INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

Z.120

Corrigendum 1

(12/2001)

SERIES Z: LANGUAGES AND GENERAL SOFTWARE ASPECTS FOR TELECOMMUNICATION SYSTEMS

Formal description techniques (FDT) – Message Sequence Chart (MSC)

Message Sequence Chart (MSC)

Corrigendum 1

ITU-T Recommendation Z.120 (1999) – 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.120

Message Sequence Chart

Corrigendum 1

Summary

This is list of changes for ITU-T Rec. Z.120 (MSC) was approved by the Study Group in Septembre 2001.

Source

Corrigendum 1 to ITU-T Recommendation Z.120 was prepared by ITU-T Study Group 10 (2001-2004) and approved under the WTSA Resolution 1 procedure on 14 December 2001.

CONTENTS

ITU-T Recommendation Z.120

Message Sequence Chart

Corrigendum 1

1 Objectives and scope

The purpose of this corrigendum is to record agreed changes to ITU-T Rec. Z.120. The scope and nature of the changes follows the classification given in Appendix II/Z.100 (SDL).

The changes come in two categories:

a) Correction of *errors* and *clarifications*;

b) *Extensions* and *modifications*.

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 MSC. 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 (*sic*) is revised." *Modification* and *extensions* included in this corrigendum are not considered to be significant changes – but are considered as useful improvements to – the MSC notation and therefore are now recommended for immediate use.

2 Terminology

An *error* is an inconsistency in ITU-T Rec. Z.120.

A *textual correction* is a change in the text or diagrams of the Recommendation 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 MSC is not clearly defined in the Recommendation.

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

A *modification* changes the semantics of MSC.

An *extension* is a new feature that does not change the semantics of MSC defined in the approved Recommendation.

3 Maintenance of ITU-T Rec. Z.120

Appendix II/Z.100 (1999) documents the procedure to be followed for the maintenance of Z.100 Recommendations, which is taken as the baseline for the maintenance of ITU-T Rec. Z.120 (MSC). This procedure requires error corrections, proposed modifications and extensions to be widely publicized 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.120 changes

4.1 Clerical error – Reference to Section 0: Section 1.2

The reference to "Section 0" is corrected to read "Section 6".

4.2 Clarification – References to non-existent productions in Meta-language examples: Section 1.3

The <msc document body> production example in section 1.3 does not exist in the grammar and so is replaced by the <using clause> production. The accompanying explanatory text is altered accordingly.

4.3 Clerical error – Spelling of "precedence": Section 1.3

The word "presidence" is corrected to read "precedence" in section 1.3.

4.4 Clerical error – Missing closing double quote in text: Section 1.3

In the final sentence of section 1.3, a missing closing double quote from the text is to be added: "{…}".

4.5 Clarification – Generic production names: Section 1.4

Addition of sentence to section 1.4 to clarify the role of generic production names <area>, <non terminal>, etc.

4.6 Clarification – Correct example production: Section 1.4

The example production rule in section 1.4 for <message end area> is corrected to match the production in the grammar.

4.7 Clerical error – Reference to non-existent <event list>: Section 4.1

The production name "<event list>" corrected to read "<instance event list>" in the Semantics part of section 4.1.

4.8 Clarification – Removal of reference to instance interface with environment: Section 4.1

Delete the sentences:

"The optional <msc inst interface> which describes the interface of the MSC with its environment consists of the <msc inst interface> and the <msc gate interface>."

and:

"Since normally one MSC only consists of a section of a system run the <msc inst interface> describes the connection points of instances to the environment."

from the Semantics section of Section 4.1, as they do not make sense, and are redundant.

4.9 Clarification – Relationship between timed and untimed semantics: Section 6

The final sentence of the second paragraph of section 6 is changed to read

"The untimed interpretation of an MSC can be derived from the timed interpretation by removing the extra time events from its traces; in doing so groups of different timed traces will reduce to the same untimed trace.".

This is a simple textual clarification.

4.10 Modification – Lexical rules: Section 2.1

Remove production for <word> as it is not used, although it appears in <lexical unit>, where it is to be replaced by <name>, as it is missing there.

Move the production <qualifier> from section 3 to section 2.1, since it has to be handled similar to <note>; furthermore <text> is not exported as a lexical unit and therefore is not visible in the parsing section.

Reference to <qualifier> is to be added in the production <lexical unit>.

Add production <misc> to contain amongst others <apostrophe>.

Move '[', ']', '#', and '@' from <national> to <special>, using their production names where defined.

Move '&', '<', and '>' from <other characters> to <special>, using their production names.

