5i’

ANNEX C2
(to Recommendation Z.100)

SDL PR syntax summary

1 *system*

Figure T1003240-88, (N), p.

2 *definition*

Figure T1003250-88, (N), p.

3 *diagram*

Diagram is defined in GR syntax summary.

4 *system definition* **|** (Basic SDL 2.4.1)

Figure T1003260-88, (N), p.

5 *block definition* | (Basic SDL 2.4.2)

Figure T1003270-88, (N), p.

6 *textual block reference* | (Basic SDL 2.4.2)

Figure T1003280-88, (N), p.

7 *process definition* **|** (Basic SDL 2.4.3)

Figure T1003290-88, (N), p.

8 *textual process reference* | (Basic SDL 2.4.3)

Figure T1003300-88, (N), p.

9 *valid input signal set*

Figure T1003310-88, (N), p.

10 *process body* | (Basic SDL 2.4.3)

Figure T1003320-88, (N), p.

11 *simple expression*

Figure T1003330-88, (N), p.

12 *procedure definition* | (Basic SDL 2.3.4)

Figure T1003340-88, (N), p.

13 *textual procedure reference* | (Basic SDL 2.3.4)

Figure T1003350-88, (N), p.

14 *channel definition* | (Basic SDL 2.5.1)

Figure T1003360-88, (N), p.

15 *signal route definition* | (Basic SDL 2.5.2)

Figure T1003370-88, (N), p.

16 *signal definition* | (Basic SDL 2.5.4)

Figure T1003380-88, (N), p.

17 *signal list definition* | (Basic SDL 2.5.5)

Figure T1003390-88, (N), p.

18 *variable definition* | (Basic SDL 2.6.1.1)

Figure T1003400-88, (N), p.

19 *view definition* | (Basic SDL 2.6.1.2)

Figure T1003410-88, (N), p.

20 *view expression* | (Data 5.5.4.4)

Figure T1003420-88, (N), p.

21 *start* | (Basic SDL 2.6.2)

Figure T1003430-88, (N), p.

22 *state* | (Basic SDL 2.6.3)

Figure T1003440-88, (N), p.

23 *input* | (Basic SDL 2.6.4)

Figure T1003450-88, (N), p.

24 *save* | (Basic SDL 2.6.5)

FigureT1003460-88, (N), p.

25 *transition* | (Basic SDL 2.6.7)

Figure T1003470-88, (N), p.

26 *nextstate* | (Basic SDL 2.6.7.2.1)

Figure T1003480-88, (N), p.

27 *join* | (Basic SDL 2.6.7.2.2)

Figure T1003490-88, (N), p.

28 *stop* | (Basic SDL 2.6.7.2.3)

Figure T1003500-88, (N), p.

29 *return* | (Basic SDL 2.6.7.2.4)

Figure T1003510-88, (N), p.

30 *task* | (Basic SDL 2.7.1)

Figure T1003520-88, (N), p.

31 *create request* | (Basic SDL 2.7.2)

Figure T1003530-88 , (N), p.

32 *procedure call* | (Basic SDL 2.7.3)

Figure T1003540-88, (N), p.

33 *output* | (Basic SDL 2.7.4)

Figure T1003550-88, (N), p.

34 *decision* | (Basic SDL 2.7.5)

Figure T1003560-88, (N), p.

35 *answer* | (Basic SDL 2.7.5)

Figure T1003570-88, (N), p.

36 *timer definition* | (Basic SDL 2.8)

Figure T1003580-88, (N), p.

37 *reset* | (Basic SDL 2.8)

Figure T1003590-88, (N), p.

38 *set* | (Basic SDL 2.8)

Figure T1003600-88, (N), p.

39 *timer active expression* **|** (Data 5.5.4.5)

Figure T1003610-88, (N), p.

40 *end*

Figure T1003620-88, (N), p.

41 *signal list*

Figure T1003630-88, (N), p.

42 *sort list*

Figure T1003640-88, (N), p.

43 *data definition*

Figure T1003650-88, (N), p.

Figure T1003660-88, (N), p.

45 *actual parameters*

Figure T1003670-88, (N), p.

46 *block substructure definition* **|** (Structural concepts 3.2.2)

Figure T1003680-88, (N), p.

