >
                                                </xsd:choice>
                                        </xsd:complexType>
                                </xsd:element>
                                <xsd:element name="transactionPhase" minOccurs = "0" >
                                        <xsd:complexType>
                                                <xsd:choice>
                                                        <xsd:element ref="iso29481p2a:Transaction
PhaseType"/ >
                                                        <xsd:element ref="iso29481p2a:Transaction
PhaseTypeRef"/ >
                                                </xsd:choice>
                                        </xsd:complexType>
                                </xsd:element>
                                <xsd:element name="group" >
                                        <xsd:complexType>
                                                <xsd:choice>
                                                        <xsd:element
ref="iso29481p2a:GroupType"/ >
                                                        <xsd:element ref="iso29481p2a:GroupTypeRe
f"/ >
                                                </xsd:choice>
                                        </xsd:complexType>
                                </xsd:element>
                                <xsd:element name="appendixTypes" minOccurs = "0" >
                                        <xsd:complexType>
                                                <xsd:choice maxOccurs="unbounded" >
                                                        <xsd:element ref="iso29481p2a:AppendixTyp
e"/ >
                                                        <xsd:element ref="iso29481p2a:AppendixTyp
eRef"/ >
                                                </xsd:choice>
                                        </xsd:complexType>
                                </xsd:element>
                        </xsd:sequence>
                </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="MessageTypeType" >
            <xsd:complexContent>
                    <xsd:extension base="iso29481p2a:elementType" >
                            <xsd:sequence>
                                    <xsd:element name="description" type = "xsd:string"/ >
                                    <xsd:element name="startDate" type = "xsd:dateTime"/ >
                                    <xsd:element name="endDate" type = "xsd:dateTime"/ >
                                    <xsd:element name="state" type = "xsd:string"/ >
                                    <xsd:element name="dateLaMu" type = "xsd:dateTime"/ >
                                    <xsd:element name="userLaMu" type = "xsd:string"/ >
                                    <xsd:element name="language" type = "xsd:string" minOc-
curs = "0"/ >
                                    <xsd:element name="category" type = "xsd:string" minOc-
curs = "0"/ >
                                    <xsd:element name="helpInfo" type = "xsd:string" minOc-
curs = "0"/ >
                                    <xsd:element name="code" type = "xsd:string" minOccurs = "0"/ >
                                    <xsd:element name="complexElements" minOccurs = "0" >
                                            <xsd:complexType>
                                                    <xsd:choice maxOccurs="unbounded" >
```

```
                                                        <xsd:element ref="iso29481p2a:ComplexElem
entType"/ >
                                                        <xsd:element ref="iso29481p2a:ComplexElem
entTypeRef"/ >
                                                </xsd:choice>
                                        </xsd:complexType>
                                </xsd:element>
                                <xsd:element name="appendixTypes" minOccurs = "0" >
                                        <xsd:complexType>
                                                <xsd:choice maxOccurs="unbounded" >
                                                        <xsd:element ref="iso29481p2a:AppendixTyp
e"/ >
                                                        <xsd:element ref="iso29481p2a:AppendixTyp
eRef"/ >
                                                </xsd:choice>
                                        </xsd:complexType>
                                </xsd:element>
                        </xsd:sequence>
                </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="OrganisationTypeType" >
        <xsd:complexContent>
                <xsd:extension base="iso29481p2a:elementType" >
                        <xsd:sequence>
                                <xsd:element name="description" type = "xsd:string"/ >
                                <xsd:element name="startDate" type = "xsd:dateTime"/ >
                                <xsd:element name="endDate" type = "xsd:dateTime"/ >
                                <xsd:element name="state" type = "xsd:string"/ >
                                <xsd:element name="dateLaMu" type = "xsd:dateTime"/ >
                                <xsd:element name="userLaMu" type = "xsd:string"/ >
                                <xsd:element name="language" type = "xsd:string" minOc-
curs = "0"/ >
                                <xsd:element name="category" type = "xsd:string" minOc-
curs = "0"/ >
                                <xsd:element name="helpInfo" type = "xsd:string" minOc-
curs = "0"/ >
                                <xsd:element name="code" type = "xsd:string" minOccurs = "0"/ >
                                <xsd:element name="complexElements" minOccurs = "0" >
                                        <xsd:complexType>
                                                <xsd:choice maxOccurs="unbounded" >
                                                        <xsd:element ref="iso29481p2a:ComplexElem
entType"/ >
                                                        <xsd:element ref="iso29481p2a:ComplexElem
entTypeRef"/ >
                                                </xsd:choice>
                                        </xsd:complexType>
                                </xsd:element>
                        </xsd:sequence>
                </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="PersonTypeType" >
        <xsd:complexContent>
                <xsd:extension base="iso29481p2a:elementType" >
                        <xsd:sequence>
                                <xsd:element name="description" type = "xsd:string"/ >
                                <xsd:element name="startDate" type = "xsd:dateTime"/ >
                                <xsd:element name="endDate" type = "xsd:dateTime"/ >
                                <xsd:element name="state" type = "xsd:string"/ >
                                <xsd:element name="dateLaMu" type = "xsd:dateTime"/ >
                                <xsd:element name="userLaMu" type = "xsd:string"/ >
                                <xsd:element name="language" type = "xsd:string" minOc-
curs = "0"/ >
                                <xsd:element name="category" type = "xsd:string" minOc-
curs = "0"/ >
                                <xsd:element name="helpInfo" type = "xsd:string" minOc-
curs = "0"/ >
                                <xsd:element name="code" type = "xsd:string" minOccurs = "0"/ >
                                <xsd:element name="complexElements" minOccurs = "0" >
                                        <xsd:complexType>
                                                <xsd:choice maxOccurs="unbounded" >
                                                        <xsd:element ref="iso29481p2a:ComplexElem
entType"/ >
                                                        <xsd:element ref="iso29481p2a:ComplexElem
entTypeRef"/ >
                                                </xsd:choice>
                                        </xsd:complexType>
                                </xsd:element>
                        </xsd:sequence>
```

```
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>
        <xsd:complexType name="ProjectTypeType" >
            <xsd:complexContent>
                <xsd:extension base="iso29481p2a:elementType" >
                    <xsd:sequence>
                        <xsd:element name="namespace" type = "xsd:string"/ >
                        <xsd:element name="description" type = "xsd:string"/ >
                        <xsd:element name="startDate" type = "xsd:dateTime"/ >
                        <xsd:element name="endDate" type = "xsd:dateTime"/ >
                        <xsd:element name="state" type = "xsd:string"/ >
                        <xsd:element name="dateLaMu" type = "xsd:dateTime"/ >
                        <xsd:element name="userLaMu" type = "xsd:string"/ >
                        <xsd:element name="language" type = "xsd:string" minOc-
curs = "0"/ >
                        <xsd:element name="category" type = "xsd:string" minOc-
curs = "0"/ >
                        <xsd:element name="helpInfo" type = "xsd:string" minOc-
curs = "0"/ >
                        <xsd:element name="code" type = "xsd:string" minOccurs = "0"/ >
                        <xsd:element name="complexElements" minOccurs = "0" >
                            <xsd:complexType>
                                <xsd:choice maxOccurs="unbounded" >
                                    <xsd:element ref="iso29481p2a:ComplexElem
entType"/ >
                                    <xsd:element ref="iso29481p2a:ComplexElem
entTypeRef"/ >
                                </xsd:choice>
                            </xsd:complexType>
                        </xsd:element>
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>
        <xsd:complexType name="RoleTypeType" >
            <xsd:complexContent>
                <xsd:extension base="iso29481p2a:elementType" >
                    <xsd:sequence>
                        <xsd:element name="description" type = "xsd:string"/ >
                        <xsd:element name="startDate" type = "xsd:dateTime"/ >
                        <xsd:element name="endDate" type = "xsd:dateTime"/ >
                        <xsd:element name="state" type = "xsd:string"/ >
                        <xsd:element name="dateLaMu" type = "xsd:dateTime"/ >
                        <xsd:element name="userLaMu" type = "xsd:string"/ >
                        <xsd:element name="language" type = "xsd:string" minOc-
curs = "0"/ >
                        <xsd:element name="category" type = "xsd:string" minOc-
curs = "0"/ >
                        <xsd:element name="helpInfo" type = "xsd:string" minOc-
curs = "0"/ >
                        <xsd:element name="code" type = "xsd:string" minOccurs = "0"/ >
                        <xsd:element name="responsibilityScope" type = "xsd:string"
minOccurs = "0"/ >
                        <xsd:element name="responsibilityTask" type = "xsd:string"
minOccurs = "0"/ >
                        <xsd:element name="responsibilitySupportTask"
type = "xsd:string" minOccurs = "0"/ >
                        <xsd:element name="responsibilityFeedback" type = "xsd:string"
minOccurs = "0"/ >
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>
        <xsd:complexType name="SimpleElementTypeType" >
            <xsd:complexContent>
                <xsd:extension base="iso29481p2a:elementType" >
                    <xsd:sequence>
                        <xsd:element name="description" type = "xsd:string"/ >
                        <xsd:element name="interfaceType" type = "xsd:string"/ >
                        <xsd:element name="state" type = "xsd:string"/ >
                        <xsd:element name="dateLaMu" type = "xsd:dateTime"/ >
                        <xsd:element name="userLaMu" type = "xsd:string"/ >
                        <xsd:element name="language" type = "xsd:string" minOc-
curs = "0"/ >
                        <xsd:element name="category" type = "xsd:string" minOc-
curs = "0"/ >
                        <xsd:element name="helpInfo" type = "xsd:string" minOc-
curs = "0"/ >
                        <xsd:element name="valueList" type = "xsd:string" minOc-
```

```
curs = ”0”/ >
                                        <xsd:element name=”userDefinedType” >
                                                <xsd:complexType>
                                                        <xsd:choice>
                                                                <xsd:element ref=”iso29481p2a:UserDefinedT
ype”/ >
                                                                <xsd:element ref=”iso29481p2a:UserDefinedT
ypeRef”/ >
                                                        </xsd:choice>
                                                </xsd:complexType>
                                        </xsd:element>
                                </xsd:sequence>
                        </xsd:extension>
                </xsd:complexContent>
        </xsd:complexType>
        <xsd:complexType name=”TransactionPhaseTypeType” >
                <xsd:complexContent>
                        <xsd:extension base=”iso29481p2a:elementType” >
                                <xsd:sequence>
                                        <xsd:element name=”description” type = ”xsd:string”/ >
                                        <xsd:element name=”startDate” type = ”xsd:dateTime”/ >
                                        <xsd:element name=”endDate” type = ”xsd:dateTime”/ >
                                        <xsd:element name=”state” type = ”xsd:string”/ >
                                        <xsd:element name=”dateLaMu” type = ”xsd:dateTime”/ >
                                        <xsd:element name=”userLaMu” type = ”xsd:string”/ >
                                        <xsd:element name=”language” type = ”xsd:string” minOc-
curs = ”0”/ >
                                        <xsd:element name=”category” type = ”xsd:string” minOc-
curs = ”0”/ >
                                        <xsd:element name=”helpInfo” type = ”xsd:string” minOc-
curs = ”0”/ >
                                        <xsd:element name=”code” type = ”xsd:string” minOccurs = ”0”/ >
                                </xsd:sequence>
                        </xsd:extension>
                </xsd:complexContent>
        </xsd:complexType>
        <xsd:complexType name=”TransactionTypeType” >
                <xsd:complexContent>
                        <xsd:extension base=”iso29481p2a:elementType” >
                                <xsd:sequence>
                                        <xsd:element name=”description” type = ”xsd:string”/ >
                                        <xsd:element name=”startDate” type = ”xsd:dateTime”/ >
                                        <xsd:element name=”endDate” type = ”xsd:dateTime”/ >
                                        <xsd:element name=”state” type = ”xsd:string”/ >
                                        <xsd:element name=”dateLaMu” type = ”xsd:dateTime”/ >
                                        <xsd:element name=”userLaMu” type = ”xsd:string”/ >
                                        <xsd:element name=”language” type = ”xsd:string” minOc-
curs = ”0”/ >
                                        <xsd:element name=”category” type = ”xsd:string” minOc-
curs = ”0”/ >
                                        <xsd:element name=”helpInfo” type = ”xsd:string” minOc-
curs = ”0”/ >
                                        <xsd:element name=”code” type = ”xsd:string” minOccurs = ”0”/ >
                                        <xsd:element name=”result” type = ”xsd:string” minOc-
curs = ”0”/ >
                                        <xsd:element name=”subTransactions” minOccurs = ”0” >
                                                <xsd:complexType>
                                                        <xsd:choice maxOccurs=”unbounded” >
                                                                <xsd:element ref=”iso29481p2a:Transaction
Type”/ >
                                                                <xsd:element ref=”iso29481p2a:Transaction
TypeRef”/ >
                                                        </xsd:choice>
                                                </xsd:complexType>
                                        </xsd:element>
                                        <xsd:element name=”initiator” >
                                                <xsd:complexType>
                                                        <xsd:choice>
                                                                <xsd:element
ref=”iso29481p2a:RoleType”/ >
                                                                <xsd:element ref=”iso29481p2a:RoleTypeRef
”/ >
                                                        </xsd:choice>
                                                </xsd:complexType>
                                        </xsd:element>
                                        <xsd:element name=”executor” >
                                                <xsd:complexType>
                                                        <xsd:choice>
                                                                <xsd:element
ref=”iso29481p2a:RoleType”/ >
```

