INTERNATIONAL TELECOMMUNICATION UNION

ITU-T Z.333

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

MAN-MACHINE LANGUAGE

METHODOLOGY FOR THE SPECIFICATION OF THE MAN-MACHINE INTERFACE - TOOLS AND METHODS

ITU-T Recommendation Z.333

(Extract from the *Blue Book*)

NOTES

1 ITU-T Recommendation Z.333 was published in Fascicle X.7 of the *Blue Book*. This file is an extract from the *Blue Book*. While the presentation and layout of the text might be slightly different from the *Blue Book* version, the contents of the file are identical to the *Blue Book* version and copyright conditions remain unchanged (see below).

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

Recommendation Z.333

METHODOLOGY FOR THE SPECIFICATION OF THE MAN-MACHINE INTERFACE TOOLS AND METHODS

1 Introduction

This Recommendation presents the tools and methods that support the general working procedure described in Recommendation Z.332. Taken together, Recommendations Z.332 and Z.333 constitute the methodology for the specification of the man-machine interface.

2 List of tools and methods[1)]

The following tools and methods are necessary to support the methodology for the specification of MML functions:

- guidelines,
- modelling,
- MML function decomposition method,
- information structure metalanguage,
- procedure description method,
- formal representation of the syntactic structure of each input and output.

3 Description of available tools

3.1 *Guidelines*

3.1.1 *For phase 1*

Determine for each job:

- the purpose of the job;
- what the system is supposed to do;
- what the user is supposed to do;
- the complexity of the job from the users' perspective (see Note);
- the frequency of the job (see Note);
- at which level in the network hierarchy the job is supposed to be performed (exchange, OMC);
- safety aspects.

Note – The following assumptions have been taken to better identify what is meant for "frequency" and "complexity" of a job.

3.1.1.1 *Frequency*

Low:

- if the job is supposed to be performed weekly or at longer intervals.

Medium:

- if the job is supposed to be performed daily.

---

1) The tools and methods may be improved on the basis of user experience leading to additions or revisions.

High:

– if the job is supposed to be performed several times in a day.

3.1.1.2 *Complexity*

Low:

– low number of parameters (in general sense) – max 0 : 3;

– most information associated with these parameters are not compound;

– there is no semantic relationship among different parameters and parameter values.

Medium:

– the number of parameters is greater than 4 but less than 6-8;

– much information associated to these parameters is compound;

– there is no semantic relationship among parameters and/or parameter values.

High:

– there are many parameters;

– most information associated to these parameters is compound;

– there are semantic relationships among parameters and/or parameter values.

3.1.2 *For Phase 4*

No specific guidelines are provided for phase 2.

3.1.3 *For phase 3*

Three main categories of outputs can be identified within the MML function semantics specification, namely:

1) Response outputs inside the dialogue to the operator inputs.

2) Result outputs whose end-user is supposed to be the operator (e.g. results of reporting or interrogation functions).

3) Result outputs whose end-user is not assumed to be the operator (e.g. data collected for further elaborations).

The partitioning of the output media to be used and its component information entities should not be pursued in detail, with the following guidelines:

– Output media and output characteristics to support the first category of output (output inside dialogue) will not appear in the diagrams.

– Output media and output characteristics to support the second category will be as shown in Figure 1/Z.333.

It is also recognized that the lower level of detail, whose definition will depend on the individual Administration's needs, could in general include the information shown in Figure 2/Z.333.

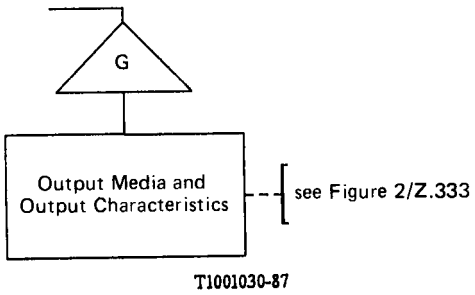


FIGURE 1/Z.333

Output media and output characteristics to support outputs whose end-user is supposed to be the operator

Output media to support outputs belonging to the third category will be shown if possible in the same way as the previous point.

