SOFTWARE ENGINEERING ENVIRONMENTS

Proceedings of the Symposium
held in Lahnstein, Federal Republic of Germany
June 16-20, 1980

organized by
Gesellschaft für Mathematik
und Datenverarbeitung mbH Bonn
Institut für Software-Technologie

ISBN: 0-444-86133-5

edited by
Horst HÜNKE

1931 N·H 1981
P C

NORTH-HOLLAND PUBLISHING COMPANY – AMSTERDAM • NEW YORK • OXFORD

1981

Håndlykken, Petter and Kristen Nygaard, Kristen and (1980), The Delta system description language, main concepts and experience from use (pp. 173-190).
In: H. Hünke (ed.) Proceedings of the Symposium on *Software Engineering Environments* in Lahnstein, Germany, June 16-20, 1980. North-Holland, Amsterdam.

Software Engineering Environments
H. Hünke, editor
North-Holland Publishing Company

THE DELTA SYSTEM DESCRIPTION LANGUAGE MOTIVATION, MAIN CONCEPTS AND EXPERIENCE FROM USE

Petter Håndlykken and Kristen Nygaard

Norwegian Computing Center
Oslo, Norway

The paper presents a set of concepts and a language for system description, which may be used in development of computer-based systems. Experiences from use of the language and direction for future work are presented.

1. INTRODUCTION.

A set of concepts and a language for system description was developed within a research project at the Norwegian Computing Center in the period 1973-75*). The concepts and the language developed are intended to be used for communicating on complex systems, and to express formal descriptions of such systems.

The language, called the DELTA language, was designed for use mainly in the development of computer-based systems in administration and research. The language and the concepts do not include specific principles for organising the system development process, nor do they assume a specific method for producing software specified in the language.

2. BACKGROUND.

The development of the language tools that led to the DELTA language, started for two main reasons:

- a need was seen for improved language tools to be used in the systems development process. Demands for such tools emerged from a research project carried out for the Norwegian Iron and Metalworkers Union in the period 1971-73 (Nygaard and Bergo, 1974).
- experience from using SIMULA as a language for programming and systems description, showed that the language could be useful for discussing the structure and operation of complex systems. SIMULA was originally designed as "a language for programming and description of discrete-event systems" (Dahl and Nygaard, 1965), (Birtwistle et. al., 1973). It was, however, obvious that SIMULA, being a programming language, has limitations as a description language for interpersonal communication.

---

*) The project was sponsored by the Royal Norwegian Council for Scientific and Industrial Research (NTNF) and to some extent also by the Norwegian Research Council for Science and the Humanities (NAVF).

For these reasons a research project on the DEvelopment of Language Tools for Administration and Research (DELTA) was undertaken. The main result from the project was a set of concepts and a language for systems description (Holbæk-Hanssen, Håndlykken and Nygaard, 1975).

3. MOTIVATION AND PURPOSE.

The DELTA language was intended to be used in a variety of communication situations:

- Communication between system analysts and those being influenced by their systems in various ways.

- Communication on specific systems between trade union members, and system analysts working for trade unions.

- Communication between computer programming experts and scientists in various fields.

- Communication between people working in interdisciplinary teams, having different ways of conceiving the part of the world considered.

A language that is to be used for such communication has to fulfill a number of demands:

- It should be possible to describe a wide class of systems in a manner that is natural to those involved in the communication. This implies that it should be possible to describe alternative views of the part of the world considered.

- The conceptual framework of the language should fit well into natural language descriptions. It should be possible to integrate descriptions in natural language with formalised descriptions. Aspects of a system which may not be portrayed formally could thus be included in the description.

- The concepts and the formal language should have a mathematically sound basis, allowing detailed formal descriptions of systems and parts of systems. Descriptions of software to be developed should serve as a good basis for implementation.

The language is intended to be learnt thoroughly by those who are to produce or study in detail formal descriptions in the language, mainly system analysts. The main concepts of the language are, however, intended to be understood by everybody taking actively part in the development process, and also by many of those using the systems developed.

The documents available today for users of the language are:

- An extensive definition and discussion of the DELTA language and its concepts (in English) (Holbæk-Hanssen, Håndlykken and Nygaard, 1975).

- A brief introduction to the DELTA language (in Norwegian) (Holbæk-Hanssen, 1978).

- A number of reports on the use of the language in various applications.

The present DELTA language has from the outset been considered by the designers as a preliminary result. Based on experience gained in use, the language will be modified and presented in a final version.