47 *block substructure reference* **|** (Structural concepts 3.2.2)

Figure T1003690-88, (N), p.

48 *channel connection* **|** (Structural concepts 3.2.2)

Figure T1003700-88, (N), p.

49 *channel to route connection* | (Structural concepts 3.2.2)

Figure T1003710-88, (N), p.

50 *channel substructure definition* | (Structural concepts 3.2.3)

Figure T1003720-88, (N), p.

51 *channel substructure body* | (Structural concepts 3.2.3)

Figure T1003730-88, (N), p.

52 *channel substructure reference* | (Basic SDL 2.5.1)

Figure T1003740-88, (N), p.

53 *channel endpoint connection* | (Structural concepts 3.2.3)

Figure T1003750-88, (N), p.

54 *signal refinement* **|** (Structural concepts 3.3)

Figure T1003760-88, (N), p.

55 *macro definition* **|** (Additional concepts 4.2.2)

Figure T1003770-88, (N), p.

56 *macro call* **|** (Additional concepts 4.2.3)

Figure T1003780-88, (N), p.

57 *external synonym definition* **|** (Additional concepts 4.3.1)

Figure T1003790-88, (N), p.

Figure T1003800-88, (N), p.

59 *transition option* **|** (Additional concepts 4.3.4)

Figure T1003810-88, (N), p.

60 *service decomposition* **|** (Additional concepts 4.10.1)

Figure T1003820-88, (N), p.

61 *service definition* | (Additional concepts 4.10.2)

Figure T1003830-88, (N), p.

62 *service reference* | (Additional concepts 4.10.1)

Figure T1003840-88, (N), p.

63 *signal route connection* | (Additional concepts 4.10.1)

Figure T1003850-88, (N), p.

64 *service signal definition* | (Additional concepts 4.10.1)

Figure T1003860-88, (N), p.

65 *priority input* | (Additional concepts 4.10.2)

Figure T1003870-88, (N), p.

66 *priority output* | (Additional concepts 4.10.2)

Figure T1003880-88, (N), p.

67 *continuous signal* | (Additional concepts 4.11)

Figure T1003890-88, (N), p.

68 *enabling condition* | (Additional concepts 4.12)

Figure T1003900-88, (N), p.

69 *import definition* | (Additional concepts 4.13)

Figure T1003910-88, (N), p.

70 *import expression* | (Additional concepts 4.13)

Figure T1003920-88, (N), p.

71 *export* | (Additional concepts 4.13)

Figure T1003930-88, (N), p.

72 *sort* | (Data 5.2.2)

Figure T1003940-88, (N), p.

73 *partial type definition* | (Data 5.2.1)

Figure T1003950-88, (N), p.

74 *properties expression* | (Data 5.2.1 and 5.5.3.3)

Figure T1003960-88, (N), p.

75 *operators* **|** (Data 5.2.2 and 5.4.1.8)

Figure T1003970-88, (N), p.

76 *literal list* **|** (Data 5.2.2)

Figure T1003980-88, (N), p.

77 *literal signature* **|** (Data 5.2.2 & 5.4.1.8)

Figure T1003990-88, (N), p.

78 *character string literal identifier*

Figure T1004000-88, (N), p.

79 *operator signature* **|** (Data 5.2.2)

Figure T1004010-88, (N), p.

Figure T1004020-88, (N), p.

81 *equation*

Figure T1004030-88, (N), p.

82 *term*

Figure T1004040-88, (N), p.

83 *ground term*

Figure T1004050-88, (N), p.

Figure T1004060-88, (N), p.

Figure T1004070-88, (N), p.

Figure T1004080-88, (N), p.

Figure T1004090-88, (N), p.

Figure T1004100-88, (N), p.

Figure T1004110-88, (N), p.

Figure T1004120-88, (N), p.

Figure T1004130-88, (N), p.

84 *extended operator identifier*

Figure T1004140-88, (N), p.

85 *infix operator*

Figure T1004150-88, (N), p.

86 *monadic operator*

Figure T1004160-88, (N), p.

87 *quoted operator*

FigureT1004170-88, (N), p.

