ETSI/MTS(34)006

European Telecommunications Standards Institute

MTS#34

10th to 12th April 2002

Rome

Title:	MI/MTS-00073: Checklists for the use of SDL and MSC in Protocol Standards
Source:	STF188
Date:	27 February 2002
Agenda item:	5.8
Document for:	Approval

The Use of SDL & MSC in Standards

These check lists are derived from EG 202 106 revised for SDL-2000 (as defined in ITU-T Z.100), MSC-2000 (as defined in ITU-T Z.120) and EG 201 383. However, it should be noted that EG 201 383 was published prior to SDL-2000 and MSC-2000 and has not been updated so that in some cases it differs from the list below.

The Use of SDL

Not Recommended – not in SDL-2000

The following SDL concepts were valid in SDL-96 (see Z.100 03/96) but are not valid in SDL-2000.

These features also make models difficult to validate or hamper the development of conformance tests and should be avoided completely.

Keywords are in uppercase (**BLOCK)** in the Concept column and lowercase bold (**block**) in the Use Instead column.

Concept	Use Instead	Z.100 (03/93)	Used
ALTERNATIVE external data type definitions	Data types defined within the system using ASN.1 or SDL.	5.4.6	
ANY data type (ASN.1) Not supported by X.680 or SDL-2000	Specific bounded data type.	N/A	
AXIOMS	Operation or explicit behaviour description	5.2.3	
Block partitioning	A number of system definitions, one for each partitioning of interest. Use one or more **package** diagrams for parts common between these systems.	3.2.2	
Channel partitioning	A simplified **system**, for example by replacing channel by a pair of channels leading to/from a block with same contents as channel substructure.	3.2.3	
Equality / Noequality	Not necessary. Only needed in conjunction with **axioms** which are not part of SDL-2000	5.3.1.3	
GENERATOR (User defined) Not valid in SDL-2000	Data types with context parameters	5.3.1.12	
Macros, graphical - behaviour, more than one inlet or outlet	The equivalent macro expansion. Note: behaviour macros with a single inlet and outlet *are* allowed.	4.2	
Macros, graphical - structural	Select or the equivalent expansion.	4.2	
Output with **VIA ALL**	Explicitly listed paths	2.7.4	
SERVICE	A **state aggregation**, but nested behaviour should be avoided.	2.4.5	
SIGNAL REFINEMENT	Simplified **signal** definitions.	3.3	
VIEW and **REVEAL**	A **remote procedure**s to access data across non-normative interfaces or explicit protocol signals to transfer data across normative interfaces. The use of data defined in a containing agent shared between other agents is not recommended.	2.6.1	

Not Recommended

The following SDL concepts make models difficult to validate or hamper the development of conformance tests and should be avoided completely.
Keywords are in uppercase (**BLOCK)** in the Concept column and lowercase bold (**block**) in the Use Instead column.

Concept	Use Instead	Z.100	Used
ANY data type (SDL)	A specific **object** data type.	12.1.5	
ANY in a decision or expression Except possibly in validation models.	A specific value, variable or procedure call. If a random value is needed, it should be calculated.	12.3.4.5	
ATLEAST constraint	Constraints can always be omitted. SDL that is valid with constraints is valid without the constraints.	8.2, 8.3.2	
BLOCK - used as system	A system diagram replacing the block.	9.2	
CREATE textual statement	Graphical create request	11.14.2	
CREATE using type without explicit BLOCK or PROCESS definition	An explicit block or process definition based on the type in the immediate surroundings of the creator.	11.13.2	
Enabling condition	**Input** and **Save**	Z.100 (03/93) 4.12	
EXCEPTION - user defined (unless implied with the same name as a remote procedure timer)	Not needed. See **Raise**.	11.16	
EXCEPTION context parameter	See **EXCEPTION**.	8.2.11	
Exception statement	See **EXCEPTION**.	11.14.9	
EXPORT in a statement list.	An **export** in task symbols on its own.	11.14.2	
Gate on BLOCK or PROCESS	Explicit channels to and from the definition of the block or process. Note: gates can be used freely on types.	8.1.6	
Implicit transition for expected signals.	Explicit empty transition rather than rely on implicit consumption of unwanted (but expected) signals	11.8	
IMPORT and **EXPORT** (**remote** variables)	Explicit protocol messages to transfer data between agents.	12.3.4.2	
Informal text (must be enclosed in single quotes to be legal SDL)	In a task symbol, a procedure call. In a decision symbol, a formal expression (of a data type or **syntype** that has a finite number of values except if there is an **else** path from the decision.	6.4	
Keywords as names (must be mixed case - otherwise taken as the keyword)	A changed name that is more distinct from the keyword.	6.1	
Nested diagram (that is, inner diagrams drawn in place rather than have a reference to the diagram).	Reference symbols for nested diagrams.	7.3 8.1.1 9, 9.4 11.11.1 11.11.2	
Name class and Name class mapping	Explicit list of literals	12.1.9.1, 12.1.9.2	