```
                                                                <xsd:element ref=”iso29481p2a:RoleTypeRef
”/ >
                                                        </xsd:choice>
                                                </xsd:complexType>
                                        </xsd:element>
                                        <xsd:element name=”appendixTypes” minOccurs = ”0” >
                                                <xsd:complexType>
                                                        <xsd:choice maxOccurs=”unbounded” >
                                                                <xsd:element ref=”iso29481p2a:AppendixTyp
e”/ >
                                                                <xsd:element ref=”iso29481p2a:AppendixTyp
eRef”/ >
                                                        </xsd:choice>
                                                </xsd:complexType>
                                        </xsd:element>
                                </xsd:sequence>
                        </xsd:extension>
                </xsd:complexContent>
        </xsd:complexType>
        <xsd:complexType name=”UserDefinedTypeType” >
                <xsd:complexContent>
                        <xsd:extension base=”iso29481p2a:elementType” >
                                <xsd:sequence>
                                        <xsd:element name=”description” type = ”xsd:string”/ >
                                        <xsd:element name=”state” type = ”xsd:string”/ >
                                        <xsd:element name=”dateLaMu” type = ”xsd:dateTime”/ >
                                        <xsd:element name=”userLaMu” type = ”xsd:string”/ >
                                        <xsd:element name=”baseType” type = ”xsd:string”/ >
                                        <xsd:element name=”xsdRestriction” type = ”xsd:string”
minOccurs = ”0”/ >
                                        <xsd:element name=”language” type = ”xsd:string” minOc-
curs = ”0”/ >
                                        <xsd:element name=”helpInfo” type = ”xsd:string” minOc-
curs = ”0”/ >
                                </xsd:sequence>
                        </xsd:extension>
                </xsd:complexContent>
        </xsd:complexType>
        <!- complex types for element reference declarations (for ENTITY definitions) →
        <xsd:complexType name=”AppendixTypeTypeRef” >
                <xsd:complexContent>
                        <xsd:extension base=”iso29481p2a:elementTypeRef”/ >
                </xsd:complexContent>
        </xsd:complexType>
        <xsd:complexType name=”ComplexElementTypeTypeRef” >
                <xsd:complexContent>
                        <xsd:extension base=”iso29481p2a:elementTypeRef”/ >
                </xsd:complexContent>
        </xsd:complexType>
        <xsd:complexType name=”ElementConditionTypeRef” >
                <xsd:complexContent>
                        <xsd:extension base=”iso29481p2a:elementTypeRef”/ >
                </xsd:complexContent>
        </xsd:complexType>
        <xsd:complexType name=”GroupTypeTypeRef” >
                <xsd:complexContent>
                        <xsd:extension base=”iso29481p2a:elementTypeRef”/ >
                </xsd:complexContent>
        </xsd:complexType>
        <xsd:complexType name=”MessageInTransactionTypeTypeRef” >
                <xsd:complexContent>
                        <xsd:extension base=”iso29481p2a:elementTypeRef”/ >
                </xsd:complexContent>
        </xsd:complexType>
        <xsd:complexType name=”MessageTypeTypeRef” >
                <xsd:complexContent>
                        <xsd:extension base=”iso29481p2a:elementTypeRef”/ >
                </xsd:complexContent>
        </xsd:complexType>
        <xsd:complexType name=”OrganisationTypeTypeRef” >
                <xsd:complexContent>
                        <xsd:extension base=”iso29481p2a:elementTypeRef”/ >
                </xsd:complexContent>
        </xsd:complexType>
        <xsd:complexType name=”PersonTypeTypeRef” >
                <xsd:complexContent>
                        <xsd:extension base=”iso29481p2a:elementTypeRef”/ >
                </xsd:complexContent>
        </xsd:complexType>
        <xsd:complexType name=”ProjectTypeTypeRef” >
```

```
                <xsd:complexContent>
                        <xsd:extension base="iso29481p2a:elementTypeRef"/ >
                </xsd:complexContent>
        </xsd:complexType>
        <xsd:complexType name="RoleTypeTypeRef" >
                <xsd:complexContent>
                        <xsd:extension base="iso29481p2a:elementTypeRef"/ >
                </xsd:complexContent>
        </xsd:complexType>
        <xsd:complexType name="SimpleElementTypeTypeRef" >
                <xsd:complexContent>
                        <xsd:extension base="iso29481p2a:elementTypeRef"/ >
                </xsd:complexContent>
        </xsd:complexType>
        <xsd:complexType name="TransactionPhaseTypeTypeRef" >
                <xsd:complexContent>
                        <xsd:extension base="iso29481p2a:elementTypeRef"/ >
                </xsd:complexContent>
        </xsd:complexType>
        <xsd:complexType name="TransactionTypeTypeRef" >
                <xsd:complexContent>
                        <xsd:extension base="iso29481p2a:elementTypeRef"/ >
                </xsd:complexContent>
        </xsd:complexType>
        <xsd:complexType name="UserDefinedTypeTypeRef" >
                <xsd:complexContent>
                        <xsd:extension base="iso29481p2a:elementTypeRef"/ >
                </xsd:complexContent>
        </xsd:complexType>
        <!- group declarations (for SELECT data type definitions) →
        <!- enumeration type declarations (for ENUMERATION data type definitions) →
        <!- simple type declarations (for TYPE defined data type definitions) →
        <!- standard declarations, elementType, simpleType, logical (for each translation) →
        <xsd:complexType name="elementType" >
                <xsd:attribute name="id" type = "xsd:ID" use = "required"/ >
        </xsd:complexType>
        <xsd:complexType name="elementTypeRef" >
                <xsd:attribute name="idref" type = "xsd:IDREF" use = "required"/ >
        </xsd:complexType>
        <xsd:simpleType name="logical" >
                <xsd:restriction base="xsd:string" >
                        <xsd:enumeration value="true"/ >
                        <xsd:enumeration value="false"/ >
                        <xsd:enumeration value="unknown"/ >
                </xsd:restriction>
        </xsd:simpleType>
</xsd:schema>
```

A.4 Element types

A.4.1 AppendixType

Attributes: id

Elements: description, startDate, endDate, state, dateLaMu, userLaMu, language, category, helpInfo, code

References: complexElements

An AppendixType contains the definition of an appendix. Which data items should be recorded with an appendix can be specified in the complex element section.

EXAMPLE

```
<AppendixType id="StandardAppendixType" >
  <description>Standard appendix type</description>
  <startDate>2007-01-23T00:00:00Z</startDate>
  <endDate>2007-12-31T00:00:00Z</endDate>
  <state>active</state>
  <dateLaMu>2007-01-23T00:00:00Z</dateLaMu>
  <userLaMu>bapa</userLaMu>
</AppendixType>
```

NOTE Reference **appendixTypes**. If an interaction framework specifies many appendix types, it can be hard for users to select the appropriate appendix type in certain situations. This reference filters the set of all appendix types to the ones that are valid for this transaction or message.

A.4.2 ComplexElementType

Attributes: id

Elements: description, startDate, endDate, state, dateLaMu, userLaMu, language, category, helpInfo

References: elements

A ComplexElementType contains a set of SimpleElementTypes. Each stated SimpleElementType occurs exactly the number of times it is mentioned.

EXAMPLE

```
<ComplexElementType id=”MenuItem” >
  <description>Item on menu</description>
  <startDate>2007-01-23T00:00:00Z</startDate>
  <endDate>2007-12-31T00:00:00Z</endDate>
  <state>active</state>
  <dateLaMu>2007-01-23T00:00:00Z</dateLaMu>
  <userLaMu>bapa</userLaMu>
  <elements>
    <SimpleElementTypeRef idref=”Name”/ >
    <SimpleElementTypeRef idref=”Price”/ >
    <SimpleElementTypeRef idref=”Description”/ >
    <SimpleElementTypeRef idref=”Calories”/ >
  </elements>
</ComplexElementType>
```

A.4.3 ElementCondition

Attributes: id

Elements: description, requiredNotify, minValue, maxValue, format, helpInfo

References: message, element

The condition of a SimpleElementType as used within a specific MessageType.

EXAMPLE

```
<ElementCondition id=”Price restriction” >
  <description>Minimal price of a menu item</description>
  <requiredNotify>0</requiredNotify>
  <minValue>5.00</minValue>
  <element>
    <SimpleElementType>Price</SimpleElementType>
  </element>
  <message>
    <MessageTypeRef idref=”ProvideWithMenuMessage”/ >
  </message>
</ElementCondition>
```

A.4.4 GroupType

Attributes: id

Elements: description, startDate, endDate, state, dateLaMu, userLaMu, language, category, helpInfo

The definition of a group to store appendices sent with a message within a transaction.

EXAMPLE

```
<GroupType id=”StandardGroupType” >
  <description>Standard group</description>
  <startDate>2007-12-20T00:00:00Z</startDate>
  <endDate>2008-12-31T00:00:00Z</endDate>
  <state>active</state>
  <dateLaMu>2007-12-20T00:00:00Z</dateLaMu>
  <userLaMu>bapa</userLaMu>
</GroupType>
```

A.4.5 MessageType

Attributes: id

Elements: description, startDate, endDate, state, dateLaMu, userLaMu, language, category, helpInfo, code

References: complexElements, appendixTypes

The definition of a type of message (MessageType), specifying the structure of the message and which set of SimpleElementType's (via ComplexElementType's) may accompany.

EXAMPLE

```
<MessageType id="ProvideWithMenuMessage" >
  <description>Message that contains the menu.</description>
  <startDate>2008-01-23T00:00:00Z</startDate>
  <endDate>2008-12-31T00:00:00Z</endDate>
  <state>active</state>
  <dateLaMu>2008-01-23T00:00:00Z</dateLaMu>
  <userLaMu>bapa</userLaMu>
  <complexElements>
    <ComplexElementTypeRef idref="Menu"/ >
  </complexElements>
  <appendixTypes>
    <AppendixTypeRef idref="Menucard" / >
  </appendixTypes>
</MessageType>
```

NOTE Element **firstMessage**. In general, a transaction will be started by the initiating role fulfilling person using a message type that is independent from any previous message type. However, this principle cannot be applied to a transaction that acts as a sub-transaction of another transaction. This element explicitly states if a message type can be used to start a new transaction.

A.4.6 MessageInTransactionType

Attributes: id

Elements: requiredNotify, dateLaMu, userLaMu, received, send, state, initiatorToExecutor, openSecondaryTransactionsAllowed, firstMessage

References: message, previous, transaction, transactionPhase, group, appendixTypes

The occurrence of a MessageType within a TransactionType related to a certain group type (GroupType).

EXAMPLE

```
<MessageInTransactionType id="MiTT_002" >
  <requiredNotify>0</requiredNotify>
  <dateLaMu>2008-01-23T00:00:00Z</dateLaMu>
  <userLaMu>bapa</userLaMu>
  <received>true</received>
  <send>true</send>
  <state>active</state>
  <initiatorToExecutor>false</initiatorToExecutor>
  <openSecondaryTransactionsAllowed>true</openSecondaryTransactionsAllowed>
 <firstMessage>false</firstMessage>
  <message>
    <MessageTypeRef idref="ProvideWithMenuMessage"/ >
  </message>
  <previous>
    <MessageInTransactionTypeRef idref="MiTT_001"/ >
  </previous>
  <transaction>
    <TransactionTypeRef idref="ObtainMenuTransaction"/ >
  </transaction>
  <transactionPhase>
    <TransactionPhaseTypeRef idref="MenuDelivered" >
  </transactionPhase>
  <group>
    <GroupTypeRef idref="StandardGroupType"/ >
  </group>
 <appendixTypes>
    <AppendixTypeRef idref="Menucard" / >
```

```
  </appendixTypes>
</MessageInTransactionType>
```

A.4.7 OrganisationType

Attributes: id

Elements: description, startDate, endDate, state, dateLaMu, userLaMu, language, category, helpInfo, code

References: complexElements

The definition of a specific group of organizations. Generally, only one instance will be present in an interaction framework defining the structure of elements that each instance of organization should expose.

EXAMPLE

```
<OrganisationType id="StandardOrganisationType" >
  <description>Standard organisation type</description>
  <startDate>2008-01-23T00:00:00Z</startDate>
  <endDate>2008-12-31T00:00:00Z</endDate>
  <state>active</state>
  <dateLaMu>2008-01-23T00:00:00Z</dateLaMu>
  <userLaMu>bapa</userLaMu>
</OrganisationType>
```

A.4.8 PersonType

Attributes: id

Elements: description, startDate, endDate, state, dateLaMu, userLaMu, language, category, helpInfo, code

References: complexElements

The definition of a specific group of persons. Generally, only one instance will be present in an interaction framework defining the structure of elements that each instance of person should expose.

EXAMPLE

```
<PersonType id="StandardPersonType" >
  <description>Standard person type</description>
  <startDate>2008-01-23T00:00:00Z</startDate>
  <endDate>2008-12-31T00:00:00Z</endDate>
  <state>active</state>
  <dateLaMu>2008-01-23T00:00:00Z</dateLaMu>
  <userLaMu>bapa</userLaMu>
</PersonType>
```

A.4.9 ProjectType

Attributes: id

Elements: namespace, description, startDate, endDate, state, dateLaMu, userLaMu, language, category, helpInfo, code

References: complexElements

The definition of a specific group of projects. Generally, only one instance will be present in an interaction framework defining the structure of elements that each instance of project should expose.

EXAMPLE

```
<ProjectType id="StandardProjectType" >
  <namespace>http://www.ISO 29481_Part_2A.com/testproject</namespace>
  <description>Standard project type</description>
  <startDate>2008-01-23T00:00:00Z</startDate>
  <endDate>2008-12-31T00:00:00Z</endDate>
  <state>active</state>
  <dateLaMu>2008-01-23T00:00:00Z</dateLaMu>
  <userLaMu>bapa</userLaMu>
</ProjectType>
```

A.4.10 RoleType

Attributes: id

Elements: description, startDate, endDate, state, dateLaMu, userLaMu, language, category, helpInfo, code, responsibilityFeedback, responsibilityScope, responsibilitySupportTask, responsibilityTask

The definition of a specific role type.

EXAMPLE

```
<RoleType id="waiter" >
  <description>Responsible for taking and posting of orders</description>
  <startDate>2008-05-04T00:00:00:00Z</startDate>
  <endDate>2008-05-04T00:00:00.00Z</endDate>
  <state>active</state>
  <dateLaMu>2008-05-04T00:00:00.00Z</dateLaMu>
  <userLaMu>MMA</userLaMu>
  <language/>
  <category/>
  <helpInfo/>
  <code/>
  <responsibilityScope/>
  <responsibilityTask/>
  <responsibilitySupportTask/>
  <responsibilityFeedback/>
</RoleType>
```

A.4.11 SimpleElementType

Attributes: id

Elements: description, interfaceType, state, dateLaMu, userLaMu, language, category, helpInfo, valueList

References: composedOf, userDefinedType

The definition of a simple element type (SimpleElementType). This element type specifies a property which may occur within various object structures, for example in MessageType (see also AppendixType, ProjectType, PersonType, and OrganisationType). A SimpleElementType is always embedded in a ComplexElementType.

EXAMPLE

