INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

Z.100

Corrigendum 1

(10/2001)

SERIES Z: LANGUAGES AND GENERAL SOFTWARE ASPECTS FOR TELECOMMUNICATION SYSTEMS

Formal description techniques (FDT) – Specification and Description Language (SDL)

Specification and description language (SDL)

Corrigendum 1

ITU-T Recommendation Z.100 – Corrigendum 1

ITU-T Z-SERIES RECOMMENDATIONS

LANGUAGES AND GENERAL SOFTWARE ASPECTS FOR TELECOMMUNICATION SYSTEMS

FORMAL DESCRIPTION TECHNIQUES (FDT)	
Specification and Description Language (SDL)	**Z.100–Z.109**
Application of formal description techniques	Z.110–Z.119
Message Sequence Chart (MSC)	Z.120–Z.129
Extended Object Definition Language (eODL)	Z.130–Z.139
Tree and Tabular Combined Notation (TTCN)	Z.140–Z.149
User Requirements Notation (URN)	Z.150–Z.159
PROGRAMMING LANGUAGES	
CHILL: The ITU-T high level language	Z.200–Z.209
MAN-MACHINE LANGUAGE	
General principles	Z.300–Z.309
Basic syntax and dialogue procedures	Z.310–Z.319
Extended MML for visual display terminals	Z.320–Z.329
Specification of the man-machine interface	Z.330–Z.349
Data-oriented human-machine interfaces	Z.350–Z.359
Human-computer interfaces for the management of telecommunications networks	Z.360–Z.369
QUALITY	
Quality of telecommunication software	Z.400–Z.409
Quality aspects of protocol-related Recommendations	Z.450–Z.459
METHODS	
Methods for validation and testing	Z.500–Z.519
MIDDLEWARE	
Distributed processing environment	Z.600–Z.609

For further details, please refer to the list of ITU-T Recommendations.

ITU-T Recommendation Z.100

Specification and description language (SDL)

CORRIGENDUM 1

Summary

This corrigendum contains a list of changes to ITU-T Rec. Z.100 (SDL) that was approved by Study Group 10 in November 2000, plus additional changes agreed to by the Expert Group at subsequent meetings, and the final list agreed to by SG 10 at the meeting in September 2001.

Source

Corrigendum 1 to ITU-T Recommendation Z.100 was prepared by ITU-T Study Group 10 (2001-2004) and approved under the WTSA Resolution 1 procedure on 29 October 2001.

CONTENTS

Page

Page

Page

ITU-T Recommendation Z.100

Specification and description language (SDL)

CORRIGENDUM 1

1 Objectives and scope

The purpose of this corrigendum is to record agreed changes to ITU-T Rec. Z.100 (1999).

The changes come in two categories:

a) Correction of *errors* and *clarifications* (see definitions in Appendix II/Z.100);

b) *Extensions* and *modifications* (see definitions in Appendix II/Z.100).

The rules for maintenance in Appendix II/Z.100 state that *errors* and *clarifications* published in the Master list of changes "come into effect immediately". Such changes should be published in a corrigendum as soon as is practical.

Modification and *extensions* imply some change to SDL. The rule in this case is "Unless there are special circumstances requiring such changes to be implemented as soon as possible, such changes will not be recommended until ITU-T Rec. Z.100 is revised." *Modification* and *extensions* included in this corrigendum are not considered to be significant changes – but are considered as useful improvements – to the SDL notation and therefore are now Recommended for immediate use.

2 Terminology

An *error* is an inconsistency in one or more ITU-T Recs. Z.100 to Z.109.

A *textual correction* is a change in the text or diagrams of Recommendations that corrects clerical or typographical errors.

An *open item* is an issue identified but not resolved.

A *deficiency* is an issue identified where the semantics of SDL are not clearly defined in the Recommendations.

A *clarification* is a change to the text (or diagrams) in a Recommendation that does not (intentionally) change the meaning of SDL, but is intended to make the Recommendations less ambiguous or easier to understand.

A *modification* changes the semantics of SDL.

An *extension* is a new feature that does not change the semantics of SDL defined in the approved Recommendations for SDL.

3 Maintenance of ITU-T Recs. Z.100 to Z.109

Appendix II/Z.100 (1999) documents the procedure to be followed for the maintenance of Z.100 Recommendations. This procedure requires error corrections, proposed modifications and extensions to be widely publicised and a Master list of changes to be maintained. Although this corrigendum contains a list of changes, the Master list of changes also contains other information not relevant to this corrigendum.

4 Z.100 changes

4.1 Clerical error – Underlining <Natural literal name>

Replace all occurrences of <Natural literal name> with <Natural literal name>.

These occur in the syntax rules <number of pages>, <priority name> and <partial regular expression> and in the description of <partial regular expression>.

4.2 Extension – Add <comment body> before <left curly bracket>

Add [<comment body>] before every occurrence of <left curly bracket> in syntax rules except in the lexical rules: that is in 9.4, 11.13.1, 11.14.1, 11.14.5, 12.1, 12.1.1, 12.1.2, 12.1.8 and 12.1.9.4 (in some clauses more than once).

4.3 Clerical error – <end> – Annex D

In Annex D the <data type heading>s should each have an <end>.

4.4 Clerical error – Loop statement keyword – Clause 1.5: Differences ...

Replacement of keyword "**for**" by keyword "**loop**".

Add "**loop**" to the list of keywords of SDL-2000 that are not keywords of SDL-92.

Add "**for**" to the list of keywords of SDL-92 that are not keywords of SDL-2000.

4.5 Clarification – Clause 5.3.2: Titled enumeration item, Model

Add the following to the end of the *Model* section:

"Transformations can be interdependent and therefore the order in which various transformations are applied can determine the validity and meaning of an SDL specification. Precise details of the order of transformation can be found in Annexes F1 to F3."

4.6 Clerical error and clarification – Clause 5.4.2: BNF

Change the four syntax example lines:

```
<diagram in package> ::=
                        |     <package diagram>
                        |     <package reference area>
                        |     <type in agent area>
```

to omit the erroneous vertical bar, and rename <type in agent area> to match a clarification, as follows:

```
                              <package diagram>
                        |     <package reference area>
                        |     <entity in agent diagram>
```

Change "<type in agent area>" to "<entity in agent diagram>" in the paragraph that follows the example to read as follows:

"expresses that a <diagram in package> is a <package diagram>, or a <package reference area>, or a <entity in agent diagram>, or a <data type reference area>, …"

4.7 Clarification – Example – Clause 5.4.3: Metalanguage for graphical grammar

To match a clarification of the actual syntax that changes <operation graph area> to <operation body area>, change the Example and following text to:

"Example:

```
{ <operation text area>* <operation body area> } set
```

is a set consisting of zero or more <operation text area>s, and one <operation body area>. The ***set*** metasymbol" …

4.8 Clerical error – Line symbols – Clause 5.4.3: Metalanguage for graphical grammar

The text "and implies a flow line symbol (see 6.5)" is placed in the wrong paragraph. Change the last two paragraphs from:

"The metasymbol ***is followed by*** means that its right-hand argument follows (both logically and in drawing) its left-hand argument.

The metasymbol ***is connected to*** means that its right-hand argument is connected (both logically and in drawing) to its left-hand argument and implies a flow line symbol (see 6.5)."

to:

"The metasymbol ***is followed by*** means that its right-hand argument follows (both logically and in drawing) its left-hand argument and implies a flow line symbol (see 6.5).

The metasymbol ***is connected to*** means that its right-hand argument is connected (both logically and in drawing) to its left-hand argument."

4.9 Modification – Comment syntax – Clause 6.1: Lexical rules

The syntax of <comment> is changed to **comment** /# … #/ to allow also /# … #/ before "{" as a comment. To support this, the following modifications are made:

<comment body> is added as an alternative of <lexical unit>

```
<lexical unit> ::=
                        <name>
                    |   <quoted operation name>
                    |   <character string>
                    |   <hex string>
                    |   <bit string>
                    |   <note>
                    |   <comment body>
                    |   <composite special>
                    |   <special>
                    |   <keyword>
```

Introduce <number sign> as an alternative in the following rules, because it is removed from <other special>:

```
<note text> ::=
                        {   <general text character>
                        |   <other special>
                        |   <number sign>
                        |   <asterisk>+ <not asterisk or solidus>
                        |   <solidus>
                        |   <apostrophe> }*
<not asterisk or solidus> ::=
                        <general text character> | <other special> | <apostrophe> | <number sign>
<special> ::=
                        <solidus>   |   <asterisk>   |   <number sign>   |   <other special>
```

Define <comment body> by the following rules:

```
<comment body> ::=
                    <solidus> <number sign>      <note text> <number sign>+ <solidus>
<comment text> ::=
                          {     <general text character>
                          |     <other special>
                          |     <asterisk>
                          |     <number sign>+ <not number or solidus>
                          |     <solidus>
                          |     <apostrophe> }*

<not number or solidus> ::=
                          <general text character> | <other special> | <apostrophe> | <asterisk>
```

4.10 Extension – Range sign – Clause 6.1: Lexical rules

Change the rule <composite special> to include the alternative <range sign> – that is add a new line after <result sign>:

```
                    |     <range sign>
```

Add, after the rule for <result sign>:

```
<range sign> ::=
                          <full stop> <full stop>
```

4.11 Clerical error – Loop statement keyword – Clause 6.1: Lexical rules

The keyword **for** should be **loop**. Delete "**for**" from <keyword> and add "**loop**". Reformat the layout of the list of words.

4.12 Clerical error – Reference – Clause 6.3: Visibility rules, names and identifiers, Concrete textual grammar

In the paragraph:

"A reference definition is an entity after the transformation step for <referenced definition> (see Annex F)."

change the reference to "Annex F" to "7.3".

4.13 Clerical errors – Content of drawings – Clause 6.6: Partitioning of drawings

The word "implicit" should be underlined in:

<heading area> ::=
        <implicit text symbol> ***contains*** <heading>

The word "drawing" should be underlined in:

<kernel heading> ::=
        [<virtuality>] [**exported**]
        <drawing kind> <drawing qualifier> <drawing name>

Insert the missing vertical bar at the start of the line:

    | **state** [**type**] | **procedure** | **operator** | **method**

The word "implicit" should be underlined in:

```
                              <implicit text symbol> contains [<page number> [ (<number of pages>) ]]
```

NOTE – Only the word "Natural" should be underlined in <number of pages> (see global change).

In the text the word "implicit" should be underlined in "<implicit text symbol>" in the paragraph starting:

"The <implicit text symbol> is not shown, but implied, in order to have a clear separation …"

4.14 Modification – Change syntax for <comment> – Clause 6.7: Comment

The syntax is changed to:

```
<comment> ::=
                              comment <comment body>
```

4.15 Clerical error – Grammar for <specification area> – Clause 7.1: Framework

There seems to have been an editorial problem with this clause, and the grammar was not completed correctly. This is therefore corrected as follows.

The start of the rule <specification area>:

```
<specification area> ::=
                              <frame symbol> contains
```

is changed by the addition of an opening curly bracket as follows:

```
<specification area> ::=
                              <frame symbol> contains {
```

4.16 Clerical error – Allow signals to environment for type-based systems – Clause 7.1: Framework

To allow signals to the environment to be defined for a type-based system, the syntax is changed to allow packages on type-based systems. The syntax rules <textual system specification> and <graphical system specification> are revised as follows:

Concrete textual grammar

```
<textual system specification> ::=
                              <agent definition>
                        |     {<package use clause>}* <textual typebased agent definition>
```

Concrete graphical grammar

```
<graphical system specification> ::=
                              <agent diagram>
                        |     <graphical typebased agent definition> [ is associated with <package use area> ]
```

and add the following paragraphs to the end of the *Model* section:

"A <package use clause> before a <textual typebased agent definition> of a <textual system specification> is derived syntax for a <package use clause> before the <system heading> in the <system definition> derived from the <textual typebased agent definition>.

A <package use area> associated with a <graphical typebased agent definition> of a <graphical system specification> is derived syntax for a <package use area> associated with the <system diagram> derived from the <graphical typebased agent definition>."

4.17 Clarification – Potentially instantiated type – Clause 7.1: Framework, Semantics

Add the following paragraph to the end of the *Semantics*:

"For an *SDL-specification* with an *Agent-definition*, a type is *potentially instantiated* if it is either instantiated in the *Agent-definition*, or instantiated in a potentially instantiated type."

4.18 Clarification – Concrete graphical syntax for <diagram in package> – Clause 7.2: Package

The alternative "<type in agent area>" is renamed "<entity in agent diagram>".

4.19 Clerical error – Grammar for <specification area> – Clause 7.2: Package, Concrete textual grammar

Change the text under the <dependency symbol> syntax:

"... for the <package diagram> or <package> or <system specification> for a <package reference area> in a <specification area>, and must be consistent ..."

to:

"... for the <package diagram> (or <package> or <system specification> for a <package reference area> in a <specification area>), and must be consistent ..."

4.20 Clerical error – Union of packages – Clause 7.2: Package, Model

Delete the paragraph:

"If a package is mentioned in several <package use clause>s of a <package definition>, this corresponds to one <package use clause> which selects the union of the definitions selected in the <package use clause>s."

4.21 Clarification – Clause 8.1.1.1: Agent types, Abstract grammar

The definition of *Agent-kind*, *Agent-formal-parameter*, *State-machine-definition* are moved to this clause from clause 9.

Add *Agent-kind* before *Agent-type-identifier* and *Agent-formal-parameter*, *State-machine-definition* after *Agent-type-identifier* as follows:

Agent-kind	=	***SYSTEM* | *BLOCK* | *PROCESS***
Agent-type-identifier	=	*Identifier*
Agent-formal-parameter	=	*Parameter*
State-machine-definition	::	*State-name*
		Composite-state-type-identifier

4.22 Clarification – Syntax simplified – Clause 8.1.1.1: Agent types, Concrete textual grammar

In the rule <agent type structure> change "<state partitioning>" to "<state partition>".