4.11 Modification – Removal of keywords: Section 2.1

Remove keywords **by** and **order** from the <keyword> list in section 2.1 as they are not used anywhere.

4.12 Modification – Replacement of keywords "startbefore", "startafter", "endbefore", "endafter": Section 2.1

The four keywords **startbefore**, **startafter**, **endbefore**, and **endafter**, are used in the concrete textual grammar to indicate where a timing constraint is placed on an in-line expression. However, only two keywords are required; furthermore, the *before* or *after* sense cannot be specified statically since timing constraints apply dynamically. That is, one cannot say statically if one end of an interval is dynamically the first or the last event in respect of the other end. Therefore, the four keywords have been removed and replaced by **top** and **bottom**, which determine if the constraint applies to the first or to the last event within the expression, respectively. All occurrences of the old keywords in the standard have been updated to reflect this change. The change in keyword reflects the possible positioning of the time constraints in the graphical grammar.

4.13 Clerical error – Typo corrected: Section 2.1

In the final sentence of the main body of the text of section 2.1, the word "define" is corrected to read "defined" as follows:

"To make the approach applicable in all circumstances, an escape is also defined to describe when parenthesis delimiters are to be considered in other ways than as parenthesis delimiters."

4.14 Clarification – Role of <space> explained: Section 2.1

The following sentence is to be added to the end of the second paragraph in the text following the formal grammar of section 2.1, as there is no explanation on the role that <space> play in the grammar. This sentence mirrors one from the Z.100 SDL standard.

"Any number of <space> may be inserted before or after any <lexical unit>. Inserted <space> or <note> have no syntactic relevance, but sometimes a <space> or a <note> is needed to separate one <lexical unit> from another."

4.15 Clarification and Modification – Production for <non par non escape>: Section 2.1

The text in the right hand side of the rule for <non par non escape> in section 2.1 is changed: a) to avoid confusion between the informal use of the term "character string", since this is the formal name of a production; b) to avoid a similar confusion on the word "delimiter"; and c) to add "]" to the list of terminating symbols, since this was incorrectly omitted (it can terminate a time interval).

Thus, "character string" becomes simply "string", "parenthesis delimiters" becomes "parenthesis string". The revised right hand side of the productions read as follows:

> string containing no escape char and no parenthesis strings and not the possible terminators "," ")" "]" ";" ":" ":=" "=:" or **comment**

Similarly, the first occurrence of the word "delimiters" is changed to "string" in the sentence immediately before the <string> production; the second occurrence is to be deleted. The change from "delimiters" to "string" is also to be made in the sentences before the <non parenthesis> production, and finally after the <non-par-non-escape> production.

4.16 Modification – Moving data part in MSC document: Section 3

Move the productions <data definition> and <parenthesis declaration> from the bottom of the production <document head> to near the top. Moving of <parenthesis declaration> facilitates lexical analysis of the document, and moving <data definition> facilitates static analysis of data usage. The example diagrams are changed to reflect the grammar change. The <data definition> reference will no longer be optional, as each of the elements in its production are optional, which removes ambiguity in the grammar.

4.17 Clarification – Environment messages in redefined MSCs: Section 3

Add the sentence "The environment messages of a **redefined** MSC must correspond exactly to those of the MSC it is redefining: no fewer or extra messages to/from the environment are permitted." to Static Requirements part of section 3.

4.18 Modification – Move <time interval> reference in <time dest list>: Section 4.1

Move <time interval> reference in production <time dest list>, which may end with a data string, to after the optional time destination <time dest> in section 4.1. This ensures that the data string is terminated either by a ',' or a ';', which can be recognized, rather than an identifier (the <event name>), the start of which cannot be recognized. The new version of the production is:

```
        <time dest list> ::= [ <time dest> ] <time interval> [ , <time dest list> ]
```

4.19 Modification – Allowing parameter blocks in any order: Section 4.1

The order of which variable, timer, message, and instance parameter declarations is currently fixed by the grammar. The grammar is to be changed to permit these blocks to be declared in any order. Specifically, the production <msc parameter decl> is to be changed, and new intermediate productions <msc parm decl list> and <msc parm decl block> are to be added. The productions are to be:

```
<msc parameter decl> ::=
                    ( <msc parm decl list>)

<msc parm decl list> ::=

        <msc parm decl block> [ <end> <msc parm decl list> ]

<msc parm decl block>h ::=
                    <data parameter decl>
                    |       <instance parameter decl>
                    |       <message parameter decl>
                    |       <timer parameter decl>
```

The following sentence is to be added to the Static Requirements part:

"The blocks of MSC parameter declarations given inside a <msc parm decl list>, viz variable, instance, message, and timer blocks, can be given in any order, but there can be at most one block of each type. For example, there cannot be two blocks of message parameters, even if separated by an instance declaration block."