Not Recommended			
The following SDL concepts make models difficult to validate or hamper the development of conformance tests and should be avoided completely. Keywords are in uppercase (**BLOCK)** in the Concept column and lowercase bold (**block**) in the Use Instead column.			
Concept	**Use Instead**	**Z.100**	**Used**
Nextstate with history - contains hyphen asterisk (Only meaningful when the transition leading to the next state follows a state symbol for a composite state with sub-states.)	Explicit entry points, preferably the default unlabelled entry point.	11.12.2.1	
OUTPUT textual statement	Graphical output symbol	11.14.2	
Predefined exceptions	Explicit checks (for example, for **null**) to ensure do not occur.	D.3.16	
PROCESS - used as system	Encapsulation of the **process** in a **system** so that channels connected to the **process** can be shown.	9.2	
Raise	A transition with the same functionality as the exception handler.	11.12.2.5	
REMOTE variables (See IMPORT and EXPORT)	Only needed if IMPORT and EXPORT used. See also use of shared data.	10.6	
Shared data variable in a block (or the system) containing other agents	Data variable that is not shared within a contained agent, and communication of the value of the variable by **remote procedure**, **remote** data or **signal** exchange.	9.2	
SIGNAL defined as virtual	Another signal defined with a different identifier that **inherits** the **signal** properties.	10.3	
SIGNAL with **OBJECT** data type parameter	Equivalent **value** data type. Can assign to **object** when received if needed.	10.3	
Spontaneous transition	Explicit stimulus (possibly from environment) to trigger the transition.	11.9	
STATE AGGREGATION	No alternative.	11.11.2	
STATE expression	Explicit variable to record the state if needed.	12.3.4.6	
State machine in System (state symbol directly enclosed in a system)	Encapsulate the whole system in a **block** or **process** so that normative channels can be connected to the state machine.	9.1	

Use With Care

The following SDL concepts can cause problems in validation and test development in certain circumstances and should be used with care.
Keywords are in uppercase (**BLOCK)** in the Concept column and lowercase bold (**block**) in the other columns.

Concept	Use when	Consider using instead	Z.100	Used
Asterisk inputlist Asterisk savelist (* in input or save symbol)	The number of different signals that could be received is small. Avoid use in a composite state.	Explicit list of input signals	11.3 11.7	
Asterisk statelist (* in state symbol)	The number of states in the process is small and each transition below the asterisk state is short and non-complex	Explicit transitions in each state. If a transition below the asterisk state is complex, it can be placed in a procedure, which could be called for each explicit state.	11.2	
BREAK and labelled statements	The control flow is discontinuous. A **break** statement can only be used in a statement that contains the label used in the **break** statement.	One or more **if**, **decision** or **loop** statements to avoid the need for a label.	11.14.7	
Composite state graph (a contained **STATE** diagram with the same name as a state symbol)	A state needs to have sub-states to hide complexity, with specific signals causing an exit from the state and transition to a new state. A comment should be used to highlight the use of a composite state.	Expanding the composite state.	11.11.1	
Composite state type context parameter		Avoiding the use of such context parameters.	8.2.12	
Composite state type (a **STATE TYPE** diagram)	Two or more states have the same set of sub-states. Each such state is defined as a **State based on composite state type**. Or, one **state type** behaviour graph is based on another,	No alternative.	8.1.1.5	
Connect from composite state (a line, optionally labelled, directly from a state symbol to a transition - distinguished from input and save etc by the symbol at the end of the line)	A state represents a composite state to connect the exits (at most one unlabelled) to the transition to be interpreted next.	Composite state that has no explicit returns. No alternative if composite state has one or more exits.	11.11.4	
Context parameter	A type or procedure should be reused in different contexts with some elements dependent on the context. See also Composite state type context parameter, Gate context parameter, Interface context parameter, Sort context parameter and Synonym context parameter.	A generic type or procedure that does not have parameters and specialised types defined for each context.	8.2	

Use With Care

The following SDL concepts can cause problems in validation and test development in certain circumstances and should be used with care.
Keywords are in uppercase (**BLOCK)** in the Concept column and lowercase bold (**block**) in the other columns.