However, <system type definition>, <block type definition> and <process type definition> are redefined in terms of <agent structure> rather than <agent type structure> and therefore the rules <agent type structure> and <agent type body> are not required. This simplifies the syntax.

4.23 Clarification – Syntax simplified – Clause 8.1.1.1: Agent types, Concrete graphical grammar

Change the rule name "<agent type diagram content>" to "<agent type structure area>".

Similar to the textual grammar, <agent type diagram content>, <type in agent area> and <agent type body area> and <type state machine graph area> are deleted, because <system type diagram>, <block type diagram>, and <process type diagram> are defined in terms of <agent structure area>.

4.24 Clerical error – Complete output set definition – Clause 8.1.1.1: Agent types, Semantics

Insert the following text at the end of the *Semantics*:

"The complete output set of an agent type is the union of all signals, remote procedures and remote variables mentioned, either directly or as part of interfaces and signal lists, in the outgoing signal lists associated with the gates of the agent type."

4.25 Clarification – Agent type model – Clause 8.1.1.1: Agent types, Model

The description of the transformation of the model was incomplete.

Replace the whole of the *Model* section by:

"An agent type with an <agent body> or an <agent body area> is shorthand for an agent type having only a state machine, but no contained agents. This state machine is obtained by replacing the <agent body> or <agent body area> by a composite state definition. This composite state definition has the same name as the agent type and its *State-transition-graph* is represented by the <agent body> or the <agent body area>.

An agent type with:

– a <state partition> with a <composite state reference> or <composite state>, or
– a <state partition area> with a <composite state reference area>, or
– <composite state area>

is a shorthand for an agent type having a state machine that is based on a virtual implied composite state type. The implied state type has the body of the <composite state reference> or <composite state> or <composite state reference area> or <composite state area>. If the agent type is a subtype, and if the supertype has a <state partition> or <state partition area>, the implied state type is a subtype that implicitly inherits the implied state type of the supertype.

Each implied type has a constraint which is itself (see 8.3.1)."

4.26 Clerical error – Missing context parameters – Clause 8.1.1.2: System type, Concrete textual grammar

Change the rule <system type reference> to:

```
<system type reference> ::=
                              system type <type reference heading> <type reference properties>
```

followed by the new paragraph:

"A <type reference heading> that is part of a <system type reference> must have a <system type name>."

4.27 Clerical error – Syntax simplified – Clause 8.1.1.2: System type, Concrete textual grammar

To simplify the syntax, change "<agent type structure>" to "<agent structure>" in <system type definition>.

```
<system type definition> ::=
                        <package use clause>*
                        <system type heading> <end> <agent structure>
                        endsystem type [ [<qualifier>] <system type name>] <end>
```

4.28 Clerical error – Omitted gates/syntax simplified – Clause 8.1.1.2: System type, Concrete graphical grammar

In the rule <system type diagram>, change "<agent type diagram content>" to "<agent type structure area>". However, the simplification of syntax below changes <agent type diagram content> to <agent structure area>.

See also 9.1.

The syntax is simplified to use <agent structure area>. Change "<system diagram>" to:

```
<system type diagram> ::=
                         <frame symbol> contains {<system type heading> <agent structure area> }
                        is connected to {{ <gate on diagram>* }set }
                        [ is associated with <package use area> ]
```

4.29 Clarification – Clause 8.1.1.2: System type, Concrete graphical grammar

Replace "<graphical type reference heading> that contains" by "<type reference heading> with" to obtain the following revised paragraph:

"The <type reference area> that is part of a <system type reference area> must have a <type reference heading> with a <system type name>."

4.30 Clarification – Syntax simplification – Clause 8.1.1.3: Block type, Concrete textual grammar

Change the rule <block type definition> to use "<agent structure>" instead of "<agent type structure>".

4.31 Clerical error – Missing context parameters – Clause 8.1.1.3: Block type, Concrete textual grammar

Change the rule <block type reference> to:

```
<block type reference> ::=
                        <type preamble>
                        block type <type reference heading> <type reference properties>
```

followed by the new paragraph:

"A <type reference heading> that is part of a <block type reference> must have a <block type name>."

4.32 Clarification – Clause 8.1.1.3: Block type, Concrete graphical grammar

In the rule <block type diagram>, change "<agent type diagram content>" to "<agent type structure area>". However, then change "<agent type structure area>" to "<agent structure area>" to simplify the syntax.

4.33 Clarification – Clause 8.1.1.3: Block type, Concrete graphical grammar

Replace "<graphical type reference heading> that contains" by "<type reference heading> with" to obtain the following revised paragraph:

"The <type reference area> that is part of a <block type reference area> must have a "<type reference heading> with a <block type name>."

4.34 Error correction – Clause 8.1.1.3: Block type, Concrete graphical grammar

Delete the sentence:

"The <gate on diagram> in a <block type diagram> may not include <external channel identifiers>."

4.35 Clarification – Syntax simplification – Clause 8.1.1.4: Process type, Concrete textual grammar

Change the rule <process type definition> to use "<agent structure>" instead of "<agent type structure>".

4.36 Clerical error – Missing context parameters – Clause 8.1.1.4: Process type, Concrete textual grammar

Change the rule <process type reference> to:

```
<process type reference> ::=
                                    <type preamble>
                                    process type <type reference heading> <type reference properties>
```

followed by the new paragraph:

"A <type reference heading> that is part of a <process type reference> must have a <process type name>."

4.37 Clarification – Clause 8.1.1.4: Process type, Concrete graphical grammar

In the rule <process type diagram> change "<agent type diagram content>" to "<agent type structure area>". However, then change "<agent type structure area>" to "<agent structure area>" to simplify the syntax.

4.38 Clarification – Clause 8.1.1.4: Process type, Concrete graphical grammar

Replace "<graphical type reference heading> that contains" by "<type reference heading> with" to obtain the following revised paragraph:

"The <type reference area> that is part of a <process type reference area> must have a "<type reference heading> with a <process type name>."

4.39 Error correction – Clause 8.1.1.4: Process type, Semantics

Delete the sentence:

"The <gate on diagram> in a <process type diagram> may not include <external channel identifiers>."

4.40 Clerical error – Align with Annex F – Clause 8.1.1.5: Composite state type, Abstract grammar

Delete "*Connect-node**-set***" in *Composite-state-type-definition* to make the rule:

```
Composite-state-type-definition    ::    State-type-name
                                         [ Composite-state-type-identifier ]
                                         Composite-state-formal-parameter*
                                         State-entry-point-definition-set
                                         State-exit-point-definition-set
                                         Gate-definition-set
                                         Data-type-definition-set
                                         Syntype-definition-set
                                         Exception-definition-set
                                         Composite-state-type-definition-set
                                         Variable-definition-set
                                         Procedure-definition-set
                                         [ Composite-state-graph | State-aggregation-node ]
```

Change "::" to "=" in *Composite-state-type-identifier* to make the rule:

```
Composite-state-type-identifier    =     Identifier
```

4.41 Error correction – Clause 8.1.1.5: Composite state type, Abstract grammar

It is harmless if a composite state type has gates that are not used; therefore, the following static condition is deleted:

"The *Gate-definition**-set*** must be empty unless the composite state is used as a *State-machine-definition*."

4.42 Clarification – Clause 8.1.1.5: Composite state type, Concrete textual grammar

Replace the rule <composite state type definition> by:

```
<composite state type definition> ::=
                        {<package use clause>}*
                        { <composite state type heading> | <state aggregation type heading> }<end>
                              <composite state structure>
                        endsubstructure state type [ [ <qualifier> ] <composite state type name> ] <end>
```

4.43 Clerical error – Missing context parameters – Clause 8.1.1.5: Composite state type, Concrete textual grammar

Change the rule <composite state type reference> to:

```
<composite state type reference> ::=
                        <type preamble>
                        state type <type reference heading> <type reference properties>
```

followed by the new paragraph:

"A <type reference heading> that is part of a <composite state type reference> must have a <composite state type name>."

4.44 Clarification – Clause 8.1.1.5: Composite state type, Concrete textual grammar

Replace the rule <composite state type diagram> by:

```
<composite state type diagram> ::=
                    <frame symbol>
                    contains {
                         { <composite state type heading> | <state aggregation type heading> }
                         <composite state structure area>
                    is associated with { <graphical state connection point>* }set
                    is connected to {{ <gate on diagram>* }set }
                    [ is associated with <package use area> ]
```

4.45 Clarification – Clause 8.1.1.5: Composite state type, Concrete graphical grammar

Replace "<graphical type reference heading> that contains" by "<type reference heading> with" to obtain the following revised paragraph:

"The <type reference area> that is part of a <composite state type reference area> must have a "<type reference heading> with a <composite state type name>."

4.46 Error correction – Clause 8.1.1.5: Composite state type, Concrete graphical grammar

Move the paragraph:

"The <gate on diagram>s in a <process type diagram> must be outside the diagram frame."

from the *Semantics* to the end of the *Concrete graphical grammar*.

Delete the sentence in the *Semantics* (which should have been in the *Concrete graphical grammar*):

"The <gate on diagram> in a <composite state type diagram> may not include <external channel identifiers>."

4.47 Clerical error – Underlined type – Clause 8.1.3.1: System definition based on system type, Concrete textual grammar

Change "<system type expression>" to "<system type expression>" in the rule <typebased system heading>.

4.48 Clerical Error – Unnecessary transformation – Clause 8.1.3.1: System definition based on system type

Change:

"*Semantics*

A typebased system definition defines an *Agent-definition* with *Agent-kind* **SYSTEM** derived by transformation from a system type.

Model

A <textual typebased system definition> or a <graphical typebased system definition> is transformed to a <system definition> which has the definitions of the system type as defined by <system type expression>."

to:

"Semantics

A typebased system definition denoted by a <textual typebased system definition> or a <graphical typebased system definition> defines an *Agent-definition* with *Agent-kind* **SYSTEM** that is an instantiation of the system type denoted by the <system type expression>."

4.49 Clerical error – Unnecessary transformation – Clause 8.1.3.2: Block definition based on block type

Change:

"Semantics

A typebased block definition defines *Agent-definition*s of *Agent-kind* **BLOCK** derived by transformation from a block type.

Model

A <textual typebased block definition> or a <graphical typebased block definition> is transformed to a <block definition> which has the definitions of the block type as defined by <block type expression>.

An <inherited block definition> can only appear in a subtype definition. It represents the block defined in the supertype of the subtype definition. Additional channels connected to gates of the inherited block may be specified."

to:

"An <inherited block definition> shall only appear in a subtype definition. It represents the block defined in the supertype of the subtype definition.

NOTE – It is allowed to specify additional channels connected to gates of an inherited block.

Semantics

A typebased block definition denoted by a <textual typebased block definition> or a <graphical typebased block definition> define an *Agent-definition* of *Agent-kind* **BLOCK** that is an instantiation of the block type denoted by the <block type expression>."

4.50 Clerical error – Unnecessary transformation – Clause 8.1.3.3: Process definition based on process type

Change:

"Semantics

A typebased process definition defines an *Agent-definition* with *Agent-kind* **PROCESS** derived by transformation from a process type.

Model

A <textual typebased process definition> or a <graphical typebased process definition> is transformed to a <process definition> which has the definitions of the process type as defined by <process type expression>.

An <inherited process definition> can only appear in a subtype definition. It represents the process defined in the supertype of the subtype definition. Additional channels connected to gates of the inherited process may be specified."

to:

"An <inherited process definition> shall only appear in a subtype definition. It represents the process defined in the supertype of the subtype definition.

NOTE – It is allowed to specify additional channels connected to gates of the inherited process.

Semantics

A typebased process definition denoted by a <textual typebased process definition> or a <graphical typebased process definition> defines an *Agent-definition* with *Agent-kind* **PROCESS** that is an instantiation of the process type denoted by the <process type expression>."

4.51 Clerical error – Missing exported – Clause 8.1.4: Abstract type, Concrete grammar

Delete the two paragraphs:

"An <operation identifier> must not represent the *Operation-identifier* for the Make operator defined for an abstract data type.

A <literal identifier> must not represent a *Literal-identifier* denoting a *Literal-signature* of an abstract data type."

4.52 Clerical error – Sort in signal parameter property – Clause 8.1.5: Type references, Concrete textual grammar

Change the production for <signal parameter property> to:

```
<signal parameter property> ::=
                        <sort>
```

and replace the paragraph:

"A <signal parameter property> corresponds to a list of signal parameters in a signal definition. The <sort list> must correspond to the <sort list> in <signal definition item> of the corresponding signal definition. There must be only one <signal parameter property> in the <type reference properties> or <type reference area>."

by:

"A <signal parameter property> corresponds to a signal parameter in a signal definition. The sort must correspond to a <sort> in the <sort list> in <signal definition item> of the corresponding signal definition. The <signal parameter property>s in <type reference properties> must occur in the same order as the corresponding properties in the referenced type definition."

4.53 Clerical error – Missing exported – Clause 8.1.5: Type references, Concrete graphical grammar

This was allowed in SDL-92 and was omitted in error in SDL-2000.

Add "**exported as**" in <type reference heading>:

```
<type reference heading> ::=
                        <type preamble>
                        [ exported [ as <remote procedure identifier> ]]
                        [<qualifier>] <name> [<formal context parameters>]
```

Add to the paragraph after this rule:

"If **exported** is given in a <type reference heading>, the referenced type has to be an exported procedure and if a <remote procedure identifier> is also given, the procedure has to identify the same remote procedure definition."

4.54 Clerical error – Endpoint constraint too restrictive – Clause 8.1.6: Gate, Concrete textual grammar

The following paragraph is modified because it contained a restriction that is no longer needed:

"The <identifier> of <textual endpoint constraint> must denote a block type, process type or state type definition."

4.55 Clerical error – Wrong identifier – Clause 8.2.1: Agent type context parameter, Concrete textual grammar

The constraint should be an <agent type identifier> rather than an <agent identifier>.

4.56 Clarification – Meaning of colon – Clause 8.2.2: Agent context parameter, Concrete textual grammar

Add the text "(<colon> <agent type identifier>)" to the paragraph:

"An actual agent parameter must identify an agent definition. Its type must be a subtype of the constraint agent type (**atleast** <agent type identifier>) with no addition of formal parameters to those of the constraint type, or it must be the type denoted by <agent type identifier> (<colon> <agent type identifier>), or it must be compatible with the formal <agent signature>."