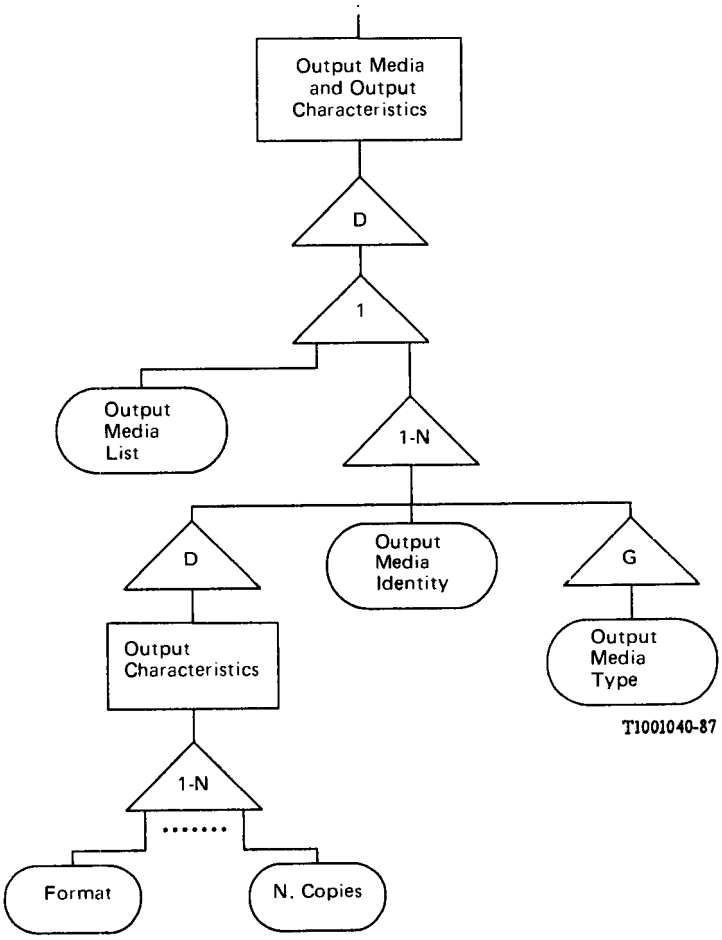


FIGURE 2/Z.333

Output media and output characteristics diagrams

3.1.4 *For phase 4*

To define individual menus and forms, follow the guidelines for the design of menus and forms as defined in Recommendation Z.323.

To define the individual inputs and outputs:

1) Consider what the system is supposed to do.
2) Select options in the function information structure.
3) Define the information to be represented by the command code or equivalent.
4) Define the information to be represented by the parameters and, if appropriate, their order.
5) For each parameter, when appropriate, identify the:
   – range of values,
   – default values,
   – information to be automatically supplied by the system.
6) Define the response outputs within dialogue, the interaction request outputs and outputs outside dialogue when applicable after considering the various mode operating sequences and the users' reactions to the outputs.

7) Define the associated syntactic structure.

8) Select terms and abbreviations for inputs and outputs.

3.1.5 *For phase 5*

1) Define a preliminary operational procedure in functional terms.

2) Finalize operational procedures.

3.1.6 *General guidelines*

1) Determine that the MML functions support the jobs to be performed.

2) It will be necessary to consider:
   - human factor aspects;
   - adequate allocation of authorities;
   - adequate definition of responsibility;
   - training of the user.

3.1.7 *Terminology harmonization guidelines for Phases 1-3*

To harmonize terminology:

1) Utilize existing CCITT vocabulary.

2) Select appropriate terms included in the general functional terminology (Appendix I).

3) Derive specific terms and their definitions pertinent to the functional area involved based on the following considerations:
   - common usage;
   - specificity;
   - translatability.

3.2 *Modelling*

Modelling involves the use of description text and/or figures drawn either with the support of formal symbology and rules (formal modelling) or without such rules (informal modelling).

3.2.1 *The need for models*

A tool available is the construction of informal models of those parts of telecommunications systems which have been selected for MML control. Also the organization of the Administration could be subject to modelling. Several models could apply when defining a job or an MML function. The use of models has the following advantages.

1) Models provide a means for the exchange of functional descriptions.

2) The validity of the derived man-machine interface can be consistently demonstrated by reference to the relevant models.

3.2.2 *Interpretation of models*

A model can be defined as an abstraction of a reality as seen from a certain viewpoint.

In Z.300 Recommendations the viewpoint assumed is that of their users, i.e. Administration specifiers and suppliers designers.

Models should therefore be interpreted as high level specifications and are not aimed to represent, suggest or imply any particular implementation.

They intend to provide only an overview in a conceptual sense of the information which is primarily relevant for the control of each particular functional area and of the main relationships among the various entities in the operator perspective.

Models produced expressly to determine the MML control structure are interpreted purely with that use in mind. Other models must lend themselves to the generation of MML control message sequences. CCITT feel bound to produce models which can be linked with the methods for determining information structure of MML functions.

3.3 *MML function decomposition*

The general MML functions are structured into component MML functions. Multiple levels of decomposition are allowed. For examples see the Annexes to this Recommendation.

3.4 *Information structure meta-language*

Each MML function identified at the lowest level of the MML function decomposition is structured into the information components needed to perform it. A top-down structuring is performed and multiple levels of information decomposition are allowed. The supporting tool is the meta-language presented below.

An aid in understanding of information structuring is to view a MML function as an action on an object(s). Information composed may therefore relate either to objects or to actions.

A general action associated with a MML function can be decomposed into subsidiary actions and modifiers to those actions. It is possible that no decomposition will take place. However, if decomposition is necessary, it should be noted that "decomposition" with respect to actions means both determining subsidiary actions *and* determining any qualifiers (modifiers, options, etc.) associated with the action. The latter is not a true decomposition.

3.4.1 *Decomposition meta-language*

3.4.1.1 *General*

The representation of the information structure associated with a MML function involves the specification of all needed information entities and their inter-relationships.