88 *extended operator name* **|** (Data 5.4.1)

Figure T1004180-88, (N), p.

89 *extended sort* **|** (Data 5.4.1.9)

Figure T1004190-88, (N), p.

90 *extended properties* **|** (Data 5.4.1)

Figure T1004200-88, (N), p.

91 *syntype definition* | (Data 5.4.1.9)

Figure T1004210-88, (N), p.

92 *range conditions* | (Data 5.4.1.9.1)

Figure T1004220-88, (N), p.

93 *constant* | (Data 5.4.1.9)

Figure T1004230-88, (N), p.

94 *structure definition* | (Data 5.4.1.10)

Figure T1004240-88, (N), p.

95 *inheritance rule* **|** (Data 5.4.1.11)

Figure T1004250-88, (N), p.

96 *literal renaming* | (Data 5.4.1.11)

Figure T1004260-88, (N), p.

97 *generator definition* | (Data 5.4.1.12.1)

Figure T1004270-88, (N), p.

98 *generator formal name* | (Data 5.4.1.12.1)

Figure T1004280-88, (N), p.

99 *generator sort* | (Data 5.4.1.12.1)

Figure T1004290-88, (N), p.

100 *generator instantiations* | (Data 5.4.1.12.2)

Figure T1004300-88, (N), p.

101 *generator instantiation* (Data 5.4.1.12.2)

Figure T1004310-88, (N), p.

102 *operator name*

Figure T1004320-88, (N), p.

103 *synonym definition* **|** (Data 5.4.1.13)

Figure T1004330-88, (N), p.

104 *name class literal* **|** (Data 5.4.1.14)

Figure T1004340-88, (N), p.

105 *regular expression* | (Data 5.4.1.14)

Figure T1004350-88, (N), p.

106 *ground expression* | (Data 5.4.2.1)

Figure T1004360-88, (N), p.

107 *active expression* | (Data 5.4.2.2)

Figure T1004370-88, (N), p.

108 *ground primary* | (Data 5.4.2.2)

Figure T1004380-88, (N), p.

109 *field selection*

Figure T1004390-88, (N), p.

110 *operator identifier*

Figure T1004400-88, (N), p.

111 *expression* | (Data 5.5.2.1)

Figure T1004410-88, (N), p.

112 *operand0* | (Data 5.5.2.1)

Figure T1004420-88, (N), p.

113 *operand1* | (Data 5.5.2.1)

Figure T1004430-88, (N), p.

114 *operand2* | (Data 5.5.2.1)

Figure T1004440-88, (N), p.

115 *operand3* | (Data 5.5.2.1)

Figure T1004450-88, (N), p.

116 *operand4* | (Data 5.5.2.1)

Figure T1004460-88, (N), p.

117 *operand5* **|** (Data 5.5.2.1)

Figure T1004470-88, (N), p.

118 *primary*

Figure T1004480-88, (N), p.

119 *active primary*

Figure T1004490-88, (N), p.

120 *active expression list* **|** (Data 5.5.2.1)

Figure T1004500-88, (N), p.

Figure T1004510-88, (N), p.

Figure T1004520-88, (N), p.

Figure T1004530-88, (N), p.

Figure T1004540-88, (N), p.

121 *assignment statement*

Figure T1004550-88, (N), p.

122 *imperative operator* | (Data 5.5.4)

Figure T1004570-88, (N), p.

123 *now expression*

Figure T1004580-88, (N), p.

124 *PId expression*

Figure T1004590-88, (N), p.

125 *literal*

Figure T1004600-88, (N), p.

126 *label*

Figure T1004610-88, (N), p.

127 *comment*

Figure T1004620-88, (N), p.

128 *identifier*

Figure T1004630-88, (N), p.

129 *qualifier*

Figure T1004640-88, (N), p.

130 *decimal integer*

Figure T1004650-88, (N), p.

Lexical rules syntax diagrams

131 *lexical unit*

Figure T1004660-88, (N), p.

132 *name*

Figure T1004670-88, (N), p.

A *name* | must contain at least one alphabetic character.

133 *end of name*

Figure T1004680-88, (N), p.

134 *alphanumeric*

Figure T1004690-88, (N), p.

135 *letter*

Figure T1004700-88, (N), p.