A corresponding change for actual parameters is to be made in section 7.3, and is covered in the appropriate section of this corrigendum.

4.20 Modification and clarification – Removal of SDL <kind denominator> keywords: Sections 4.1, 4.2, and 2.1

The explicit reference to SDL entities is to be removed as they may change, as is the case with the removal of SDL services. There is an existing mechanism in the MSC grammar to permit a user to write any identifier in the same place as the SDL specific keywords so that the connection can still be made informally. The SDL related keywords **system**, **block**, **process**, and **service** are to be removed from the production <kind denominator> in Section 4.1 and from the list of keywords in the production <keyword> of section 2.1.

In the initial text of section 4.2, remove the sentence "Related to SDL, an instance maybe an SDL-system, block, process, or service".

The explanation of the roles of <instance kind> identifier and <kind denominator> has been improved in the Semantics part of section 4.2. The second paragraph has been split into two, the second part on instance definitions remains unchanged, but the first becomes:

"In the context of SDL an instance may refer to entities such as processes, blocks, or systems, and outside of SDL it may refer to any kind of entity. The <kind denominator> permits the user to name the kind of entity that the instance relates to, such as process, block, etc., and the <instance kind> identifier can be used to give the particular name for that entity class. Multiple instances can share the same <instance kind> identifier to denote that they are of the common kind."

4.21 Clarification – Replacing <textual msc document> by "MSC document": Section 4.1

The reference to <textual msc document> in the Static Requirements part for the textual grammar is replaced by "MSC document". As the textual grammar is meant to be derived from a graphical representation, it is wrong to require that anything come from the textual grammar when there is a graphical equivalent.

4.22 Modification – Addition <label names> to <time dest>, and change of keywords: Section 4.1

The production for <time dest> in the Concrete Textual Grammar part does not currently permit time constraints to refer to HMSC nodes, which is not the intention. Therefore, <label name> is to be added to the rule. Furthermore, the keywords **begin** and **end** are to be replaced by **top** and **bottom** for uniformity with other rules, and their optionality is to be removed – so that one of the keywords has always to be given. The rule for <time dest> becomes:

```
<time dest> ::=
                <event name> | { top | bottom } { <reference identification> | <label name> }
```

An additional static requirement is to be added as a consequence of the change to this rule to ensure that the destination node of a <time dest> in an HMSC is a <timeable node>, and not a non-timeable <node>. The following sentence is to be added at the end of the Static Requirements part, following the Concrete Textual Grammar:

"The <**label** name> used in a <time dest> must refer only to <timeable node>."

4.23 Clarification – Defining when static instances come into existence: Section 4.2

The following sentence is added to the end of the first paragraph of the Semantics part to clarify when ordinary, non-dynamically created, instances come into existence.

"Static instances of an MSC document are those instances that appear within its defining part but which are not dynamically created. The static instances are created when their enclosing instance is created – the enclosing instance (kind) being the name of the MSC document."

4.24 Clerical error – Typo: Section 4.2

In the Concrete Graphical Grammar part , the spelling of "occuring", is corrected to read "occurring".

4.25 Clarification – Creating and terminating instances inside MSC reference: Section 4.2

Replace the final sentence before the Semantics part with by:

"If an instance is created but not terminated inside a referenced MSC, then the instance head and axis must be included on the referring MSC, and no events may be placed between the head and the reference. This permits the continuation of the instance axis below the reference to be identified with the correct instance name. Conversely, if an instance is terminated (stopped) but not created within a referenced MSC, then the instance axis must be terminated on the referencing MSC, and no events may be placed between the reference and the stop symbol. These rules ensure that a reference always has the same number of instance axes attached below as it does above."

This explanation expands and clarifies how instances that would otherwise emerge from only one side of an MSC reference symbol are represented.

4.26 Clerical error – Missing Static Requirements title: Section 4.2

The text immediately following the Concrete Graphical Grammar part, "The <instance heading> may be placed …" should have the title "Static Requirements" inserted above it.

4.27 Clerical error – Spelling of AccessPoint in Figure 28: Section 4.2

Correct the msc document name in Figure 28 from "AcessPoint" to read "AccessPoint".

4.28 Clerical error – Reference to non-existent productions: Section 4.3

Change production reference from <name> to <gate name> in text accompanying Concrete Textual Grammar part. Change reference <address> to "<input address> or <output address>" in the text of the Concrete Textual Grammar part. Change reference <bindings> to "<binding>s" in the text of the Static Requirements part. Change reference <output event area> to "output <event area>" in the text of the Static Requirements part. Similarly, <input event area> becomes "input <event area>".