4.57 Clarification – Link to abstract syntax – Clause 8.3.1: Adding properties, Concrete textual grammar

Add the following text before the paragraph starting with "The concrete syntax for specialization of data types":

"The <type expression> of the <specialization> in:

a) <agent additional heading> represents the *Agent-type-identifier* of *Agent-type-definition* in 8.1.1.1.

b) <composite state type heading> or <state aggregation type heading> represents the *Composite-state-type-identifier* of *Composite-state-type-definition* in 8.1.1.5.

c) <procedure heading> represents the *Procedure-identifier* of *Procedure-definition* in 9.4."

4.58 Clerical error – Reference – Clause 8.3.3: Virtual transition/save, Concrete textual grammar

Correct cross-reference to "11.2.2 (virtual input)" to "11.3 (virtual input)".

4.59 Clerical error – Reference to Annex F – Clause 8.3.3: Virtual transition/save, Semantics

Replace the paragraph:

"The transformation of virtual transitions and saves in asterisk state is elaborated in Annex F."

with:

"In the subtype, it is the definition from the subtype that defines the virtual transition or save. A virtual transition or save that is not redefined in a subtype definition has the definition as given in the supertype definition."

4.60 Clerical error – Missing context parameters – Clause 8.4: Associations, Concrete textual grammar

Replace the rule <association end> by:

```
<association end> ::=
                    [<visibility>] [as <role name>] [size <multiplicity>] [ordered]
                    { <agent type reference> | <interface reference> | <data type reference> }
```

Delete the rule <linked type identifier> which is not needed.

Delete "<linked type identifier> in" in the paragraph starting with "If an <association end> is preceded …".

4.61 Clarification – Clause 9: Agents, Abstract grammar

The abstract syntax rule for *State-transition-graph* is deleted and moved to 11.11.1.

The abstract syntax rules for *State-machine-definition*, *Agent-kind* and *Agent-formal-parameter* are deleted and moved to 8.1.1.1.

4.62 Clarification – Clause 9: Agents, Concrete textual grammar

In the rule <agent structure> change "<state partitioning>" to "<state partition>".

Replace the two occurrences of "<state partitioning>" with "<state partition>" in the paragraph after the rules.

4.63 Clerical errors – Clause 9: Agents, Concrete textual grammar

Add the missing condition on connections. Insert the following new paragraph before the rule <agent instantiation>:

"A <channel to channel connection> must not be contained within an <agent type definition>."

In the rule <entity in agent>, insert "<composite state>" as an alternative; this was omitted in error.

4.64 Clarification – Clause 9: Agents, Concrete textual grammar

Remove the rule <state machine graph> by deleting the lines:

```
                    <state machine graph>
<state machine graph>::=
```

4.65 Clarification – Simplification – Clause 9: Agents, Concrete textual grammar

In the rule <agent body>, to support the syntax simplification that <agent body> can be used in both agent types and agents, make the [<on exception>] <start> optional:

```
                    [ [<on exception>] <start> ]
```

Before the paragraph "The use and syntax of <valid input signal set> is defined in clause 9.", insert the new paragraph:

"<start> may be omitted in <agent body> only in agent type definitions."

4.66 Clarification – How described – Clause 9: Agents, Concrete graphical grammar

Change the rule name "<agent diagram content>" to "<agent structure area>" and change "<type in agent area>" in this rule to "<entity in agent diagram>".

Redefine <agent body area> as:

```
<agent body area> ::=
                              {      [ [<on exception association area>] <start area> ]
                                     { <state area> | <exception handler area> | <in connector area> }* }set
```

This incorporates the rule <state machine graph area> which is therefore deleted.

Change the rule name "<type in agent area>" (from 8.1.1.1 Agent types where it is deleted) to "<entity in agent diagram>" and insert before <interaction area> as:

```
<entity in agent diagram> ::=
                              <agent type diagram>
                         |    <agent type reference area>
                         |    <composite state area>
                         |    <composite state type diagram>
                         |    <composite state type reference area>
                         |    <procedure diagram>
                         |    <procedure reference area>
                         |    <data type reference area>
                         |    <signal reference area>
                         |    <association area>
```

In the rule <interaction area>, change "<state machine area>" to "<state partition area>".

Delete the rule named "<state machine area>".

In the rule <create line endpoint area>, change "<state machine area>" to "<state partition area>".

Change the first sentence after the rules to:

"An <agent text area> is permitted to contain an <agent reference> only if the directly enclosing <agent structure area> contains an <interaction area>."

Change "<state machine area>" to "<state partition area>" in the sentence after NOTE 1 to read:

"The <state partition area> of <interaction area> identifies the state machine (composite state) of the agent, which may be given directly as an agent graph or by reference to a state definition."

Insert after this paragraph a new paragraph:

"<start area> can only be omitted in an agent type diagram."

4.67 Clerical error – How described – Clause 9: Agents, Concrete graphical grammar

Change the paragraph starting with "If there is an <agent reference area> ..." as follows:

"If there is an <agent reference area> for the agent, the <gate property area>s associated with the <agent reference area> correspond to the <gate on diagram>s associated with the <agent diagram>. No <gate property area> associated with the <agent reference area> must contain <signal list item>s not contained in the corresponding <gate on diagram>s associated with the <agent diagram>. A corresponding rule applies if there is an <agent reference> for the agent."

4.68 Clerical error – Unnecessary restriction on gate connection – Clause 9: Agents, Semantics

The following paragraph is not correct because there can be implicit channels and is therefore deleted. The text originates from SDL-92 where it was correct.

"If an <agent definition> or an <agent type definition>, which is used in a <textual typebased agent definition>, contains <channel definition>s and <textual typebased agent definition>s, then each gate of the <agent type definition>s of the contained <textual typebased agent definition>s must be connected to a channel."

4.69 Clarification – Stopping condition for initial agents – Clause 9: Agents, Semantics

To clarify the lifetime of an agent that has no state machine, insert the following paragraph after the paragraph about stopping conditions that starts with "When the state machine of an agent …"

"If an agent has no explicit or implicit state machine, as soon as all the initial contained agents have been created the agent enters a stopping condition. An agent with contained initial instances and no contained state machines, therefore ceases to exist as soon as it is created."

4.70 Clerical error – Complete valid input signal set – Clause 9: Agents, Semantics

"a) the set of signals in all channels or gates leading to the set of agent instances;

b) the <valid input signal set>;

c) the implicit input signals introduced as in 10.5, 10.6, 11.4, 11.5 and 11.6; and

d) "

is replaced with:

"a) the set of signals in all channels or gates leading to the state machine of the agent;

b) the <valid input signal set> of the agent;

c) the <valid input signal set> of the state machine of the agent;

d) the implicit input signals introduced as in 10.5 and 10.6, and

e) "

4.71 Clerical error – Incorrect plural – Clause 9: Agents, Semantics

At the end of NOTE 2, replace "names clashes" by "name clashes".

4.72 Clerical error – Complete output set definition – Clause 9: Agents, Semantics

Insert the following text at the end of the *Semantics*:

"The complete output set of an agent set is the same as the complete output set of the type of the agent set."

4.73 Clarification – Pid value of contained agents – Clause 9: Agents, Model

Generalize the penultimate paragraph of the *Model* to cover initial instances of any created agent (not just the system), by replacing it with:

"In all initial instances created when the containing instance is created, **parent** is initialized to Null."

4.74 Clerical error – Omitted gates – Clause 9.1: System, Concrete graphical grammar

Change the rule <system diagram> to:

```
<system diagram> ::=
                    <frame symbol> contains {<system heading> <agent structure area> }
                    is connected to { {<gate on diagram>}* <external channel identifiers> }set
                    [ is associated with <package use area> ]
```

4.75 Clerical error – Incorrect condition – Clause 9.1: System, Concrete graphical grammar

Change the last sentence to:

"A <system reference area> must only be used as part of a <specification area>."

4.76 Clarification – Clause 9.2: Block, Concrete graphical grammar

In rule <block diagram>, change "<agent diagram content>" to "<agent structure area>" and change:

is connected to { {<gate on diagram>}* <external channel identifiers> }***set***

to:

is connected to { {<gate on diagram> | <external channel identifiers>}* }***set***

4.77 Clarification – Clause 9.3: Process, Concrete graphical grammar

In the rule <process diagram>, change "<agent diagram content>" to "<agent structure area>" and change:

is connected to { {<gate on diagram>}* <external channel identifiers> }***set***

to:

is connected to { {<gate on diagram> | <external channel identifiers>}* }***set***

4.78 Clerical error – Start node/base procedure identifier – Clause 9.4: Procedures, Abstract grammar

In the definition of *Procedure-definition*, replace "*Procedure-identifier*" with "[*Procedure-identifier*]".

In the definition of *Procedure-graph*, replace "*Procedure-start-node*" with "[*Procedure-start-node*]".

At the end of *Abstract grammar*, add the paragraph:

"In an *SDL-specification*, all potentially instantiated procedures must have a *Procedure-start-node*."

4.79 Clerical error – Missing exported – Clause 9.4: Procedure, Concrete textual grammar

This was allowed in SDL-92 and was omitted by error in SDL-2000.

Change the rule <procedure reference> to:

<procedure reference> ::=
<type preamble> [**exported** [**as** <remote procedure identifier>]]
procedure <procedure identifier> <type reference properties>

Add a new paragraph before the *Concrete graphical grammar*:

"If **exported** is given in a procedure reference, the referenced procedure has to be an exported procedure and if a <remote procedure identifier> is also given, the procedure has to identify the same remote procedure definition."

4.80 Clerical error – Missing context parameters – Clause 9.4: Procedure, Concrete textual grammar

Change the rule <procedure reference> to:

<procedure reference> ::=
<type preamble>
procedure <type reference heading> <type reference properties>

followed by the new paragraph:

"A <type reference heading> that is part of a <procedure reference> must have a <procedure name>."

4.81 Clarification – Clause 9.4: Procedure, Concrete graphical grammar

In rule <procedure diagram> change "<procedure graph area>" to "<procedure body area>" because the rule has been renamed (see below).

4.82 Clerical error – Missing composite state – Clause 9.4: Procedure, Concrete graphical grammar

Change the rule <procedure area> to allow composite state type diagrams and references:

```
<procedure area> ::=
                        <procedure diagram>
                   |    <procedure reference area>
                   |    <composite state type diagram>
                   |    <composite state type reference area>
```

4.83 Clarification – Clause 9.4: Procedure, Concrete graphical grammar

Replace "<graphical type reference heading> that contains" by "<type reference heading> with" to obtain the following revised paragraph:

"The <type reference area> that is part of a <procedure reference area> must have a <type reference heading> with a <procedure name>."

4.84 Clarification – Clause 9.4: Procedure, Concrete graphical grammar

Rename the rule <procedure graph area> as "<procedure body area>".

Replace "<procedure graph area>" by "<procedure body area>" to obtain the following revised paragraph:

"The <on exception association area> of a <procedure body area> identifies the exception handler associated to the whole graph. The originating end must not be connected to any symbol."

4.85 Clerical error – Clause 9.4: Procedure, Semantics

In the sentence:

"a) A local variable is created for each *In-parameter*, having the *Name* and *Sort* of the *In-parameter...*",

replace the hyphen in the last "*In-parameter*." with a normal hyphen.

4.86 Clerical error – Reference to Annex F – Clause 9.4: Procedure, Semantics

Replace the paragraph:

"An external procedure is a procedure whose <procedure body> is not included in the SDL description. Conceptually, an external procedure is assumed to be given a <procedure body> and will be transformed into a <procedure definition> as part of the generic system transformation (see Annex F)."

by:

"An external procedure is a procedure whose <procedure body> is not included in the SDL description (see clause 13)."

4.87 Clerical error – Reference to Annex F – Clause 10.1: Channel, Concrete textual grammar

Replace the paragraph:

"If <gate> is specified, the channel is connected to that gate. The gate and the channel must have at least one common element in their signal lists in the same direction. If no <gate> is specified, the following rules apply:

a) if the channel endpoint is an agent or state machine and that agent/state contains a <channel to channel connection> for the channel, the channel is connected to the implicit gate introduced by the <channel to channel connection>;

b) if the channel endpoint is a state, the channel is connected to the implicit gate of that state machine (see Annex F),

otherwise the channel introduces an implicit gate on the agent or state mentioned in <channel endpoint>. This gate obtains the <signal list> of the respective <channel path>s as its corresponding gate constraint. The channel is connected to that gate.″

with:

"If <gate> is specified, the channel is connected to that gate. The gate and the channel must have at least one common element in their signal lists in the same direction. If no <gate> is specified, the following applies: If the channel endpoint is an agent or state machine and that agent/state contains a <channel to channel connection> for the channel, the channel is connected to the implicit gate introduced by the <channel to channel connection>. Otherwise, the channel introduces an implicit gate on the agent or state mentioned in <channel endpoint>. This gate obtains the <signal list> of the respective <channel path>s as its corresponding gate constraint. The channel is connected to that gate."

Add the following paragraph at the end of the section:

"If a <channel definition> contains two <channel path>s, then the following conditions must be satisfied:

– The <channel endpoint> following ***from*** in the first <channel path> must be the same as the <channel endpoint> following ***to*** in the second <channel path>.

– The <channel endpoint> following ***to*** in the first <channel path> must be the same as the <channel endpoint> following ***from*** in the second <channel path>."

4.88 Clarification – How described – Clause 10.1: Channel, Concrete graphical grammar

Change the paragraph starting with "For each arrowhead on the <channel symbol>," to:

"For each arrowhead on the <channel symbol>, there must be at most one <signal list area>. Each <signal list area> must be unambiguously close enough to one of the arrowheads. This arrowhead indicates the direction of the channel path the signal list with to which it is associated."