This representation can be achieved in a consistent manner by means of Information Structure Diagrams, drawn using the meta-language described below. Such meta-language consists of a set of symbols and drawing conventions.

A diagram represents the information structure in a top-down approach, starting from the identification of the MML function to be structured and ending with all the information components felt necessary in the man-machine interworking for that function.

The decomposition process is performed by the use of sequences, selections and iterations, by means of which any type of structure can be obtained.

Unless otherwise stated, the sequence of information is not implied by the order in which different elements are presented in the diagram.

3.4.1.2 *Information entities*

3.4.1.2.1 *Composite parts*

A composite part is an information entity that can be further structured into smaller parts.

The following symbol is used:

3.4.1.2.2 *Component*

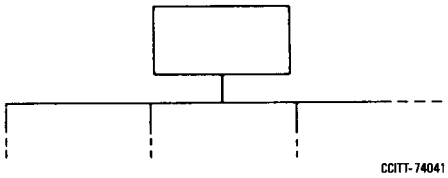
A component is an information entity that is not structured further.

The following symbol is used:

3.4.1.3 *Structuring*

3.4.1.3.1 *Subdivision*

Subdivision in Information Structure Diagrams is shown in the following way:

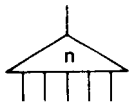


3.4.1.3.2 *Sequence*

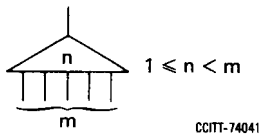
When the order between information entities is relevant, these are specified in sequence. A left-to-right sequence is indicated by the use of arrowheads as follows:

3.4.1.3.3 *Selection*

When a composite part is structured into a number of information entities, of which one or only some are relevant in any one case, a selection mechanism is used, represented by the following symbol:



In the general selection case, *m* possibilities exist from which selection must be made. Of these *m* possibilities a *specified* number, *n*, is to be selected, which implies $n < m$.



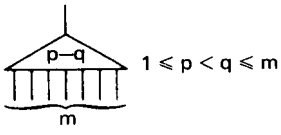
The number of possibilities to be selected, *n*, is given explicitly within the selection symbol, while the total number of possibilities, *m*, is given implicitly by the number of outlets from the selection symbol.

The following cases are allowed:

$n = 1, m > 1$ This is the most common selection case implying that one and only one of the possibilities is to be selected.

$n > 1, m > n$ Multiple selection of *n* of *m* possibilities.

If the number of choices to be made are variable between specified lower and upper limits, a number of possibilities are implied. In this case, both limits are given in the selection symbol:

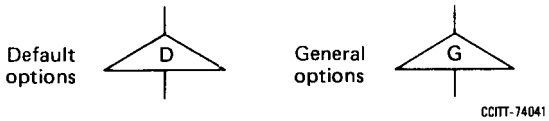


The lower limit *p* indicates the smallest number and *q* the largest number of *different* choices to be made out of the *m* possibilities. It should be noted that each choice may be selected only once.

3.4.1.3.4 *Options*

In some cases, options may be required such as default options or general options.

In these cases, the type of option is indicated by the appropriate capital letter only within the selection symbol, i.e. D for default options, G for general options. Only one outlet from the symbol is allowed:

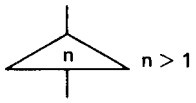


The use of a default option implies that the value taken by an information entity will be provided automatically if the user does not supply a value in the input.

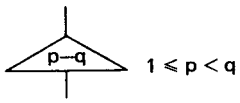
A general option is to be used for various reasons reflecting the needs of manufacturers and the needs of Administrations. The information entities that can be deduced from the outlet of this box can optionally be part of the man-machine interworking. This means either that the information exists in the system in a predetermined manner or that it is not needed. If this distinction must be made an annotation to the information structure diagrams should be made.

3.4.1.3.5 *Iteration*

When a composite part is structured into a number of information entities that can be repeated an arbitrary number of times, an iteration mechanism is used, represented by the following symbol, which has only one outlet:



If a number of interactions can vary within a range, the number of times a part is to be repeated is given as the lower limit *p* and the upper limit *q*.



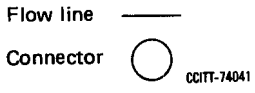
3.4.1.4 *Drawing conventions*

3.4.1.4.1 *Flow lines and connectors*

Every symbol is connected to the symbol it follows by a solid flow line.