4.29 Clerical error – Reference to a non-existent production: Section 4.4

Change production reference from <call out event> to "<call out> event" in teh text of the Static Requirements part.

4.30 Clerical error – Spelling mistakes: Section 4.5

Change "incompleted" to "incomplete" in the fourth paragraph.

Change "Correspondance" to "Correspondence" in the Static Requirements part.

4.31 Clerical error – Figures 23, 24, 25, and 26: Section 8.1

The quotes enclosing the message names *PleaseEnter*, *TryAgain*, are to be removed from Figures 23, 24, 25, and 26. The message names *Cardid*, *try_again*, and '*Please enter*' in Figures 24 (and the latter in Figure 23) are to be corrected to read *CardId*, *TryAgain*, and *PleaseEnter*, respectively.

4.32 Clerical error – Spelling mistake: Section 4.7

Change "optinonal" to "optional" in the Concrete Textual Grammar part.

4.33 Clarification – Instances covered by a condition: Section 4.7

The relationship between the instances covered by a graphical condition symbol and those defined in its enclosed <shared> list is clarified.

The following sentence in teh Concrete Graphical Grammar part:

"The <shared> instances have no <instance area> in the diagram since there are no events on these instance, but these <shared> instances are still covered by the condition."

is replaced by:

"The instances covered by a condition consist of those attached to it in the diagram plus those listed explicitly in the <shared> clause."

The last sentence in the Concrete Graphical Grammar part is moved to the Static Requirements part and is clarified to read:

"If there are any ambiguities regarding event ordering on <shared> instances due to the implicit condition symbol, they must be represented in the diagram explicitly with an <instance area> rather than via the <shared list> list".

The following sentence is added to the Static Requirements part following the Concrete Graphical Grammar part:

"An instance that is graphically covered by a condition symbol cannot be included also via its <shared> list."

4.34 Clarification – Type of timer duration expressions: Section 4.8

The following sentence to be added to the Static Requirements part. This clarifies that the type of expressions used to define a timer's duration are of the assumed type Time.

"The <expression string> used in specifying a <duration> must have the assumed type Time."

4.35 Clerical error – Hyphens in syntax rules changed to spaces: Section 5.2

All occurrences of the following production names are changed to remove hyphens:

<non-parenthesis>	becomes	<non parenthesis>.
<non-nestable par>	becomes	<non nestable par>.
<non-parenthesis>	becomes	<non parenthesis>.
<non-par-non-escape>	becomes	<non par non escape>.
<non-orderable event>	becomes	<non orderable event>.
<non-nestable par pair>	becomes	<non nestable par pair>.
<extra-global>	becomes	<extra global>.

4.36 Modification – Addition of delimiters in <escapechar>: Section 5.2

The production <escapechar> is changed from just <character string> to a <character string> enclosed in <delim>. This serves two purposes: a) a character string can contain spaces and so determining the end of the string is difficult, particularly as any spaces would be included in the escape string; there were no static rules restricting the number or make-up of the characters in the string; and b) it is not in harmony with the other declarations in the parameters block.

The new production is to be:

```
<escapechar> ::= <delim> <character string> <delim>
```

4.37 Clarification – Change of the word "delimiter": Section 5.2

In the Semantics part of Section 5.2, the word "delimiter" is to be replaced by "parenthesis characters" in the sentence "The lexical analyzer will search for the matching right delimiter." This change is made since *delimiter* has a formal meaning in this section as the characters surrounding the declarations of the parenthesis strings, and the reference is to the parenthesis characters, not to these delimiter characters.

4.38 Modification – Removal of comma in <par decl list>: Section 5.2

The production <par decl list> is to be changed to remove the superfluous comma that separated the statements that make up the <par decl list>, since each statement is terminated by a semi-colon anyway. So in the rule <par decl list>, the final part is changed from:

    [, <par decl list>]

to:

    [<par decl list>].

4.39 Modification – Grammar for parenthesis declarations: Section 5.2

The grammar for declaring the parenthesis characters does not capture the intention and is revised. In addition to the superfluous commas removed per modification 4.38, every pair of matching parenthesis characters had to be declared in a separate statement. The intention was that all nestable parenthesis pairs, etc., are to be declared in the one statement.

To accommodate this, the syntax for declaring the bracket pair changes to remove the comma that separates the opening and closing bracket. To achieve this, the delimiter character becomes the separator, so that in place of " '(' , ')' ", we shall have " '(')' ".