4.89 Clarification – How described – Clause 10.1: Channel, Concrete graphical grammar

In the rule <channel definition area>, change both instances of "<state machine area>" to "<state partition area>" and replace the sentence following the syntax rules by:

"When a <channel symbol> is connected to a <state partition area>, the <state partition area> denotes the state machine of the agent directly enclosing the channel definition."

4.90 Clerical error – Reference to Annex F – Clause 10.1: Channel, Model

The text:

"The details of this are described in Annex F."

was first replaced by

"The details of this are described with the model for implicit channels."

Subsequently, however, the model was moved from draft Annex F to 10.1, and the model text from Annex F was revised to obtain the following text:

"*Model*

If the <channel name> is omitted from a <channel definition> or <channel definition area>, the channel is implicitly and uniquely named.

A channel with both endpoints being gates of one <textual typebased agent definition> represents individual channels from each of the agents in this set to all agents in the set, including the originating agent. Any resulting bidirectional channel connecting an agent in the set to the agent itself is split into two unidirectional channels.

If an agent or agent type contains explicit or implicit gates not connected by explicit channels, implicit channels are derived according to the following three transforms, which must be applied after the transform for type-based creation in 11.13.2 is applied.

1) *Transform 1:* insertion of channels between instance sets inside the agent or agent type and between the instance sets and the agent state machine;
2) *Transform 2:* insertion of channels from a gate on the agent or agent type to gates on instance sets inside the agent or agent type and to gates on the agent state machine;
3) *Transform 3:* insertion of channels from gates on instance sets inside the agent or agent type and from gates on the agent state machine to gates on the agent or agent type.

These transforms are described in detail below. They are applied in the order given.

In the transforms, one signal list element (interfaces, signals, remote procedures or remote variables) matches another signal list element if:

a) both denote the same interface, signal, remote procedure or remote variable; or
b) the first denotes a signal or remote procedure or remote variable, and the second denotes an interface and the interface includes the signal or remote procedure or remote variable; or
c) both denote interfaces, and the second signal list element inherits the first signal list element.

Transform 1: insertion of implicit channels between entities inside one agent or agent type

a) if an element of the outgoing signal list associated with a gate of an instance in an agent (or agent type) matches an element of an incoming signal list associated with a gate of another instance in the same agent (or agent type respectively); and
b) if neither of these gates has an explicit channel connected to it,

then:

i) if no implicit channel exists between the two gates, a unidirectional implicit channel is created from the gate where the element is outgoing to the gate where the element is incoming, and this channel is non-delaying if it is within a process (or process type) and otherwise it is delaying; and
ii) the element is added to the signal list of the implied channel.

Transform 2: insertion of implicit channels from the gates on an agent or agent type

a) if an element of the incoming signal list associated with a gate outside an agent (or agent type) matches an element of an incoming signal list associated with a gate of an instance in the agent (or agent type respectively); and

b) if there is no explicit channel inside the agent (or agent type respectively) connected to the gate outside the agent (or agent type respectively) and no explicit channel connected to the gate of the instance inside the agent (or agent type respectively),

then:

i) if no implicit channel exists between the two gates, a unidirectional implicit channel is created from the gate outside the agent (or agent type respectively) to the gate of the instance inside the agent (or agent type respectively), and this channel is non-delaying if it is within a process (or process type) and otherwise it is delaying; and

ii) the element is added to the signal list of the implied channel.

Transform 3: insertion of implicit channels from the gates on instances

The following is applied for insertion of implicit channels from the gates on instance sets within the agent or agent type to the gates on the agent or agent type:

a) if an element of the outgoing signal list associated with a gate outside an agent (or agent type) matches an element of an outgoing signal list associated with a gate of an instance in the agent (or agent type respectively); and

b) if there is no explicit channel connected to the gate outside the agent (or agent type respectively) and no explicit channel connected to the gate of the instance inside the agent (or agent type respectively),

then:

i) if no implicit channel exists between the two gates in the direction to the gate outside the agent (or agent type respectively), a unidirectional implicit channel is created from the gate of the instance inside the agent (or agent type respectively) to the gate outside the agent (or agent type respectively), and this channel is non-delaying if it is within a process (or process type) and otherwise it is delaying; and

ii) the element is added to the signal list of the implied channel."

4.91 Clerical error – Clause 10.2: Connection, Concrete graphical grammar

Replace:

"Graphically, the connect construct is represented by the graphical connection of a <channel symbol> in a <channel definition area> to an <external channel identifiers> in a <gate on diagram>."

by:

"Graphically, the connect construct is represented by the graphical connection of a <channel symbol> in a <channel definition area> to an <external channel identifiers> connected to the enclosing frame symbol."

4.92 Clerical error – Clause 10.2: Connection, Model

Add the following paragraph at the end of the section:

"When a diagram is directly contained within another diagram (that is, it is not referenced), each <external channel identifiers> is omitted, because the external channels connected to the same point on the frame of the diagram from outside the diagram are shown directly."

4.93 Clerical error – Missing context parameters – Clause 10.3: Signal, Concrete textual grammar

Change the rule the procedure <procedure reference> to:

```
<signal reference> ::=
                                <type preamble>
                                signal <type reference heading> <type reference properties>
```

followed by the new paragraph:

"A <type reference heading> that is part of a <signal reference> must have a <signal name>."

4.94 Clarification – Clause 10.3: Signal, Concrete graphical grammar

Replace "<graphical type reference heading> that contains" by "<type reference heading> with" to obtain the following revised paragraph:

"The <type reference area> that is part of a <signal reference area> must have a "<type reference heading> with a <signal name>."

4.95 Clerical error – Missing interface – Clause 10.4: Signal list definition, Concrete textual grammar

In the paragraph starting with:

"A <signal list item> which is an <identifier> denotes a <signal identifier> or <timer identifier> if ..."

change "<timer identifier>" to "<timer identifier> or <interface identifier>".

4.96 Clerical error – Missing interface – Clause 10.4: Signal list definition, Concrete textual grammar

Delete the paragraph:

"It is only allowed to use **this** in <signal list>s that are part of <gate constraint>s (see 8.1.6)."

The paragraph is erroneous, because the syntax does not allow "**this**".

4.97 Clerical error – Missing exception – Clause 10.5: Remote procedures, Concrete textual grammar

Add an optional "<on exception>" to "<remote procedure call>" to obtain the rule:

```
<remote procedure call> ::=
                                call <remote procedure call body> [ <on exception> ]
```

4.98 Clerical Error – Complete output set definition – Clause 10.5: Remote procedures, Concrete textual grammar

In the following paragraph, change "Annex F" to "8.1.1.1 and clause 9" (see change to 8.1.1.1):

"A remote procedure mentioned in a <remote procedure call> must be in the complete output set (see Annex F) of an enclosing agent type or agent set."

4.99 Clerical error – Complete output set definition – Clause 10.5: Remote procedures, Concrete textual grammar

It should not be allowed to specify that a procedure definition has nodelay.

Delete "[**nodelay**]" in the syntax for <remote procedure definition>.

Delete the sentence:

"The use of **nodelay** is described in 10.1."

4.100 Clerical errors – Extra return, extra tab – Clause 10.5: Remote procedures, Model

From the SDL:

```
decision newn = n;
        (true):
        (false): nextstate pWAIT;
        enddecision;
        return;
```

delete the line:

```
return:
```

Delete a tab at the start of the line with two tabs after " … is added:"

```
input pCALL(fpar,n);
```

4.101 Clerical error – Nodelay – Clause 10.5: Remote procedures, Model

It should not be allowed to specify that a remote procedure definition has nodelay.

Delete the sentence:

"When a remote procedure is conveyed on explicit channels, the **nodelay** keyword from the <remote procedure definition> is ignored."

4.102 Clerical error – Missing exception – Clause 10.6: Remote variables, Concrete textual grammar

Replace the rule <export> by the two rules:

```
<export> ::=
                    export <export body> [ <on exception> ]
<export body> ::=
                    ( <variable identifier> { , <variable identifier> }* )
```

4.103 Clerical error – Annex F reference and nodelay – Clause 10.6: Remote variables

It should not be allowed to specify that a remote variable has **nodelay**.

Delete "[**nodelay**]" in the syntax for <remote variable definition> (twice).

Delete the sentence:

"The use of **nodelay** is described in Annex F."

In the following paragraph, "Annex F" is changed to "8.1.1.1 and clause 9".

Delete the sentence:

"A remote variable mentioned in an <import expression> must be in the complete output set (see Annex F) of an enclosing agent type or agent set."

In the *Model,* delete the sentence:

"When a remote variable is conveyed on explicit channels, the **nodelay** keyword from the <remote variable definition> is ignored."

4.104 Clerical error – Presentation mistake – Clause 11.2: State, Abstract grammar

Add "*Connect-node**-set***" to *State-node* as follows:

State-node	::	*State-name*
		[*On-exception*]
		Save-signalset
		*Input-node**-set***
		*Spontaneous-transition**-set***
		*Continuous-signal**-set***
		*Connect-node**-set***
		[*Composite-state-type-identifier*]

4.105 Clarification – Presentation of basic and composite states – Clause 11.2: State, Abstract grammar

Change the paragraph:

"A *State-node* with *Composite-state-type-identifier* represents a <composite state application>."

to:

"A *State-node* with *Composite-state-type-identifier* represents a composite state <composite state application>."

4.106 Clarification – Presentation of basic and composite states – Clause 11.2: State, Concrete textual grammar

Replace:

Concrete textual grammar

```
<state> ::=
                    <basic state> | <composite state application>
```

A <composite state application> represents a state with a *Composite-state-type-identifier.*

Concrete graphical grammar

```
<state area> ::=
                    <basic state area> | <composite state application area>
```

A <composite state area> represents a state with a *Composite-state-type-identifier.*

11.2.1 Basic State

By:

Concrete textual grammar

```
<state> ::=
                              state <state list> <end> [<on exception>]
                                   {      <input part>
                                   |      <priority input>
                                   |      <save part>
                                   |      <spontaneous transition>
                                   |      <continuous signal>
                                   |      <connect part> }*
                              [ endstate [<state name>] <end> ]

<state list> ::=
                              { <basic state name> | <composite state item> }
                                   { , { <basic state name> | <composite state item> } }*
                         |    <asterisk state list>

<basic state name> ::=
                              <state name>

<asterisk state list> ::=
                              <asterisk> [ ( <state name> { , <state name>}* ) ]

<composite state item> ::=
                              <composite state name> [<actual parameters>]
                         |    <typebased composite state>

<composite state name> ::=
                              <state name>
```

A <basic state name> is the name of a state that does not have a <composite state> or <composite state area>, and is not defined in a <typebased composite state>. A <composite state name> is the name of a state that has a <composite state> or <composite state area>, or is defined in a <typebased composite state>.

----

and insert before the Concrete graphical grammar the following two paragraphs (for composite state application):

"The <connect part> is only allowed for a <state> with <state list> that contains a <composite state item>.

A <composite state item> or <typebased composite state> shall only contain <actual parameters> if it is in a <state area> that coincides with a <nextstate area>. In this case the <state area> must only contain one <composite state name> and, optionally, <actual parameters>."

4.107 Clarification – Presentation of basic and composite states – Clause 11.2: State, Concrete graphical grammar

Change the rule name <basic state area> to "<state area>" and add a new alternative associated item "<connect association area>" to the end of the rule to make the rule (which replaces also "<composite state application>") as follows:

```
<state area> ::=
                              <state symbol> contains <state list>
                              [ is connected to <on exception association area> ]
                              is associated with
                                   {      <input association area>
                                   |      <priority input association area>
                                   |      <continuous signal association area>
                                   |      <spontaneous transition association area>
                                   |      <save association area>
                                   |      <connect association area> }*
```

Add at the end of the syntax just before the paragraph starting with "A <state are> represents …", the rule (from composite state application):

```
<connect association area> ::=
                        <solid association symbol> is connected to <connect area>
```

Add, before *Semantics*, the paragraph:

"<connect association area> is only allowed for a <state area> with <state list> that contains a <composite state item>."

4.108 Clarification – Presentation of basic and composite state – Clause 11.2: State, Semantics

Add at the start of the *Semantics* the following two paragraphs:

"A state represents either a basic state or a composite state application.

The semantics for composite state application is given in 11.11."

Change the start of the following paragraph from "A state represents …" to "A basic state represents …".

4.109 Clarification – Transition selection – Clause 11.2.1: Basic state, Semantics

Change:

"... in the following order:

a) ..."

to:

"... in the following steps. Each time the steps are repeated, the set of signals considered is updated to the signals on the input port, otherwise the same set is considered in each step.

a) ..."

and replace item d) by:

"d) if no enabled signal was found, as soon as the signals on the input port differs from set of signals already considered, or if there is an *Input-node* with a *Provided-expression* that could have changed, or *Continuous-expression* that could have changed, these steps are repeated. A *Provided-expression* or *Continuous-expression* can change only if it contains a *NOW-expression*, *Timer-active-expression*, *Any-expression*, or *Variable-access* to a variable defined in an enclosing process that is changed by assignment in another agent instance or another state partition."

4.110 Clarification – Presentation of basic and composite state – Clause 11.2.1: Basic state, Model

In the first paragraph of the *Model*, change "<state name>, a copy …" to "<state name> item, a copy …".

In the third paragraph of the *Model* after "one for each <state name>", add "and <composite state name>", and after "those <state name>s", add "and <composite state name>s".

Delete the whole of the old clause **11.2.2, Composite state application**.

4.111 Clarification – <action statement> to <action> – Clause 11.3: Input, Model

Change "<action statement>" to "<action>" (see change 4.155).

4.112 Clerical error – Omitted on-exception – Clause 11.5: Continuous Signal, Abstract grammar

Change:

Continuous-signal	::	*Continuous-expression* [*Priority-name*] *Transition*

to:

Continuous-signal	::	[*On-exception*] *Continuous-expression* [*Priority-name*] *Transition*

4.113 Clarification – Transition selection – Clause 11.5: Continuous signal, Semantics

Change the *Semantics* text to:

"The *Continuous-expression* is interpreted as part of the state to which its *Continuous-signal* is associated (see 11.2). If the *Continuous-expression* returns the predefined Boolean value true, the continuous signal is enabled.