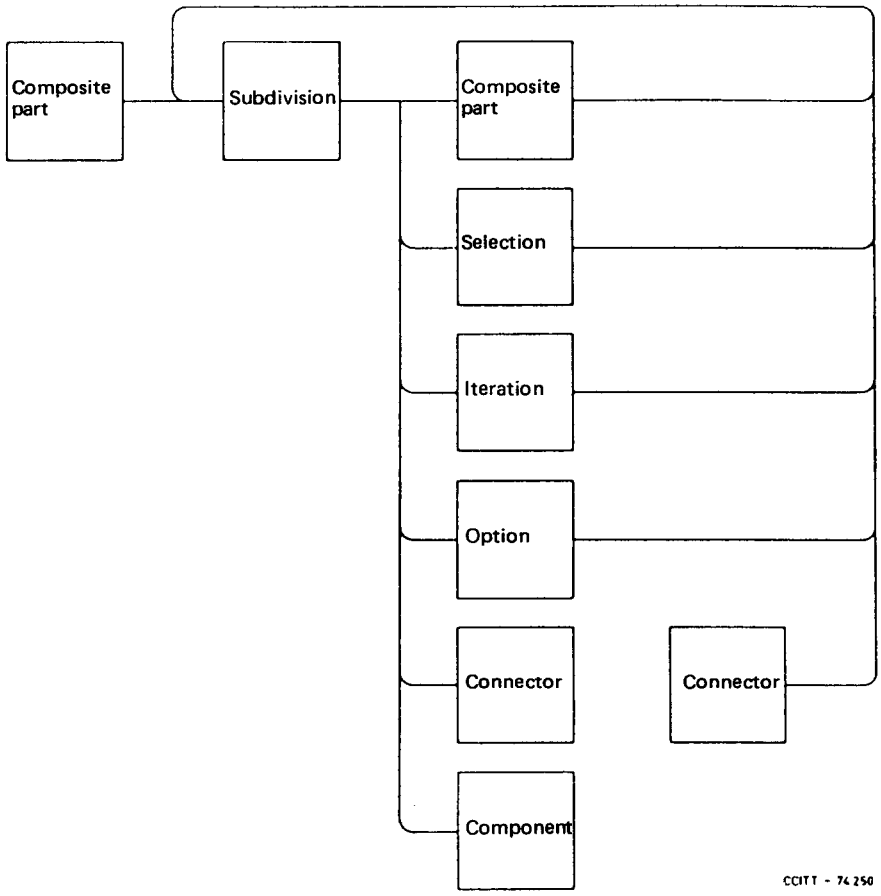
A solid flow line may be broken by a pair of associated connectors, with the flow assumed to be from the out-connector to its associated in-connector. Several out-connectors can be associated with the same in-connector.

Crossed flow lines should be avoided wherever possible.



.4.1.4.2 *Connectivity rules*

Each information structure diagram begins with a composite part symbol and each path of a diagram ends with a component symbol. The drawing of diagrams must follow the connectivity rules represented below.



Note 1 – The symbol types and possible subdivision of flow lines are shown in square boxes.
Note 2 – Subdivision includes the trivial case of single continuous flow line.

3.4.1.4.3 *Annotations*

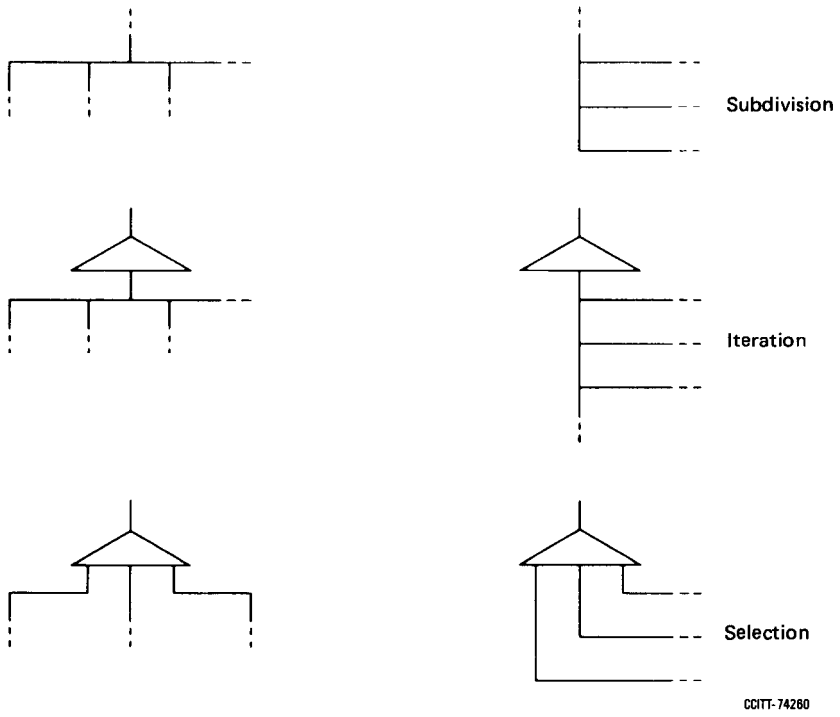
Annotations are denoted by the following symbol, where *n* is a number referencing a note providing descriptive and/or explanatory information.