4. MAIN CONCEPTS.

The DELTA language is concerned with the description of systems. Many definitions of the term system exist, we will use the term defined in the following way:

A system is a part of the world,
which a person or a group of persons,
during some time and for some reason,
choose to regard as a whole
consisting of parts called components.
Each component is characterised by
selected properties and by actions which may involve itself and other components.

An important aspect of this definition is that nothing is a system by itself, as an inherent quality. We may regard a part of the environment as a system, but we may also choose to regard it in other ways. The same part of the environment may be regarded as a system in many different ways.

According to the definition a system is always a part of the physically existing world. An important class of such systems is those which exist in our minds, as a result of our thought processes. Systems physically materialised as states of the cells of our brains, are called mental systems. Systems external to the human mind are called manifest systems.

A person will consider a part of the world as a system in order to get a better understanding of it. The part of the world considered will be called the referent system. The understanding of the referent system is based upon mental systems created and manipulated in the mind of the person. The mental systems are in some sense similar to the referent system, and we will call these mental systems mental model systems or mental models.

The mental model system is generated by the person, who in this case acts as a system generator.

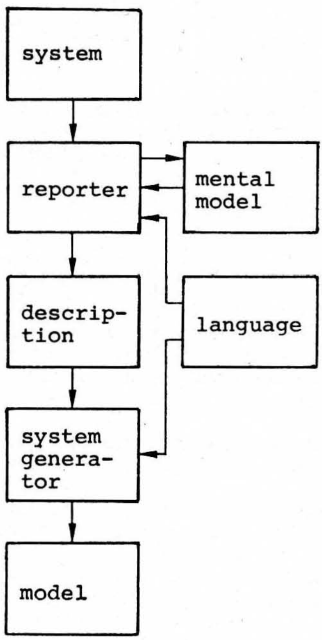
The human mind is not able to manipulate concurrently a large number of components of a mental model with a complex structure. For this reason the person may decide to generate a manifest model system to portray the properties of the referent system. The manifest model system may consist of symbols and drawings on pieces of paper, a database within a computer, a computer simulation model or a model on a different scale of size (e.g. a ship model). In our use of the word, a model is always itself a system. It is related to another system, the referent system.

Information about a system may be communicated by means of various techniques and tools. A classification proposed in (Bjerg and Nielsen 1977), illustrates the variety of approaches:

1. Information by system exposure
   - 1.1. Exposure to similar systems
   - 1.2. Exposure to partial models
     - 1.2.1. Static (Pictures, graphs)
     - 1.2.2. Dynamic (Movies, simulators)

2. Information by system description
   - 2.1. Static description (snapshots, dumps, invariants)
   - 2.2. Dynamic description
     - 2.2.1. Narratives (Traces etc.)
     - 2.2.2. Generating descriptions used by generators (programs, plays etc.)

A computer program is a detailed generating description, a prescription for the behaviour of computing equipment. In the DELTA project we have considered higher level and more comprehensive generating descriptions, which can be related to the program components and used as their specification. DELTA is intended as a language for making such descriptions. This does not imply that we regard generating descriptions as being the only, or even the most important ones in all cases. Usually a variety of approaches are necessary.



The system reporter is observing a system, generating a mental model and communicating this model in a description. Someone is receiving the description and acts as a system generator producing a model of the system.

DELTA systems.

When designing complex computerized systems, people have some model of the systems (program executions, databases etc.) more or less clearly visualised, in their minds. Models of this kind were precisely formulated and standardised as a basis for the development, understanding and definition of the SIMULA (Nygaard and Dahl, 1978 and Birtwistle et.al. 1979) language.

In DELTA the concept of computer program executions has been generalised, since continuous interactions and changes of state are taken into account. This family of model systems are called DELTA systems. The components of a DELTA system are called objects. The objects exist as a changing collection of rectangles containing sequences of characters, and which are interconnected by directed lines. The rectangles are called entities. An object may exist as one or more entities.

A DELTA system generator is a system generator which is able to generate a DELTA system upon a substrate. A DELTA system generator may e.g. be a person materialising the DELTA system by making drawings on a blackboard, or by means of a computer and a display screen.

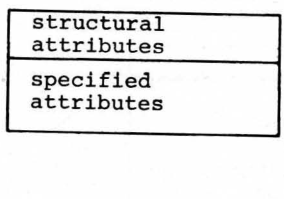
The behaviour of DELTA systems can be specified formally in a language called the DELTA language. An abstract interpreter called the idealized DELTA system generator gives a mathematical definition of the generation of DELTA systems according to a formal description. As we will see the abstract interpreter will not generally be a digital computer. The DELTA language is thus not a programming language.

