

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

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

Z.103

(04/2016)

SERIES Z: LANGUAGES AND GENERAL SOFTWARE
ASPECTS FOR TELECOMMUNICATION SYSTEMS

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

Specification and Description Language – Shorthand notation and annotation in SDL-2010

Recommendation ITU-T Z.103

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
User Requirements Notation (URN)	Z.150–Z.159
Testing and Test Control Notation (TTCN)	Z.160–Z.179

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-machine interfaces for the management of telecommunications networks	Z.360–Z.379

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

Processing environment architectures	Z.600–Z.609
--------------------------------------	-------------

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

Recommendation ITU-T Z.103

Specification and Description Language – Shorthand notation and annotation in SDL-2010

Summary

Scope/Objective

Recommendation ITU-T Z.103 defines the shorthand and annotation features of the Specification and Description Language. Together with Recommendations ITU-T Z.100, ITU-T Z.101, ITU-T Z.102, ITU-T Z.104, ITU-T Z.105, ITU-T Z.106 and ITU-T Z.107, this Recommendation is part of a reference manual for the language. The language defined in this document covers features of the language not included in Basic SDL-2010 in Recommendation ITU-T Z.101 or Comprehensive SDL-2010 in Recommendation ITU-T Z.102. Features defined in this Recommendation either do not have their own abstract grammar and are transformed to concrete grammar defined by Recommendations ITU-T Z.101, ITU-T Z.102 and ITU-T Z.104 (and ITU-T Z.107 for object-oriented data), or are annotations with no formal meaning.

Coverage

The Specification and Description Language has concepts for behaviour, data description and (particularly for larger systems) structuring. The basis of behaviour description is extended finite state machines communicating by messages. Data description is based on data types for values and objects. The basis for structuring is hierarchical decomposition and type hierarchies. A distinctive feature of the Specification and Description Language is the graphical representation. This Recommendation covers the features of the language such as shorthand and alternative graphical concrete syntax and macros that make SDL-2010 easier and more practical to use. The concrete grammar given is the graphical representation. The alternative textual programming representation is given in Recommendation ITU-T Z.106. This Recommendation does not provide a canonical syntax, but by applying the *Model* descriptions given a specification can be transformed to Comprehensive SDL-2010 defined in ITU-T Z.102, or (if no Comprehensive SDL-2010 features are used) to Basic SDL-2010 as defined in SDL-2010. It should be noted that in both Basic and Comprehensive SDL-2010 the details of expressions, data definitions and action language, are defined in Recommendation ITU-T Z.104 (and ITU-T Z.107 for object-oriented data).

Applications

The Specification and Description Language is applicable within standard bodies and industry. The main application areas for which the Specification and Description Language has been designed are stated in ITU-T Z.100, but the language is generally suitable for describing reactive systems. The range of application is from requirement description to implementation. The features of the language defined in Recommendation ITU-T Z.103 are cumulative with, and rely on, the features defined in Recommendations ITU-T Z.101, ITU-T Z.102 and ITU-T Z.104 (plus ITU-T Z.107 for object-oriented data). Therefore the language defined in Recommendation ITU-T Z.103 is practical for application to all kinds of real time systems (except if ASN.1 is also required in which case features defined in Recommendation ITU-T Z.105 should also be used, or if the specification needs to be in text only or interchange format in which case features defined in Recommendation ITU-T Z.106 should also be used)

History

Edition	Recommendation	Approval	Study Group	Unique ID*
1.0	ITU-T Z.103	2011-12-22	17	11.1002/1000/11390
2.0	ITU-T Z.103	2016-04-29	17	11.1002/1000/12857

Keywords

Specification and Description Language, SDL-2010, shorthand notation, annotation, macros

* To access the Recommendation, type the URL <http://handle.itu.int/> in the address field of your web browser, followed by the Recommendation's unique ID. For example, <http://handle.itu.int/11.1002/1000/11390-en>.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications, information and communication technologies (ICTs). The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure, e.g., interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had not received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementers are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database at <http://www.itu.int/ITU-T/ipr/>.