136 *decimal digit*

Figure T1004710-88, (N), p.

137 *national*

Figure T1004720-88, (N), p.

138 *character string literal*

Figure T1004730-88, (N), p.

139 *special*

Figure T100474600-88, (N), p.

140 *composite special*

Figure T1004750-88, (N), p.

141 *text*

Figure T1004760-88, (N), p.

142 *note*

Figure T1004770-88, (N), p.

143 *formal name*

Figure T1004780-88, (N), p.

Figure T1004790-88, (N), p.

Figure T1004800-88, (N), p.

INDEX

Fin premi`ere partie

ANNEX E

(to Recommendation Z.100)

State-oriented representation and pictorial elements

E.1 *Introduction*

SDL is based on an ‘‘extended’’ Finite State Machine (FSM) model. That is, an FSM is extended with objects, such as variables, resources, etc. A machine stays in some state. On receiving a signal, a machine executes a transition, in which relevant actions (e.g. resource allocation and/or deallocation, resource control, signal sending, decision, etc.) are taken. Therefore, the dynamic behaviour of an extended FSM can be explained by describing action sequences on objects for each transition of the FSM in a procedural way.

As a consequence of the state transition, the machine arrives in a new state. The state of an extended FSM can be characterized by objects associated with the state, additional object information (e.g. the value of variables, states of resources, relations between the resources), and signals which can be received in that state. For example, the ‘‘await-first-digit state’’ in telephone call processing is characterized as follows:

Caller: handset-off

Dial tone-sender: dialtone sending

Digit receiver: ready for receiving

Timer: supervising permanent-signal timing

Path: Caller is connected to dial tone-sender and digit receiver, etc.

As can be seen, each state can be defined statically by objects and additional information (qualifying text) associated with that state.

The SDL/GR is extended with **pictorial elements** to define objects associated with each state. The state definitions in terms of pictorial elements are called **state pictures** . The SDL/GR state symbol may include a state picture. This is an optional part of SDL/GR. Figure E-1 shows a state definition example of the ‘‘await-first-digit state’’.

Figure E-1, (N), p.

In many cases, actions on each object, which are required in the transition, can be derived from the difference between state definitions before and after the transition. For example, if some resource appears only after transition, it means that resource allocation action is necessary in the transition. Therefore, if detailed state definitions are given, total actions in the extended FSM transition can usually be derived from the difference between pre-and post-state definitions. However, the sequence of actions in the transition may not be derived from the state definition difference. Therefore, in SDL diagrams, when the sequence of actions is less important, those transition actions which can be derived from the state definition need not be described explicitly. Otherwise, it is desirable to describe action sequences explicitly.

An SDL diagram, in which transitions are described exclusively by explicit action symbols, is called a **transition-oriented version of SDL/GR** .

An SDL/GR diagram, in which states are described using state pictures and transition actions are minimized, is called the **state-oriented version of SDL/GR** or **state-oriented SDL with pictorial elements (SDL/PE).** State pictures can be used advantageously when applied to certain system definitions, resulting in more compact, declarative and less verbal process diagrams.

A combined version is also possible. Thus, these are 3 SDL/GR versions:

a) Transition-oriented version

— Transition sequences are described exclusively by explicit action symbols.

— This is, as it were, a procedural explanation of the extended FSM.

— This version is suitable when the sequence of actions is important and detailed state descriptions are not important.

b) State-oriented version

— The state is described uniquely using pictorial elements. This picture is called a state picture.

— The transition action sequence is implied by the difference between pre-and post-state definitions.

— This is, as it were, a declarative specification of the extended FSM.

— This vesion is suitable when the sequence of actions within each transition is of low importance, when pictorial explanation is desirable, or when a compact representation is desirable.

c) Combined version

— The combined version is suitable when both the sequence of actions within each transition and the detailed state descriptions are under consideration.

Examples of these three versions are given in Figure E-2, E-3 and E-4.

Figure E-2, (N), p.

Figure E-3, (N), p.

Figure E-4, (N), p.