Objects.

An object in a DELTA system is represented by one or more entities in the DELTA system.

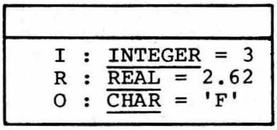
The properties of an object are partly given by the DELTA language definition, we call these properties structural. Partly the properties are specified, by the user of DELTA, which means that they are defined in the description of the system, and may only be altered by altering the description. Finally there are the actual properties referring to the state of the object at a specific moment of time.

The properties of an object are represented by attributes, which are sequences of characters contained in the entity (or entities) constituting the object.

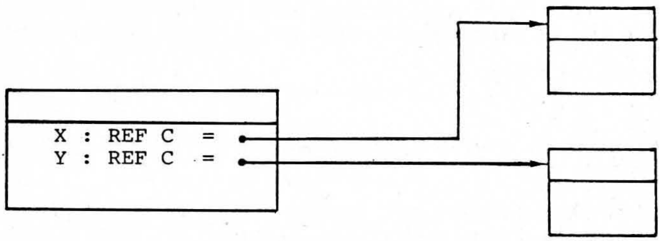


The specified attributes of an object may be quantities, references or patterns.

Quantities are data items characterising the object. A quantity is the association of a name with a type and a value. The name is generally specified so that it associates the quantity in the model system with a corresponding property of the component in the referent system. The type indicates the way in which an amount or quality is measured, a set of possible states which may be observed and a set of operations possible upon these states. The value is the state actually observed at a specific moment of time. The value may be variable or constant.

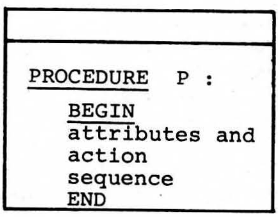


References are data items indicating another object in the system. A reference is an association of a name with a qualification and a value. The name identifies the reference and is generally specified to associate the reference with a corresponding property of the referent system. The qualification defines a class of objects which may be indicated by the reference. The value is the object actually indicated by the reference. A reference may have the value NONE i.e. indicating no object.

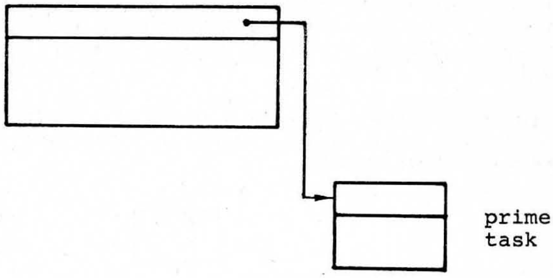


Patterns are definitions of categories. The categories defined by a pattern may be:

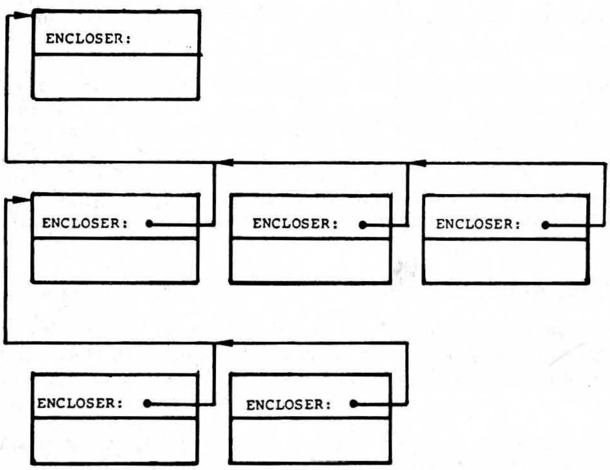
- types of quantities,
- classes of objects enclosed within the object,
- procedures giving a rule for the execution of actions,
- functions giving a rule for defining a value of a specific type of quantity or reference qualification.



The actions associated with an object are specified as a prime task that the object is to carry out in the period of time the system is considered. The prime task may consist of a set of attributes characterising the task and a sequence of actions to be performed. The prime task is represented by a separate entity in the DELTA system.



The objects contained in a DELTA system are ordered in a hierarchy: The system will always have one unique object called the system object representing the system as a whole. All other objects in the system are enclosed by the system object or by some other object in the system. An object is enclosed by one and the same object throughout its lifetime, and may not exist without its enclosing object.