© ITU 2016

All rights reserved. No part of this publication may be reproduced, by any means whatsoever, without the prior written permission of ITU.

Table of Contents

		Page
1	Scope and objective	1
	1.1 Objective.....	1
	1.2 Application	1
2	References.....	1
3	Definitions	2
	3.1 Terms defined elsewhere	2
	3.2 Term defined in this Recommendation	2
4	Abbreviations and acronyms	2
5	Conventions	2
6	General rules	3
	6.1 Lexical rules – text in a comment area	3
	6.2 End terminator, comment and comment area.....	3
	6.3 Text extension	4
	6.4 Solid association symbol	5
	6.5 The metasymbol <i>is followed by</i> and flow line symbol without arrowhead	5
	6.6 Names and identifiers, name resolution and visibility – additional diagrams	6
	6.7 Macro.....	6
	6.8 Informal text	6
	6.9 Text symbol	6
	6.10 Frame symbol, page numbers and multiple page diagrams	6
	6.11 Drawing rules	7
7	Organization of Specification and Description Language specifications.....	8
	7.1 Framework.....	8
	7.2 Package.....	8
	7.3 Referenced definition	10
8	Structural concepts.....	11
	8.1 Types, instances and gates.....	11
	8.2 Type references and operation references	13
	8.3 Context parameters.....	15
	8.4 Specialization	15
9	Agents	15
	9.1 System	19
	9.2 Block.....	19
	9.3 Process.....	19
	9.4 Procedure	20
	9.5 Agent and composite state reference	21
10	Communication.....	23



		Page
	10.1 Channel.....	23
	10.2 Connection.....	25
	10.3 Signal.....	26
	10.4 Signal list area	26
11	Behaviour.....	26
	11.1 Start.....	26
	11.2 State	26
	11.3 Input.....	27
	11.4 Priority input.....	29
	11.5 Continuous signal	30
	11.6 Enabling condition.....	30
	11.7 Save	30
	11.8 Implicit transition	30
	11.9 Spontaneous transition.....	31
	11.10 Label (connector name).....	31
	11.11 State machine and composite state	31
	11.12 Transition.....	34
	11.13 Action	36
	11.14 Statement lists.....	38
	11.15 Timer	40
12	Data.....	40
	12.1 Data definitions	41
	12.2 Use of data	41
	12.3 Active use of data	41
13	Generic system definition	41
	13.1 Optional definition.....	41

Introduction

This Recommendation is part of the ITU-T Z.100 to ITU-T Z.107 series of Recommendations that give the complete language reference manual for SDL-2010. The text of this Recommendation is stable. For more details see Recommendation ITU-T Z.100.

Copyright International Telecommunication Union/ITU Telecommunication Sector
Provided by IHS Markit under license with ITU-T
No reproduction or networking permitted without license from IHS

Recommendation ITU-T Z.103

Specification and Description Language – Shorthand notation and annotation in SDL-2010

1 Scope and objective

This Recommendation defines features of the Specification and Description Language that make the language practical to use. The features defined in this document cover the shorthand notation and annotation for the language, which is defined further in the other Recommendations of the ITU-T Z.100 series. Together with Recommendations [ITU-T Z.100], [ITU-T Z.101], [ITU-T Z.102], [ITU-T Z.104], [ITU-T Z.105], [ITU-T Z.106] and [ITU-T Z.107], this Recommendation forms a reference manual for the language.

1.1 Objective

The objective of this Recommendation is to define the features of the Specification and Description Language that are written using shorthand notation or are used to add annotation. The features defined in this Recommendation add to SDL-2010 features defined for Basic SDL-2010 or Comprehensive SDL-2010 or the data and action language in SDL-2010. The features do not rely on features defined for using ASN.1 or the use of the common interchange format.

Shorthand notation is a concrete grammar that is transformed by a model into further concrete grammar before the representation as abstract grammar is considered. The use of shorthand notation is important to make the use of the language practical, because the shorthand is easy to read, write and understand as well as being concise. A construct written using the concrete grammar defined by this Recommendation is allowed to contain more than one shorthand; in some cases a shorthand is transformed through one (or more) other shorthand forms before a concrete grammar that does not contain any shorthand is obtained from which the abstract grammar is determined. Where the order of transformation is important, this is defined.