The following example accords with the published syntax:

```
parenthesis
    nestable        '(' , ')' ; ,
    nestable        '[' , ']' ; ,
    nonnestable     '/*' , '*/' ; ,
    equalpar        '[' ; ,
    escape          \ ;
;
```

In the modified grammar, this example becomes:

```
parenthesis
    nestable        '(')' , '[']' ;
    nonnestable     '/*'*/' ;
    equalpar        '[' ;
    escape          '\' ;
```

The productions constituting the Concrete Textual Grammar part of section 5.2 are replaced by the following:

```
<parenthesis declaration> ::=
                    parenthesis <par decl list> <end>

<par decl list> ::=
                    { <nestable par pair> | <non nestable par pair> | <equal par decl> | <escape decl> }
                    [<par decl list>]

<nestable par pair> ::=
                    nestable <pair par list> <end>

<non nestable par pair>::=
                    nonnestable <pair par list> <end>
```

<equal par decl> ::=
    equalpar <equal par list> <end>

<escape decl> ::=
    escape <escapechar>

<pair par list> ::=
    <pair par> [, <pair par list>]

<pair par> ::=
    <delim> <open par> <delim> <close par> <delim>

<equal par list> ::=
    <equal par> [, <equal par list>]

<equal par> ::=
    <delim> <par> <delim>

<delim> ::=
    <apostrophe>
  | <alphanumeric>
  | <other character>
  | <special>
  | <full stop
  | <underline

<par> ::=
    <character string>

<escapechar> ::=
    <delim> <character string> <delim>

4.40 Clarification and clerical Error – Grammar for parenthesis declarations: Section 5.2

The missing heading "Static Requirements" is inserted following the productions of the Concrete Textual Grammar part. The original sentence explaining the static requirements is replaced to provide an expanded and clearer explanation. The new text will read:

> "The delimiter character must be the same character used on the left, centre, and right in a <pair par> production, and likewise on the left and right of a <equal par> or an <escapechar> production, but they may be different between different occurrences of these productions. The delimiter character must not be contained in the character strings they enclose. For example, the following is legal:
>
> **nestable** '(')' , /[/]/ **;**

In parsing the parenthesis character declarations, an MSC analyzer will read the first delimiter character and then read all characters up to but not including the next occurrence of this delimiter and take that as the parenthesis string. In the case of a paired parenthesis the analyzer will then take all characters following the middle delimiter up to but not including the third occurrence of the delimiter as the matching closing parenthesis string. No escape mechanism is provided for the character strings between the delimiters since a delimiter character not contained in the parenthesis strings must be selected by the user to prevent any conflict."

4.41 Modification – Changing terminator ';' to separator ';': Section 7.3

The productions <actual instance parameters>, <actual message parameters>, and <actual timer parameters> are changed to have the <end> removed. This is because <actual parameters> has changed to allow the blocks of declarations to occur in any order.

4.42 Modification – Optional variables keyword in <data parameter decl>: Section 5.5

The change to <msc parm decl> means that <data parameter decl> need not be the first declaration block, so an optional leading keyword, **variables**, is to be added to its production. The same is true of <actual data parameters>. The keyword will indicate that a previous block of declarations is terminated; otherwise, the grammar would be ambiguous. The addition of the keyword is to be made in the Concrete Textual Grammar, so the productions become:

```
<data parameter decl> ::=
                              [ variables ] <variable decl list>

<actual data parameters> ::=
                              [ variables ] <actual data parameter list>
```

The following sentence is to be added after the first sentence of the Static Requirements part:

"If the variables declarations follow after timer, or message, etc. declaration block, then the keyword **variables** is used to indicate that the previous block is complete. If the variable declarations occur first amongst the parameter declarations, then **variables** keyword is optional."

The following sentence is to be added to the end of the Static Requirements part:

"The use of the keyword **variables** in the <actual data parameter list> follow, the rules as for <data parameter decl>."

4.43 Modification – Addition of default time unit: Section 6.4

No default unit for time was defined. The following sentence is to be added at the end of section 6.4 that fixes the unit to be in seconds:

"The units of, and representation of elements from, the time domain can be specified via the MSC data interface (see section 5.4). However if no time unit is defined there, then the time units are in seconds by default."

4.44 Clerical error – Incorrect time marks: Section 6.9

In the opening text of section 6.9, the absolute time mark is incorrectly given as '&', which is to be changed to '@'. Similarly, the relative time mark is changed from '?' to '&'.

4.45 Clerical error – Typo: Section 6.10

In the Concrete Graphical Grammar part of section 6.10, "if the arrow heads" is changed to "of the arrow heads". Also the spellings of "fulfill", "fulfillment", and "distinuguishable" are to be corrected.