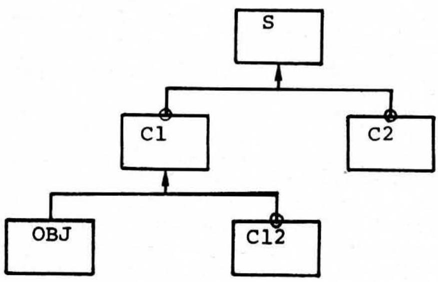


(In many systems permanent or changing aggregates of objects being at the same hierarchical level represent natural units in a decomposition process (e.g. queues). Such aggregates should often not be portrayed by a hierarchical encloser relationship.)

Classes and subclasses of objects.

A referent system is often considered as containing a number of components which may be described by a common description. Therefore the notion of classes of objects is introduced in DELTA systems. A class of objects is a set of objects having common specified properties. The common specification is a class pattern attribute of the object enclosing all the members of the class. The class pattern specifies a set of attributes, and the actions associated with objects belonging to the class. The actual properties may, however, vary from object to object within the class.

It is often convenient to draw a diagram illustrating the classes of objects existing in a DELTA system (a structure diagram). The names of the classes are indicated.



The diagram should not be confused with the diagrams representing DELTA objects.

Within a class of components we often find components which we want to distinguish from others by specified properties in addition to those shared by all the members of the class. To model this situation the (SIMULA 67) concept of a subclass is introduced. A subclass is a subset of objects within a class which have common specified properties in addition to those common to all objects in the class. A subclass is defined by a pattern attribute in a similar manner to a class.

The execution of actions.

The actions to be executed by an object in the period of time under consideration, are specified in its prime task. Within the prime task there is a sequence of actions which the object should execute one by one.

The concept of time is represented in DELTA models by a quantity named TIME which is available everywhere in the system. The value of the quantity portrays the moment of time considered in the referent system. The time unit may be chosen in each case. The value of TIME will increase continuously, from the value indicating the beginning of the period of time considered, to the value indicating the end of the period.

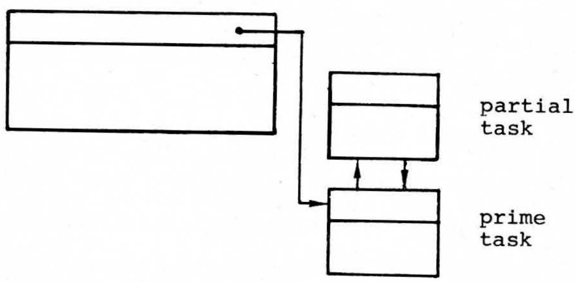
The DELTA notion of TIME represents the view of one observer of the system in question. Petri Net Theory has a different time concept and the next version of DELTA will also be related to Petri Net concepts, see (Jensen, Kyng and Madsen, 1979).

An action may be specified in two different forms:

- by specifying the name of a pattern defining a partial task (procedure).
- by specifying a basic action defined in the language.

A partial action specified by the name of a task pattern, is executed by generating a partial task according to the pattern. The specification of the partial task may include values which should be assigned to attributes of the generated task. The partial task is connected to the prime task and the action sequence of the partial task is executed. When the actions of the partial task are finished, it is discarded.

The action sequence of the partial task may again be specified by partial tasks. The execution of tasks may thus be represented by a stack of tasks which are under execution, for each of the objects in the system.



Of the basic actions defined within the language, we will here consider only the basic time consuming action. This action is specified by:

- The duration of the action, by stating a condition which has to be fullfilled during the action.
- A set of property descriptors defining a set of relations which have to be fullfilled during the action. The descriptors may involve the variable TIME.
- A set of operated variables which may be assigned values in order to fullfill the property descriptors.

In the syntax of the DELTA language such a timeconsuming action is described by an imperative of the form:

WHILE condition LET descriptors DEFINE variables;

The underlined words are reserved words in the DELTA language.

All acting objects within a system may concurrently execute time consuming actions. The total set of descriptors of the actions being executed at a specific moment of time is called the set of effective descriptors. The total set of variables that may be modified by these actions is called the set of operated variables. The system generator of the DELTA model has (at any moment of time) to find a set of values for the operated variables that satisfies the set of effective descriptors. If such a set of values does not exist the specified model is self-contradictory. If many such sets of values exist, one of the sets is chosen indeterministically.