Annotation – – – – – – [n

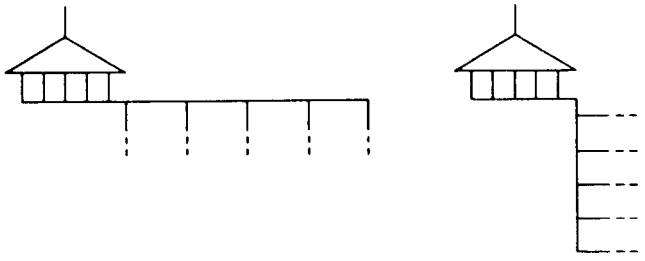
Annotations may be connected by a dashed line to any symbol or flow line.

3.4.1.4.4 *Special Notations*

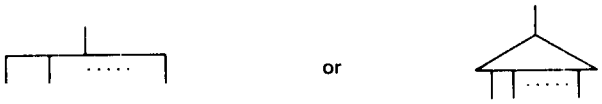
Instead of the normal structuring symbology where the structuring is shown horizontally, a vertical symbology may be used where this is advantageous, i.e. for saving of space. This vertical symbology may be used with all types of structuring.



For the selection symbol, in case of a high number of possibilities, the following drawing conventions are also allowed:



Where the number of information entities in a structure is undetermined, this could be shown as



depending on the type of structuring used.

3.5 *Procedure description method*

Man-machine dialogue may be considered a feature of an SPC system and may be represented by means of two processes: one related to the user, the other related to the system. These two processes exchange information by means of signals that for MML purposes are intended to be mostly inputs and outputs.

In particular, the description of MML operational procedures may be made by focusing the attention on one of the machine logic functions, the associated MML function, and describing the process that performs this function.

To reduce the complexity of drawings, it seems useful to limit the description to the main signals between user and system, i.e. inputs and outputs, and to omit showing features such as timing, error reporting, editing procedures, etc., that may be described elsewhere by means of SDL if needed. For an example see Appendix II.

3.5.1 *Features to be used in the description*

A MML operational procedure can be considered as a process whose behaviour may be specified in terms of inputs, states, transitions, decisions, outputs and tasks.

In the following paragraphs, basic concepts of SDL are interpreted in the context of MML applications.

3.5.1.1 *Input*

An input is a set of data which is input by the user and which is recognized by the MML operational procedure. Input may be, e.g commands in direct information entry, or other types of data.

3.5.1.2 *State*

A state is a condition in which the action of the MML procedure is suspended awaiting an input.

3.5.1.3 *Transition*

A transition is a sequence of actions which occurs when a MML operational procedure changes from one state to another as a reaction to an input.

3.5.1.4 *Decision*

A decision is an action within a transition which asks a question to which the answer can be obtained at that instant and chooses one of several possible paths to continue the transitions.

3.5.1.5 *Output*

Output is a set of data which is output by the MML operational procedure and which in turn is used as an input to the operational process.

3.5.1.6 *Task*

A task is any action within a transition that is neither a decision nor an output.

3.5.1.7 *Symbols and rules*

Symbols and rules are those defined in the SDL Z.100 series Recommendations.

3.6 *Formal representation of the syntactic structure of specific inputs and outputs*

The formal representation of the syntactic structure of specific inputs and outputs may be provided by the use of the existing syntax metalanguage in Recommendation Z.302. The use of the Backus Naur Form (BNF) has also been suggested as possibly being more effective. As advanced terminal capabilities are being considered by the MML Sub-Working Party, additional methods may be needed. The suitability of these methods must be studied further, and if possible, a single method recommended.

3.6.1 *Backus Naur Form (BNF)*

Inputs and outputs are defined as sequences of terminal elements and/or non-terminal elements.

Terminal elements are characters belonging to the MML character set as defined in Recommendation Z.314 and the syntactic elements as defined in Recommendations Z.314, Z.315 and Z.316. Syntactic elements are indicated by means of their name written with small letters between angular brackets (< and >).

Non-terminal elements are elements that must be further defined again as sequences of terminal and/or non-terminal elements. They are indicated by one or more words written with small letters between angular brackets (< and >).

3.6.1.1 *Notation*

Definitions are indicated by writing commands or non-terminal elements on the left hand of the symbol ::= (double colon, equal sign) and, on the right hand side, one or more sequences of terminal and/or non-terminal elements.

Alternative choices are indicated separated by | (vertical bar).

Terminal and non-terminal elements may be grouped together by using braces ({ and }). Repetition of these groups is indicated by means of two subscripts after the braces, one for the minimum, one for the maximum number of times the group may be repeated.

If a group of terminal and non-terminal elements is written between brackets ([and]) the group is optional.

For an example see Appendix III.

APPENDIX I

(to Recommendation Z.333)

Glossary of common terms used in the

specification of the man-machine interface

This glossary of common terms is to be utilized where applicable by CCITT bodies when applying phases 1-3 of the methodology. It is expected to evolve as the methodology is applied to a wider range of areas. This document is not intended to constrain manufacturers' and administrations' choice of terms to represent these concepts at the actual man-machine interface.

It has been noted in Recommendation Z.332 that it is useful to view MML functions as actions upon objects. The concepts represented by the terms in the present collection are limited to action concepts. It is expected that as this glossary evolves, most action concepts will be defined here since they generally apply to more than one functional areas. Conversely, object concepts will generally be specific to a functional area and thus defined in the glossary associated with a functional area.