An annotation adds information to a specification in the language that does not change the formal meaning of the specification. While such information does not change the formal meaning as far as the Specification and Description Language is concerned, annotations are important to aid understanding of the purpose and intention of a specification or how an SDL-2010 specification interacts with or is interpreted by another language (such as the Message Sequence Chart language).

1.2 Application

This Recommendation is part of the reference manual for the Specification and Description Language. The part of the language defined by this Recommendation includes additional concrete syntax for shorthand notations and *Model* sections for the shorthand. It also includes the concrete syntax for additional annotation not included Basic SDL-2010 or Comprehensive SDL-2010 or the data and action language in SDL-2010. The part of the language defined in this Recommendation is required to make the use of the language practical.

2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

- [ITU-T Z.100] Recommendation ITU-T Z.100 (2016), *Specification and Description Language – Overview of SDL-2010*.
- [ITU-T Z.101] Recommendation ITU-T Z.101 (2016), *Specification and Description Language – Basic SDL-2010*.
- [ITU-T Z.102] Recommendation ITU-T Z.102 (2016), *Specification and Description Language – Comprehensive SDL-2010*.
- [ITU-T Z.104] Recommendation ITU-T Z.104 (2016), *Specification and Description Language – Data and action language in SDL-2010*.
- [ITU-T Z.105] Recommendation ITU-T Z.105 (2016), *Specification and Description Language – SDL-2010 combined with ASN.1 modules*.
- [ITU-T Z.106] Recommendation ITU-T Z.106 (2016), *Specification and Description Language – Common interchange format for SDL-2010*.
- [ITU-T Z.107] Recommendation ITU-T Z.107 (2016), *Specification and Description Language – Object-oriented data in SDL-2010*.
- [ITU-T Z.111] Recommendation ITU-T Z.111 (2016), *Notations and guidelines for the definition of ITU-T languages*.

3 Definitions

3.1 Terms defined elsewhere

This Recommendation uses the following terms defined elsewhere: the definitions of [ITU-T Z.100] apply.

3.2 Term defined in this Recommendation

None.

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms: the abbreviations and acronyms defined in Recommendation [ITU-T Z.100] apply.

5 Conventions

The conventions defined in [ITU-T Z.100] apply, which includes the conventions defined in [ITU-T Z.111].

Where an abstract or concrete syntax rule is defined in this Recommendation with the same name as a rule in [ITU-T Z.101] or [ITU-T Z.102], the rule given here replaces the rule in [ITU-T Z.101] or [ITU-T Z.102]. Any *Abstract grammar* or *Concrete grammar* conditions, *Semantics* and *Model* defined on a named rule in [ITU-T Z.101] or [ITU-T Z.102] apply to the redefined rule, unless specifically defined otherwise in this Recommendation. Any contradiction between [ITU-T Z.101] or [ITU-T Z.102] and this Recommendation is an error in the definition of SDL-2010 that needs to be resolved by further study.

6 General rules

6.1 Lexical rules – text in a comment area

An additional lexical rule is defined for text in a <comment area>:

<text> ::=
 { <general text character> | <special> | <semicolon> | <apostrophe> }*

If the extended character set is used (see [ITU-T Z.102]), all printing characters are permitted to appear freely in <text> in a <comment area>.

A <text> sequence of lexical units is used in a <comment area> where it is treated as annotation. In a <comment area> the <text> is a single lexical unit and the sequence of characters for the <text> is terminated by reaching the end of the text contained in the <comment symbol> or <text extension symbol> of the <comment area>.

6.2 End terminator, comment and comment area

End terminator and comment are as defined in [ITU-T Z.101] and [ITU-T Z.102].

A comment area is a notation to represent comments associated with symbols.

Concrete grammar

With text, three forms of comments are used. The first form is the <note>, which is considered only as a unit at the lexical level and therefore does not appear outside lexical rules (see [ITU-T Z.101]). The second and third forms are a <comment> in an <end> and <comment body> as defined in [ITU-T Z.101]. The places where <end> (and therefore <comment>) and <comment body> are allowed are defined in the concrete syntax rules.