The continuous signal having the lowest value for *Priority-name* has the highest priority."

4.114 Clarification – Transition selection – Clause 11.6: Enabling condition, Semantics

Changed the *Semantics* text to:

"The *Provided-expression* of an *Input-node* is interpreted as part of the state this *Input-node* is attached to (see 11.2).

A signal in the input port is enabled, if all the *Provided-expressions* of an *Input-node* return the predefined Boolean value true, or if the *Input-node* does not have a *Provided-expression*. The *Provided-expression* of a *Spontaneous-transition* can be interpreted at any time while the agent is in the state."

4.115 Clarification – Clause 11.7: Save, Abstract grammar

In the grammar:

Save-signalset	::	*Signal-identifier**-set***

change "::" to "=" to obtain:

Save-signalset	=	*Signal-identifier**-set***

4.116 Clarification – body area instead of graph area – Clause 11.10: Label, Concrete textual grammar

Change the paragraph after the syntax starting with "The term …" to:

"The term "body" is used to refer to a state machine graph, possibly after transformation from a <statement list> and after transformation from a type. A body therefore refers to <agent body>, <procedure body>, <operation body>, <agent body area>, <procedure body area>, <operation body area>, or <composite state body>."

4.117 Clarification – <action statement> to <action> – Clause 11.10: Label, Concrete textual grammar

In the last paragraph of the *Concrete textual grammar*, change "<action statement>" to "<action>" (see change 4.155).

4.118 Clarification – body area instead of graph area – Clause 11.10: Label, Concrete graphical grammar

Change the paragraph after the syntax starting with "An <in-connector area> …" to:

"An <in connector area> represents the continuation of a <flow line symbol> from a corresponding <out connector area> with the same <connector name> in the same <state machine graph area> or <agent body area> or <procedure body area>."

4.119 Clarification – <action statement> to <action> – Clause 11.10: Label, Model

Change "<action statement>" to "<action>" (see change 4.155).

4.120 Clarification – <terminator statement> to <terminator> – Clause 11.10: Label, Model

Change "<terminator statement>" to "<terminator>" (see change 4.155).

4.121 Clerical Error – Missing "by" – Clause 11.11: State machine and Composite state

In the introduction before the *Abstract grammar*, in the second paragraph change "and specification of a <state>" to "and by specification of a <state>".

4.122 Clarification – Clause 11.11: State machine and Composite state, Semantics

In the last paragraph, change the penultimate sentence from:

"The exit procedure is invoked after a *Return-node* of the *Composite-state-graph* is interpreted, or when a transition attached directly to the *State-node* is interpreted."

to:

"The exit procedure is invoked after a *Return-node* of the *Composite-state-graph* is interpreted and before a transition attached directly to the *State-node* is interpreted, if there are such transitions."

4.123 Clerical error – Missing Model – Clause 11.11: State machine and Composite state, Model

Add the *Model* section:

"Model

A *Composite-state* has an implied anonymous composite state type that defines the properties of the composite state.

A *Composite-state* that is a specialization is shorthand for defining an implicit composite state type and one typebased composite state of this type."

4.124 Clerical error – Optional start node – Clause 11.11.1: Composite state graph, Abstract grammar

Add the abstract syntax for *State-transition-graph* from clause 9.

In the definition of *State-transition-graph*, replace "*State-start-node*" with "*[State-start-node]*". The new syntax is therefore:

```
State-transition-graph        ::   [On-exception]
                                   [State-start-node]
                                   State-node-set
                                   Free-action-set
                                   Exception-handler-node-set
```

At the end of the *Abstract grammar*, add the paragraph:

"In an *SDL-specification*, all potentially instantiated agents must have a *State-start-node*."

4.125 Clarification – Clause 11.11.1: Composite state graph, Concrete textual grammar

Replace the rule <composite state graph> by the following two rules:

```
<composite state graph> ::=
                    {<package use clause>}*
                    <composite state heading> <end> <composite state structure>
                    endsubstructure [ [<qualifier>] <composite state name> ] <end>

<composite state structure> ::=
                    substructure
                         [<valid input signal set>]
                         {<gate in definition>}*
                         <state connection points>*
                         <entity in composite state>*
                         { <composite state body> | <state aggregation body> }
```

Add the following paragraph after the new rule <composite state structure>:

"A <composite state structure> shall contain a <state aggregation body> only if it is directly contained in a <state aggregation> or a <composite state type definition> with a <state aggregation type heading>; otherwise, it contains a <composite state body>. A <composite state structure> that contains a <state aggregation body> shall not have a <valid input signal set>."

4.126 Clerical error – Omitted specialization – Clause 11.11.1: Composite state graph, Concrete textual grammar

Add an optional <specialization> to the rule:

```
<composite state heading> ::=
                    state [<qualifier>] <composite state name>
                    [<agent formal parameters>] [<specialization>]
```

4.127 Clerical error – Omitted items – Clause 11.11.1: Composite state graph, Concrete textual grammar

Add <procedure reference> as an alternative to the production for <entity in composite state> after <procedure definition>.

Add <composite state> as an alternative to the production for <entity in composite state> after <exception definition>.

4.128 Clerical error – Misplaced rule – Clause 11.11.1: Composite state graph, Concrete textual grammar

Delete the rule <composite state reference>, which is better placed under **11.11.2, State aggregation**.

4.129 Clarification – Misplaced sentence – Clause 11.11.1: Composite state graph, Concrete textual grammar

After the paragraph:

"Exactly one of the <start>s shall be unlabelled. Each additional labelled entry and exit point must be defined by a corresponding <state connection points>."

add the following paragraph from 11.11.3 (reworded):

"A <start> with a <state entry point name> (a labelled start) in a <composite state body> shall refer only to <state entry point>s of the <composite state graph> directly enclosing the <composite state body>. A <return> with a <state exit point> (a labelled return) in a <composite state body> shall refer only to <state exit point>s of the <composite state graph> directly enclosing the <composite state body>."

4.130 Clarification – Gate use – Clause 11.11.1: Composite state graph, Concrete textual grammar

Add the following sentence about gates just before the *Concrete graphical grammar*:

"A <gate> of a <composite state graph> may only be a <channel endpoint> in the case the <composite state graph> is the <state partitioning> representing the state machine of an agent."

4.131 Clarification – Clause 11.11.1: Composite state graph, Concrete graphical grammar

Replace the rule <composite state graph area> by the following two rules:

```
<composite state graph area> ::=
                              <frame symbol> contains
                                    { <composite state heading> <composite state structure area> }
                              is associated with {<graphical state connection point>* } set
                              is connected to { {<gate on diagram> | <external channel identifiers>}* }set
                              [ is associated with <package use area> ]

<composite state structure area> ::=
                                    {     <composite state text area>*
                                          <entity in composite state area>*
                                          { <composite state body area> | <state aggregation body area> } } set
```

Add the following sentence after the new rule <composite state structure area>:

"A <composite state structure area> shall contain a <state aggregation body area> only if it is directly contained in a <state aggregation area> or a <composite state type diagram> with a <state aggregation type heading>; otherwise, it contains a <composite state body area>."

4.132 Clerical error – Omitted items – Clause 11.11.1: Composite state graph, Concrete graphical grammar

Add "<exception definition>" as an alternative in the <text symbol> to the production for <composite state text area> after "<procedure reference>".

4.133 Clarification – Clause 11.11.1: Composite state graph, Concrete graphical grammar

Rename the rule "<type in composite state area>" as "<entity in composite state area>", which is a more correct name.

Elaborate the paragraph:

"There shall be at most one <valid input signal set> in the <composite state text area>s of a <composite state graph area>."

to:

"There shall be at most one <valid input signal set> in the <composite state text area>s of a <composite state graph area> (or the corresponding composite state type definition). A <valid input signal set> must not be contained in a <composite state text area> of a <state aggregation area> (or the corresponding composite state type definition)."

4.134 Clarification – Misplaced sentence – Clause 11.11.1: Composite state graph, Concrete graphical grammar

Add the following paragraphs (after the one elaborated on above):

"At most, one of the <start area>s shall be unlabelled. Each additional labelled entry and exit point must be defined by a corresponding <state connection points>. Each additional labelled <start area> shall contain a different <state entry point name>.

A <start area> with a <state entry point name> (a labelled start) in a <composite state body area> shall refer only to <state entry point>s of the <composite state graph area> directly enclosing the <composite state body area>. A <return area> with a <state exit point> (a labelled return) in a <composite state body area> shall refer only to <state exit point>s of the <composite state graph area> directly enclosing the <composite state body area>."

4.135 Clarification – Clause 11.11.1: Composite state graph, Semantics

Change the last paragraph of the *Semantics* to:

"It is possible to specify a <composite state> that only consists of transitions associated with an asterisk state, without <start> and without any substates. These transitions may either be terminated by <dash nextstate> or by <return>. These transitions apply when the agent or procedure is in the composite state. The nextstate of such a transition terminated by <dash nextstate> is the composite state; however, the Exit-procedure-definition and Entry-procedure-definition of the composite state are not called."

4.136 Clarification – Clause 11.11.2: State aggregation

In the second sentence of the first paragraph, change "that is interleaved at the transition level" to "of alternating transitions" and delete the last sentence: "State aggregations can be used to partition the graph of a state.".

4.137 Clarification – Conditions – Clause 11.11.2: State aggregation, Abstract grammar

The combination of conditions in the *Abstract grammar* and *Concrete graphical grammar* is not clear. The conditions should be moved to the *Abstract grammar* only.

In the *Abstract grammar*, replace:

"All entry and exit points of the container state and the state partitions must appear in exactly one *Connection-definition*."

by

"For each *State-partition* each of the entry points of the container state shall appear in exactly one *Connection-definition*. For each *State-partition* each of the exit points of the *State-partition* shall appear in exactly one *Connection-definition*."

4.138 Clarification – Clause 11.11.2: State aggregation, Abstract grammar

Change "*State-partition-**set***" to "*State-partition*s" in the sentence:

"The input signal sets of the *State-partition-**set*** within a composite state must be disjoint."

to obtain:

"The input signal sets of the *State-partition*s within a composite state must be disjoint."

4.139 Clarification – Clause 11.11.2: State aggregation, Concrete textual grammar

Replace <state aggregation> by the rule:

```
<state aggregation> ::=
                        {<package use clause>}*
                        <state aggregation heading> <end> <composite state structure>
                        endsubstructure [ [<qualifier>] <composite state name> ] <end>
```

4.140 Clerical error – Omitted specialization – Clause 11.11.2: State aggregation, Concrete textual grammar

Add an optional <specialization> to the rule:

```
<state aggregation heading> ::=
                        state aggregation [<qualifier>] <composite state name>
                        [<agent formal parameters>][<specialization>]
```

4.141 Clarification – Clause 11.11.2: State aggregation, Concrete textual grammar

In the rule <state aggregation body>, change "<state partitioning>" to "<state partition>".

Change the rule name "<state partitioning>" to "<state partition>".

4.142 Clerical error – Allow empty body – Clause 11.11.2: State aggregation, Concrete textual grammar

In the rule <state aggregation body>, change "+" to "*" to make the repetition zero or more as follows:

```
<state aggregation body> ::=
                  {     <state partition>
                  |     <state partition connection> }*
```

4.143 Clarification – Composite state reference – Clause 11.11.2: State aggregation, Concrete textual grammar

<composite state reference> is moved here from 11.11.1. The syntax is:

```
<composite state reference> ::=
                        state substructure <composite state name> referenced <end>
```

4.144 Clerical error – Composite state reference – Clause 11.11.2: State aggregation, Concrete textual grammar

The identifiers for entry points should be for composite states rather than state partitions. The revised syntax is:

```
<outer entry point> ::=
                          <composite state identifier> via <point>
<inner entry point> ::=
                          <composite state identifier> via <point>
```

4.145 Clarification – Clause 11.11.2: State aggregation, Concrete graphical grammar

Replace <state aggregation area> by the rule:

```
<state aggregation area> ::=
                          <frame symbol> contains {
                                <state aggregation heading>
                                <composite state structure area>
                          is associated with {<graphical state connection point>* } set
                          is connected to { {<gate on diagram> | <external channel identifiers>}* }set
                          [ is associated with <package use area> ]
```

4.146 Clerical error – Allow empty body – Clause 11.11.2: State aggregation, Concrete graphical grammar

In the rule <state aggregation body area>, change "<connection definition area>" to "<state partition connection area>" and change "+" to "*" to make the repetition zero or more as follows:

```
<state aggregation body area> ::=
                          { { <state partition area> | <state partition connection area>}* }set
```

(The rule <connection definition area> is renamed <state partition connection area> – see below.)

4.147 Clarification – Clause 11.11.2: State aggregation, Concrete graphical grammar

In the rule <state partition area> change "<state partition reference area>" to "<composite state reference area>".

Change the rule name "<state partition reference area>" to "<composite state reference area>".

4.148 Clerical error – Identifier – Clause 11.11.2: State aggregation, Concrete graphical grammar

Modify the rule <inherited state partition definition> such that it refers to <composite state identifier> rather than to <state partition identifier>.

4.149 Clerical error – Connection points syntax – Clause 11.11.2: State aggregation, Concrete graphical grammar

Replace the rules:

```
<connection definition area> ::=
                          <solid association symbol> is connected to <graphical point>
```

<graphical point> ::=
<graphical state connection point>
| { <state entry points> | <state exit points> } ***is associated with*** <state partition area>
| <state connection point symbol> ***is connected to*** <frame symbol>

by the following where the rule <connection definition area> is renamed "<state partition connection area>":

<state partition connection area> ::=
<solid association symbol> ***is connected to***
[<outer graphical point> <inner graphical point>]

<outer graphical point> ::=
{ <state entry points> | <state exit points> } ***is associated with*** <frame symbol>

<inner graphical point> ::=
{ <state entry points> | <state exit points> } ***is associated with*** <state partition area>

4.150 Clerical error – <gate>s omitted – Clause 11.11.2: State aggregation, Concrete graphical grammar

It was omitted to allow gates in the graphical syntax of states (though they were allowed in the corresponding textual form). The rules <composite state reference area> and <graphical type based state partition definition> are updated as follows, and the text below is added after the rules:

<composite state reference area> ::=
<state symbol> ***contains*** { <<u>state</u> name> {<gate>*}***set*** }

<graphical typebased state partition definition> ::=
<state symbol> ***contains*** { <typebased state partition heading> {<gate>*}***set*** }

The <gate>s contained in <state symbol>s are placed near the border of the symbols and associated with the connection point to channels.