Among the concepts for actions that may be performed at the man-machine interface are concepts for which the proper object of the action is:

- data only
- equipment only
- either data or equipment.

These three categories of actions correspond to the three major divisions of this glossary.

A number of the concepts below are best understood and normally utilized in complementary pairs; these cases will be indicated by notation such as CREATE/DELETE.

I.1 *Data management actions*

The term **data set** is defined to be a user-accessible set of one or more data items characterized by a particular use, and also by the constraints on data format and/or values that make it suitable for this use.

I.1.1 *CREATE/DELETE*

The following concepts concern user control of the existence of data sets within the system.

CREATE: Establish in the system a new data set.

Examples: CREATE A MEASUREMENT SET, CREATE AN OBJECT LIST.

DELETE: Eliminate a data set from the system.

Examples: DELETE A MEASUREMENT SET, DELETE AN OBJECT LIST.

I.1.2 *CHANGE and EDIT*

The modification of data is normally accomplished by one of two basic methods. The first method (CHANGE) is through the use of functionally specific inputs and outputs intended to be used to modify particular data set types or even particular data items within those data sets. The second method of data modification (EDIT) allows the user to perform changes directly to a display of the data that is to be modified.

Taking this into account, CCITT organizations applying the methodology described in this recommendation should employ the term CHANGE for any data modification requirements, except in cases where the capability to EDIT would have clear advantages, such as in the example given below.

CHANGE: Modify specified data items in a data set via an input or inputs intended for that purpose.

Examples: CHANGE ANALYSIS THRESHOLDS.

EDIT: Display a specified data set and subsequently modify the data set. A common system capability, e.g., editor, is normally used to support such an action.

Examples: EDIT TRAFFIC DATA RECORDS.

I.1.3 *ACTIVATE/DEACTIVATE*

The creation of a data set does not necessarily imply that that data is immediately available for use by the system for its intended purpose. The following concepts make a previously created data set available or unavailable to the system.

ACTIVATE: Initiate a system process that requires preliminary data entry, or make a previously entered data set available to the system for its intended use.

Examples: ACTIVATE A MEASUREMENT, ACTIVATE A ROUTINE TEST.

DEACTIVATE: Terminate a system process initiated by an ACTIVATE action, or make a data set unavailable for use by the system.

Examples: DEACTIVATE A MEASUREMENT, DEACTIVATE A ROUTINE TEST.

I.1.4 *FILTER and SORT*

These concepts allow the user to manipulate data to be subsequently accessed or stored.

FILTER: Form a subset of a data set consisting of all data items in the set meeting specified criteria. The original data set is unaffected by this action.

Example: FILTER TROUBLE OR RESTORAL REPORTS.

SORT: Rearrange the order of a data set according to specified (or default) criteria. The contents of the original set is not affected by this action, only its order.

Example: SORT A FILE OF NAMES (e.g. in alphabetical order).

I.1.5 *INTERROGATE and BROWSE*

The concepts below describe system actions that allow user access to specified portions of the data that has been created by the user or by the system.

INTERROGATE: Provide a display of the current values of the items in one or more data sets.

Examples: INTERROGATE A MEASUREMENT, INTERROGATE A MEASUREMENT TYPE.

BROWSE: Display sequentially the current values of items in a data set. The user may examine the data items in either the forward or backward direction.

Example: BROWSE REPORT FILES.

I.1.6 *INPUT/OUTPUT and ROUTE*

The concepts in this section concern the transfer of data from one location to another.

INPUT: Enter data by means of a user terminal into the system.

Example: INPUT TROUBLE OR RESTORAL REPORT.

OUTPUT: Transfer specified data from the system to the user terminal (e.g. VDT, printer).

Example: OUTPUT SUMMARY REPORT.

The distinction between OUTPUT and INTERROGATE (I.1.5) is that INTERROGATE simply gives a read-back of user-created data, whereas OUTPUT refers to data upon which the system itself has acted in some way, e.g. reports.

ROUTE: Instruct the system that any subsequent messages, classes of data, or message types indicated should be output to specified media.

Example: ROUTE OUTPUT OF REPORTS.

I.2 *Equipment Management Actions*

I.2.1 *REMOVE/RESTORE and SET*

Equipment units can often simply be placed either out of service or in service under software control. The pair REMOVE/RESTORE represents this pair of actions. Manipulation of the status of objects with a more complicated set of maintenance states is expressed by the system action SET, which normally also covers the out of service and in service states. The REMOVE/RESTORE pair is used frequently and is sufficient for a large range of equipment, hence is singled out here as an important special case of the SET action.

REMOVE: Take specified equipment units out of service. The system still retains knowledge of the units so that they may be returned to service by the RESTORE action defined below, automatic recovery, or manual override.

Example: REMOVE CIRCUIT.

RESTORE: Return specified units to service.

Example: RESTORE CIRCUIT.