4.46 Clerical error – Incorrect references to productions: Section 6.10

Replace the reference <int_symbol 1> by <int symbol 1> in the text accompanying the Concrete Graphical Grammar of section 6.10. Similarly, replace <int_symbol 2> by <int symbol 2>.

4.47 Clerical error – Typo, "what" corrected to "which": Section 6.10

In the final paragraph of the opening part of section 6.10, "what is determined dynamically" has been corrected to "which is determined dynamically".

4.48 Modification – Keyword int_boundary made optional: Section 6.10

The keyword **int_boundary** is only required in the ASCII syntax, and so is to be made optional in the production <interval label> of section 6.10. The following sentence to be added to the Static Requirements part to explain this.

"Where an interval label is used, the keyword **int_boundary** must appear in the programming representation, and must be absent in the graphical representation."

4.49 Clarification – Sense of mirroring symbols: Section 6.10

In the Concrete Graphical Grammar part of section 6.10, following the production rule for <int symbol 2>, the phrase "may be mirrored horizontally and vertically" is to be replaced by "may be mirrored about a horizontal or vertical axis". Likewise, the phrase "may be mirrored vertically" is to be replaced by "may be mirrored about a horizontal axis" following the definition of the production <cont int symbol>. These changes clarify the allowed sense of mirroring permitted.

4.50 Modification – Addition of absolute time intervals: Section 6.10

The existing graphical grammar does not permit absolute time intervals, although the textual grammar does. Therefore the graphical grammar is to be extended to include the use of absolute time intervals for time constraints. Thus, two new productions <abs time interval> and <abs bounded time> are to be added. The latter defines the syntax for an absolute time interval, and the former replaces explicit choices in the <abs time are> production. That is, the "{ <abs time expression> | <abs measurement> } part of the rule <abs time area> is replaced by <abs time interval>. The <abs time interval> extends the former choice with <abs bounded time>. The new grammar is given by:

```
<abs time area>::=
                    <abs time symbol>
                    is associated with <abs time interval>
                    is attached to
                    { <message out symbol>  |< message in symbol>  | <action symbol>  |
                    <timer start symbol> | <timer stop symbol> | <timeout symbol> |
                    <inline expression symbol> | <separator symbol> |
                    <msc reference symbol> | <par frame symbol> |
                    <call in symbol> | <call out symbol> | <reply symbol> }

<abs time interval>::=
                    <abs time expr> | <abs bounded time> | <abs measurement>

<abs bounded time>::=
                    <abs time mark> { <left open> | <left closed> }
                    [ <time point>] , [ <time point>]
                    { <right open> | <right closed }
```

4.51 Modification – Addition of sequence for in-line expressions: Section 7.2

Sequences were not included in the grammar for in-line expressions, although the text did refer to them. Since sequences can usefully be employed to mark off regions of events to apply timing constraints, they are to be added to the grammar.

In the opening sentence of section 7.2, "parallel composition" becomes "parallel and sequential composition". In the opening sentence of the Semantics part of Section 7.2, "alternative composition, parallel composition" becomes "alternative composition, sequential composition, parallel composition". In the final paragraph of the Semantics part, "**alt end**, and **par end**" becomes "**alt end**, **seq end**, and **par end**".

The following two productions are added to the Concrete Textual Grammar of section 7.2.

```
<shared seq expr> ::=
                    seq begin [ < inline expr identification>] <shared><end>
                    [ <inline gate interface>] [<instance event list>]
                    { seq <end> [ <inline gate interface> ] [<instance event list>]}*
                    seq end <end>

<seq expr> ::=
                    seq begin [ <inline expr identification> ] <end>
                    [ <inline gate interface>] <msc body>
                    { seq <end> [ <inline gate interface> ] <msc body> }*
                     seq end <end>
```

In addition, a reference to <shared seq expr> is added between <shared alt expr> and <shared par expr> in the <shared inline expr> production rule. Similarly, a reference to <seq expr> is added between <alt expr> and <par expr> in the <inline expr> production rule.

The following production is added to the Concrete Graphical Grammar of section 7.2.

<seq area> ::=

<inline expression symbol> [***is attached to*** <time interval area>] ***contains***
{ **seq** <operand area>
{ ***is followed by*** <separator area> ***is followed by*** <operand area> }* }
is attached to { <instance axis symbol>* } ***set***
is attached to { <inline gate area>* | <inline order gate area>* } ***set***

In addition, a reference to <seq area> has been added to the <inline expression area> production rule between <opt area> and <par area>.