With symbols a <comment area> with <text> is used.

<comment area> ::=
 <dashed association symbol> **is connected to**
 { <comment symbol> | <text extension symbol> } **contains** <text>

<comment symbol> ::=



<dashed association symbol> ::=



A <dashed association symbol> is a line symbol.

It is allowed that a <comment area> is connected to any graphical symbol except a <comment symbol> or <text extension symbol>. One end of the <dashed association symbol> shall be connected to the middle of the vertical segment of the <comment symbol>. The graphical symbol shall connect to the other end of the <dashed association symbol>.

The <comment symbol> is considered as a closed symbol by completing (in imagination) the rectangle to enclose the text. It contains comment text related to the graphical symbol.

Model

A <text extension symbol> in a <comment area> is treated as a <comment symbol>, and the <comment area> is distinguished from a <text extension area> by the use of the <dashed association symbol>.

Copyright International Telecommunication Union/ITU

In principle the non-terminal <comment area> is added to the concrete syntax for pages (such as <package page> or <block type page>) so that it is possible to add <comment area> to any symbol on the page (except a <comment symbol> or <text extension symbol>). For example, a modified form of <package page> is:

```
<package page> ::=
    <frame symbol> contains
    { <package heading> <page number area>
      {
        { <package text area> { is connected to <comment area> }* }*
        { <diagram in package> { is connected to <comment area> }* }* } set }
    [ is associated with <package use area> ]
```

For conciseness, it is assumed that this example is used (as further extended to include the syntax for <text extension area> as given in clause 6.3) as a template to derive the modified syntax for other pages such as <block type page>.

A <comment area> connected to a symbol has no formal meaning and is therefore removed before determining the abstract grammar. Each <comment area> for a <package page> is connected to the symbol for the <package text area> or the <diagram in package> (not to any other <comment area>).

6.3 Text extension

Concrete grammar

```
<text extension area> ::=
    <solid association symbol> is connected to
    <text extension symbol> contains
    { {
      | <name>
      | <integer name>
      | <real name>
      | <character string>
      | <hex string>
      | <bit string>
      | <note>
      | <comment body>
      | <composite special>
      | <special>
      | <semicolon>
      | <other character>
      | <quoted operation name>
      | <keyword> } *
    }
```

```
<text extension symbol> ::=
```



It is allowed to connect a <text extension area> to any graphical symbol that contains text to be analysed as lexical units (except another <text extension symbol>). One end of the <solid association symbol> shall be connected to the middle of the vertical segment of the <text extension symbol>. The graphical symbol shall connect to the other end of the <solid association symbol>.

The <text extension symbol> is considered as a closed symbol by completing (in imagination) the rectangle.

The <text extension symbol> is allowed to contain a list of lexical items (such as <name>, <note> or <keyword>). Whether a particular lexical item is allowed will depend on the graphical symbol that connects to the <text extension area> and the model given below.

Model

In principle, the non-terminal <text extension area> is added to the concrete syntax for pages (such as <package page> or <block type page>) so that <text extension area> is allowed to be added to any symbol (except a <comment symbol> or <text extension symbol>) on the page. It is allowed for a symbol to have both a <text extension area> and a <comment area>. For example, a modified form of <package page> is:

```
<package page> ::=
    <frame symbol> contains
        { <package heading> <page number area>
            { <package text area>
                [ is connected to <text extension area> ]
                { is connected to <comment area> }* }*
            { <diagram in package>
                [ is connected to <text extension area> ]
                { is connected to <comment area> }* }*
            } set }
        [ is associated with <package use area> ]
```

Each <comment area> of a <package page> is connected to the symbol for the <package text area> or the <diagram in package> (not to the optional <text extension area> or any other <comment area>). The <comment area> is removed as described in clause 6.2 before determining the abstract grammar.

NOTE – Only one <text extension area> is allowed.

For conciseness, it is assumed that this example is used as a template to derive the modified syntax for other pages such as <block type page>.