SET: Place equipment in a specified state (number of states >2). Possible states include in service and out of service.

Example: SET EQUIPMENT UNIT.

I.2.2 *ALLOW/INHIBIT*

Modern systems (e.g. for maintenance or control) utilize many system functions which occur automatically or dependent only upon the detection of certain conditions. Often it is essential to be able to instruct the system not to perform these functions, even should the appropriate set of conditions arise. The complementary capability to return the automatically controlled function to its normal state must then also be provided.

ALLOW: Permit specified system actions, system responses or functions to occur. These functions may be inhibited by system design or by the INHIBIT system action defined below.

Example: ALLOW THRESHOLD.

INHIBIT: Prevent the specified system actions, system responses or functions from occurring. These functions may normally be allowed by the system design, or by the ALLOW action defined above.

Example: INHIBIT THRESHOLD.

I.3 *Management actions that may apply to data or equipment*

INITIALIZE: Put specified data or equipment into a predefined initial (normal) condition or value.

Examples: INITIALIZE THRESHOLD COUNTER, INITIALIZE OUTPUT DEVICE.

EXECUTE: Perform a predefined procedure.

VERIFY: Perform the enforcement of a consistency rule on a specified set of data.

CONNECT: Make a connection between two existing entities.

DISCONNECT: Break a previously established connection.

START: Initiate a procedure or process.

STOP: Terminate the specified activity and leave the system in a defined state.

SUSPEND: Hold an activity temporarily.

RESUME: Continue an activity previously suspended.

APPENDIX II

(to Recommendation Z.333)

Procedure description example

The job “create a new traffic measurement” is described as a procedure in which two different SDL processes, the user process and the system process, are shown.

Only the relevant aspects of the procedure are represented in the diagrams; some features are omitted such as a rejection output due to syntactic errors and related correction procedures, etc., which are common to the other procedures.