```
<SimpleElementType id="Name" >
  <description>Name of the menu item</description>
  <interfaceType/>
  <state>active</state>
  <dateLaMu>2008-01-23T00:00:00Z</dateLaMu>
  <userLaMu>bapa</userLaMu>
  <userDefinedType>
    <UserDefinedTypeRef idref="String"/ >
  </userDefinedType>
</SimpleElementType>
```

A.4.12 TransactionPhaseType

Attributes: id

Elements: description, startDate, endDate, state, dateLaMu, userLaMu, language, category, helpInfo, code

The definition of a phase related to a transaction. Examples are 'assignment accepted' or 'result part received'.

EXAMPLE

```
<TransactionPhaseType id="WaitingForMenu" >
  <description>Menu requested but not yet delivered</description>
  <startDate>2008-01-23T00:00:00Z</startDate>
  <endDate>2008-12-31T00:00:00Z</endDate>
  <state>active</state>
  <dateLaMu>2008-01-23T00:00:00Z</dateLaMu>
  <userLaMu>bapa</userLaMu>
</TransactionPhaseType>
```

A.4.13 TransactionType

Attributes: id

Elements: description, startDate, endDate, state, dateLaMu, userLaMu, language, category, helpInfo, code, result

References: initiator, executor, subTransactions

The definition of a type of transaction. A transaction type may reference again other transaction types. A transaction will be initiated by a person belonging to an organization fulfilling a certain role. At this level, the role type of the initiator should be stated (the promoted schema will enforce this). The same holds for the executor.

EXAMPLE

```
<TransactionType id="MenuObtainingTransaction" >
  <description>The transaction to obtain the proper menu</description>
  <startDate>2008-01-23T00:00:00Z</startDate>
  <endDate>2008-12-31T00:00:00Z</endDate>
  <state>active</state>
  <dateLaMu>2008-01-23T00:00:00Z</dateLaMu>
  <userLaMu>bapa</userLaMu>
  <initiator>
    <RoleTypeRef idref="Customer"/ >
  </initiator>
  <executor>
    <RoleTypeRef idref="Employee"/ >
  </executor>
</TransactionType>
```

A.4.14 UserDefinedType

Attributes: #id

Elements: description, state, dateLaMu, userLaMu, baseType, xsdRestriction, language, helpInfo

A specification of a data type (to be used in a SimpleElementType). This data type encapsulates fill-in areas in the final message.

EXAMPLE

```
<UserDefinedType id="String" >
  <description>Standard string</description>
  <state>active</state>
  <dateLaMu>2007-12-20T00:00:00Z</dateLaMu>
  <userLaMu>bapa</userLaMu>
  <baseType>STRING</baseType>
  <xsdRestriction/>
</UserDefinedType>
```

A.5 Attributes

A.5.1 id

Unique 'short' name of the occurrence. After promotion, this name will be an object name.

EXAMPLE

```
Framework object
<OrganisationType id="TNO Building and Construction" >
  <description>The attributes of an organisation</description>
  <startDate>2007-12-12T00:00:00Z</startDate>
  <endDate>9999-12-31T00:00:00Z</endDate>
  <state>active</state>
  <dateLaMu>2007-12-12T00:00:00Z</dateLaMu>
  <userLaMu>testmanagement</userLaMu>
  <language>NL</language>
  <category>S</category>
  <helpInfo>http://.../</helpInfo>
  <code>DEFAULT</code>
```

```
</OrganisationType>
Message object
  <name>TNO Building & Construction</name>
  <state>active</state>
  <dateLaMu>2007-12-02T00:00:00Z</dateLaMu>
  <userLaMu>Peter Bonsma</userLaMu>
</Organisation>
```

A.6 Elements

A.6.1 baseType

Holds the base type of a SimpleElementType.

EXAMPLE

```
<SimpleElementType id="Height" >
  ...
  <userDefinedType>
    <UserDefinedType id="..." >
      ...
      <baseType>INTEGER</baseType>
      ...
    </UserDefinedType>
  </userDefinedType>
</SimpleElementType>
```

Here, the SimpleElementType *Height* always holds an integer number (possibly with a restriction: xsdRestriction).

A.6.2 category

Category associated with this object (optional value).

A.6.3 code

Interaction framework agreed code for this object.

EXAMPLE

```
<... id="..." >
  ...
  EAN 33156
  ...
</...>
```

A.6.4 dateLaMu

Date and time of the last mutation to this object.

```
<... id="..." >
  ...
  <dateLaMu>2007-12-02T00:00:00Z</dateLaMu>
  ...
</...>
```

A.6.5 description

Description of the object.

EXAMPLE

```
<... id="Doorleaf" >
  ...
  <description>The leaf of a flat door.</description>
  ...
</...>
```

A.6.6 endDate

End date and time of validity of this object.

EXAMPLE

```
<... id="..." >
  ...
  <endDate>2007-02-03T00:00:00Z</endDate>
  ...
</...>
```

A.6.7 firstMessage

Optional Boolean value to indicate that this MessageInTransactionType object is allowed to be used as a start message for a new transaction. Default value is false.

EXAMPLE

```
<MessageInTransactionType id="..." >
  ...
  <firstMessage>true</firstMessage>
  ...
</MessageInTransactionType>
```

A.6.8 format

The layout of an element (optional).

A.6.9 helpInfo

A URL/URI linking to further information about this object.

EXAMPLE

```
<... id="..." >
  ...
  <helpInfo> http://www.ISO 29481_Part_2A.com/helpInfo_object0001.html</helpInfo>
  ...
</...>
```

A.6.10 initiatorToExecutor

A Boolean value representing the direction the message is supposed to be sent.

EXAMPLE

```
<MessageInTransactionType id="..." >
  ...
  <initiatorToExecutor>false</initiatorToExecutor>
  ...
  <message>
    <MessageType id="TenderAcceptance" >
      ...
    </MessageType>
  </message>
  <transaction>
    <TransactionType id="TenderTraject" >
    ...
      <initiator>
        <RoleType id="Performer" >
          ...
        </RoleType>
      </initiator>
      <executor>
        <RoleType id="Client" >
          ...
        </RoleType>
      </executor>
        ...
    </TransactionType>
  </transaction>
    ...
  <MessageInTransactionType id="..." >
```

It is anticipated that the message *TenderAcceptance* will be sent from Client (executor) to Performer (initiator).

A.6.11 interfaceType

Type interface or view on this SimpleElementType for a specific message.

A.6.12 language

Language to be used after promotion of this object.

EXAMPLE

```
<... id="..." >
  ...
  <language>NL</language>
  ...
</...>
```

A.6.13 namespace

Namespace target name to identify messages belonging to this interaction framework.

EXAMPLE

```
<ProjectType id="..." >
  <namespace>http://www.visi.nl/testraamwerk</namespace>
  ...
</ProjectType>
```

NOTE Element **namespace**. Previously, all interaction framework schemas referred to the same target namespace, which happened to be the same as the namespace of the interaction schema itself. Using this element, each interaction schema can specify its own namespace.

A.6.14 openSecondaryTransactionsAllowed

Optional Boolean value that allows continuation of the primary transaction even if not all secondary transactions are completed. A "TRUE" value is interpreted as a green light for continuation (OR gate). A "FALSE" value is interpreted as a red light for continuation until all secondary transactions are completed (AND gate). If openSecondaryTransactionsAllowed is not specified, the interpretation equals "TRUE".

A.6.15 received

Boolean value indicating that the previous message is received.

A.6.16 requiredNotify

No semantics are associated with this element.

A.6.17 responsibilityFeedback

Expected feedback in the direction of other roles (optional string).

A.6.18 responsibilityScope

Scope/frame of the defined responsibilities of the role (optional string).

A.6.19 responsibilitySupportTask

Tasks to be executed to support other roles. For example, delegated responsibilities (optional string).

A.6.20 responsibilityTask

Tasks originating from the responsibilities of this role (optional string).

A.6.21 result

Expected result from the execution of the transaction.

A.6.22 send

Boolean value indicating whether the message has been sent or not.

A.6.23 startDate

Start date and time of validity of this object.

EXAMPLE

```
<... id="..." >
  ...
  <startDate>2007-02-03T00:00:00Z</startDate>
  ...
</...>
```

A.6.24 state

Visibility state of this object, possible values:

EXAMPLE

```
<... id="..." >
  ...
  <state>active</state>
  ...
</...>
```

and

```
<... id="..." >
  ...
  <state>passive</state>
  ...
</...>
```

A.6.25 userLaMu

User of the last mutation to this object.

EXAMPLE

```
<... id="..." >
  ...
  <userLaMu>Peter Bonsma</userLaMu>
  ...
</...>
```

A.6.26 valueList

Semicolon separated list of values which an occurrence at message level can take. Originally, this element was intended to support enumerations. Currently, this is implemented with the element type UserDefinedType and the element xsdRestriction. In the xsdRestriction, enumeration values are specified. No semantics are linked to this element anymore.

EXAMPLE

```
<SimpleElementType id="..." >
  ...
  <valueList>Groen;Rood;Oker Geel</valueList>
  ...
</SimpleElementType>
```

A.6.27 xsdRestriction

This element specifies a restriction to be performed at message level on the SimpleElementType's of this UserDefinedType.

A.7 References

A.7.1 appendixTypes

Set of selectable appendix types.

A.7.2 complexElements

A reference to a set of SimpleElementType's (collected in a ComplexElementType).

EXAMPLE

```
<... id="Abc" >
...
  <complexElements>
    <ComplexElementType id="ElementsSet1" >
     ...
      <elements>
        <SimpleElementType id="Element_A" >
         ...
        </SimpleElementType>
        <SimpleElementType id="Element_B" >
         ...
        </SimpleElementType>
      </elements>
    </ComplexElementType>
    <ComplexElementType id="ElementsSet2" >
     ...
    </ComplexElementType>
    <ComplexElementType id="ElementsSet3" >
     ...
    </ComplexElementType>
  </complexElements>
</...>
```

At message level, this is elaborated in:

```
<Abc id="..." >
 ...
  <elementsSet1>
    <ElementsSet1>
      <Element_A>...</Element_A>
      <Element_B>...</Element_B>
    </ElementsSet1>
    ...
    <ElementsSet1>
      <Element_A>...</Element_A>
      <Element_B>...</Element_B>
    </ElementsSet1>
  </elementsSet1>
  <elementsSet2>
    <ElementsSet2>
     ...
    </ElementsSet2>
     ...
    <ElementsSet2>
     ...
    </ElementsSet2>
  </elementsSet2>
  <elementsSet3>
    <ElementsSet3>
     ...
    </ElementsSet3>
     ...
    <ElementsSet3>
     ...
    </ElementsSet3>
  </elementsSet3>
</...>
```

A.7.3 composedOf

A SimpleElementType may consist of a set of SimpleElementType's (collected in a ComplexElementType).

EXAMPLE

```
<SimpleElementType id=”Doorleaf” >
  ...
  <composedOf>
    <ComplexElementType id=”Measurement” >
      ...
      <elements>
        <SimpleElementType id=”Height” >
          ...
        </SimpleElementType>
        <SimpleElementType id=”Width” >
          ...
        </SimpleElementType>
        <SimpleElementType id=”Thickness” >
          ...
        </SimpleElementType>
      </elements>
    </ComplexElementType>
    <ComplexElementType id=”Material-selection” >
      ...
      <elements>
        <SimpleElementType id=”Type-of-wood” >
          ...
        </SimpleElementType>
      </elements>
    </ComplexElementType>
  </composedOf>
</SimpleElementType>
```

At message level, this is elaborated to:

```
<Doorleaf>
  ...
  <Measurement id=”...” >
    <Height>...</Height>
    <Width>...</Width>
    <Thickness>...</Thickness>
  </Measurement>
  <Material-selection id=”...” >
    <Type-of-wood>...</Type-of-wood>
  </Material-selection>
</Doorleaf>
```

A.7.4 element

The condition on a SimpleElementType to be used within a MessageType.

EXAMPLE

```
<ElementCondition id=”...” >
  ...
  <minValue>2000</minValue>
  ...
  <element>
    <SimpleElementTypeRef idref=”Doorheight” >
  </element>
  <message>
    <MessageType id=”M” >
      ...
      <complexElements>
        <ComplexElementType>
            ...
          <elements>
            <SimpleElementType id=”Doorheight” >
              ...
            </SimpleElementType>
          </elements>
        </ComplexElementType>
      </complexElements>
    </MessageType>
  </message>
</ElementCondition>
```

It is specified that within MessageType *M*, the *Doorheight* should be at least 2000, though this is not enforced at SimpleElementType level.

A.7.5 elements

Set of SimpleElementType’s available within a ComplexElementType.

EXAMPLE

```
<ComplexElementType id=”Door” >
  ...
  <elements>
    <SimpleElementType id=”Doorleaf” >
      ...
    </SimpleElementType>
    <SimpleElementType id=”Fastenings” >
      ...
    </SimpleElementType>
    <SimpleElementType id=”UpperWindow” >
      ...
    </SimpleElementType>
  </elements>
</ComplexElementType
```

At message level, this is elaborated as follows:

```
<... id=”...” >
  ...
  <door>
    <Door>
      <Doorleaf>...</Doorleaf>
      <Fastenings>...</Fastenings>
      <UpperWindow>...</UpperWindow>
    </Door>
    ...
    <Door>
      <Doorleaf>...</Doorleaf>
      <Fastenings>...</Fastenings>
      <UpperWindow>...</UpperWindow>
    </Door>
  </door>
</...>
```

Compulsory is that all elements are precisely stated only once. These doors always have an upper window. The number for doors are unrestricted.

A.7.6 executor

The ‘executing’ role (RoleType) associated with a particular transaction.

EXAMPLE