Example: A FURNACE heating metal may be described as executing the action:

```
WHILE TEMPERATURE < MELTING_POINT
LET {TEMPERATURE = START_TEMP + F(RESOURCE.ENERGY,TIME)}
DEFINE TEMPERATURE;
```

The RESOURCE supplying energy may simultaneously execute an action which determines the energy supplied:

```
WHILE FURNACE.TEMPERATURE < MELTING_POINT
LET {ENERGY = G(FURNACE.TEMPERATURE,TIME)}
DEFINE ENERGY;
```

The joint evaluation of the two property descriptors will determine the temperature and the energy supplied to the furnace as long as the temperature has not reached the melting point.

At the first moment of time when the condition for an action is no longer fullfilled, the time consuming action is ended. At this moment there will be an instantaneous transition which terminates the action and proceeds to the next time consuming action specified in the action sequence of the object. The transition is called an event.

An event may be specified by an algorithm taking the system from the state when the preceding time consuming action is ended to the state when the next time consuming action is started.

All events in the total DELTA model (consisting of many objects) are assumed to take place in sequence. Two or more independent events taking place at the same moment of time will be executed in a sequence which is unpredictable. As a part of an event an object may send an interrupt to another object. An interrupt is a request to execute a specific task. The sender of an interrupt generates a task according to a task pattern (procedure), gives values to attributes of the task and appends the task to a list called the agenda of the object receiving the interrupt.

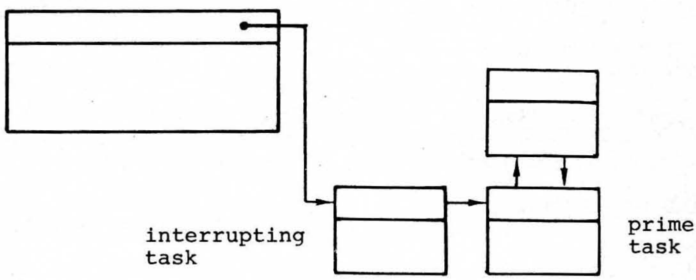
The receiver will compare the power priority of the interrupting task with the resistance priority of the action it is currently executing. If the interrupt penetrates, the execution of the current action will be postponed and the object will start executing the interrupting task.

As soon as the interrupting task is executed, the object will return to the execution of the action that was postponed.

An object being interrupted may execute an EXIT-action before it starts executing the interrupt requested, and it may execute a REENTRY-action before it resumes the interrupted time consuming action.

If an interrupt does not penetrate, the interrupting task will remain on the agenda, until the object starts executing an action that is penetrated by the interrupt.

The definition of priorities and of postponement and restart of actions is discussed in detail in (Holbæk-Hanssen, Håndlykken and Nygaard, 1975).



DELTA System Descriptions.

A DELTA system may be described by a text with a specific formal structure: a DELTA description. A DELTA description may be used by a system generator to generate a DELTA model of the system described.

A formal DELTA description is structured as nested text blocks. The system is described by a text block starting with the words SYSTEM BEGIN and ending with END SYSTEM. Within this block the attributes and actions of the system object are specified. Among the attributes are patterns defining the objects contained in the system object. The objects in the system are described in a similar way to the system object. A system description will thus have the structure indicated by the example below:

```
SYSTEM BEGIN

     CLASS A    :   OBJECT BEGIN
                    descriptions of attributes
                    (data items and patterns)
                    and prime task
                    END OBJECT ;

     CLASS B    :   OBJECT BEGIN
                    .....
                    END OBJECT ;

     M      : OBJECT BEGIN                 NB! M is a singular
                .....                          object not belonging
              END OBJECT ;                     to a class.

     I,J,K : INTEGER ;
     X     : REF A ;

     PRIME TASK BEGIN
          attributes and
          action sequence

     END PRIME TASK ;

END SYSTEM ;
```

Conceptually the system description is a single sequence of text starting with SYSTEM BEGIN and ending with END SYSTEM. In practice this organisation of the description will seldom be used. The description will be organised in several layers of detail where the first layer gives an overview of the attributes of the system object. The layer will have references to detail layers where the attributes are given a more comprehensive description. The organisation of these layers may be chosen according to a number of criteria:

- following the hierarchy of enclosing/enclosed objects.
- following the class/subclass hierarchy.
- describing components and properties which are closely interrelated in the same part of the description.
- following the detailing of actions into partial actions.

The actual choice must be made according to what seems to best fullfill the purpose of the description. However, it should always be possible to reorganise the description into a single sequence by replacing each reference to a detail by the corresponding description.