The list of lexical items contained in the <text extension symbol> is a continuation of text within the graphical symbol. The graphical symbol is therefore transformed by adding the list of lexical items in the <text extension symbol> to the end of the items in the graphical symbol, and the <text extension symbol> is removed. The validity and meaning of any particular lexical item therefore depends on the particular graphical symbol and the context in which it is used. This transformation is applied before any transformation relating to the graphical symbol or its contents.

Sometimes a symbol contains more than one item of text. For example, the <block symbol> for a <typebased block definition> contains text for the <typebased block heading> and for each <gate> item. In this case, it is the text item that appears first in the syntax that is extended. The <text symbol> is a special case, because the contents are always treated as a single text item; therefore the text extends the whole contents of the <text symbol>.

6.4 Solid association symbol

See [ITU-T Z.101].

6.5 The metasymbol *is followed by* and flow line symbol without arrowhead

The flow line symbol without arrowhead is added to make specifications more readable by omitting unnecessary arrowheads.

Concrete grammar

The concrete syntax for <flow line symbol> is extended to allow a <flow line symbol without arrowhead> as an alternative.

```
<flow line symbol> ::=
    <flow line symbol without arrowhead>
    | <flow line symbol with arrowhead>

<flow line symbol without arrowhead> ::=
```

<flow line symbol without arrowhead> is a line symbol.

A <flow line symbol without arrowhead> shall not be used in a diagram where it merges with another <flow line symbol>, or an <out connector symbol> or a <nextstate area>.

A <flow line symbol without arrowhead> has the same representation as a <solid association symbol>. If there is any ambiguity in a diagram, the symbol is taken to be a <solid association symbol>, and if a <flow line symbol> is required instead (representing *is followed by*) a <flow line symbol with arrowhead> should be used.

Model

A <flow line symbol without arrowhead> is transformed to a <flow line symbol with arrowhead>.

6.6 Names and identifiers, name resolution and visibility – additional diagrams

An identifier is used to identify an item of a particular entity kind with a specific name defined in a specific scope unit context. As well as a *Name*, an *Identifier* has a *Qualifier* that gives the path to the definition of the identifier. In the concrete syntax the <qualifier> is usually minimized or omitted if it is possible to resolve the identity without full qualification, to avoid the specification becoming unreadable due to all the qualification. The minimization or omission of qualifiers is a shorthand notation that should in principle appear in this Recommendation, but is fully described in [ITU-T Z.101] because of the importance of this shorthand for usage of the language.

This Recommendation introduces concrete syntax for the following scope unit kinds in addition to those defined in Basic SDL-2010:

system	<system diagram>
block	<block diagram>
process	<process diagram>

6.7 Macro

See [ITU-T Z.102].

6.8 Informal text

See [ITU-T Z.101].

6.9 Text symbol

See [ITU-T Z.101].

6.10 Frame symbol, page numbers and multiple page diagrams

Basic SDL-2010 is extended to allow multiple page diagrams.

General rules are described here.

Every page of a diagram has a heading, which identifies the diagram kind (such as **process type**) and the identity (usually as an unqualified name if this is unambiguous) for the corresponding abstract grammar item. The concrete grammar for multiple pages is the same for every page, and it is required that the information given is consistent on every page. Consistency means that in a heading if an item (such as **abstract** as a <type preamble>) appears on one page, it should appear identically on all other pages, but for convenience it is permitted to omit additional information on all but one page and the combined heading formed from a combination of all pages. It is suggested that at least one page (with an appropriate <page number>) has the full heading. The formal correct presentation is for every diagram page to have the same identical full heading.

If there is a <package use area> the formal correct presentation is for every diagram page to have the same <package use area>, but for convenience it is permitted to omit information in the <package use area> of some pages or to omit the <package use area> altogether on a page, provided the package use information for the diagram is given in at least one <package use area>. It is suggested that one page with the full heading for the diagram also has a <package use area> that gives the full package use information.

Model

If a diagram has several pages, and the headings are different on different pages a consistent heading is derived (if possible; otherwise the diagram is probably invalid).