A <gate> is allowed in a <state symbol> of a <graphical typebased state partition definition> or <composite state reference area> of a <state partition area> only if the <state partition area> represents the state machine of an agent or agent type.

4.151 Clarification – Conditions – Clause 11.11.2: State aggregation, Concrete graphical grammar

The combination of conditions in the *Abstract grammar* and *Concrete graphical grammar* is not clear. The conditions should be moved to the *Abstract grammar* only.

In the *Concrete graphical grammar*, delete the conditions:

"The same state exit point of a state partition must not be connected to more than one state exit point of the enclosing state.

One state entry point of the enclosing state must not be connected to two different state entry points of the same state partition."

4.152 Clarifications – Clause 11.11.2: State aggregation, Semantics

In the first paragraph, third sentence, change "creation of the contained" to "creation of each contained" and change "*State-partition-**set*** and their connections" to "*State-partition* and its connections".

4.153 Clarification – Misplaced sentence – Clause 11.11.3: State connection point, Semantics

Delete the following sentence from 11.11.3 (additions to 11.11.1, and the text of 11.12.2.4 have the same effect):

"A <composite state> may only refer to its own <state entry point>s in labelled <start>s, and own <state exit point>s in labelled <return>s."

4.154 Extension – More flexible connect list – Clause 11.11.4: Connect

To allow a more flexible syntax, <connect list> and <asterisk connect list> are changed, and <state exit point list> is added as follows:

```
<connect list> ::=
                        <state exit point list>
                   |    <asterisk connect list>

<state exit point list> ::=
                        { <state exit point name> | default } { , { <state exit point name> | default}*

<asterisk connect list> ::=
                        <asterisk> [ ( <state exit point list> ) ]
```

The *Model* section is replaced by:

"**default** in a <state exit point list> represents an unlabelled <return>.

When the <connect list> of a certain <connect part> contains more than one <state exit point name>, a copy of the <connect part> is created for each such <state exit point name>. Then the <connect part> is replaced by these copies.

A <connect list> that contains an <asterisk connect list> is transformed into a list of <state exit point>s, one for each <state exit point> of the <composite state> in question (including the unlabelled <return>) except those mentioned in parentheses after the <asterisk>. The list of <state exit point>s is then transformed as described above."

4.155 Clerical error – Exception handling – Clause 11.12.1: Transition body, Concrete textual grammar

In the *Concrete textual grammar*, rename "<terminator statement>" and "<action statement>" to "<terminator>" and "<action>" respectively and eliminate the rules <action 1>, <action 2>, <terminator 1> and <terminator 2> to obtain the following syntax rules:

```
<transition> ::=
                             { <transition string> [ <terminator> ] }
                        |    <terminator>

<transition string> ::=
                        { <action> }+

<action> ::=
                        [ <label> ]
                        {         <task>
                             |    <output>
                             |    <create request>
                             |    <decision>
                             |    <set>
                             |    <reset>
                             |    <export>
                             |    <procedure call>
                             |    <remote procedure call>
                             |    <transition option> } <end>
```

```
<terminator > ::=
                              [<label>]
                              {          <return>
                                    |    <raise>
                                    |    <nextstate>
                                    |    <join>
                                    |    <stop>}end>
```

and replace "<terminator statement>" by "<terminator>" in the text after the syntax rules.

In the *Concrete graphical grammar*, change the rules as follows:

```
<transition area> ::=
                              [ <transition string area> is followed by ]
                              <terminator area>

<terminator area> ::=
                                    <state area>
                              |     <nextstate area>
                              |     <decision area>
                              |     <stop symbol>
                              |     <merge area>
                              |     <out connector area>
                              |     <return area>
                              |     <transition option area>
                              |     <raise area>

<transition string area> ::=
                              <action area>
                              [ is followed by <transition string area> ]

<action area> ::=
                                    <task area>
                              |     <output area>
                              |     <create request area>
                              |     <procedure call area>
                              |     <remote procedure call area>
```

4.156 Clerical error – Body of operators – Clause 11.12.1: Transition body, Concrete graphical grammar

Add to the end of the section the paragraph:

"A <transition area> in an <operation body area> shall not contain a <state area> or a <nextstate area>."

4.157 Clerical error – Annex F reference – Clause 11.12.1: Semantics

In the paragraph starting:

"A transition in one process of a block can …"

replace "by the rules in Annex F" with "by the rules given in the *Model* sections of this Recommendation".

4.158 Clarification – <action statement> to <action> – Clause 11.12.2.2: Join

In the first sentence, change "<action statement>" to "<action>" (see change 4.155).

4.159 Clarification – <agent body> – Clause 11.12.2.2: Join, Concrete textual grammar

To match clarification to refer to drop "type" from "bodies" change "<agent type body>" to "<agent body> in a type definition" in the sentence after the syntax.

4.160 Clarification – <agent body area> – Clause 11.12.2.2: Join, Concrete graphical grammar

To match PR and other clarifications change, "<state machine graph area>" to "<agent body area>" (twice) in the first sentence after the syntax.

4.161 Extension – Allow in procedures, etc. – Clause 11.12.2.3: Stop

See also 11.14 and 11.14.2. Allow stop in procedures, etc.

In the *Abstract grammar*, delete the condition "A *Stop-node* must not be contained in a *Procedure-graph*.".

In the *Semantics*, add at the end:

"The interpretation of a *Stop-node* in a *Procedure-graph* or *State-transition-graph* causes the agent interpreting that *Procedure-graph* to stop. Interpretation of the procedure, operation, compound statement, or composite state terminates and the stop propagates outwards to the caller and is treated as if a *Stop-node* were interpreted at the place of the procedure call, operation application, invocation of the compound statement, or entrance to the composite state. Termination propagates outwards until the containing agent is reached."

4.162 Clerical error – Sort of return – Clause 11.12.2.4: Return, Abstract grammar

Add, at the end of the *Abstract grammar*:

"The *Expression* of a *Value-return-node* must be sort compatible with the sort of the *Result* of the enclosing *Procedure*."

4.163 Clerical errors – PR and GR consistency – Clause 11.12.2.4: Return, Concrete grammars

Change <return> in the *Concrete textual grammar* to:

```
<return> ::=
                        return [<return body>] [ <end> <on exception> ]
```

Change <return area> in the *Concrete graphical grammar* to:

```
<return area> ::=
                        <return symbol>
                        [ is connected to <on exception association area> ]
                        [ is associated with <return body> ]
```

and add:

```
<return body> ::=
                             <expression>
                        |    via <state exit point>
```

4.164 Clerical error – Sort of return – Clause 11.12.2.4: Return, Concrete graphical grammar

Add at the end of the *Concrete graphical grammar*:

"The <expression> in <return> or <return area> shall not be omitted if the enclosing scope is an operator or method with an <operation result> or a value returning procedure with a <procedure result> without a <variable name>.

NOTE – If the <expression> is omitted in an operator or method with an <operation result> or a value returning procedure with a named <procedure result>, the model in 9.4 adds the procedure result variable as the <expression>."

4.165 Clerical error – Sort of return – Clause 11.12.2.4: Return, Semantics

Change:

"c) If a *Value-return-node* is interpreted, the result of *Expression* is returned to the calling context."

to:

"c) If a *Value-return-node* is interpreted, the result of *Expression* is interpreted in the same way as an *Expression* assigned to a variable with the sort of the result (see 12.3.3), but without the result being associated with a variable or a range check taking place; then, the object or value result is returned to the calling context."

4.166 Clerical error – Exception handling – Clause 11.13.1: Task, Concrete textual grammar

Change the rule <task> to:

```
<task> ::=
                    task <textual task body> [ <end> <on exception> ]
```

4.167 Clarification and modification – Clause 11.13.1: Task

In the *Concrete textual grammar*, change the last alternative of <textual task body> to:

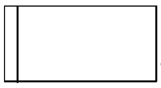
```
                |   [ <comment body> ] <left curly bracket> <statement list> <right curly bracket>
```

In the *Concrete graphical grammar*, change <task area> to:

```
<task area> ::=
                    <task symbol> contains <graphical task body>
                    [ is connected to <on exception association area> ]
                |   <macro symbol> contains { <macro name> [<macro call body>] }
```

and add, at the end of the rules:

"<macro symbol> ::=

"

In the *Concrete graphical grammar*, insert at end before the Semantics:

"A <task area> or <task> containing a single <assignment> represents an *Assignment* or *Assignment-attempt*.

A <task area> or <task> containing a single <set> represents a *Set-node*.

A <task area> or <task> containing a single <reset> represents a *Reset-node*.

A <task area> or <task> containing any other <statement list> represents a *Compound-node*. The *Connector-name* is represented by a newly created anonymous name. The *Variable-definition-**set*** is represented by the list of all <variable definition>s in <statement list>. The *Transition* is represented by the transform of <statements> in <statement list>, or by the transform of <statements> in <statement list> followed by a *Break-node* with *Connector-name*, if the <statement list> is not terminating."

In the *Semantics*, insert after first paragraph a new paragraph:

"The interpretation of a *Compound-node* is given in 11.14.1. The interpretation of *Assignment* and *Assignment-attempt* is given in 12.3.3. The interpretation of *Set-node* and *Reset-node* is given in 11.15."

In the *Model*, replace all the existing paragraphs including the Note by:

"If the <statement list> of <textual task body> or <graphical task body> is empty, the <task> or <task area>, respectively, is removed. Any syntactic item leading to such an empty <task> or <task area> shall then lead directly to the item following the <task> or <task area>, respectively.

A <macro symbol> containing <macro name> [<macro call body>] is transformed into a <task symbol> containing **macro** <macro name> [<macro call body>] <end>."

4.168 Clerical error – Exception handling – Clause 11.13.2: Create, Concrete textual grammar

Change the rule <create request> to:

```
<create request> ::=
                              create <create body> [ <end> <on exception> ]
```

4.169 Clarification – Static determination of instance sets – 11.13.2, Create, Model

Item c) of the *Model* implied that instance sets can be created dynamically, in which case the implicit channels to the instance set could not be derived in the same way as static instance sets. Instead it is statically determined that an instance may be created, and therefore if no instance set is explicitly given, an instance set (with no initial instances) is implied. The text for item c) of the model is changed to:

"c) If there is no instance set of the indicated agent type in the containing agent, then:
   - i) an instance set of the given type with a unique name is implied in the containing agent; and
   - ii) the <agent type identifier> in the <create request> is the derived syntax for this implicit instance set."

4.170 Clerical error – Operation/procedure inconsistency – Clause 11.13.3: Procedure call, Abstract grammar

Change:

"Each *Expression* corresponding by position to an *Inout-parameter* or *Out-parameter* must be a *Variable-identifier* which is sort compatible to the sort identified by the *Sort-reference-identifier* of the *Procedure-formal-parameter*."

to:

"Each *Expression* corresponding by position to an *Inout-parameter* or *Out-parameter* must be a *Variable-identifier* with the same *Sort-reference-identifier* as the *Procedure-formal-parameter*."

4.171 Clerical error – Exception handling – Clause 11.13.3: Procedure call, Concrete textual grammar

Change the rule <procedure call> to:

```
<procedure call> ::=
                              call <procedure call body> [ <end> <on exception> ]
```

4.172 Clerical error – Operation/procedure inconsistency – Clause 11.13.3: Procedure call, Semantics

The *Semantics* section is replaced by:

"The interpretation of a procedure *Call-node* or *Value-returning-call-node* interprets the actual parameter expressions in the order given. If no exceptions are raised by the parameter interpretation, interpretation is then transferred to the procedure definition referenced by the *Procedure-identifier*, and that procedure graph is interpreted (the explanation is contained in 9.4).

If an <expression> in <actual parameters> is omitted, the corresponding formal parameter has no data item associated; that is, it is "undefined".

If an argument sort of the *Call-node* or *Value-returning-call-node* for an *In-parameter* or *Inout-parameter* of the procedure is a syntype, the range check defined in 12.1.9.5 is applied to the result of the *Expression*. If the range check is the predefined Boolean value False at the time of interpretation, then the predefined exception OutOfRange (see D.3.16) is raised instead of interpreting further actual parameters or the procedure definition.

If OutOfRange is not raised, the interpretation of the transition containing a *Call-node* continues when the interpretation of the called procedure is finished.

If OutOfRange is not raised, the interpretation of the transition containing a *Value-returning-call-node* continues when the interpretation of the called procedure is finished. The result of the called procedure is returned by the *Value-returning-call-node*.

A *Value-returning-call-node* has a sort, which is the sort of the result obtained by the interpretation of the procedure.

If the result sort of a value returning procedure call is a syntype, the range check defined in 12.1.9.5 is applied to the result of the procedure call. If the range check is the predefined Boolean value False at the time of interpretation, then the predefined exception OutOfRange (see D.3.16) is raised."

4.173 Clerical error – Exception handling – Clause 11.13.4: Output, Concrete textual grammar

Change the rule <output> to:

```
<output> ::=
                        output <output body> [ <end> <on exception> ]
```

4.174 Extension – Inline data types – Clause 11.13.4: Output, Concrete textual grammar

Change the rule <destination> to:

```
<destination> ::=
                        <pid expression0> | <agent identifier> | this
```

Change each corresponding "<pid expression>" to "<pid expression0>" in the text of the *Concrete textual grammar* (5 times), and in the text of *Semantics* (once).

4.175 Error correction – Evaluation of decision answers – Clause 11.13.5: Decision

Change the following sentence in the *Abstract grammar*:

"The *Range-condition*s of the *Decision-answer*s must be mutually exclusive, and the *Constant-expression*s of the *Range-condition*s must be of a compatible sort."

to:

"The *Constant-expression*s of the *Range-condition*s must be of a compatible sort."