5. SOME EXAMPLES OF USE OF THE LANGUAGE.

Since the DELTA language was defined, it has been used for a number of applications, mainly at universities and at the Norwegian Computing Center. The applications have aimed at exploring strengths and weaknesses of the language. No efforts have till now been made to encourage wide spread use of DELTA. The next version of DELTA will, according to present plans, be intended to be a practical tool supported by software.

The applications fall in three main categories:

- descriptions for analysis and design of worktasks within an organisation.
- communication within an interdisciplinary group of scientists, studying the same phenomena.
- descriptions of formal models to be implemented on a computer.

The language has been taught to students in informatics at the universities at Aarhus and Oslo. Reactions from the students have been valuable for the further development of the language.

Description of work tasks within an organisation.

Use of the DELTA language for these purposes is described in (Håndlykken, 1977), (Eriksen, 1977) and (Bjerg & Nielsen, 1977).

For such purposes the descriptions are to a large extent formed in natural language. The DELTA concepts are used in the descriptions in the following manner:

- identification and naming of components and of classes and subclasses of components within the system.
- identification and naming of properties of each component. The properties may be data characterising the component or patterns for work tasks carried out in the organisation.

It is important that identification and naming is if possible done in accordance with concepts already in use within the organisation. A consistent identification and naming of components and properties enriches the natural language and makes it possible to discuss the system in question more precisely.

The level of detail and precision used in such a description will vary considerably. Within an organisation there are a number of tasks, often important, which are not repetitive and which are planned and executed as need or idea arises. E.g. writing a memo to management on a specific problem. A detail description of such tasks in advance is not possible. It might be possible to identify the kind of task, its goals and some of the aids available for doing it, but often not even that is possible.

For other tasks which are repetitive, it might not be possible or desirable to describe a uniform way in which the task is or should be executed. The detailing in the description may be sufficient when it is assured that it is possible to execute the task satisfactorily.

There are also, however, repetitive work tasks where it is crucial that the task is executed in a specific manner. Very detailed descriptions of these tasks may be necessary. Tasks supplying data to an important database will often tend to be of this kind.

The descriptions of work tasks made during the development of an information system will not only contain information processing aspects of the work. For meaningful discussion and evaluation of proposals other aspects important to the work tasks and jobs must be included in the description. Such descriptions are used for communicating on an existing or proposed system, and in material like handbooks and manuals used in a system.

The layout of the descriptions will then be organised in named chapters or paragraphs, ordered according to the alternatives suggested for DELTA descriptions, supplemented by pictures, drawings and diagrams.

Example:

We may describe a system of OPERATORS working at TERMINALS in informal DELTA-language in the following manner:

<u>Common description for all OPERATORS</u>:

The PRIME TASK of an operator is to do WORKSESSIONS at a TERMINAL. Each WORKSESSION will be interleaved by a period of other kind of work.

An operator is currently working at a TERMINAL called his WORKSTATION.

A WORKSESSION consists of entering data on a TERMINAL from a pile of documents by the following sequence of actions:

- Choose one of the TERMINALS not currently in use as WORKSTATION.
- Make the WORKSTATION ready for use and go through a login procedure.
- Repeat these actions until the data from the documents are entered:
  - . type a transaction on the WORKSTATIONs KEYBOARD
  - . press the transmit-key on the WORKSTATIONs KEYBOARD
  - . wait for response message
  - . check the response message.
- Go through the logout procedure and leave the WORKSTATION.

<u>Common description for all TERMINALS</u>:

A terminal has a KEYBOARD and a DISPLAY unit. It is connected to a central computer which it may send messages to and receive messages from.

The OPERATOR currently using the TERMINAL is called its USER.

<u>Interdisciplinary communication</u>.

The language has been used for the purpose of providing a common framework for communication within an interdisciplinary group of scientists. (Møller-Pedersen, 1977). The group consisted of hydrologists, biologists, engineers and chemists cooperating on the analysis of a lake.

The lake was decomposed into singular components and classes of components and data items and dynamic properties were assigned to the components. This formed a basis for working out a common description, and for structuring the discussion between the groups of scientists.

The language gives a large degree of freedom with respect to the kinds of models that may be described, and the level of detail used in the model. This implies that the language itself to little extent forces the scientists to use a particular kind of model. However, it also means that the actual choice of model and the way the language is used in such a group is important. According to Møller-Pedersen the most important benefit in using the language was to define and describe a common model of the lake as a basis for communication between the members of the group. For detailed discussions within each group the concepts and notations of the relevant discipline were still used.