4.52 Modification – Replacement of time constraint keywords and addition of time statement in <shared inline expr>: Section 7.2

The rule <shared inline expr> in the Concrete Textual Grammar part of section 7.2 replaces the four statements connected with the keywords **startbefore**, **startafter**, **endbefore** and **endafter**, by two statements using the keywords **top** and **bottom**. A time interval statement has been added, which enables the occurrence of <time interval> to be removed from each of the following productions for shared expressions: <shared loop expr>, <shared opt expr>, <shared exc eppr>, <shared alt expr>, and <shared par expr> (Note that <shared seq expr>, being a new rule, has taken this change into account). The new rule for <shared inline expr> becomes:

<shared inline expr> ::=

[<extra global>]{ <shared loop expr> | <shared opt expr> |
<shared alt expr> | <shared seq expr> | <shared par expr> | <shared exc expr> }
[**time** <time interval> <end>]
[**top** <time dest list> <end>]
[**bottom** <time dest list> <end>]

4.53 Modification – Replacement of time constraint keywords and addition of time statement in <inline expr>: Section 7.2

The rule <inline expr> in the Concrete Textual Grammar part of section 7.2 replaces the four statements connected with the keywords **startbefore**, **startafter**, **endbefore** and **endafter**, by two statements using the keywords **top** and **bottom**. A time interval statement is added, which enables the occurrence of <time interval> to be removed from each of the following productions for shared expressions: <loop expr>, <opt expr>, <exc eppr>, <alt expr>, and <par expr> (Note that <seq expr>, being a new rule, has taken this change into account). The new rule for <inline expr> becomes:

<inline expr> ::=

[<extra global>]{ <loop expr> | <opt expr> | <alt expr> |
<seq expr> | < par expr> | <exc expr> }
[**time** <time interval> <end>]
[**top** <time dest list> <end>]
[**bottom** <time dest list> <end>]

4.54 Clarification – Explanation of time constraints on in-line expressions: Section 7.2

The penultimate sentence in the Semantics part of section 7.2 is replaced to reflect the modified grammar in which the keywords **startbefore**, **startafter**, **endbefore**, and **endafter** have been reduced to **top** and **bottom**, and also to clarify the relationship between the time constraint and the events it is constraining.

The existing sentence "In the textual syntax … before or after certain events, respectively" is replaced by:

"In the textual syntax, the keywords **top** and **bottom** are used to locate a time constraint. A time constraint attached to the top of an in-line expression refers dynamically to the first event that occurs within the expression, and conversely a time constraint attached to the bottom of an in-line expression refers dynamically to the last event that occurs within the in-line expression."

4.55 Modification – Replacement of time constraint keywords and moving of time statement in <shared msc reference>: Section 7.3

The rule <shared msc reference> in the Concrete Textual Grammar part of section 7.3 replaces the four statements connected with the keywords **startbefore**, **startafter**, **endbefore** and **endafter**, by two statements using the keywords **top** and **bottom**. The time interval statement is moved within the rule to make the grammar similar to that for in-line expressions. The new rule becomes:

```
<shared msc reference> ::=
                        reference [ <msc reference identification> : ]
                        <msc ref expr> <shared> <end>
                        [ time <time interval> <end>]
                        [ top <time dest list> <end>]
                        [ bottom <time dest list> <end>]
                        <reference gate interface>
```

4.56 Modification – Addition of a time constraint statement and moving the time statement in <msc reference>: Section 7.3

The rule <msc reference> in the Concrete Textual Grammar part of section 7.3 was missing the time constraint statements; therefore, two statements using the keywords **top** and **bottom** are to be added. The time interval statement is moved within the rule to make the grammar similar to that for in-line expressions and shared msc references. The new rule becomes:

```
<msc reference> ::=
                        reference [ <msc reference identification> : ]
                        <msc ref expr> <end>
                        [ time <time interval> <end>]
                        [ top <time dest list> <end>]
                        [ bottom <time dest list> <end>]
                        <reference gate interface>
```

4.57 Clarification – Change of time constraint keywords: Section 7.3

The final sentence in the Semantics part of section 7.3 is to be changed to reflect the changed keywords in the grammar. Thus, the list of keywords "**startbefore**, **startafter**, **endbefore**, and **endafter**" is replaced by "**top** and **bottom**".

4.58 Clerical error – Missing reference to 'seq': Section 7.3

In the opening part of Section 7.3, in the sentence "The **alt**, **par**, **loop**, **opt** and **exc** operators are described in 7.2.", the missing "**seq**" is added and the sentence becomes "The **alt**, **par**, **seq**, **loop**, **opt** and **exc** operators are described in 7.2."