E.2 *Pictorial elements in SDL/GR*

The syntax and semantics defined in Recommendation Z.100 SDL applies to pictorial elements. However, these semantics and syntax are extended as follows:

Pictorial elements represent various objects. The repertoire of pictorial elements is in principle unlimited because new pictorial elements can be invented to suit any new application of the SDL. However, in applications to telecommunications switching and signalling functions, the following repertoire of pictorial elements has been found to have considerable versatility:

— functional block boundary (left or right),

— terminal equipment (various),

— signalling receiver,

— signalling sender,

— combined signalling sender and receiver,

— supervising timer,

— switching path (connected, reserved),

— switching modules,

— charging in progress,

— control elements,

— uncertainty symbol.

Standard symbols for these pictorial elements are recommended in section E.2.2.

E.2.1 *Rules of interpretation*

1) A state symbol may include a **state picture** . A state picture defines de state using pictorial elements and qualifying text.

2) Each pictorial element in a state picture represents an object associated with the state, such as:

— resources,

— variables,

— internal and external boundaries,

— the relations between objects,

— signals which can be received in that state,

— etc.

3) Each pictorial element may have accompanying **qualifying text** . Qualifying text can be used to explain:

— detailed resource name,

— the resource state,

— value for a variable,

— signals relevant to the object,

— etc.

4) Function block boundary:

a) A **function block boundary** is used to express whether a pictorial element is ‘‘internal’’ or ‘‘external’’ to the process. An internal pictorial element represents objects which are owned by the process. An external pictorial element represents objects which are owned by another process under consideration.

b) Rule a) also applies to the distinction between internal and external qualifying text, by substituting the term ‘‘qualifying text’’ for pictorial elements in the rule.

5) Transition interpretation rule:

The total processing involved when a process goes from one state to the following state is the combination of:

— The processing to effect changes to all relevant objects which are derived from the state definition difference.

— The processing explicitly described in the transition, e.g. outputs or tasks.

Thus:

a) The absence from one state if a pictorial element which represents a resource with its presence in the next state implies the allocation of the resource in all transitions joining the two states. This can be equivalently represented by a task(s) showing allocation of the resource in transition(s).

b) If ‘‘presence’’ and ‘‘absence’’ are interchanged in rule a), then ‘‘allocation’’ is replaced by ‘‘deallocation’’.

c) In rule a) if ‘‘pictorial element’’ is replaced by ‘‘external pictorial element’’ then the task should be replaced by an output signal requesting the process which owns the resource to allocate it or simply an input signal from that process saying that it has been allocated.

d) If in rule a) ‘‘presence’’ and ‘‘absence’’ are interchanged and also ‘‘pictorial element’’ is replaced by ‘‘external pictorial element’’ then follow rule c) with ‘‘allocate’’ replaced with ‘‘deallocate’’.

e) Rules a), b), c) and d) also apply to the appearance or disappearance in the state picture of qualifying text, by substituting the term ‘‘qualifying text’’ for pictorial elements in those rules.

6) For a given process diagram, particular pictorial elements (or a particular combination of pictorial elements and qualifying text) should always be placed in the same position within the state picture whenever they appear, so that the presence or absence of this pictorial element (or combination) in a state symbol can be quickly determined by comparing the state picture with other state pictures in the process diagram.

7) When a signal sender appears in a state picture, its qualifying text identifies a signal which is sent during the following transitions.

8) When a sender of a permanent signal (e.g. a ringtone) appears in a state picture, its qualifying text identifies a signal which has been started to be sent during the following transition and in this state.

9) Such transition actions that cannot be derived from the difference of pre- and post-state definitions should be explicitly described in the transition. For example, if a resource with an exported variable does not appear in the pre- and post-states, the necessary actions are better to be described in the transition.

E.2.2 *Recommended symbols for pictorial elements*

When using pictorial elements, each state is represented by a state symbol containing a state picture with the format shown in Figure E-5:

Figure E-5, (MC), p.

A basic set of pictorial elements is recommended for use in SDL/GR with application to the system description of telecommunications call handling processes, including signalling protocols, network services and signalling interworking processes. Many of these pictorial elements are capable of being applied in applications of SDL/GR to other than call handling processes.

The recommended symbols for the basic set of pictorial elements is shown in Figure E-6, and the recommended proprotions for pictorial element symbols are shown in Figure E-7.