Description of systems implemented on a computer.

The language has been used in the system development process to describe the models to be implemented on a computer (Håndlykken, 1977). A database containing information on the organisation and personnel of a large Norwegian enterprise was described and implemented using DELTA as a description tool.

In this case a detailed formalisation of the model to be implemented was necessary. The database was considered as a DELTA-model and it was described in DELTA language to people who were to use it in their work.

The concepts of objects and classes and subclasses of objects were used. Nesting of objects has similarities to hierarchical databases, however, relations between objects in general were described by references. Incorporated in the description were specifications of the main events taking place in the database. A description of important aspects of the dynamics of the database was thus included in the database description.

The TIME-concept of DELTA models was used, and a variable defining the moment of time modelled by the database was introduced. Information relating to moments before that time was considered historic, information relating to a time beyond that time was considered future. As the value of TIME increases there are automatic updates in order to make the database portray the new moment of time.

The description of the database was used for discussing which properties should be included in the model and how the model could be used and supported by worktasks in the organisation.

The database was itself described as a component within the organisation in which it is used. The communication between the organisation and the database was described by the interrupt concept. A transaction to the computer system was described as an interrupt i.e. a request to execute a specific task with a specified set of attribute values.

The database was implemented using COBOL, supplemented by a transaction handling system. The conversion from the DELTA description to a COBOL description of the model implied a number of decisions: access methods, storage organisation, representation of data and detailed organisation of the programs had to be chosen. The conversion was therefore by no means automatic, however, few changes in the DELTA description was necessary to implement the system.

Simulation may be a useful tool for evaluating the design of a system. For complex real-time system it may be the main tool available. Since DELTA is based on SIMULA, which is a simulation language, a DELTA description is a good starting point for simulations. We have no experiences with this in DELTA. In SIMULA there are, however, examples that a real-time program in SIMULA may be executed in a simulated real-time environment without making changes to the program (Belsnes, Løvdal, 1977).

6. EXPERIENCES WITH DELTA.

The experiences from using the language so far are briefly summed up in the following paragraphs.

The concepts of the language are well suited for use in natural language and for defining concepts to be used in natural language descriptions. The integration of formal descriptions and informal description comes easily.

No new users knowledge of how to apply a formal language in systems development seems to be very important. Increased formalisation increases the precision and compactness of the description, but it also means that the description will concentrate on formalisable aspects of the system considered. Too much formalisation reduces the readability of the description, especially to readers not familiar with formal languages. Too little formalisation may give specifications which are ambiguous. The degree to which a description should be formalized will often be a matter of judgement.

The language is suited for describing formal models to be implemented on a computer. There are, however, a number of improvements to be made:

- The data types and control structures which are inherited from SIMULA/ALGOL should be improved.

- A notation for relating different views of one generated model system should be introduced.

- Concepts which in some form introduces modification of the attribute structure or introduction of new attributes during the generation of a DELTA system, ought to be included in the language.

- Some changes in the semantics of the language as proposed by (Jensen, Kyng and Madsen, 1979) should be included.

DELTA descriptions and the DELTA systems have so far been generated with little support from the computer. A standard representation on a data screen and a standard dialogue with a DELTA description and a DELTA model by means of a data screen should be defined. Use of a graphical data screen will be highly desirable for this purpose. Work carried out at the XEROX labs in Palo Alto gives ideas to how such a dialogue could be defined (Ellis and Nuff, 1979).

There are no tools specifically designed for realizing a DELTA model on a computer. The programming language, BETA, is, however, intended to become a programming language meeting this need. At the moment an existing programming language has to be used. Using a programming language which is not close to DELTA introduces a distance between the specification and the programs which is clearly a drawback.

7. THE BETA LANGUAGE AND FUTURE WORK.

BETA started as, and still is a research project with purpose of contributing to the development of concepts and tools in programming languages. BETA will, however, also be implemented. The language is intended for a wide range of applications, like e.g. the ADA language (the two languages are, however, very different in structure).

BETA is very much inspired by DELTA in its general system concepts, and will probably in turn influence many aspects of the next DELTA version.

Our goal is to bring DELTA and BETA very close to each other. Then DELTA could be used for the overall system development and BETA for the programming of the computer-implemented subsystems, using the DELTA descriptions of these subsystems as specifications.

Such a correspondence also will make it possible for DELTA and BETA to share tools supporting the system development - at this conference called software engineering environment.