4.59 Clerical error – Incorrect reference to production: Section 7.3

Replace reference to the non-existent production <parameter declaration> by <msc parameter decl> in the Static Requirements part of section 7.3.

4.60 Modification – Allowing actual parameter blocks in any order: Section 7.3

The order in which actual variable, timer, message, and instance parameter blocks are supplied is currently fixed by the grammar. The grammar is to be changed to permit these blocks to be declared in any order. Specifically; the production <actual parameters> is to be changed, and new intermediate productions <actual parameters list> and <actual parameters block> added. The productions are to be:

```
<actual parameters> ::=
                    ( <actual parameters list> )

<actual parameters list> ::=
                    <actual parameters block> [ <end> <actual parameters list> ]

<actual parameters block> ::=
                          <actual data parameters>
                    |     <actual instance parameters>
                    |     <actual message parameters>
                    |     <actual timer parameters>
```

The following paragraph in the Static Requirements part of section 7.3:

"MSC actual parameters must match the corresponding parameter declarations of the MSC definition. The elements separated by <end> in the <actual parameters> must correspond one-to-one with the elements separated by <end> in the <parameter declaration> of the MSC definition."

is to be modified as follows:

"MSC actual parameters must match the corresponding parameter declarations of the MSC definition. The elements separated by <end> in the <actual parameters> must correspond one-to-one with the elements separated by <end> in the <msc parameter decl> of the MSC definition, but the order of blocks may be different. Within a parameter block, the order of individual parameters must be the same between declaration and use."

4.61 Clerical error – Incorrect instance declarations, Figures 12 & 16: Section 7.4

Declaration of instances in the MSC documents depicted in Figures 12 and 16 is incorrect. "**containing** i, j;" is to be replaced by "**inst** i; **inst** j;". The message declarations also use the wrong keyword, but they are to be deleted rather than corrected, since message declarations are optional for unparameterised messages.

4.62 Modification – Addition of time constraint statements in HMSC reference and parallel nodes: Section 7.5

The rule <node> in the Concrete Textual Grammar part of section 7.5 was missing the time constraint statements for msc reference nodes and parallel nodes. Therefore, the rule <node expression> is to be changed to distinguish non-timeable <node> and the new <timeable node>. The modified rule for <node> removes reference expressions and parallel nodes, which are put into the new rule <timeable node>. The latter has time constraint statements expressed in the same way as for the equivalent basic MSC constructs. The new rules are:

```
<node expression> ::=
                          <label name> : { {<timeable node> | <node> } seq ( <label name list> )
                          | end }<end>

<timeable node> ::=
                          { ( <msc ref expr>) | <par expression> }
                          [ time <time interval> <end>]
                          [ top <time dest list> <end>]
                          [ bottom <time dest list> <end>]

<node> ::=
                          <condition identification>
                          | connect
```

4.63 Clerical error – Incorrect reference to production: Section 7.5

Replace references <label name> by <label name> in the Static Requirements part of section 7.5.

4.64 Clerical error – Grammar correction: Section 11

In the final paragraph of section 11, "resetted" is corrected to read "reset".

4.65 Clerical error – Removal of time units from Figures 43, 44 and 46: Section 11

The 's' following some of the time expressions is removed from Figures 43 and 44; likewise the 'ms' in Figure 46 is removed. These were there to indicate time units, but units are not normally part of expressions in most languages. More confusing was that the unit suffixes were only used on some of the time expressions, and not others. The accompanying text to the Figures is also changed to remove the time unit indicators, and the words "time units" have been explicitly used in their place.

4.66 Clerical error – Case of msc keyword in Figures 42 to 46: Section 11

The **msc** keyword in Figures 42 through to 46 is corrected to appaear in lower case instead of the incorrect upper case **MSC**.

4.67 Clerical error – Revealing missing text in Figure 42: Section 11

The parameters associated with the messages in Figure 42 are incorrectly truncated. These are to be revealed to show the parameters in their entirety.

4.68 Clerical error – Incorrect reference to <keywords>: Annex A

Remove the words "the <keyword> and" from the opening sentence of Annex A, since the keywords are not included in the index.

4.69 Open item – Production <end> vs. semicolon in textual grammar

The production <end> is used throughout the concrete grammar where a semicolon would be more accurate. The <end> production allows a comment before the semicolon and is intended to be the concrete version of a graphical comment. However, there are many places that the concrete grammar is not derived from the graphical grammar, but part of it. That is, the concrete grammar is used as text in the graphical part, and so a graphical comment could never occur. In such places it would be better to have a semicolon rather than an <end>. This is not strictly necessary, but it would be a clarification. This modification requires careful searching of the grammar to discover all those places the change should be made: it has not been done as yet.