Concept	Use when	Consider using instead	Z.100	Used
Dash nextstate (state symbol containing a hyphen) *Note: the use of nextstate with history (hyphen asterisk) is Not Recommended.*	The originating states are clear	Explicit transitions for each state where, in addition, common processing could be placed into a procedure.	11.12.2.1	
DECISION and **IF statements**	A graphical decision is not practical (e.g., within a loop statement).	Graphical decision.	11.14.5	
Delay (arrow head NOT at the end of the channel)	There is no more than one delaying channel between two agents.	If there is more than one delaying channel between two agents, merging these into one channel.	10.1	
Gate context parameter	A type will be used in different contexts and needs to refer to a gate of the context.	See Context parameter	8.2.13	
Interface context parameter	A state type or exception is used and the state or exception needs to refer to an interface of the context in which the state or exception is used.	See Context parameter	8.2.14	
Macro, textual	The replacement text can be kept simple. For StartTimer and StopTimer.	Procedure call or long-hand specification of behaviour	6.2	
Nextstate with parameters	State symbol at the end of a transition denotes a composite state with formal parameters	No alternative.	11.12.2.1	
OBJECT data type definition	The modelling of object data within processes is desirable. Be careful that the objects do not grow indefinitely in size. The passing of objects between agents should be avoided.	Value types	12.1.1	
Optional definition (Select)	Parts of the SDL structure or definitions should be selectively included or excluded.	Separate **system** diagrams for each case using common **package** diagrams.	13.1	
Pid	The identity of a process is to be communicated or stored; should not be included in normative signalling.	Application specific identifier such as Call_Reference. A pid sort specific to the relevant agents.	12.1.6	
Powerset	A mathematical set of data values that have a limited range is needed. A Powerset should not be included as a signal parameter.	The Array, Vector or Bag data types.	D.3.10	

Use With Care

The following SDL concepts can cause problems in validation and test development in certain circumstances and should be used with care.
Keywords are in uppercase (**BLOCK)** in the Concept column and lowercase bold (**block**) in the other columns.

Concept	Use when	Consider using instead	Z.100	Used
Predefined data types (**Integer**, **Real**, **Time**, **Natural**, **Duration**, **Bitstring**, **Octetstring**, **Charstring**, **String**)	Defining another data type in a specification with **constants** or a **size** specified so that it that has finite number of values. Avoid using these predefined types directly.	User defined type of finite size such as enumerated **literals**, or type of finite size defined using the predefiend types..	D.3	
Remote procedure	It is certain that the remote procedure call will not result in implicit signalling across a normative interface.	Explicit signalling	10.5	
RETURN textual statement	For the return of textually defined procedure or operation.	Graphical return symbol (except in textually expressed procedures and methods).	11.14.2	
Sort context parameter	The body of a type depends on data defined in the context where the type is used.	See Context parameter	8.2.10	
Specialization	Several types are very similar. It is suggested to: - minimize the number of virtual components; - have minimal changes in the inherited type; - limit layers of virtuality.	Types defined explicitly. Direct agent definitions thus eliminating types.	8.3 12.1.3	
State based on composite state type (state name followed by colon and state type identity)	See **Composite state type** and **Composite state graph**.		8.1.3.4	
State connection point - named	Additional entries and exits from a composite state are essential.	Passing a value in a parameter that is tested on entry. Returning a value in a variable that is tested on exit.	11.11.3	
STATE or **STATE TYPE** diagram	A composite state that has sub-states as in hierarchical state charts makes the higher level easier to understand and the consumption of some signals in the composite state is not confusing.	A procedure.	11.2	
Stop in procedure	It is obvious that the procedure call may terminate the calling agent. Such procedure calls should be annotated with a comment.	Stop after the procedure call, possibly dependent on testing a value returned from the procedure.	11.12.2.3	
Synonym context parameter	The actual parameter can be a constant expression.		8.2.9	

Use With Care				
The following SDL concepts can cause problems in validation and test development in certain circumstances and should be used with care. Keywords are in uppercase (**BLOCK)** in the Concept column and lowercase bold (**block**) in the other columns.				
Concept	**Use when**	**Consider using instead**	**Z.100**	**Used**
Virtual method	A method needs to be **redefined** in subtypes of a data type.	A method in the subtype that is not **redefined**.	8.3.4	
Virtual transition/save	Redefinition of the behaviour for the signals in a type needs to be permitted in a subtype.	Super type (possibly **abstract**) without **virtual** transitions or saves, that does not include behaviour for these signals and a subtype **adding** the behaviour for each case.	8.3.3	
Virtual type	Redefinition of a component type of another (container) type needs to be permitted in subtypes of the container.	A container type with a context parameter for the component type (previously virtual), or an **abstract** super type that does not include the component, and adding different components in different subtypes.	8.3.2	

Use of MSC

Not Recommended

The following MSC concepts make models difficult to validate or hamper the development of conformance tests and should be avoided completely.