```
<TransactionType id=”TenderTraject” >
  ...
  <executor>
    <RoleType id=”Client” >
      ...
    </RoleType>
  </executor>
  ...
</TransactionType>
```

A.7.7 group

The GroupType that is associated with a message within a particular transaction.

EXAMPLE

```
<MessageInTransactionType id=”...” >
  ...
  <message>
    <MessageType id=”M” >
      ...
    </MessageType>
  </message>
  <group>
    <GroupType id=”G” >
      ...
    </GroupType>
  </group>
  <transaction>
    <TransactionType id=”T” >
      ...
    </TransactionType>
  </transaction>
```

```
  ...
</MessageInTransactionType>
```

Message *M* within transaction *T* belongs to group *G* (there can be more messages *M* within transaction *T* that belong to the same or another group).

A.7.8 initiator

The 'initiating' role (RoleType) which belongs to a particular transaction.

EXAMPLE

```
<TransactionType id="TenderTraject" >
  ...
  <initiator>
    <RoleType id="Performer" >
      ...
    </RoleType>
  </initiator>
  ...
</TransactionType>
```

A.7.9 message

The message within a MessageInTransactionType.

EXAMPLE

```
<MessageInTransactionType id="..." >
  ...
  <message>
    <MessageType id="..." >
      ...
    </MessageType>
  </message>
  ...
</MessageInTransactionType
```

A.7.10 previous

A MessageInTransactionType may enforce that a previous message within a particular transaction should be executed (this previous MessageInTransactionType need not belong to the same TransactionType).

EXAMPLE

```
<MessageInTransactionType id="..." >
  ...
  <previous>
    <MessageInTransactionType id="..." >
      ...
      <message>
        <MessageType id="Tender" >
          ...
        </MessageType>
      </message>
      ...
    </MessageInTransactionType>
  </previous>
  ...
  <message>
    <MessageType id="TenderAcceptance" >
      ...
    </MessageType>
  </message>
  ...
</MessageInTransactionType>subTransactions
```

Transactions that could operate within this transaction.

EXAMPLE

```
<TransactionType id="AcquireWork" >
  ...
  <subTransactions>
    <TransactionType id="AcquisitionTrack" >
      ...
    </TransactionType>
```

```
    <TransactionType id=”TenderTrack” >
      ...
    </TransactionType>
  </subTransactions>
</TransactionType>
```

The TransactionType *AcquireWork* consists of the TransactionType's *AcquisitionTrack* and *TenderTraject*.

A.7.11 simpleElements

Set of simple element types belonging to this complex element type.

EXAMPLE

```
<ComplexElementType id=”Door” >
  ...
  <simpleElements>
    <SimpleElementType id=”Doorleaf” >
      ...
    </SimpleElementType>
    <SimpleElementType id=”HingesAndLocks” >
      ...
    </SimpleElementType>
    <SimpleElementType id=”UpperWindow” >
      ...
    </SimpleElementType>
  </simpleElements>
</ComplexElementType>
```

At message level, this could lead to:

```
<... id=”...” >
  ...
  <door>
    <Door>
      <Doorleaf>...</Doorleaf>
      < HingesAndLocks >...</ HingesAndLocks >
      < UpperWindow >...</ UpperWindow >
    </Door>
    ...
    <Door>
      < Doorleaf >...</ Doorleaf >
      < HingesAndLocks >...</ HingesAndLocks >
      < UpperWindow >...</ UpperWindow >
    </Door>
  </door>
</...>
```

A.7.12 subTransactions

Transactions that can be started from within this transaction.

EXAMPLE

```
<TransactionType id=”AcquisitionOfWork” >
  ...
  <subTransactions>
    <TransactionType id=”AcquisitionTraject” >
      ...
    </TransactionType>
    <TransactionType id=”TenderTraject” >
      ...
    </TransactionType>
  </subTransactions>
</TransactionType>
```

A.7.13 transaction

A transaction within a MessageInTransactionType.

EXAMPLE

```
<MessageInTransactionType id=”...” >
  ...
  <transaction>
    <TransactionType id=”...” >
      ...
    </TransactionType>
```

```
  </transaction>
  ...
</MessageInTransactionType>
```

A.7.14 transactionPhase

A TransactionPhase for a specific MessageType within a specific TransactionType.

EXAMPLE

```
<MessageInTransactionType id="..." >
  ...
  <message>
    <MessageType id="M" >
      ...
    </MessageType>
  </message>
  <transaction>
    <TransactionType id="T" >
      ...
    </TransactionType>
  </transaction>
  ...
  <transactionPhase>
    <TransactionPhaseType id="TP" >
      ...
    </TransactionPhaseType>
  </transactionPhase>
  ...
</MessageInTransactionType>
```

Where a MessageType *M* within a specific TransactionType *T*, determines TransactionPhaseType *TP.*

A.7.15 userDefinedType

Reference to a UserDefinedType, specifies the form of the SimpleElementType.

EXAMPLE

```
<SimpleElementType id="Height" >
  ...
  <userDefinedType>
    <UserDefinedType id="..." >
      ...
      <baseType>INTEGER</baseType>
      ...
    </UserDefinedType>
  </userDefinedType>
</SimpleElementType>
```

The SimpleElementType *Height* specifies an integer for every occurrence (possibly with a restriction as xsdRestriction).

Annex B
(normative)

Templates definition

B.1 Introduction

Once a valid interaction framework is available, an interaction schema file that includes rules for the actual communication is needed. Rules that are independent of the actual communication are included in the templates file. In this annex, the templates definition in EXPRESS format is given. Also, descriptions are given of Element types, attributes, elements, and references.

B.2 Templates definition

SCHEMA ISO 29481_Part_2B;

ENTITY GroupTemplate;

name: STRING;

description: STRING;

creationDate: DATETIME;

startDate: DATETIME;

endDate: DATETIME;

state: STRING;

dateLaMu: DATETIME;

userLaMu: STRING;

versionNo: STRING;

transaction: TransactionTemplate;

END_ENTITY;

ENTITY AppendixGroup;

state: STRING;

dateLaMu: DATETIME;

userLaMu: STRING;

group: GroupTemplate;

END_ENTITY;

ENTITY AppendixTemplate;

name: STRING;

fileLocation: STRING;

fileType: STRING;

fileVersion: STRING;

documentIdentification: STRING;

documentVersion: STRING;

documentReference: STRING;

objectCode: OPTIONAL STRING;

startDate: DATETIME;

endDate: DATETIME;

state: STRING;

dateLaMu: DATETIME;

userLaMu: STRING;

language: OPTIONAL STRING;

message: MessageTemplate;

appendixGroup: AppendixGroup;

template: ComplexElementTemplate;

END_ENTITY;

ENTITY MessageTemplate;

identification: STRING;

dateSend: DATETIME;

dateRead: DATETIME;

state: STRING;

dateLaMu: DATETIME;

userLaMu: STRING;

initiatingTransactionMessageID: OPTIONAL STRING;

initiatorToExecutor: BOOLEAN;

messageInTransaction: MessageInTransactionTemplate;

transaction: TransactionTemplate;

template: ComplexElementTemplate;

END_ENTITY;

ENTITY MessageInTransactionTemplate;

identification: STRING;

dateSend: DATETIME;

dateRead: DATETIME;

state: STRING;

dateLaMu: DATETIME;

userLaMu: STRING;

END_ENTITY;

ENTITY TransactionTemplate;

number: INTEGER;

name: STRING;

description: STRING;

startDate: DATETIME;

endDate: DATETIME;

state: STRING;

dateLaMu: DATETIME;

userLaMu: STRING;

result: OPTIONAL STRING;

initiator: PersonInRole;

executor: PersonInRole;

project: ProjectTypeInstance;

END_ENTITY;

ENTITY TransactionPhaseTemplate;

name: STRING;

description: STRING;

dateReached: DATETIME;

state: STRING;

dateLaMu: DATETIME;

userLaMu: STRING;

transaction: TransactionTemplate;

END_ENTITY;

ENTITY PersonTemplate;

userName: STRING;

name: STRING;

state: STRING;

dateLaMu: DATETIME;

userLaMu: STRING;

template: ComplexElementTemplate;

END_ENTITY;

ENTITY OrganisationTemplate;

name: STRING;

abbreviation: STRING;

state: STRING;

dateLaMu: DATETIME;

userLaMu: STRING;

contactPerson: PersonTemplate;

template: ComplexElementTemplate;

END_ENTITY;

ENTITY PersonInRole;

state: STRING;

dateLaMu: DATETIME;

userLaMu: STRING;

successor: OPTIONAL PersonInRole;

substituting: OPTIONAL PersonInRole;

contactPerson: PersonTemplate;

organization: OrganisationTemplate;

role: RoleTemplate;

END_ENTITY;

ENTITY RoleTemplate;

name: STRING;

description: STRING;

state: STRING;

dateLaMu: DATETIME;

userLaMu: STRING;

category: OPTIONAL STRING;

END_ENTITY;

ENTITY ProjectTypeInstance;

name: STRING;

description: STRING;

startDate: DATETIME;

endDate: DATETIME;

state: STRING;

dateLaMu: DATETIME;

userLaMu: STRING;

template: ComplexElementTemplate;

END_ENTITY;

ENTITY ComplexElementTemplate;

template: SimpleElementVirtual;

END_ENTITY;

END_SCHEMA;

B.3 Templates

B.3.1 AppendixGroup

Attributes: id

Elements: state, dateLaMu, userLaMu

References: group

The mapping table for the n:m relations between appendices and groups.

EXAMPLE Example at message level:

```
<AppendixGroup id="AppendixGroup_1" >
  <state>active</state>
  <dateLaMu>2008-02-04T00:00:00Z</dateLaMu>
  <userLaMu>bapa</userLaMu>
  <group>
    <StandardGroupType id="..." >
      ...
    </StandardGroupType>
  </group>
</AppendixGroup>
```

Associated part of the interaction framework:

```
<GroupType id="StandardGroupType" >
  <description>Standard group</description>
  <startDate>2007-12-20T00:00:00Z</startDate>
  <endDate>2008-12-31T00:00:00Z</endDate>
  <state>active</state>
  <dateLaMu>2007-12-20T00:00:00Z</dateLaMu>
  <userLaMu>bapa</userLaMu>
</GroupType>
```

B.3.2 AppendixTemplate

Attributes: id

Elements: name, fileLocation, fileType, fileVersion, documentIdentification, documentVersion, documentReference, objectCode, startDate, endDate, state, dateLaMu, userLaMu, language

References: appendixGroup, message

Registration of the attachement files.

EXAMPLE Example at message level:

```
<Appendix id=”ExampleDocument” >
  <name>Voorbeeld</name>
  <fileLocation>\\srv-bouw\Public\project\docs\msword\</fileLocation>
  <fileType>application/msword</fileType>
  <fileVersion>2007</fileVersion>
  <documentIdentification>345899</documentIdentification>
  <documentVersion>1</documentVersion>
  <documentReference>FG783990</documentReference>
  <startDate>2008-02-04T00:00:00Z</startDate>
  <endDate>2008-12-31T00:00:00Z</endDate>
  <state>active</state>
  <dateLaMu>2008-02-04T00:00:00Z</dateLaMu>
  <userLaMu>bapa</userLaMu>
  <language>EN</language>
  <appendixGroup>
    <AppendixGroup id=”...” >
      ...
    </AppendixGroup>
  </appendixGroup>
</Appendix>
```

Associated part of the interaction framework:

```
  <description>Standard appendix definition (no user defined data fields)</description>
  <startDate>2008-02-04T00:00:00Z</startDate>
  <endDate>2008-12-31T00:00:00Z</endDate>
  <state>active</state>
  <dateLaMu>2008-02-04T00:00:00Z</dateLaMu>
  <userLaMu>bapa</userLaMu>
  <language>NL</language>
</AppendixType>
```

B.3.3 ComplexElementTemplate

Attributes: id

B.3.4 GroupTemplate

Attributes: id

Elements: name, description, creationDate, startDate, endDate, state, dateLaMu, userLaMu, versionNo

References: transaction

The grouping of appendices of a message to recover the associated documents.

EXAMPLE Example at message level:

```
<StandardGroupType id=”MenuBackgrounds” >
  <name>Menu Pictures</name>
  <description>A set of background pictures to decorate the menu</description>
  <creationDate>2008-02-04T00:00:00Z</creationDate>
  <startDate>2008-02-04T00:00:00Z</startDate>
  <endDate>2008-12-31T00:00:00Z</endDate>
  <state></state>
  <dateLaMu>2008-02-04T00:00:00Z</dateLaMu>
  <userLaMu>bapa</userLaMu>
  <versionNo>1</versionNO>
  <transaction>
    <MenuObtainingTransaction id=”...” >
      ...
    </MenuObtainingTransaction>
  </transaction>
</StandardGroupType
```

Associated part of the interaction framework:

```
<GroupType id=”StandardGroupType” >
  <description>Standard group</description>
  <startDate>2007-12-20T00:00:00Z</startDate>
  <endDate>2008-12-31T00:00:00Z</endDate>
  <state>active</state>
  <dateLaMu>2007-12-20T00:00:00Z</dateLaMu>
  <userLaMu>bapa</userLaMu>
</GroupType>
<TransactionType id=”MenuObtainingTransaction” >
  <description>The transaction to obtain the proper menu</description>
  <startDate>2008-01-23T00:00:00Z</startDate>
```

```
  <endDate>2008-12-31T00:00:00Z</endDate>
  <state>active</state>
  <dateLaMu>2008-01-23T00:00:00Z</dateLaMu>
  <userLaMu>bapa</userLaMu>
  <initiator>
    <RoleTypeRef idref="Consumer"/ >
  </initiator>
  <executor>
  <RoleTypeRef idref="Employee"/ >
  </executor>
</TransactionType>
```

B.3.5 MessageInTransactionTemplate

Attributes: id

Elements: identification, dateSend, dateRead, state, dateLaMu, userLaMu, initiatorToExecutor

Holds the actual MessageInTransactionType within the message to retrieve the workflow state of the transaction.

B.3.6 MessageTemplate

Attributes: id

Elements: identification, dateSend, dateRead, state, dateLaMu, userLaMu, initiatingTransactionMessageID, initiatorToExecutor

References: transaction, messageInTransaction, template

An instance of the MessageType. This entity carries the actual information-exchange between OrganisationTemplate's (organisations).

EXAMPLE Example at message level:

```
<HandOutOfMenuMessage id="a002" >
  <identification>id a002</identification>
  <dateSend>2008-01-23T00:00:00Z</dateSend>
  <dateRead>2008-01-23T00:00:00Z</dateRead>
  <state>active</state>
  <dateLaMu>2008-01-23T00:00:00Z</dateLaMu>
  <userLaMu>bapa</userLaMu>
  <initiatingTransactionMessageID>a009</initiatingTransactionMessageID>
  <initiatorToExecutor>false</initiatorToExecutor>
  <messageInTransaction>
    <MessageInTransaction12Ref idref="BiT001"/ >
  </messageInTransaction>
  <transaction>
    <MenuObtainingTransaction id="..." >
      ...
    </MenuObtainingTransaction>
  </transaction>
  <menu>
    <Menu id="..." >
      ...
    </Menu>
    ...
    <Menu id="..." >
      ...
    </Menu>
  </menu>
</HandOutOfMenuMessage>
```

Associated part of the interaction framework:

```
<TransactionType id="MenuObtainingTransaction" >
  <description>The transaction to obtain the proper menu</description>
  <startDate>2008-01-23T00:00:00Z</startDate>
  <endDate>2008-12-31T00:00:00Z</endDate>
  <state>active</state>
  <dateLaMu>2008-01-23T00:00:00Z</dateLaMu>
  <userLaMu>bapa</userLaMu>
  <initiator>
    <RoleTypeRef idref="Consumer"/ >
  </initiator>
```

```
  <executor>
    <RoleTypeRef idref="Employee"/ >
  </executor>
</TransactionType>
<MessageType id="HandOutOfMenuMessage" >
  <description>Message that contains the menu.</description>
  <startDate>2008-01-23T00:00:00Z</startDate>
  <endDate>2008-12-31T00:00:00Z</endDate>
  <state>active</state>
  <dateLaMu>2008-01-23T00:00:00Z</dateLaMu>
  <userLaMu>bapa</userLaMu>
  <complexElements>
    <ComplexElementTypeRef idref="Menu"/ >
  </complexElements>
</MessageType>
<ComplexElementType id="Menu" >
  <description>Available menu items</description>
  <startDate>2008-01-23T00:00:00Z</startDate>
  <endDate>2008-12-31T00:00:00Z</endDate>
  <state>active</state>
  <dateLaMu>2008-01-23T00:00:00Z</dateLaMu>
  <userLaMu>bapa</userLaMu>
  <elements>
    <SimpleElementTypeRef idref="MenuItems"/ >
  </elements>
</ComplexElementType>
```

B.3.7 OrganisationTemplate

Attributes: id

Elements: name, abbreviation, state, dateLaMu, userLaMu

References: contactPerson, template

The organization that participates in the project by initiating or executing a TransactionTemplate (transaction).

EXAMPLE Example at message level:

```
<StandardOrganisationType id="TNO" >
  <name>Netherlands organisation for Applied Scientific Research<name>
  <abbreviation>TNO</abbreviation>
  <state>active</state>
  <dateLaMu>2008-01-23T00:00:00Z</dateLaMu>
  <userLaMu>bapa</userLaMu>
  <contactPerson>
    <StandardPersonType id="..." >
      ...
    </StandardPersonType>
  </contactPerson>
</StandardOrganisationType>
```

Associated part of the interaction framework:

```
<PersonType id="StandardPersonType" >
  <description>Standard person type</description>
  <startDate>2008-01-23T00:00:00Z</startDate>
  <endDate>2008-12-31T00:00:00Z</endDate>
  <state>active</state>
  <dateLaMu>2008-01-23T00:00:00Z</dateLaMu>
  <userLaMu>bapa</userLaMu>
</PersonType>
<OrganisationType id="StandardOrganisationType" >
  <description>Standard organisation type</description>
  <startDate>2008-01-23T00:00:00Z</startDate>
  <endDate>2008-12-31T00:00:00Z</endDate>
  <state>active</state>
  <dateLaMu>2008-01-23T00:00:00Z</dateLaMu>
  <userLaMu>bapa</userLaMu>
</OrganisationType>
```

B.3.8 PersonInRole

Attributes: id

Elements: state, dateLaMu, userLaMu

References: organization, contactPerson, role, substituting, successor

A person fulfilling a particular role for a certain organization.

EXAMPLE Example at message level:

```
<PersonInRole id="JohnAsCustomer" >
  <state>active</state>
  <dateLaMu>2008-01-23T00:00:00Z</dateLaMu>
  <userLaMu>bapa</userLaMu>
  <contactPerson>
    <StandardPersonType id="..." >
      ...
    </StandardPersonType>
  </contactPerson>
  <organisation>
    <StandardOrganisationType id="..." >
      ...
    </StandardOrganisationType>
  </organisation>
  <role>
    <Consumer idref="..." >
      ...
    </Consumer>
  </role>
</PersonInRole>
```

Associated part of the interaction framework:

```
<PersonType id="StandardPersonType" >
  <description>Standard person type</description>
  <startDate>2008-01-23T00:00:00Z</startDate>
  <endDate>2008-12-31T00:00:00Z</endDate>
  <state>active</state>
  <dateLaMu>2008-01-23T00:00:00Z</dateLaMu>
  <userLaMu>bapa</userLaMu>
</PersonType>
<OrganisationType id="StandardOrganisationType" >
  <description>Standard organisation type</description>
  <startDate>2008-01-23T00:00:00Z</startDate>
  <endDate>2008-12-31T00:00:00Z</endDate>
  <state>active</state>
  <dateLaMu>2008-01-23T00:00:00Z</dateLaMu>
  <userLaMu>bapa</userLaMu>
</OrganisationType>
<RoleType id="Consumer" >
  <description>Consuming person</description>
  <startDate>2008-01-23T00:00:00Z</startDate>
  <endDate>2008-12-31T00:00:00Z</endDate>
  <state>active</state>
  <dateLaMu>2008-01-23T00:00:00Z</dateLaMu>
  <userLaMu>bapa</userLaMu>
</RoleType>
```

B.3.9 PersonTemplate

Attributes: id

Elements: userName, name, state, dateLaMu, userLaMu

The person that participates in the project by fulfilling a certain role or as contact person of a particular organization.

EXAMPLE Example at message level:

```
<StandardPersonType id="PBonsma" >
  <userName>bapa<userName>
  <name>Peter Bonsma<name>
  <state>active</state>
  <dateLaMu>2008-02-04T00:00:00Z</dateLaMu>
  <userLaMu>bapa</userLaMu>
</StandardPersonType>
```

Associated part of the interaction framework:

```
<PersonType id="StandardPersonType" >
  <description>Standard person type</description>
  <startDate>2008-01-23T00:00:00Z</startDate>
```

```
  <endDate>2008-12-31T00:00:00Z</endDate>
  <state>active</state>
  <dateLaMu>2008-01-23T00:00:00Z</dateLaMu>
  <userLaMu>bapa</userLaMu>
</PersonType>
```

B.3.10 ProjectTemplate

Attributes: id

Elements: name, description, startDate, endDate, state, dateLaMu, userLaMu

The project to be accommodated by this framework.

EXAMPLE Example at message level:

```
<StandardProjectType id="IDM2" >
  <name>The project IDM2</name>
  <description>Formalising of the IDM2 Systematics</description>
  <startDate>2008-02-04T00:00:00Z</startDate>
  <endDate>2008-12-31T00:00:00Z</endDate>
  <state>active</state>
  <dateLaMu>2008-02-04T00:00:00Z</dateLaMu>
  <userLaMu>bapa</userLaMu>
</StandardProjectType>
```

Associated part of the interaction framework:

```
<ProjectType id="StandardProjectType" >
  <description>Standard project type</description>
  <startDate>2008-02-04T00:00:00Z</startDate>
  <endDate>2008-12-31T00:00:00Z</endDate>
  <state>active</state>
  <dateLaMu>2008-02-04T00:00:00Z</dateLaMu>
  <userLaMu>bapa</userLaMu>
</ProjectType>
```

B.3.11 RoleTemplate

Attributes: id

Elements: name, description, state, dateLaMu, userLaMu, category

The role on behalf of an organization fulfilled by a PersonTemplate (person).

EXAMPLE Example at message level.

```
<Consumer id="Customer" >
  <name>Role as customer</name>
  <description>The role as customer</description>
  <state>active</state>
  dateLaMu>2008-01-23T00:00:00Z</dateLaMu>
  <userLaMu>bapa</userLaMu>
</Consumer>
```

Associated part of the interaction framework:

```
<RoleType id="Consumer" >
  <description> Consuming person</description>
  <startDate>2008-01-23T00:00:00Z</startDate>
  <endDate>20083-12-31T00:00:00Z</endDate>
  <state>active</state>
  <dateLaMu>2008-01-23T00:00:00Z</dateLaMu>
  <userLaMu>bapa</userLaMu>
</RoleType>
```

B.3.12 TransactionPhaseTemplate

Attributes: id

Elements: name, description, dateReached, state, dateLaMu, userLaMu

References: transaction

The actual phase of a transaction.

EXAMPLE Example at message level:

```
<WaitForMenu id=”tp003” >
  <name>...</name>
  <description>Transaction Phase ...</description>
  <dateReached>2008-02-04T00:00:00Z</dateReached>
  <state>active</state>
  <dateLaMu>2008-02-04T00:00:00]]</dateLaMu>
  <userLaMu>bapa</userLaMu>
</WaitForMenu>
```

Associated part of the interaction framework:

```
<TransactionType id=”ObtainMenuTransaction” >
  <description>The transaction to obtain the proper menu</description>
  <startDate>2008-01-23T00:00:00Z</startDate>
  <endDate>2008-12-31T00:00:00Z</endDate>
  <state>active</state>
  <dateLaMu>2008-01-23T00:00:00Z</dateLaMu>
  <userLaMu>bapa</userLaMu>
  <initiator>
    <RoleTypeRef idref=”Consumer”/ >
  </initiator>
  <executor>
    <RoleTypeRef idref=”Employee”/ >
  </executor>
</TransactionType>
<TransactionPhaseType id=”WaitForMenu” >
  <description>Menukaart gevraagd maar nog niet gegeven</description>
  <startDate>2008-01-23T00:00:00Z</startDate>
  <endDate>2008-12-31T00:00:00Z</endDate>
  <state>active</state>
  <dateLaMu>2008-01-23T00:00:00Z</dateLaMu>
  <userLaMu>bapa</userLaMu>
</TransactionPhaseType>
```

B.3.13 TransactionTemplate

Attributes: id

Elements: number, name, description, startDate, endDate, state, dateLaMu, userLaMu, result

References: initiator, executor

The transaction that enables MessageTemplates (messages) to be exchanged to execute certain tasks.

EXAMPLE Example at message level:

```
<ObtainMenuTransaction id=”TheTransaction” >
  <number>001</number>
  <name>...</name>
  <description>...</description>
  <startDate>2008-01-23T00:00:00Z</startDate>
  <endDate>2008-01-23T00:00:00Z</endDate>
  <state>active</state>
  <dateLaMu>2008-01-23T00:00:00Z</dateLaMu>
  <userLaMu>bapa</userLaMu>
  <initiator>
    <PersonInRole id=”...” >
      ...
    </PersonInRole>
  </initiator>
  <executor>
    <PersonInRole id=”...” >
      ...
    </PersonInRole>
  </executor>
</ObtainMenuTransaction>
```

Associated part of the interaction framework:

```
<TransactionType id=”ObtainMenuTransaction” >
  <description>The transaction to obtain the proper menu</description>
  <startDate>2008-01-23T00:00:00Z</startDate>
  <endDate>2008-12-31T00:00:00Z</endDate>
  <state>active</state>
```

```
  <dateLaMu>2008-01-23T00:00:00Z</dateLaMu>
  <userLaMu>bapa</userLaMu>
  <initiator>
    <RoleTypeRef idref="Consumer"/ >
  </initiator>
  <executor>
    <RoleTypeRef idref="Employee"/ >
  </executor>
</TransactionType>
```

B.4 Attributes

B.4.1 id

Unique code to identify an object instance within a message.

NOTE The ID value should be unique within the XML document. Consequently, a simple serial number would already satisfy this condition. As a result, there is no convincing reason to limit the ID values to GUIDs only (although most implementations do apply GUIDs). Mark that the XML (http://www.w3.org/XML/) standard requires an ID to start with a letter or '_'.

B.5 Elements

B.5.1 Abbreviation

Abbreviation for the name of the organization. This item will be used as a prefix to the transaction number for transaction identification.

NOTE Element **abbreviation**. Abbreviation holds the short name of an organization. One of its applications is to mark transactions (in combination with a sequence number) that were started by this organization to be helpful for informal communication between human participants.

B.5.2 category

Category to classify this object instance.

EXAMPLE Example at message level:

```
<... id="..." >
  ...
  <category>...</category>
  ...
</...>
```

B.5.3 creationDate

Creation date of a specific object instance in this message.

EXAMPLE Example at message level:

```
<... id="..." >
  ...
  <creationDate>2007-12-03T00:00:00Z</creationDate>
  ...
</...>
```

B.5.4 dateLaMu

Last mutation date.

EXAMPLE Example at message level:

```
<... id="..." >
  ...
  <dateLaMu>2007-12-03T00:00:00Z</dateLaMu>
  ...
</...>
```

B.5.5 dateReached

Date “dateReached” is a date-time attribute of “TransactionPhaseTemplate”. It is the date-time when a particular phase was reached.

EXAMPLE Example at message level:

```
<... id=”...” >
  ...
  <dateReached>2008-02-04T00:00:00Z</dateReached>
  ...
</...>
```

B.5.6 dateRead

Date of reading this message.

EXAMPLE Example at message level:

```
<... id=”...” >
  ...
  <dateRead>2007-12-03T00:00:00Z</dateRead>
  ...
</...>
```

B.5.7 dateSend

Date of sending this message.

EXAMPLE Example at message level:

```
<... id=”...” >
  ...
  <dateSend>2007-12-03T00:00:00Z</dateSend>
    ...
</...>
```

B.5.8 description

Description of this object instance.

EXAMPLE Example at message level:

```
<Projectexecutor id=”TNO” >
  ...
  <description>...</description>
  ...
</Projectexecutor>
```

B.5.9 documentIdentification

Unique number or reference of a document or file to identify the document.

NOTE This part of ISO 29481 gives the freedom to distinguish between document versions and file versions (see B.5.13).

B.5.10 documentReference

Reference to identify a file or document.

B.5.11 documentVersion

Version of a document or file.

B.5.12 endDate

Expiration date of a specific object instance in this message.

EXAMPLE Example at message level:

```
<... id="..." >
  ...
  <endDate>2007-12-03</endDate>
  ...
</...>
```

B.5.13 fileLocation

Location of the file, preferably an internet address or shared server path.

EXAMPLE Example at message level:

```
<... id="..." >
  ...
  <fileLocation>\\srv-path\Public\project-x\docs\</fileLocation>
  ...
</...>
```

NOTE This part of ISO 29481 gives the freedom to distinguish between document versions and file versions (see B.5.9).

B.5.14 fileType

Type of the file, preferrably the MIME (Multipurpose Internet Mail Extensions) type.

EXAMPLE Example at message level:

```
<... id="..." >
  ...
  <fileType>text/plain</fileType>
  ...
</...>
```

B.5.15 fileVersion

Version of the file, an increasing integer number or the date.

EXAMPLE Example at message level:

```
<... id="..." >
  ...
  <fileVersion>20071202</fileVersion>
  ...
</...>
```

B.5.16 identification

Human-oriented identification of this message.

EXAMPLE Example at message level:

```
<AssignmentConfirmation id="X" >
  <identification>This message contains the assignment X related to part Y</identification>
  ...
</AssignmentConfirmation>
```

B.5.17 initiatingTransactionMessageID

Reference to the message belonging to a secondary transaction that initiated this message.

B.5.18 initiatorToExecutor

The communication direction of this specific message.

EXAMPLE Example at message level:

```
<ExampleMessage id="..." >
  ...
  <initiatorToExecutor>false</initiatorToExecutor>
  ...
  <transaction>
    <ExampleTransaction id="..." >
      ...
      <initiator>
        <PersonInRole id="A" >
```

```
          ...
        </PersonInRole>
      </initiator>
      <executor>
        <PersonInRole id="B" >
          ...
        </PersonInRole>
      </executor>
    </ExampleTransaction>
  </transaction>
  ...
</ExampleMessage>
```

In this example, *ExampleMessage* will travel from *B* (executor) to *A* (initiator).

B.5.19 language

The language of the content of the appendix.

EXAMPLE Example at message level:

```
<... id="..." >
  ...
  <language>EN</language>
  ...
</...>
```

B.5.20 name

naming (STRING)

B.5.21 number

Transaction number.

NOTE Element **number**. The sequence number of a transaction (see also previous item).

B.5.22 objectCode

Possibility to relate to an externam index. For example, a workbreakdown structure, workpackages, or the building specifications.

B.5.23 result

Result of this transaction.

B.5.24 startDate

Commencing date of a specific object instance in this message.

EXAMPLE Example at message level:

```
<... id="..." >
  ...
  <startDate>2007-12-03</startDate>
  ...
</...>
```

B.5.25 state

The current state of this object instance.

EXAMPLE Example at message level:

```
<... id="..." >
  ...
  <state>active</state>
  ...
</...>
```

B.5.26 userLaMu

User that performed the last modification (MINS: this is no reference to a PersonTypeInstance instantiation).

EXAMPLE Example at message level:

```
<... id="..." >
  ...
  <userLaMu>Peter Bonsma</userLaMu>
  ...
</...>
```

B.5.27 userName

Initials of the user, for example a trigram.

EXAMPLE Example at message level:

```
<... id="..." >
  ...
  <userName>BAP</userName>
  ...
</...>
```

B.5.28 versionNo

Version number of the object instance connected to this project; after a modification to this object instance, this field should also be updated.

EXAMPLE Example at message level:

```
<... id="..." >
  ...
  <versionNo>23</versionNo>
  ...
</...>
```

B.6 References

B.6.1 appendixGroup

A subdivision to identify a certain appendix.

EXAMPLE Example at message level:

```
<Appendix id="..." >
  ...
  <appendixGroup>
    <AppendixGroup id="..." >
      ...
    </AppendixGroup>
  </appendixGroup>
  ...
</Appendix>
```

The associated framework is considered to have defined an AppendixType *Appendix*.

B.6.2 contactPerson

The person connected to a PersonInRole object or connected to a specific organization.

EXAMPLE Example at message level:

```
<Organisation id="..." >
  ...
  <contactPerson>
    <Person id="..." >
      ...
    </Person>
  </contactPerson>
</Organisation>
```

The associated framework is considered to have defined an OrganisationType *Organization* and a PersonType *Person*.

B.6.3 executor

The PersonInRole that will act as the executor of this transaction.

B.6.4 group

The general group to assemble a set of appendices.

EXAMPLE Example at message level:

```
<AppendixGroup id="..." >
  ...
  <group>
    <Group id="..." >
      ...
    </Group>
  </group>
  ...
</AppendixGroup>
```

The associated framework is considered to have defined a GroupType *Group*.

B.6.5 initiator

The PersonInRole that will act as the initiator of this transaction.

B.6.6 message

The message that contains a specific appendix.

EXAMPLE Example at message level:

```
<Appendix id="..." >
  ...
  <message>
    <Message id="..." >
      ...
    </Message>
  </message>
  ...
</Appendix>
```

The associated framework is considered to have defined an AppendixType *Appendix* and a MessageType *Message*.

B.6.7 messageInTransaction

Reference to the position of this message in the flow of the transaction.

B.6.8 organization

The organization belonging to a PersonInRole object.

EXAMPLE Example at message level:

```
<PersonInRole id="..." >
  ...
  <organisation>
    <Organisation id="..." >
      ...
    </Organisation>
  </organisation>
  ...
</PersonInRole>
```

The associated framework is considered to have defined an OrganisationType *Organization*.

B.6.9 role

Reference to a role on behalf of an organization, fulfilled by a PersonTemplate (person).

B.6.10 substituting

PersonInRole on behalf of this person is authorized to send messages.

NOTE Reference **substituting**. A reference to the formal PersonInRole for this transaction to enable another person to act in the absence of the formal PersonInRole fulfilling person. Both should refer to the same role type.

B.6.11 successor

Successor of another person in a certain role.

B.6.12 transaction

The transaction that holds a specific group, message, or transactionphase.

EXAMPLE Example at message level:

```
<Message id="..." >
  ...
  <transaction>
    <Transaction id="..." >
      ...
    </Transaction>
  </transaction>
  ...
</Message>
```

The associated framework is considered to have defined a MessageType *Message* and a TransactionType *Transaction*.

Annex C
(informative)

Example interaction map of a simplified design office

C.1 General

In our example of creating an interaction framework, the scope of interaction has been restricted to the simplified interaction within a design office (see the example in Clause 4). The interaction is defined in the framework by the declaration of

— roles,

— transactions,

— message in transactions,

— the order of messages in transaction,

— data elements in messages.

C.2 Roles and transactions

To illustrate the scope of interaction in our example, the communicative actions are depicted in an 'interaction map', showing the roles linked by transactions.

Roles:

— CR_1: Client;

— R_1: Project leading;

— R_2: System engineering;

— R_3: 3D engineering;

— R_4: Cost engineering.

The interactions can be summarized in a transaction-role table.

Table C.1 — Transaction-role table of a simplified design office

Transaction type	Initiating role	Executing role
T_1 Deliver design	CR_1 Client	R_1 Project leading
T_2 Deliver system specification	R_1 Project leading	R_2 System engineering
T_3 Deliver 3D model	R_1 Project leading	R_3 3D engineering
T_4 Deliver cost calculation	R_1 Project leading	R_4 Cost engineering

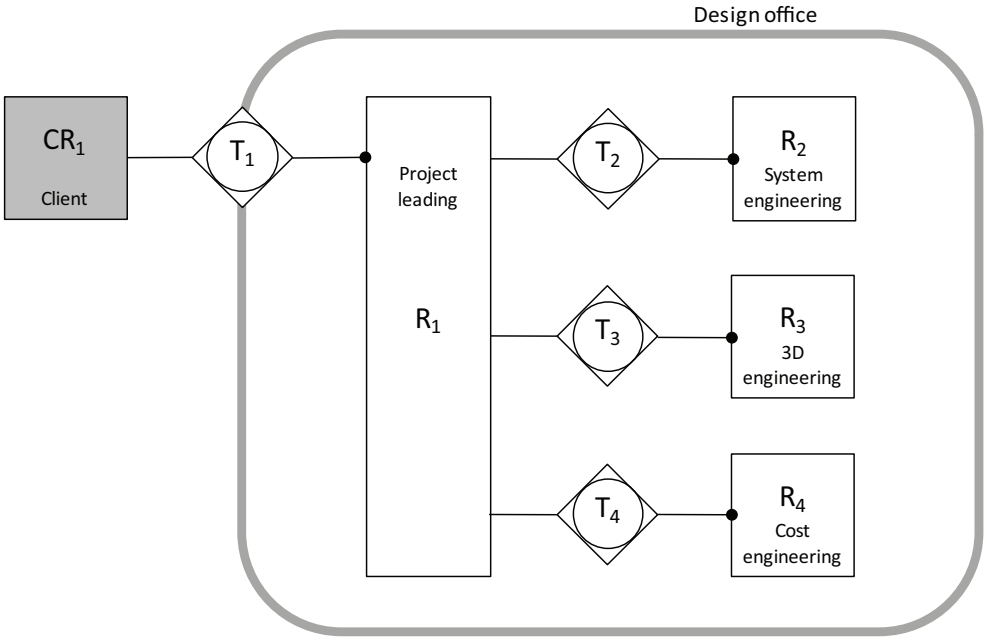

Figure C.1 — Interaction map, identifying the relevant role types and transaction types

C.2.1 RoleType

The roles are defined as 'RoleTypes' in the framework. Each RoleType can be defined in terms of

— an ID (identification of roleType),

  — [Should be a valid XML ID value (qualified name). There are certain restrictions: * Unique within the model, * Certain characters are not allowed ('/', '$', '#', '@', ...), * No spaces, * Should not start with a digit]

— description (specification of roleType),

— responsibilities (scope, task, support task, and feedback),

— metadata (last mutation, helpinfo, etc.).

The four roles of our example have been defined in the framework (see the example below in XML view):

```
    <RoleType id="CR1_Client" >
        <description>CR1: Client</description>
        <startDate>2010-10-29T00:00:00.000+02:00</startDate>
        <endDate>2099-10-29T00:00:00.000+02:00</endDate>
        <state>active</state>
        <dateLaMu>2010-10-29T00:00:00.000+02:00</dateLaMu>
        <userLaMu>Ronald</userLaMu>
        <language></language>
        <category></category>
        <helpInfo></helpInfo>
        <code></code>
        <responsibilityScope>Is responsible for the realisation of the project in terms of time,
quality and
          cost < /responsibilityScope>
        <responsibilityTask></responsibilityTask>
        <responsibilitySupportTask></responsibilitySupportTask>
        <responsibilityFeedback></responsibilityFeedback>
    </RoleType>
    <RoleType id="R1_Project_Leading" >
        <description>R1: Project Leading</description>
        <startDate>2010-10-29T00:00:00.000+02:00</startDate>
        <endDate>2099-10-29T00:00:00.000+02:00</endDate>
        <state>active</state>
        <dateLaMu>2010-10-29T00:00:00.000+02:00</dateLaMu>
        <userLaMu>Ronald</userLaMu>
```

```
    <language></language>
    <category></category>
    <helpInfo></helpInfo>
    <code></code>
    <responsibilityScope>Has to deliver the project according to the contract
      requirements</responsibilityScope>
    <responsibilityTask></responsibilityTask>
    <responsibilitySupportTask></responsibilitySupportTask>
    <responsibilityFeedback></responsibilityFeedback>
</RoleType>
<RoleType id="R2_System_Engineering" >
    <description>R2: System Engineering</description>
    <startDate>2010-10-29T00:00:00.000+02:00</startDate>
    <endDate>2099-10-29T00:00:00.000+02:00</endDate>
    <state>active</state>
    <dateLaMu>2010-10-29T00:00:00.000+02:00</dateLaMu>
    <userLaMu>Ronald</userLaMu>
    <language></language>
    <category></category>
    <helpInfo></helpInfo>
    <code></code>
    <responsibilityScope>Has to deliver a system specification according to contractual
      agreements</responsibilityScope>
    <responsibilityTask></responsibilityTask>
    <responsibilitySupportTask></responsibilitySupportTask>
    <responsibilityFeedback></responsibilityFeedback>
</RoleType>
<RoleType id="R3_3D_Engineering" >
    <description>R3: 3D Engineering</description>
    <startDate>2010-10-29T00:00:00.000+02:00</startDate>
    <endDate>2099-10-29T00:00:00.000+02:00</endDate>
    <state>active</state>
    <dateLaMu>2010-10-29T00:00:00.000+02:00</dateLaMu>
    <userLaMu>Ronald</userLaMu>
    <language></language>
    <category></category>
    <helpInfo></helpInfo>
    <code></code>
    <responsibilityScope>Has to deliver a 3D model according to contractual
       agreements</responsibilityScope>
    <responsibilityTask></responsibilityTask>
    <responsibilitySupportTask></responsibilitySupportTask>
    <responsibilityFeedback></responsibilityFeedback>
</RoleType>
<RoleType id="R4_Cost_Engineering" >
    <description>R4: Cost Engineering</description>
    <startDate>2010-10-29T00:00:00.000+02:00</startDate>
    <endDate>2099-10-29T00:00:00.000+02:00</endDate>
    <state>active</state>
    <dateLaMu>2010-10-29T00:00:00.000+02:00</dateLaMu>
    <userLaMu>Ronald</userLaMu>
    <language></language>
    <category></category>
    <helpInfo></helpInfo>
    <code></code>
    <responsibilityScope>Has to deliver a cost calculation according to contractual
       agreements</responsibilityScope>
    <responsibilityTask></responsibilityTask>
    <responsibilitySupportTask></responsibilitySupportTask>
    <responsibilityFeedback></responsibilityFeedback>
</RoleType>
```

C.3 Message in transaction

A transaction contains a set of messages that are exchanged for a particular purpose. The transaction also stipulates the participating roles, point in the life cycle, and the sequence in which messages should be delivered (if appropriate).

C.3.1 TransactionType

The transactions are defined as 'TransactionTypes' in the framework. Each TransactionType can be defined in terms of

— an ID (identification of TransactionType),

— description (specification of TransactionType),

— metadata (result of TransactionType, helpinfo, etc.),

— the RoleTypes that are involved in the TransactionType.

The four TransactionTypes in our example are depicted below in an XML view:

— T_1: Deliver design;

— T_2: Deliver system specification;

— T_3: Deliver 3D model;

— T_4: Deliver cost calculation.