The full formal correct <package use area> is derived from combining the content of every <package use area> on all the pages. Any duplication of package use information is removed.

A diagram with several pages is logically replaced by a diagram with one larger page with the consistent heading and an arbitrary <page number>. Forming the single page diagram is a topological transformation, where to join two pages together, a break is made in the frame of each and the line of the frames are joined so there is a single line around the content of both original pages. Additional diagrams are added in the same way until a single line surrounds all the contents of all the original pages. The items related to the frames remain connected, attached or associated with the new frame, and are moved with the frame as it is reshaped to make it rectangular again. The headings and page numbers are removed and replaced by the consistent heading and arbitrary <page number> in the top left and right corners respectively.

6.11 Drawing rules

The size of the graphical symbols is chosen by the user (in principle, but tools possibly limit user choice).

Symbol boundaries shall not overlay or cross except line symbols, which are allowed to cross each other. There is no logical association between symbols that do cross. The following are line symbols:

- <channel symbol 1>
- <channel symbol 2>
- <create line symbol>
- <dashed association symbol>
- <flow line symbol>
- <flow line symbol without arrowhead>
- <solid association symbol>

Vertical mirror images of <input symbol>, <output symbol>, <internal input symbol>, <internal output symbol>, <priority input symbol>, <comment symbol> and <text extension symbol> are allowed.

Text within a graphical symbol is read from left to right, starting from the upper left corner. The right-hand edge of the symbol is interpreted as a newline character, indicating that the reading continues at the leftmost point of the next line (if any). Text within a graphical symbol should be contained within the symbol (that is, not overlapping the lines forming the border of the symbol or boundary of the symbol for unclosed symbols such as the <comment symbol>), so that no part of the text is read as associated with some other symbol. However, a symbol often contains more than one text item, for example, a <typebased block definition> contains text for both a <typebased block heading> and a number of <gate> items. Different text items should not overlap, should be sufficiently separated to be distinct, and in a diagram should be unambiguously nearer to any construct they are related to than any other text item.

7 Organization of Specification and Description Language specifications

7.1 Framework

The concrete syntax of a system specification is extended to allow agent diagrams as well as typebased agent definitions.

Concrete grammar

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

Model

A <system specification> being an <agent diagram> that is a <process diagram> or a <typebased agent definition> that is a <typebased process definition> is derived syntax for a <system diagram> having the same name as the process, containing implicit channels and containing the <process diagram> or <typebased process definition> as the only definition. Using this model satisfies the constraint the *Agent-kind* is **SYSTEM**. The transformation of a <process diagram> into a <system diagram> containing the <process diagram> is carried out before the transformation of the contained <process diagram> into a <typebased process definition> and a corresponding <process type diagram>.

A <system specification> being an <agent diagram> that is a <block diagram> or a <typebased agent definition> that is a <typebased block definition> is derived syntax for a <system diagram> having the same name as the block, containing implicit channels and containing the <block diagram> or <typebased block definition> as the only definition. Using this model satisfies the constraint the *Agent-kind* is **SYSTEM**. The transformation of a <block diagram> into a <system diagram> containing the <block diagram> is carried out before the transformation of the contained <block diagram> into a <typebased block definition> and a corresponding <block type diagram>.

A <system diagram> is derived syntax for a <typebased system definition> and a corresponding <system type diagram>.

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

7.2 Package

Package is extended to allow textual references (to agent types, contained packages, procedures, composite state types and interface definitions), contained procedure definitions and create line annotation. Basic SDL-2010 is extended to allow more than one page for a <package diagram>. Package reference is extended to allow it to be used as annotation that references a package defined elsewhere.

Concrete grammar

```
<package diagram> ::=
    <package page>+
<package reference area> ::=
    <package symbol> contains [ <qualifier> ] <package name>
```

The <package reference area> is extended compared to Basic SDL-2010 to include an optional <qualifier>. The optional <qualifier> and <package name> of a <package reference area> shall be contained in the lower rectangle of <package symbol>.

If in a <package reference area> there is a <qualifier> before the <name> of the referenced package and the <qualifier> does not identify the scope unit directly enclosing the package reference, the