4.176 Clarification – <terminator/action statement> to <terminator/action> – Clause 11.13.5: Decision, Concrete textual grammar

In the first paragraph after the syntax, change "<terminator statement>" to "<terminator>" and change "<action statement>" to "<action>" (see change 4.155).

4.177 Error correction – Evaluation of decision answers – Clause 11.13.5: Decision

In the first paragraph of the *Semantics*, change the sentence:

"A decision transfers the interpretation to the outgoing path whose range condition contains the result given by the interpretation of the question."

to:

"A decision transfers the interpretation to an outgoing path whose *Range-condition* contains the result given by the interpretation of the question. The determination of whether the *Decision-question* is contained in each *Decision-answer* is carried out once for each *Decision-answer* in an arbitrary order until a *Range-condition* containing the *Decision-question* is identified, or until this determination requires interpretation of an operation application that raises an exception, or an *Informal-text* is chosen."

4.178 Clerical error – Wrong syntax – Clause 11.13.5: Decision, Model

Change the specification of the anonymously defined syntype as follows:

```
syntype data_type_N =
    <<package Predefined>>Integer constants 1:N
endsyntype;
```

4.179 Extension – Allow in procedures, etc. – Clause 11.14: Statement list, Concrete textual grammar

See also 11.12.2.3 and 11.14.2. Allow stop in procedures, etc.

Add an alternative <stop statement> to <terminating statement>.

4.180 Clerical error – Align with Annex F – Clause 11.14.1: Compound statement, Abstract grammar

Add an optional *Exception-handler-node* to *Compound-node*:

```
Compound-node          ::   Connector-name
                            Variable-definition-set
                            [ Exception-handler-node ]
                            Init-graph-node*
                            Transition
                            Step-graph-node*
```

4.181 Clerical error – Incorrect model – Clause 11.14.1: Compound statement, Model

The *Model* was inconsistent with the direct semantics for <compound statement>.

Remove the Note:

"NOTE – The transformed non-empty <statement list> becomes a list of <action>s and <terminator statement>s separated by semicolons and ending in a semicolon and therefore can be treated as a <transition>."

and change the sentence:

"If the <statement list> is empty, the result of its transformation is the empty text."

into the Note:

"NOTE – If the <statement list> is empty, it is represented by a *Break-node* as explicated in the *Concrete graphical grammar*."

4.182 Error correction – Allow remote procedure – Clause 11.14.2: Transition actions and terminators as statements, Concrete textual grammar

Remote procedure calls have been omitted in error. Change the rule <call statement> to:

```
<call statement> ::=
                    call { <procedure call body> | <remote procedure call body> }
```

4.183 Clarification – <action statement> to <action> – Clause 11.14.2: Transition actions and terminators as statements

In the first paragraph, change "<action statement>" to "<action>" (see change 4.155).

4.184 Clerical error – Exception handling – Clause 11.14.2: Transition actions and terminators as statements, Concrete textual grammar

Change the following rules as given below:

```
<algorithm action statement> ::=
                          output <output body> <end>
                    |     create <create body> <end>
                    |     set <set body> <end>
                    |     reset <reset body> <end>
                    |     export <export body> <end>
<return statement> ::=
                          return <return body> <end>
<raise statement> ::=
                          raise <raise body> <end>
```

4.185 Extension – Allow stop in procedures, etc. – Clause 11.14.2: Transition actions and terminators as statements

See also 11.12.2.3 and 11.14. Allow stop in procedures, etc.

In the *Concrete textual grammar*, add a production for <stop statement>:

```
<stop statement> ::=
                          stop <end>
```

In the *Model*, change the last paragraph starting "The transform of ..." to:

"The transform of an <algorithm action statement>, a <return statement>, a <stop statement>, and <raise statement> is obtained by dropping the trailing <end>."

4.186 Clarification – Clause 11.14.4: If statement, Model

Simplification of use the *Model* for <decision statement>.

Change:

"The <if statement> is equivalent to the following <action statement> involving a <decision>:

```
decision <Boolean expression> ;
    ( true ) : task { <consequence statement>-transform };
    ( false ) : task { <alternative statement>-transform };
 enddecision ;
```

The transform of <alternative statement> is only inserted if <alternative statement> was present."

to:

"The <if statement> is equivalent to the following <decision statement>:

```
decision ( <Boolean expression> ) {
    ( true ) : <consequence statement>
    ( false ) : <alternative statement>
}
```

If <alternative statement> was not present, an <empty statement> is inserted in its place."

4.187 Clerical error – Clause 11.14.5: Decision statement

The alternative **any** was omitted in error. The syntax should be:

```
<decision statement> ::=
                        decision ( { <expression> | any } ) [ <comment body> ] <left curly bracket>
                             <decision statement body>
                        <right curly bracket>
```

4.188 Clerical error – Keyword – Clause 11.14.6: Loop statement, Concrete textual grammar

See also 5.1, 6.1 and D.2.1.

The keyword should be **loop**.

Replace "**for**" by "**loop**" in <loop statement>.

4.189 Clerical error – Clause 11.14.9: Exception statement, Concrete textual grammar

Replace "*Local-scope-node*" by "*Compound-node*". (wrong reference)

4.190 Clerical error – Exception handling – Clause 11.15: Timer, Concrete textual grammar

Replace the rules <reset> and <set> by the following four rules:

```
<reset> ::=
                        reset <reset body> [ <end> <on exception> ]
<reset body> ::=
                        ( <reset clause> { , <reset clause> }* )
<set> ::=
                        set <set body> [ <end> <on exception> ]
<set body> ::=
                        <set clause> { , <set clause> }*
```

4.191 Clerical error – Exception handling – Clause 11.16.2: On-exception, Concrete textual grammar

Remove the "<end>" from the rule <on exception> to obtain:

```
<on exception> ::=
                    onexception <exception handler name>
```

4.192 Clarification – <action statement> to <action> – Clause 11.16.3: Handle, Model

In the penultimate paragraph, change "<action statement>" to "<action>" (see change 4.155).

4.193 Clerical error – Missing parts, align with Annex F – Clause 12.1: Data definition, Abstract grammar

Change the following rules by adding "*Data-type-definition-**set***", "*Syntype-type-definition-**set***", and "*Exception-definition-**set***", to obtain:

Value-data-type-definition	::	*Sort*
		Data-type-identifier
		*Literal-signature-**set***
		*Static-operation-signature-**set***
		*Dynamic-operation-signature-**set***
		*Data-type-definition-**set***
		*Syntype-definition-**set***
		*Exception-definition-**set***
Object-data-type-definition	::	*Sort*
		Data-type-identifier
		*Literal-signature-**set***
		*Static-operation-signature-**set***
		*Data-type-definition-**set***
		*Syntype-definition-**set***
		*Exception-definition-**set***
Interface-definition	::	*Sort*
		*Data-type-identifier**
		*Data-type-definition-**set***
		*Syntype-definition-**set***
		*Exception-definition-**set***

Change the following rules in order to distinguish between expanded and referenced sorts:

Sort-reference-identifier	=	*Sort-identifier*
	|	*Syntype-identifier*
	|	*Expanded-sort*
	|	*Reference-sort*
Sort-identifier	=	*Identifier*
Expanded-sort	::	*Sort-identifier*
Reference-sort	::	*Sort-identifier*

4.194 Extension – Inline data types – Clause 12.1: Data definitions, Concrete textual grammar

Replace the rules <sort> and <basic sort> by:

```
<sort> ::=
                    <basic sort> [ ( <range condition> ) ]
            |       <anchored sort>
            |       <expanded sort>
            |       <reference sort>
            |       <pid sort>
            |       <inline data type definition>
            |       <inline syntype definition>

<inline data type definition> ::=
                    { value | object } [<data type specialization>]
                    [ <left curly bracket> <data type definition body> <right curly bracket> ]
            |       { value | object } [<data type specialization>]
                    <end> <data type definition body> <data type closing> <end>

<inline syntype definition> ::=
                    syntype <basic sort>
                    [ <left curly bracket>
                           { <default initialization> [ [<end>] <constraint> ] | <constraint> } <end>
                     <right curly bracket> ]
            |       syntype <basic sort>
                           { <default initialization> [ [<end>] <constraint> ] | <constraint> } <end>
                     <syntype closing> <end>

<basic sort> ::=
                    <datatype type expression>
            |       <syntype>
```

4.195 Error correction – Value/object for anchored sort – Clause 12.1: Data definitions, Concrete textual grammar

It is not allowed to specify directly that <anchored sort> is always inherited as the value or as the object variant of sort of the data type. This is facilitated by the following changes:

```
<expanded sort> ::=
      value { <basic sort> | <anchored sort> }

<reference sort> ::=
      object { <basic sort> | <anchored sort> }
```

4.196 Extension – Inline data types – Clause 12.1: Data definitions, Model

Add the following paragraphs at the end of the section.

"A <sort> that is a <basic sort> with a <range condition> is derived concrete syntax for a <syntype> of an implied <syntype definition> having an anonymous name. This anonymously named <syntype definition> is defined with its elements restricted by the <range condition>, if the <basic sort> has been constructed using the literal data type constructor; otherwise, the <range condition> is part of a <size constraint>.

An <inline data type definition> is derived concrete syntax for a <basic sort> of an implied <data type definition> having an anonymous name. This anonymously named <data type definition> is derived from the <inline data type definition> by inserting **type** and the anonymous name after **value** or **object** in the <inline data type definition>. Each <inline data type definition> defines a different implied <data type definition>.

An <inline syntype definition> is derived concrete syntax for for a <basic sort> of an implied <syntype definition> having an anonymous name. This anonymously named <syntype definition> derived from the <inline syntype definition> by inserting the anonymous name and <equals sign> after syntype in the <inline syntype definition>."

4.197 Clarification – Clause 12.1.1: Data type definition

To clarify the meaning of data type construction with respect to literal values, insert in 12.1.1 just after the heading:

"A data type definition has a body that usually contains a data type constructor and an indication whether the data type is a **value** or **object** data type.

The data type constructor defines how to construct sets of values (structured values, literal values, and choice values). If the data type definition is a **value** type, these values are the elements of the sort. If the data type definition is an **object** type, these values are what is referenced by the elements of the sort."

4.198 Syntax error – <end> – Clause 12.1.1: Data type definitions, Concrete textual grammar

See also 12.1.9.4 and. 12.3.3.2.

Change the rule <data type definition body> to:

```
<data type definition body> ::=
                              {<entity in data type>}* [<data type constructor>] <operations>
                              [<default initialization> <end> ]
```

4.199 Clerical error – Missing context parameters – Clause 12.1.1: Data type definitions, Concrete textual grammar

Change the rule <data type reference> to:

```
<data type reference> ::=
                              <type preamble>
                              { value | object } type <type reference heading> <type reference properties>
```

followed by the new paragraph:

"A <type reference heading> that is part of a <data type reference> must have a <data type name>."

4.200 Clarification – Clause 12.1.1: Data type definitions, Concrete textual grammar

Add at the end:

"For each <operation signature> of <operation signatures> there shall be one and only one corresponding definition (<operation definition> or <textual operation reference> or <external operation definition>) in the <operation definitions> of the <operations>."

4.201 Clarification – Clause 12.1.1: Data type definitions, Concrete graphical grammar

Replace "<graphical type reference heading> that contains" by "<type reference heading> with" to obtain the following revised paragraph:

"The <type reference area> that is part of a <data type reference area> must have a "<type reference heading> with a <data type name>."

4.202 Clerical error – Extra vertical bars – Clause 12.1.2: Interface definition, Concrete textual grammar

Remove the vertical bars for alternatives before <virtuality> in:

```
                |     {<package use clause>}*
                |     [<virtuality>] <interface heading>
                      [<interface specialization>] <end>
                |     {<package use clause>}*
                |     [<virtuality>] <interface heading>
```

to obtain the text:

```
                |     {<package use clause>}*
                      [<virtuality>] <interface heading>
                      [<interface specialization>] <end>
                |     {<package use clause>}*
                      [<virtuality>] <interface heading>
```

4.203 Clerical error – Missing context parameters – Clause 12.1.2: Interface definition, Concrete textual grammar

Change the rule <interface reference> to:

```
<interface reference> ::=
                      [<virtuality>]
                      interface <type reference heading> <type reference properties>
```

followed by the new paragraph:

"A <type reference heading> that is part of a <interface reference> must have an <interface name>."

4.204 Clarification – Clause 12.1.2: Interface definition, Concrete graphical grammar

Replace "<graphical type reference heading> that contains" by "<type reference heading> with" to make the revised paragraph:

"The <type reference area> that is part of a <interface reference area> must have a "<type reference heading> with a <interface name>."

4.205 Clarification – Interfaces – Clause 12.1.2: Interface definition, Semantics

To define interface inheritance, the content of an interface needs to be defined.

In the *Semantics*, insert after the first paragraph the following new paragraph:

"The content of an interface is the set of all signals, remote procedures and remote variables that are defined in an <entity in interface> of the interface or referenced in the <interface use list> or included in the interface by specialization (that is, inheritance or context parameterization)."

However, in 12.1.2 interface definition should not have *Semantics* here as it has no *Abstract grammar* here. Therefore change the heading "*Semantics*" to "*Model*" and delete the heading "*Model*".

4.206 Clarification – Implicit interface – Clause 12.1.2: Interface definition, Model

See also 12.1.6.

The model did not work correctly. Replace the *Model* text by:

"Interfaces are implicitly defined by agent and agent type definitions and by the state machines of agent and agent type definitions. The implicitly defined interface for an agent or on agent type has the same name and are defined in the same scope unit as the agent or agent type that defined it. The implicitly defined interface for a state machine has the same name as the containing agent or agent type but is defined in the same scope unit as the state machine that defined it, i.e. inside the agent or agent type.

NOTE 1 – Because every agent and agent type has an implicitly defined interface with the same name, any explicitly defined interface must have a different name from every agent and agent type defined in the same scope; otherwise, there are name clashes.