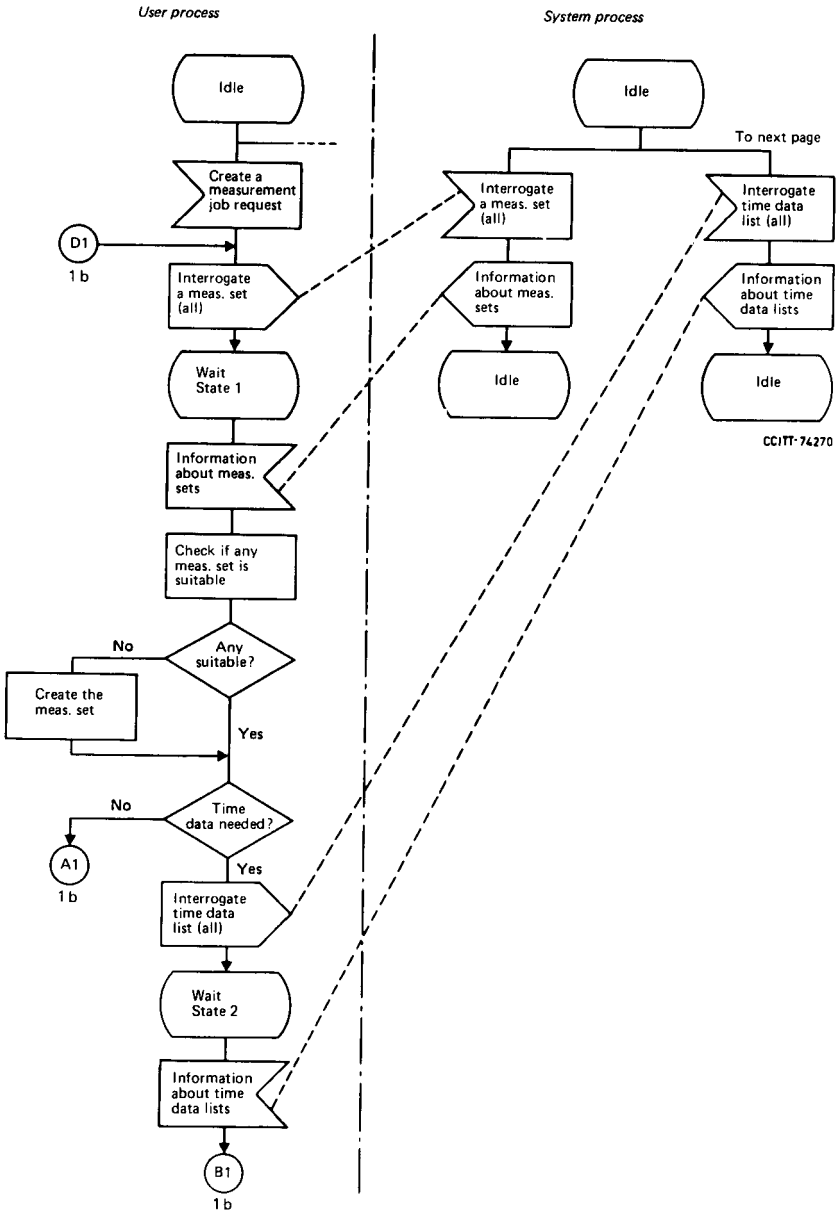


FIGURE II-1a/Z.333

Procedure description example

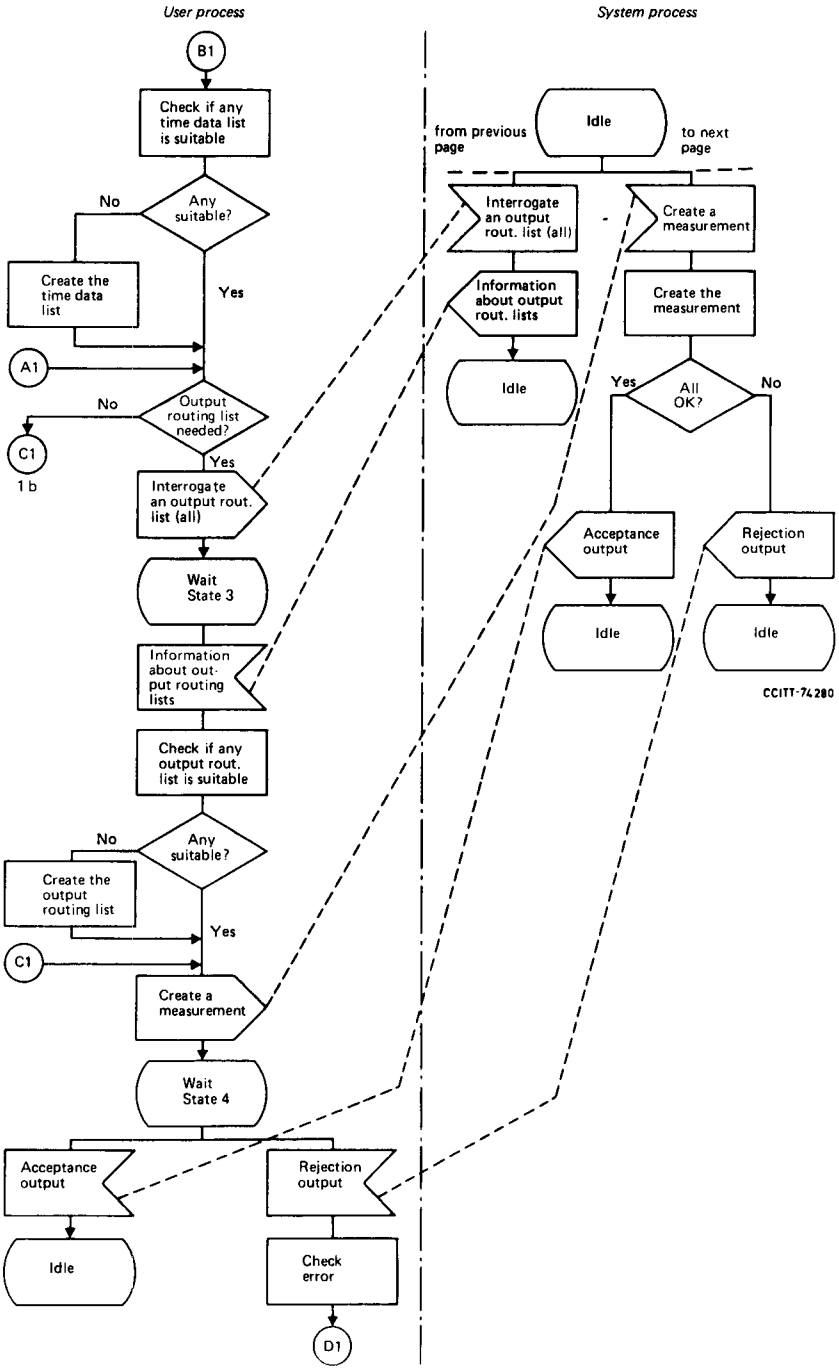


FIGURE II-1b/Z.333

Procedure description example *(cont.)*

APPENDIX III

(to Recommendation Z.333)

Examples of the use of the Backus Naur Form (BNF)

Applying the BNF meta-language described in § 2.6.1 to the traffic measurement functions (see Recommendation Z.336, Annex A, (Figures B-9/Z.336 and B-14/Z.336), the following BNF examples are derived with the assumption of a one to one relationship between the MML function and associated command:

a) *Function "create an object list":*

<create an object list> ::= <command code>:
<object list indentity>
{,<list of objects of one type>}; (1 - N)

<object list identity> ::= <parameter name> = <symbolic name>

<list of object of one type> ::= <type of objects> = <objects identity>

<type of objects> ::= <parameter name>

<object identity> ::= <decimal numeral> {{&<decimal numeral>} | {&&<decimal numeral>}} | (O – N)
<symbolic name> {&<symbolic name>} (O – N)

b) *Function "delete an object list"*

<delete an object list> ::= <command code>:
<list of identities of object list>;

<list of identities of object list> ::= <parameter name> =
<symbolic name> {&<symbolic name>}