Examples of the use of the basic set of pictorial elements are shown in Figure E-8.

E.2.3 *Special conventions and interpretations used in the state oriented extension of SDL/GR*

A number of special conventions and interpretations have been defined in this section with regard to the state-oriented version of SDL/GR. These includes:

— The special interpretation required for process diagrams according to the so-called TRANSITION INTERPRETATION RULE (see § E.2.1, rule 5).

— The unique position of pictorial elements (or pictorial elements and qualifying text) within a state picture that is required when using pictorial elements (see § E.2.1, rule 6).

— The special interpretation required for the variables represented by external pictorial elements and external qualifying text, as proxy variables associated with other processes.

E.3 *Selection criteria for pictorial elements*

The choice of symbols for pictorial elements has been based upon the following considerations and general selection criteria. These should be consulted before developing additional pictorial element symbols for wider applications of the SDL.

1) Ease of reproduction

In order to permit convenient reproduction of SDL diagrams using the dye-line or blue-print methods of reproduction as well as photocopying and photo-printing, pictorial element symbols should consist of clear lines without shading or coloration.

2) Ease of comprehension

a) Appropriateness — The shape of each symbol should be appropriate to the concept that the symbol represents.

b) Distinctiveness — When choosing a basic set of symbols, care should be taken to permit each symbol to be readily distinguishable from others in the set.

c) Affinity — The shapes of pictorial elements representing different but related functions, e.g. receivers and senders, should be related in some obvious way.

d) Association of abbreviated qualifying text with symbols — In some cases it is expected that abbreviated text will be associated with a pictorial element in order to indicate the class of pictorial element; e.g. the letters MFC associated with a receiver symbol to indicate that multi-frequency coded signals are to be received. In these cases, the pictorial elements should incorporate enclosed space to permit the use of a very small number of alphanumerical characters.

e) Limited set — The total number of symbols should be kept to a minimum in order to permit easy learning of the pictorial method.

Figure E-6, (MC), p.

Figure E-7, (MC), p.

Figure E-8, (MC), p.

Figure E-8 (suite), (MC), p.

Figure E-8 (suite), (MC), p.

Figure E-8 (fin), (MC), p.

Recommendation Z.110

CRITERIA FOR THE USE AND APPLICABILITY OF FORMAL DESCRIPTION TECHNIQUES

1 Support for formal description techniques (FDTs)

In view of the complexity and widespread use of Recommendations it is imperative that advanced methods for the development and implementation of these Recommendations be used.

Formal description techniques provide an important approach toward such advanced methods.

In some areas, the use of FDTs is still relatively new and phased procedures are required to introduce their use. This Recommendation proposes the procedures to accomplish this task.

2 FDTs

2.1 *Definitions*

A **formal description technique (FDT)** is a specification method based on a description language using rigorous and unambiguous rules both with respect to developing expressions in the language (formal syntax) and interpreting the meaning of these expressions (formal semantics). FDTs are intended to be used in the development, specification, implementation and verification of Recommendations (or parts thereof).

A **natural language description** is an example of an informal description technique using one of the languages used by CCITT to publish Recommendations. It may be supplemented with mathematical and other accepted notation, figures, etc.

2.2 *Objectives of an FDT*

The goal of an FDT is to permit precise and unambiguous specifications. FDTs are also intended to satisfy objectives such as:

— a basis for analyzing specifications for correctness, efficiency, etc.;

— a basis for determining completeness of specifications;

— a basis for verification of specifications against the requirement of the Recommendation;

— a basis for determining conformance of implementations to Recommendations;

— a basis for determining consistency of specifications between Recommendations;

— a basis for implementation support.

In the current state of the art, in some areas more than one FDT may be needed to accomplish all objectives.

The content of this Recommendation is also published as ISO Resolution ISO/IEC JTC 1/N 145. The statement on precedence in case of several descriptions contained in the JTC 1 document is omitted in this Recommendation.

2.3 *Benefits of an FDT*

The application of an FDT can provide benefits such as:

— improving the quality of Recommendations, which in turn reduces maintenance costs to both CCITT and to users of Recommendations;

— reducing dependency on the natural language to communicate technical concepts in a multilingual environment;

— reducing development time of implementations by using tools that are based on the properties of the FDT;

— making the implementation easier, resulting in better products.