```
<TransactionType id="T1" >
     <description>T1 Deliver design </description>
     <startDate>2010-10-29T00:00:00.000+02:00</startDate>
     <endDate>2099-10-29T00:00:00.000+02:00</endDate>
     <state>active</state>
     <dateLaMu>2010-10-29T00:00:00.000+02:00</dateLaMu>
     <userLaMu>Ronald</userLaMu>
     <language></language>
     <category></category>
     <helpInfo></helpInfo>
     <code></code>
     <result>Design is delivered</result>
     <initiator>
         <RoleTypeRef idref="CR1_Client"/ >
     </initiator>
     <executor>
         <RoleTypeRef idref="R1_Project_Leading"/ >
     </executor>
</TransactionType>
<TransactionType id="T2" >
     <description>T2 Deliver system specification</description>
     <startDate>2010-10-29T00:00:00.000+02:00</startDate>
     <endDate>2099-10-29T00:00:00.000+02:00</endDate>
     <state>active</state>
     <dateLaMu>2010-10-29T00:00:00.000+02:00</dateLaMu>
     <userLaMu>Ronald</userLaMu>
     <language></language>
     <category></category>
     <helpInfo></helpInfo>
     <code></code>
     <result>System specification is delivered</result>
     <initiator>
         <RoleTypeRef idref="R1_Project_Leading"/ >
     </initiator>
     <executor>
         <RoleTypeRef idref="R2_System_Engineering"/ >
     </executor>
</TransactionType>
<TransactionType id="T3" >
     <description>T3 Deliver 3D model</description>
     <startDate>2010-10-29T00:00:00.000+02:00</startDate>
     <endDate>2099-10-29T00:00:00.000+02:00</endDate>
     <state>active</state>
     <dateLaMu>2010-10-29T00:00:00.000+02:00</dateLaMu>
     <userLaMu>Ronald</userLaMu>
     <language></language>
     <category></category>
     <helpInfo></helpInfo>
     <code></code>
     <result>3D model is delivered</result>
     <initiator>
         <RoleTypeRef idref="R1_Project_Leading"/ >
     </initiator>
     <executor>
         <RoleTypeRef idref="R3_3D_Engineering"/ >
     </executor>
</TransactionType>
<TransactionType id="T4" >
     <description>T4 Deliver cost calculation</description>
     <startDate>2010-10-29T00:00:00.000+02:00</startDate>
     <endDate>2099-10-29T00:00:00.000+02:00</endDate>
```

```
        <state>active</state>
        <dateLaMu>2010-10-29T00:00:00.000+02:00</dateLaMu>
        <userLaMu>Ronald</userLaMu>
        <language></language>
        <category></category>
        <helpInfo></helpInfo>
        <code></code>
        <result>Cost calculation is delivered</result>
        <initiator>
            <RoleTypeRef idref="R1_Project_Leading"/ >
        </initiator>
        <executor>
            <RoleTypeRef idref="R4_Cost_Engineering"/ >
        </executor>
    </TransactionType>
```

IDM draws a distinction between a role that makes a request (the initiator) and the role that gives effect to that request (the executor). A transaction shall have only one initiating role and only one executing role. For instance, TransactionType 'T3 Deliver 3D model' can only be initiated by role R1 Project leading and executed by the 3D engineer in role R3.

By using transactions, the information transfer is brought in a process context.

C.3.2 TransactionPhaseType

To indicate the state of interaction within a transaction, phases can be related to the TransactionTypes. As a matter of fact, the TransactionPhaseTypes within an TransactionType represent the transaction pattern. The TransactionPhaseType definitions provide for an identification of the TransactionPhaseType and some metadata. The number of the phases and its description are not restricted.

In our example, the following TransactionPhaseTypes have been defined, which refer to Figure 2 in Clause 3:

— desired result;

— result requested;

— result promised;

— result produced;

— result stated;

— result accepted.

The example below shows in XML view the definition of a TransactionPhaseType:

```
    <TransactionPhaseType id="Desired_Result" >
        <description>Desired result</description>
        <startDate>2010-10-01T00:00:00.000+02:00</startDate>
        <endDate>2099-10-01T00:00:00.000+02:00</endDate>
        <state>active</state>
        <dateLaMu>2010-10-01T00:00:00.000+02:00</dateLaMu>
        <userLaMu>Ronald</userLaMu>
        <language></language>
        <category></category>
        <helpInfo></helpInfo>
        <code>-</code>
    </TransactionPhaseType>
    <TransactionPhaseType id="Result_Accepted" >
        <description>Result accepted</description>
        <startDate>2010-11-04T00:00:00.000+01:00</startDate>
        <endDate>2099-11-04T00:00:00.000+01:00</endDate>
        <state>active</state>
        <dateLaMu>2010-11-04T00:00:00.000+01:00</dateLaMu>
        <userLaMu>Ronald</userLaMu>
        <language></language>
        <category></category>
        <helpInfo></helpInfo>
        <code>-</code>
    </TransactionPhaseType>
    <TransactionPhaseType id="Result_Produced" >
        <description>Result produced</description>
        <startDate>2010-10-01T00:00:00.000+02:00</startDate>
```

```
        <endDate>2099-10-01T00:00:00.000+02:00</endDate>
        <state>active</state>
        <dateLaMu>2010-10-01T00:00:00.000+02:00</dateLaMu>
        <userLaMu>Ronald</userLaMu>
        <language></language>
        <category></category>
        <helpInfo></helpInfo>
        <code>-</code>
    </TransactionPhaseType>
    <TransactionPhaseType id="Result_Promised" >
        <description>Result promised</description>
        <startDate>2010-10-01T00:00:00.000+02:00</startDate>
        <endDate>2099-10-01T00:00:00.000+02:00</endDate>
        <state>active</state>
        <dateLaMu>2010-10-01T00:00:00.000+02:00</dateLaMu>
        <userLaMu>Ronald</userLaMu>
        <language></language>
        <category></category>
        <helpInfo></helpInfo>
        <code>-</code>
    </TransactionPhaseType>
    <TransactionPhaseType id="Result_Requested" >
        <description>Result requested</description>
        <startDate>2010-10-01T00:00:00.000+02:00</startDate>
        <endDate>2099-10-01T00:00:00.000+02:00</endDate>
        <state>active</state>
        <dateLaMu>2010-10-01T00:00:00.000+02:00</dateLaMu>
        <userLaMu>Ronald</userLaMu>
        <language></language>
        <category></category>
        <helpInfo></helpInfo>
        <code>-</code>
    </TransactionPhaseType>
    <TransactionPhaseType id="Result_Stated" >
        <description>Result stated</description>
        <startDate>2010-11-04T00:00:00.000+01:00</startDate>
        <endDate>2099-11-04T00:00:00.000+01:00</endDate>
        <state>active</state>
        <dateLaMu>2010-11-04T00:00:00.000+01:00</dateLaMu>
        <userLaMu>Ronald</userLaMu>
        <language></language>
        <category></category>
        <helpInfo></helpInfo>
        <code>-</code>
    </TransactionPhaseType>
```

C.3.3 MessageType

A message is a populated information model and contains data. The MessageType defines the structure and provides the elements for the content of the message.

Within each message, there are groups of segments (composite elements). These composite elements are defined by ComplexElementTypes. Within each segment, there can be a mix of composite elements and/or singular elements. Singular elements are defined by SimpleElementTypes.

The example below shows in XML view the definition of the MessageTypes that are graphically depicted in Figure 5 in Clause 3:

— request for 3D model;

— work done and request approval;

— request adjustments;

— work approved;

— work not approved.

```
    <MessageType id="Request_for_3D_model" >
        <description>Request for 3D model</description>
        <startDate>2010-11-04T00:00:00.000+01:00</startDate>
        <endDate>2099-11-04T00:00:00.000+01:00</endDate>
        <state>active</state>
        <dateLaMu>2010-11-04T00:00:00.000+01:00</dateLaMu>
        <userLaMu>Ronald</userLaMu>
```

```
        <language></language>
        <category></category>
        <helpInfo></helpInfo>
        <code>-</code>
        <complexElements>
            <ComplexElementTypeRef idref="ResultRequestedComplexElement"/ >
            <ComplexElementTypeRef idref="WorkIdentificationComplexElement"/ >
            <ComplexElementTypeRef idref="WorkSpecificationComplexElement"/ >
            <ComplexElementTypeRef idref="RemarksComplexElement"/ >
            <ComplexElementTypeRef idref="DeadlineComplexElement"/ >
        </complexElements>
    </MessageType>
    <MessageType id="Work_done_and_request_approval" >
        <description>Work done and request approval</description>
        <startDate>2010-11-04T00:00:00.000+01:00</startDate>
        <endDate>2099-11-04T00:00:00.000+01:00</endDate>
        <state>active</state>
        <dateLaMu>2010-11-04T00:00:00.000+01:00</dateLaMu>
        <userLaMu>Ronald</userLaMu>
        <language></language>
        <category></category>
        <helpInfo></helpInfo>
        <code>-</code>
        <complexElements>
            <ComplexElementTypeRef idref="RequestApprovalComplexElement"/ >
            <ComplexElementTypeRef idref="RemarksComplexElement"/ >
            <ComplexElementTypeRef idref="DeadlineComplexElement"/ >
            <ComplexElementTypeRef idref="WorkIdentificationComplexElement"/ >
            <ComplexElementTypeRef idref="WorkSpecificationComplexElement"/ >
        </complexElements>
    </MessageType>
    <MessageType id="Request_adjustments" >
        <description>Request adjustments</description>
        <startDate>2010-11-04T00:00:00.000+01:00</startDate>
        <endDate>2099-11-04T00:00:00.000+01:00</endDate>
        <state>active</state>
        <dateLaMu>2010-11-04T00:00:00.000+01:00</dateLaMu>
        <userLaMu>Ronald</userLaMu>
        <language></language>
        <category></category>
        <helpInfo></helpInfo>
        <code>-</code>
        <complexElements>
            <ComplexElementTypeRef idref="RequestAdjustmentsComplexElement"/ >
            <ComplexElementTypeRef idref="RemarksComplexElement"/ >
            <ComplexElementTypeRef idref="DeadlineComplexElement"/ >
            <ComplexElementTypeRef idref="WorkIdentificationComplexElement"/ >
            <ComplexElementTypeRef idref="WorkSpecificationComplexElement"/ >
        </complexElements>
    </MessageType>
    <MessageType id="Work_approved" >
        <description>Work approved</description>
        <startDate>2010-11-04T00:00:00.000+01:00</startDate>
        <endDate>2099-11-04T00:00:00.000+01:00</endDate>
        <state>active</state>
        <dateLaMu>2010-11-04T00:00:00.000+01:00</dateLaMu>
        <userLaMu>Ronald</userLaMu>
        <language></language>
        <category></category>
        <helpInfo></helpInfo>
        <code>-</code>
        <complexElements>
            <ComplexElementTypeRef idref="ApprovalComplexElement"/ >
            <ComplexElementTypeRef idref="RemarksComplexElement"/ >
            <ComplexElementTypeRef idref="WorkIdentificationComplexElement"/ >
            <ComplexElementTypeRef idref="WorkSpecificationComplexElement"/ >
        </complexElements>
    </MessageType>
    <MessageType id="Work_not_approved" >
        <description>Work not approved</description>
        <startDate>2010-11-04T00:00:00.000+01:00</startDate>
        <endDate>2099-11-04T00:00:00.000+01:00</endDate>
        <state>active</state>
        <dateLaMu>2010-11-04T00:00:00.000+01:00</dateLaMu>
        <userLaMu>Ronald</userLaMu>
        <language></language>
        <category></category>
        <helpInfo></helpInfo>
        <code>-</code>
        <complexElements>
```

```
            <ComplexElementTypeRef idref="DisapprovalComplexElement"/ >
            <ComplexElementTypeRef idref="RemarksComplexElement"/ >
            <ComplexElementTypeRef idref="WorkIdentificationComplexElement"/ >
            <ComplexElementTypeRef idref="WorkSpecificationComplexElement"/ >
        </complexElements>
    </MessageType>
```

The MessageType is also used to identify messages within a transaction. The relation between a messageType and a transactionType is defined in another Type: the MessageInTransactionType.

C.3.4 MessageInTransactionType

A transaction contains a set of messages that are exchanged for a particular purpose. The MessageInTransactionType definitions provide for linking MessageTypes to the TransactionTypes. An identical MessageType can be linked to all defined TransactionTypes. It is also possible to link an identical MessageType more than once (even unlimited) to the same TransactionType.

Each MessageInTransactionType can be defined in terms of

— an ID (identification of the MessageInTransactionType),

— the direction the message should be sent (From Initiator to Executor = true; From Executor to Initiator = false),

— metadata of the MessageInTransactionType,

— the reference to the MessageType (which position is defined in the MessageInTransactionType),

— the reference to the TransactionType(s) for linking the referred MessageTypes,

— the reference to the TransactionPhase in which state the process is brought (after sending the MessageType within the TransactionType as has been defined by the MessageInTransactionType),

— the reference to the GroupType.

The example below shows in XML view the definition of two MessageInTransactionTypes in our example of delivering a 3D model. The example can be understood by referring to Figure 5 (in ISO/WD 29481-2 IDM Part 2 Interaction Map; Chapter 3).

The example in XML view below illustrates that the TransactionType T3 starts with the MessageType 'Request for 3D model'. It is the first message within the transaction because there is no previous MessageInTransactionType defined within MessageInTranactionType 'MessageInTransaction1'. The direction of MessageInTransaction1 is pointed from Initiator to Executor (value = true).

The second MessageInTransactionType defined by 'MessageInTransaction2' refers to the MessageType 'Work done and request approval'. Figure 5 shows that this message should follow on two previous messages:

— Request for 3D model;

— Request adjustments.

These two previous messages have been defined as MessageInTransactionTypes (nr. One and 3) within the MessageInTransactionType2. The direction of MessageInTransaction1 is pointed from Executor to Initiator (value = false).