The interface defined by an agent or agent type contains in its <interface specialization> all interfaces given in the incoming signal list associated with explicit or implicit gates of the agent or agent type such that the gates are connected via implicit or explicit channels to the gates of the state machine of the agent or agent type. The interface also contains in its <interface use list> all signals, remote variables and remote procedures given in the incoming signal list associated with explicit or implicit gates of the agent or agent type such that the gates are connected via implicit or explicit channels to the gates of the state machine of the agent or agent type. In addition the interface for an agent type that inherits another agent type also contains in its <interface specialization> the implicit interface defined by the inherited agent type.

The interface defined by the state machine of an agent or agent type contains in its <interface specialization> the interface defined by the agent or agent type itself. In addition the interface contains in its <interface specialization> all interfaces given in the incoming signal list associated with explicit or implicit gates of the state machine such that gates are not connected by implicit or explicit channels to explicit or implicit gates of the agent or agent type. The interface also contains in its <interface use list> all signals, remote variables and remote procedures given in the incoming signal list associated with explicit or implicit gates of the state machine such that gates are not connected by implicit or explicit channels to explicit or implicit gates of the agent or agent type. If the containing entity is an agent type that inherits another agent type, then the interface will also contain in its <interface specialization> the implicit interface of the state machine of the inherited agent type.

The interface defined by a type based agent contains in its <interface specialization> the interface defined by its type.

NOTE 2 – To avoid cumbersome text, the convention is used that the phrase "the pid sort of the agent A" is often used instead of "the pid sort defined by the interface implicitly defined by the agent A" when no confusion is likely to arise."

4.207 Clerical error – Align with Annex F – Clause 12.1.4: Operations, Abstract grammar

Add *Identifier* to:

```
Operation-signature          ::    Operation-name
                                   Formal-argument*
                                   [Result]
                                   Identifier
```

and add the paragraph after the grammar rules:

"The *Identifier* in a operator signature is an anonymous identifier for the anonymous procedure corresponding to the operation."

4.208 Clerical error – Syntax ambiguity – Clause 12.1.4: Operations, Concrete textual grammar

Change the rule <operation preamble> to:

```
<operation preamble> ::=
                                        [ <virtuality> [<visibility>] | <visibility> [<virtuality>] ]
```

4.209 Clerical error – Placement of text – Clause 12.1.5: Any

Delete the last two lines before the heading for *Concrete textual grammar*.

Move all the remaining text, except the first paragraph, of the non-normative explanatory section to the beginning of the *Semantics* section.

4.210 Clarification – Implicit interfaces – Clause 12.1.6: Pid and pid sorts, Semantics

See also 12.1.2.

Change:

"b) the signal is contained on an inwards gate connected to the state machine of the agent that implicitly defined the interface; or

c) the compatibility check …"

to:

"b) the compatibility check …".

4.211 Clerical error – Clause 12.1.7.3: Choice data types, Model

In item f), delete the word "an" before "<operation signature>s".

4.212 Clerical error – Optional sorts – Clause 12.1.8: Behaviour of operations, Concrete graphical grammar

Delete the paragraph:

"For each <operation diagram>, there must exist an <operation signature> in the same scope unit … Likewise, the <sort> may be omitted in <operation result>."

4.213 Clerical error – Clause 12.1.8: Behaviour of operations, Semantics

Change:

"An <external operation definition> is an operator or method whose behaviour is not included in the SDL description. Conceptually, an <external operation definition> is assumed to be given behaviour and to be transformed into an <operation definition> as part of the generic system transformation (see Annex F)."

to:

"An <external operation definition> is an operator or method whose behaviour is not included in the SDL description (see clause 13)."

4.214 Extension – Range sign – Clause 12.1.9.1: Name class, Concrete textual grammar

See also 6.1 and 12.1.9.5.

Change the rule <regular interval> to:

```
<regular interval> ::=
                    <character string> { <colon> | <range sign> } <character string>
```

4.215 Syntax error – <end> – Clause 12.1.9.4: Syntypes, Concrete textual grammar

See also 12.1.1 and. 12.3.3.2.

Change the first alternative in the production for <syntype definition> and remove the vertical bar before "<type preamble>" as an alternative to obtain:

```
<syntype definition> ::=
               {<package use clause>}*
               syntype <syntype name> <equals sign> <parent sort identifier>
                    { { <default initialization> [ [<end>] <constraint> ] | <constraint> } <end>
                      <syntype closing>
                    | [<syntype closing>] } <end>
          |    {<package use clause>}*
               <type preamble> <data type heading> [<data type specialization>]
```

4.216 Clerical error – Omitted alternative – Clause 12.1.9.4: Syntypes, Concrete textual grammar

Add the alternative omitted in error in <syntype definition> as follows:

```
          |    syntype <syntype name> <equals sign> <parent sort identifier>
               <left curly bracket>
                    [ { <default initialization> [ [<end>] <constraint> ] | <constraint> } <end> ]
               <right curly bracket>
```

4.217 Modification – Parsing difficulty – Clause 12.1.9.4: Syntypes, Concrete textual grammar

Change the second production for <syntype definition> as follows:

```
          |    {<package use clause>}*
               <type preamble> <data type heading> [<data type specialization>]
               {     <end> <data type definition body> <constraint> <end> <data type closing> <end>
                 |   <left curly bracket> <data type definition body> <constraint> <end>
                     <right curly bracket> }
```

4.218 Clerical error – Ambiguity – Clause 12.1.9.4: Syntypes, Concrete textual grammar

Add the following paragraph after the syntax at the end of *Concret textual grammar*:

"If a <constraint> could be interpreted as either belonging to the <default initialization> or the <syntype definition>, it shall be considered part of the <default initialization>."

4.219 Extension – Range sign – Clause 12.1.9.5: Constraint, Concrete textual grammar

See also 6.1 and 12.1.9.1.

Change the rule <closed range> to:

```
<closed range> ::=
                        <constant> { <colon> | <range sign> } <constant>
```

4.220 Clerical error – Clause 12.1.9.6: Synonym definition, Concrete textual grammar

Change:

"An <external synonym definition item> defines a <synonym> whose result is not defined in a specification. Conceptually, an <external synonym definition item> is assumed to be given a result and to be transformed into <internal synonym definition item> as part of the generic system transformation (see Annex F)."

to:

"An <external synonym definition item> defines a <synonym> whose result is not defined in a specification (see clause 13)."

4.221 Clerical error – Clause 12.2.1: Expressions, Abstract grammar

Add "*State-expression*" as an alternative to "*Active-expression*".

4.222 Extension – Inline data types – Clause 12.2.1: Expressions, Concrete textual grammar

Replace "<expression>" by the following two rules:

```
<expression> ::=
                        <expression0>
                  |     <range check expression>
<expression0> ::=
                        <operand>
                  |     <create expression>
                  |     <value returning procedure call>
```

Delete "<range check expression>" in <operand 2> to obtain the rule:

```
<operand2> ::=
                        <operand3>
                  |     <operand2> { <greater than sign>
                                   | <greater than or equals sign>
                                   | <less than sign>
                                   | <less than or equals sign>
                                   | in } <operand3>
                  |     <equality expression>
```

Change "<constant expression>" to:

```
<constant expression> ::=
                        <constant expression0>
```

Change "An <expression> that …" to "An <expression0> that …"

4.223 Clerical error – Clause 12.2.1: Expressions, Concrete textual grammar

In the first paragraph, change:

"An <expression> that does not contain any <active primary>, a <create expression>, or a <value returning procedure call> with a <remote procedure call body> is a <constant expression>. A <constant expression> represents a *Constant-expression* in the abstract syntax. ..."

to obtain the following, with "constant" underlined:

"An <expression> that does not contain any <active primary>, a <create expression>, or a <value returning procedure call> is a <constant expression>. A <constant expression> represents a *Constant-expression* in the abstract syntax. ..."

Underline "constant" in the second paragraph as follows:

"An <expression> that is not a <constant expression> represents an *Active-expression*."

4.224 Clerical error – Spurious sentence – Clause 12.2.7: Operation application, Concrete textual grammar

Delete the paragraph "If **this** is used, <operation identifier> must denote an enclosing operator or method."

4.225 Clarification – Parameter handling – Clause 12.2.7: Operation application, Semantics

Delete the paragraph starting "If an <expression> in <actual parameters> is omitted, …"

4.226 Error correction – Restriction on parameters – Clause 12.2.7: Operation application, Model

To allow a method to be applied to a primary that is not a variable, the following needs to be added to the *Model* section:

"If the <primary> of a <method application> is not a variable or **this**, there is an implicit assignment of the <primary> to an implicit variable with the sort of the first parameter of the operation (that is the method sort). The assignment is placed before the action in which the <method application> occurs. The implicit variable replaces the <primary> in the <method application>."

4.227 Error correction – Rules too restrictive for data initialization – Clause 12.3.1: Variable definition

The rules for initialization were too strict and prevented sensible initialization of objects.

In the *Abstract grammar* condition:

"If the Constant-expression is present ..."

is changed to:

"If the *Constant-expression* is present, it must be one of following:

1) the same sort as the Sort-reference-identifier denoted; or
2) if the denoted sort is an object sort OS the sort denoted by value OS; or
3) if the denoted sort is a value sort VS the sort denoted by object OS."

Change the first paragraph of the *Semantics* to:

"When a variable is created and the *Constant-expression* is present, the variable is associated with:

1) the result of the *Constant-expression*, if the sort of the variable and the *Constant-expression* are the same;
2) an object (distinct from any other object) that references the *Constant-expression*, if the variable has an object sort and the *Constant-expression* has a value sort;
3) the value referenced by the *Constant-expression*, if the variable has an value sort and the *Constant-expression* has a object sort.

Otherwise, if no *Constant-expression* applies, the variable has no data item associated: that is, the variable is "undefined"."

4.228 Syntax error – <end> – Clause 12.3.3.2: Default initialization, Concrete textual grammar

See also 12.1.9.4 and 12.1.1.

Change the rule for <default initialization> as follows:

```
<default initialization> ::=
                              default [<virtuality>] [<constant expression>]
```

4.229 Clerical error – Clause 12.3.4.6: State expression

See also the *Abstract grammar* in 12.2.1, Expressions.

Replace the content of the whole section with the following text:

"Abstract grammar

State-expression :: *()"*

Concrete textual grammar

```
<state expression> ::=
                              state
```

Semantics

A state expression denotes the Charstring literal that contains the spelling of the name of the most recently entered state of the nearest enclosing scope unit. If there is no such state, <state expression> denotes the empty string (")."

4.230 Clerical error – Omitted and duplicated items – Clause 13.1: Optional definition, Concrete textual grammar

Add "<agent reference>", "<signal reference>", and "<interface reference>" as alternatives to the production for <select definition>.

Remove the alternative for "<textual typebased state partition definition>" from <select definition>.

4.231 Clarification – Clause 13.1: Optional definition, Concrete textual grammar

In the rule <select definition>, change "<state partitioning>" to "<state partition>".

4.232 Clarification – <action statement> to <action> – Clause 13.2: Optional transition string, Model

Change "<action statement>" to "<action>" in items a) and b) (see change 4.155).

4.233 Clarification – <terminator statement> to <terminator> – Clause 13.2: Optional transition string, Model

Change "<terminator statement>" to "<terminator>" in item a) (see change 4.155).

4.234 Clerical error – Quantification keyword – Clause D.2.1: Axioms, Concrete textual grammar

See also 5.1, 6.1 and D.2.1.

Add the following Note after the rule <quantification>:

"NOTE – **for** is considered an SDL keyword for the purpose of this annex."

4.235 Clarification – Remove implicit quantification – Clause D.2.1: Axioms, Concrete textual grammar

Replace:

"c) a <value identifier> if there is a definition of that name in a <quantification> of <quantified equations> enclosing the <term>, which then must have a suitable sort for the context; otherwise

d) a <value identifier> which has an implied quantified equation for the <unquantified equation>.

Two or more occurrences of the same unbound value identifier within one <unquantified equation> (or in case the <unquantified equation> is contained in a <conditional equation>, within the <conditional equation>), imply one <quantification>.

Within one <unquantified equation> (or in case the <unquantified equation> is contained in a <conditional equation>, within the <conditional equation>), there must be exactly one sort for each implicitly quantified value identifier which is consistent with all its uses."

with:

"c) a <value identifier> if there is a definition of that name in a <quantification> of <quantified equations> enclosing the <term>, which then must have a suitable sort for the context."

4.236 Clarification – Informal text not used – Clause D.2.1: Axioms, Semantics

Delete the paragraph:

"If any axioms contain informal text, then the interpretation of expressions is not formally defined … specification has not been given in SDL."

4.237 Clarification – Remove implicit quantification – Clause D.2.1: Axioms, Semantics

Delete the sentences:

"The sort of the implied quantifications is the sort required by the context(s) of the occurrence of the unbound identifier. If the contexts of a value identifier that has implied quantification allow different sorts, the identifier is bound to a sort that is consistent with all its uses in the equation."

4.238 Clarification – Case of operator names – Clause D.3: Package predefined

The operator names "Make", "Extract" and "Modify" should be modified throughout the annex:

to start with an upper case letter rather than with a lower case letter.

4.239 Clerical error – Missing inherits – Clause D.3.13.2: Usage

Insert missing **inherits** in:

```
value type Boolset inherits Bag< Boolean > endvalue type Boolset;
```

4.240 Clerical error – Upper and lower case keywords – Appendix III

In Appendix III, modify:

"a) Replace all keywords with the corresponding <lowercase keyword> (or <uppercase keyword>);"

by replacing "<lowercase keyword>" with "lowercase <keyword>" and "<uppercase keyword>" with "uppercase <keyword>" to obtain the following text:

"a) Replace all keywords with the corresponding lowercase <keyword> (or uppercase <keyword>);"

and modify item d) to obtain the text:

"d) If <name>s conflict with lowercase <keyword>s, replace the first character with an uppercase character."

4.241 Clarifications – Appendix III

In item 1 b), change "<national>" to "national".

4.242 Clarifications – Appendix III

Delete "the" before "one of the duplicate transitions".

Geneva, 2002