2.4 *Problem with FDTs*

FDTs are advanced techniques which have not yet been widely introduced. In addition, there is a lack of resources in the development of FDTs and formal descriptions (FDs), as well as a lack of expertise within the CCITT Study Groups both to assess the technical merits of the formally described Recommendations and to reach consensus on them.

2.5 *Solutions*

The development of tutorial and educational materials will help to provide widespread understanding of the complexities of FDTs. Nevertheless, time must be permitted for their assimilation.

3 Development and standardization of FDTs

It is important to avoid unnecessary proliferation of FDTs. The following criteria should be met before adopting a new FDTs:

— the need for the FDT should be demonstrated;

— evidence that it is based on a significantly different model from that of an existing FDT should be provided, and

— the usefulness and capabilities of the FDT should be demonstrated.

4 Development and acceptance of formal descriptions

4.1 In future, only standard FDTs or FDTs in the process of being standardized should be used in formal descriptions of Recommendations.

4.2 It is considered that the development of a FD of any particular Recommendation is a decision of the Study Group (in consultation with ISO for collaborative standards). If a FD is to be developed for a new Recommendation, the FD should be progressed, as far as possible, according to the same timetable as the rest of the Recommendation.

4.3 For the evolutionary introduction of FDs into Recommendations three phases can be identified. It is the responsibility of the Study Group to decide which phase initially applies to each FD and the possible evolution of the FD toward another phase. It is not mandatory for a FD to go through the three phases described below and, more generally, it is not mandatory for a FD to evolve.

Phase 1

This phase is characterized by the fact that widespread knowledge of FDTs, and experience in formal descriptions, are lacking; there may not be sufficient resources in the Study Groups to produce or review formal descriptions.

The development of Recommendations has to be based on conventional natural language approaches, leading to Recommendations where the natural language description is the definitive Recommendation.

Study Groups are encouraged to develop FDs of their Recommendations since these efforts may contribute to the quality of the Recommendations by detecting defects, may provide additional understanding to readers, and will support the evolutionary introduction of FDTs.

A formal description produced by a Study Group that can be considered to represent faithfully a significant part of the Recommendation or the complete Recommendation should be published as an appendix to the Recommendation.

Meanwhile Study Groups should develop and provide educational material for the FDTs to support their widespread introduction in the CCITT and Liaison Organizations.

Phase 2

This phase is characterized by the fact that knowledge of FDTs and experience in formal descriptions is more widely available; Study Groups can provide enough resources to support the production of formal descriptions. However, it cannot be assured that enough CCITT Members can review formal descriptions in order to enable them to approve a proposed formally described Recommendation.

The development of Recommendations should still be based on conventional natural language approaches, leading to Recommendation where the natural language description is the definitive standard. However, these developments should be accompanied and supported by the development of formal descriptions of these standards with the objective of improving and supporting the structure, consistency, and correctness of the natural language description.

A formal description, produced by Study Group, that is considered to represent faithfully a significant part of the Recommendation or the complete Recommendation should be published as an annex to the Recommendation.

Meanwhile educational work should continue.

Phase 3

This phase is characterized by the fact that a widespread knowledge of FDTs may be assumed; CCITT Members can provide sufficient resources both to produce and review formal descriptions, and assurance exists that the application of FDTs does not unnecessarily restrict freedom of the implementations.

Study Groups should use FDTs routinely to develop their Recommendations, and the FD(s) become part of the Recommendation together with natural language descriptions.

Whenever a dicrepancy between a natural language description and a formal description, or between two formal descriptions, is detected, the discrepancy should be resolved by changing or improving the natural language description or the FDs without necessarily giving preference to one over the other(s).

4.4 The above procedures for phased development of FDs are intended to aid the progression of FDs within the standards process, not to hinder their progression. However, since there has been little or no actual experience with these procedures, any Study Group having to use them is urged to identify one or more pilot cases and carefully monitor the progress of each within the framework of the procedures. Should procedural problems arise, the Study Group responsible for Formal Description Techniques should be informed and, where possible, recommended procedural modifications should be proposed to alleviate the problems.