Concept	Use Instead	Z.120	Used
Absolute time	Relative time.	6.3	
Create	Static instances.	4.10	
Dynamic expression in loop boundary	Constant expression (i.e. not involving variables) in loop boundary.	5.6, 7.2, 7.3	
Dynamic instances	Static instances.	4.10, 4.11	
Environment as the diagram frame	Distinct instance(s) with kind name 'environment'.	4.5	
Found message, call and **reply**	Explicit sending of the message, call or reply from a specified instance.	4.3	
Gate	Messages with both events within the same diagram. Messages with both events within the same inline expression.	4.5	
General ordering between different instances.	There is no direct replacement, but explicit synchronization by means of messages should be used to achieve the effect of ordering events on different instances.	4.6	
General ordering in coregion	Simple alternative inline expression, distinct MSCs or splitting the MSC into smaller MSCs.	4.6	
HMSC reference in **basic MSC**	Splitting the MSC into distinct MSCs that, together with the HMSC reference, are referenced from another HMSC.	7.5	
Inheritance	Direct reuse of MSCs (without modifications or additions) or the use of distinct MSCs.	3, 8.3	
Instance decomposition	The same instance granularity throughout all MSCs in the specification.	7.4	
Unrestricted **loop expression**	Restricted upper boundary of loop expression.	7.2, 7.3	
Graphical **loop** in **HMSC**	Loop expression in MSC reference with restricted upper boundary.	7.5	
Lost message, call and **reply**	Indicate by an annotation at the sending instance that message communication is expected but not achieved.	4.3	
MSC parameters	Direct reuse of MSCs (without modifications or additions) or the use of distinct MSCs.	7.3	
Redefined MSC	Direct reuse of MSCs (without modifications or additions) or the use of distinct MSCs.	4.1	
Stop	Static instances	4.11	
Time measurement	There is no direct replacement, but time measurements should not be necessary in standard specifications.	6.9	
Virtual MSC	Direct reuse of MSCs (without modifications or additions) or the use of distinct MSCs.	4.1	

Use With Care				
The following MSC concepts can cause problems in validation and test development in certain circumstances and should be used with care.				
Concept	**Use when**	**Consider using instead**	**Z.120**	**Used**
Alternative inline expression	there is a limited number of alternatives and the number of events in each alternative can be kept low.	Distinct MSCs to express the different scenarios.	7.2	
Coregion	a limited number of events may occur independently.	Simple alternative inline expression or distinct MSCs.	7.1	
Dynamic data expression	annotations of data are not sufficient.	Static data or no data at all.	5.6 – 5.9	
Exception expression	the overall expression complexity can be kept low.	Distinct MSCs to express the different scenarios.	7.2, 7.3	
Inline expression	the combined effect of inline expressions does not cause unnecessary complexity.	Distinct MSCs to express the different scenarios.	7.2	
Loop expression with different lower and upper boundaries	the overall expression complexity can be kept low.	Distinct MSCs to express the different scenarios.	7.2, 7.3	
Method call	the call symbol and the reply symbol appear together on the same page and clearly indicate the synchronized nature of the communication.	Normal asynchronous messages.	4.4	
MSC reference expression	the combined effect of MSC reference expressions does not cause unnecessary complexity. it enables a diagram to be described within one page.	MSC References, each containing only one name.	7.3	
Message parameter	the complexity of parameter data is low and the parameter data is vital for the scenario.	Messages with informal or no parameters.	4.3, 5.8	
Optional expression	the overall expression complexity can be kept low.	Distinct MSCs to express the different scenarios.	7.2, 7.3	
Parallel expression	there is no other choice than to use a parallel expression and the overall expression complexity can be kept low.	The alternative expression (with less complexity).	7.2, 7.3, 7.5	

Concept	Make sure that	Z.120	Yes
MSC Document / the complete MSC specification	the number of sequences covered by the overall MSC document are limited to just those that are necessary and sufficient to cover the required behaviour.	3	