```
<MessageInTransactionType id="MessageInTransactie1" >
    <requiredNotify>0</requiredNotify>
    <dateLaMu>2010-11-05T00:00:00.000+01:00</dateLaMu>
    <userLaMu>Ronald</userLaMu>
    <received>0</received>
    <send>0</send>
    <state></state>
    <initiatorToExecutor>true</initiatorToExecutor>
    <message>
        <MessageTypeRef idref="Request_for_3D_model"/ >
    </message>
    <transaction>
        <TransactionTypeRef idref="T3"/ >
```

```
        </transaction>
        <transactionPhase>
            <TransactionPhaseTypeRef idref="Result_Requested"/ >
        </transactionPhase>
        <group>
            <GroupTypeRef idref="StandardGroup"/ >
        </group>
    </MessageInTransactionType>
    <MessageInTransactionType id="MessageInTransactie2" >
        <requiredNotify>0</requiredNotify>
        <dateLaMu>2010-11-05T00:00:00.000+01:00</dateLaMu>
        <userLaMu>Ronald</userLaMu>
        <received>0</received>
        <send>0</send>
        <state></state>
        <initiatorToExecutor>false</initiatorToExecutor>
        <message>
            <MessageTypeRef idref="Work_done_and_request_approval"/ >
        </message>
        <previous>
            <MessageInTransactionTypeRef idref="MessageInTransactie1"/ >
            <MessageInTransactionTypeRef idref="MessageInTransactie3"/ >
        </previous>
        <transaction>
            <TransactionTypeRef idref="T3"/ >
        </transaction>
        <transactionPhase>
            <TransactionPhaseTypeRef idref="Result_Produced"/ >
        </transactionPhase>
        <group>
            <GroupTypeRef idref="StandardGroup"/ >
        </group>
    </MessageInTransactionType>
```

C.3.5 AppendixTypes

Attachments may be linked to messages. As an attachment, an exchange requirement can be transferred to the executing role and the result (contribution to the BIM) is delivered to the initiating role.

AppendixType definitions provide for constraining element information items related to documents by adding metadata to documents that are attached to a message.

The XML view below shows the definition of the AppendixType. The metadata is defined by SimpleElementTypes within the ComplexElementType 'AppendixComplexElement'.

```
    <AppendixType id="StandardAppendixType" >
        <description>Standard Appendix</description>
        <startDate>2010-10-01T00:00:00.000+02:00</startDate>
        <endDate>2099-10-01T00:00:00.000+02:00</endDate>
        <state>active</state>
        <dateLaMu>2010-10-01T00:00:00.000+02:00</dateLaMu>
        <userLaMu>Ronald</userLaMu>
        <language></language>
        <category></category>
        <helpInfo></helpInfo>
        <complexElements>
            <ComplexElementTypeRef idref="AppendixComplexElement"/ >
        </complexElements>
    </AppendixType>
```

C.3.6 Constraining the content of messages

The third and last step of defining the interaction framework is to define the components within the MessageTypes in order to:

— structure the MessageTypes;

— restrict the input of content.

The definition of a MessageType already contains the composite elements of a message. These composite elements are defined as ComplexElementTypes.

The ComplexElementTypes consist of SimpleElementTypes (or other ComplexElementTypes are also possible). The SimpleElementTypes are related to UserDefinedTypes for defining restrictions to the input of contents. This structure is graphically depicted below.

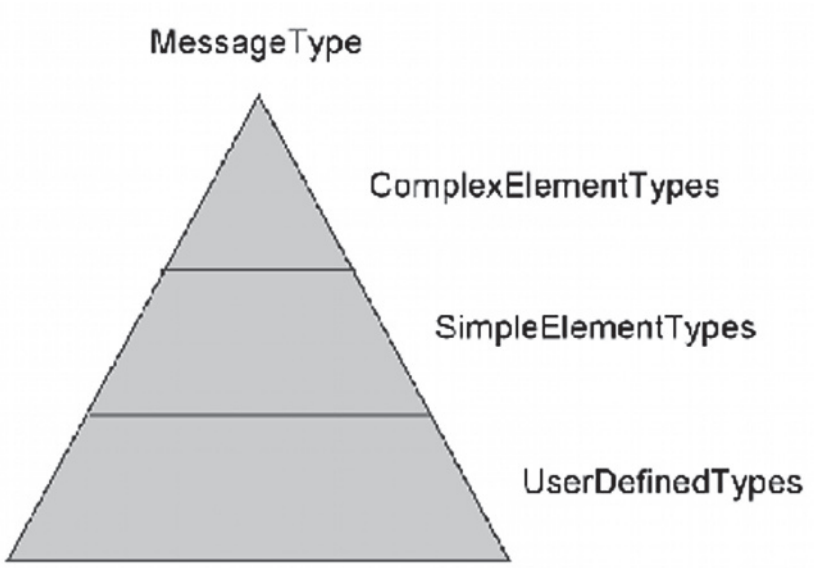

Figure C.2 — Components of modelling a message

C.3.7 ComplexElementTypes

ComplexElementType definitions provide for

— constraining element information items,

— using the mechanisms of Type Definition Hierarchy (within XML) to derive a complex type from another simple or complex type,

— specifying post-schema-validation infoset contributions for elements,

— limiting the ability to derive additional types from a given complex type,

— controlling the permission to substitute, in an instance, elements of a derived type for elements declared in a content model to be of a given complex type.

Referring to our example of the MessageType 'Request for 3D Model', related to transaction T3 (Deliver 3D model), this MessageType contains the following ComplexElementTypes:

— resultRequestedComplexElement;

— workIdentificationComplexElement;

— workSpecificationComplexElement;

— remarksComplexElement;

— deadlineComplexElement.

The ComplexElementType 'WorkSpecification' is depicted below in the XML view. A ComplexElementType defines its identification, the reference to the simple elements that give content to the complexElementType and its metadata.

The XML view of ComplexElementType 'WorkSpecificationComplexElement' shows that ComplexElementType contains two SimpleElementTypes:

— ScopeOfWork;

— CostEstimation.

```
<ComplexElementType id="WorkSpecificationComplexElement" >
    <description>Work specification</description>
    <startDate>2010-10-01T00:00:00.000+02:00</startDate>
    <endDate>2099-10-01T00:00:00.000+02:00</endDate>
    <state>active</state>
    <dateLaMu>2010-10-01T00:00:00.000+02:00</dateLaMu>
    <userLaMu>Ronald</userLaMu>
    <language></language>
    <category></category>
    <helpInfo></helpInfo>
    <simpleElements>
        <SimpleElementTypeRef idref="ScopeOfWork"/ >
        <SimpleElementTypeRef idref="CostEstimation"/ >
    </simpleElements>
</ComplexElementType>
```

C.3.8 SimpleElementTypes

SimpleElementType definitions provide for

— defining its identification,

— establishing the 'value space' and input restrictions.

The XML view below shows the definition of one SimpleElementType: CostEstimation. This SimpleElementType is part of the ComplexElementType 'WorkSpecificationComplexElementType' (previous example).

```
<SimpleElementType id="CostEstimation" >
        <description>Cost estimation</description>
        <interfaceType/>
        <state>active</state>
        <dateLaMu>2010-12-10T12:36:02.527+01:00</dateLaMu>
        <userLaMu>Ronald</userLaMu>
        <language/>
        <category/>
        <helpInfo>Indication of costs </helpInfo>
        <valueList/>
        <userDefinedType>
                <UserDefinedTypeRef idref="EURO"/ >
        </userDefinedType>
</SimpleElementType>
```

C.3.9 UserDefinedTypes

UserDefinedTypes are used to define the restrictions for the input of content (by defining dataTypes). All possible XML restrictions are allowed to define the UserDefinedTypes within an interaction framework. In our example, only the 'STRING' datatype and 'EURO' have been used.

The example below shows in XML view the definition of a UserDefinedType provided for

— identification,

— metadata,

— baseType,

— XSD restriction (in order to define a restriction more specific or to define enumeration types).

```
- <UserDefinedType id="EURO" >
  <description>Euro</description>
  <state>active</state>
  <dateLaMu>2010-12-10T12:35:19.485+01:00</dateLaMu>
  <userLaMu>Ronald</userLaMu>
  <baseType>DECIMAL</baseType>
  <xsdRestriction><xs:fractionDigits value="2"/ >  < /xsdRestriction >
```

```
  <helpInfo>amounts in Euro, specified in two decimals</helpInfo>
  </UserDefinedType>
- <UserDefinedType id="STRING" >
  <description>Character strings</description>
  <state>active</state>
  <dateLaMu>2010-10-01T00:00:00</dateLaMu>
  <userLaMu>Ronald</userLaMu>
  <baseType>STRING</baseType>
  <xsdRestriction />
  <language />
  <helpInfo>The string datatype represents character strings in XML. The 'value
    space' of string is the set of finite-length sequences of
    characters.</helpInfo>
  </UserDefinedType>
```

C.3.10 Completion

All ElementTypes have been defined to complete the interaction framework. In order to check the correctness, the framework shall be validated against it XSD Schema. The next step is to promote the interaction framework in order to obtain an Interaction schema. These actions have been described in 4.8.

Annex D
(informative)

Principles for Promotor algorithm

D.1 Principles for Promotor algorithm

A Promotor is an algorithm that turns an interaction framework into an interaction schema. Since an interaction framework is recorded in XML and an interaction schema in XSD, one may conclude that a Promotor abstracts the data elements of the interaction framework into the entity-relation structure of an interaction schema.

The basic algorithm of a Promotor raises all "*Type" data elements () of the interaction framework (RoleType, TransactionType, MessageType, ComplexElementType, SimpleElementType, etc.) into XSD complex type elements. Of course, certain rules must be obeyed to guarantee that the resulting XSD is a meaningful XML schema. For example, the ID attribute value of a *Type data element is interpreted as the name of its XSD complex element counterpart. As a result, the ID attribute in an interaction framework should not be something like ID = "Role003" but more like ID = "Project_Manager".

The XSD is organized in such a way that the MessageTypes are the primary complex elements to structure the contents of a message at the level of the actual exchange in a transaction. Such a message should contain enough information to trace the actual position of this message in the total flow of messages and to determine which succeeding messages can be dispatched from this position.

To generate the correct attribute types and relation types for the resulting XSD, the Promotor consults a file that describes which template should be used for which complex element type (templates file).

Bibliography

[1] Dietz J.L.G. (2006). *Enterprise Ontology: Theory and Methodology*, Springer; 1 edition, 240 pages, ISBN: 3540291695

ICS 91.010.01
Price based on 74 pages

This page deliberately left blank

British Standards Institution (BSI)

BSI is the national body responsible for preparing British Standards and other standards-related publications, information and services.

BSI is incorporated by Royal Charter. British Standards and other standardization products are published by BSI Standards Limited.

About us

We bring together business, industry, government, consumers, innovators and others to shape their combined experience and expertise into standards -based solutions.

The knowledge embodied in our standards has been carefully assembled in a dependable format and refined through our open consultation process. Organizations of all sizes and across all sectors choose standards to help them achieve their goals.

Information on standards

We can provide you with the knowledge that your organization needs to succeed. Find out more about British Standards by visiting our website at bsigroup.com/standards or contacting our Customer Services team or Knowledge Centre.

Buying standards

You can buy and download PDF versions of BSI publications, including British and adopted European and international standards, through our website at bsigroup.com/shop, where hard copies can also be purchased.

If you need international and foreign standards from other Standards Development Organizations, hard copies can be ordered from our Customer Services team.

Subscriptions

Our range of subscription services are designed to make using standards easier for you. For further information on our subscription products go to bsigroup.com/subscriptions.

With **British Standards Online (BSOL)** you'll have instant access to over 55,000 British and adopted European and international standards from your desktop. It's available 24/7 and is refreshed daily so you'll always be up to date.

You can keep in touch with standards developments and receive substantial discounts on the purchase price of standards, both in single copy and subscription format, by becoming a **BSI Subscribing Member**.

PLUS is an updating service exclusive to BSI Subscribing Members. You will automatically receive the latest hard copy of your standards when they're revised or replaced.

To find out more about becoming a BSI Subscribing Member and the benefits of membership, please visit bsigroup.com/shop.

With a **Multi-User Network Licence (MUNL)** you are able to host standards publications on your intranet. Licences can cover as few or as many users as you wish. With updates supplied as soon as they're available, you can be sure your documentation is current. For further information, email bsmusales@bsigroup.com.

BSI Group Headquarters

389 Chiswick High Road London W4 4AL UK

Revisions

Our British Standards and other publications are updated by amendment or revision.

We continually improve the quality of our products and services to benefit your business. If you find an inaccuracy or ambiguity within a British Standard or other BSI publication please inform the Knowledge Centre.

Copyright

Useful Contacts:

Customer Services
Tel: +44 845 086 9001
Email (orders): orders@bsigroup.com
Email (enquiries): cservices@bsigroup.com

Subscriptions
Tel: +44 845 086 9001
Email: subscriptions@bsigroup.com

Knowledge Centre
Tel: +44 20 8996 7004
Email: knowledgecentre@bsigroup.com

Copyright & Licensing
Tel: +44 20 8996 7070
Email: copyright@bsigroup.com

bsi.

...making excellence a habit.™