The BETA programming language is being developed by Bent Bruun Kristensen, Ålborg, Denmark, Ole Lehrmann Madsen, Århus, Denmark and Birger Møller-Pedersen and Kristen Nygaard, NCC, Oslo, Norway. Dag Belsnes, NCC, recently joined the BETA team. The work till date is reported in a series of Working Notes, the most recent are (Kristensen et. al., 1979) and (Kristensen et. al., 1980).

Finally, it should be remarked that our work on the design and use of DELTA should be regarded as a component of the wider task of contributing to the understanding of and methods for development of information systems (Nygaard and Håndlykken, 1980).

REFERENCES.

Birtwistle, G.M. et al. 1973: "SIMULA BEGIN". Studentlitteratur, Lund and Auerbach Publ. Inc., Philadelphia, Pa.

Bjerg, L. and Nielsen, L.V. 1977: "EDB-systemer indenfor avisproduksjon. Beskrivelse af systemerne og vurdering af beskrivelsesmetoder ud fra de grafiske arbeideres synspunkt". ("Computer Systems in Newspapers. Description of Systems and Evaluation of Description Methods - from the Typographers "Point of View".) DAIMI, University of Aarhus, Aarhus, 1977.

Dahl, O.-J. and Nygaard, K. 1965: "SIMULA - a Language for Programming and Description of Discrete Event Systems". Norwegian Computing Center, Oslo, 1965.

Ellis, C.A. and Nutt, G.J. 1979: "Computer Science and Office Information Systems". XEROX, Palo Alto Research Center, June 1979.

Eriksen, A.M. 1977: "Problemer i systemudvikling, specielt vedrørende sprogformer i systembeskrivelser". ("Problems in System Development, Particularly Relating to Language Used in System Description".)
DAIMI, University of Aarhus, Aarhus, 1977.

Holbæk-Hanssen, E., Håndlykken, P. and Nygaard, K. 1975: "System Description and the DELTA Language".
Publ. no. 523, Norwegian Computing Center, Oslo, 1975.

Holbæk-Hanssen, E. 1978: "En kort innføring i DELTA-språket og DELTA-strukturerte systemer". ("A Brief Introduction to the DELTA Language and DELTA Structural Systems".)
Publ. no. 599, Norwegian Computing Center, Oslo, 1978.

Håndlykken, P. 1977: "Rapport fra et systemutviklingsprosjekt der resultater fra DELTA-prosjektet er brukt i praksis". ("Report from a System Development Project in which Results from the DELTA Project is Used".)
Publ. no. 558, Norwegian Computing Center, Oslo, 1977.

Jensen, K., Kyng, M. and Madsen, O.L. 1979: "DELTA Semantics Defined by Petri Nets".
DAIMI, University of Aarhus, Aarhus, 1979.

Kristensen et al., 1979: "BETA Language Proposal as of August 1979".
Internal Report IR-11
DAIMI, Aarhus University, Aarhus, 1979.

Kristensen et al., 1980: "An Introduction to the BETA Programming Language (As of August 1979)".
Internal Report IR-20
DAIMI, Aarhus University, Aarhus, 1980.

Løvdal, E. and Belsnes, D. 1977: "Use of Simulation Techniques in Actual Network Implementation".
The EIN Symposium on Simulation and Modelling of Data Networks, Oslo, 1977.

Møller-Pedersen, B. 1977: "Communicating Concepts in an Interdisciplinary Project. Four Models of a Lake Described in the DELTA Language".
DAIMI, University of Aarhus, Aarhus, 1977.

Nygaard, K. and Bergo, O.T. 1974: "Planlegging, styring og databehandling. Grunnbok for fagbevegelsen. Del II". ("Planning, Control and Data Processing. Basic Reader for Trade Unions. Part II.")
Tiden Norsk Forlag, Oslo, 1974.

Nygaard, K. 1976: "DELTA-prosjektet og dets tilknytning til problemene i systemutvikling". ("The DELTA Project and its Relation to Problems in System Development".)
Publ. no. 539. Norwegian Computing Center, Oslo, 1976.

Nygaard, K. and Dahl, O.-J. 1978: "The Development of the SIMULA Languages." Paper presented at the ACM History of Programming Languages Conference, Los Angeles, 1-3 June 1978.

Nygaard, K. and Håndlykken, P. 1980: "The System Development Process". Paper presented at the GMD Symposium on Software Engineering Environments, Lahnstein, Germany, 16-20 June 1980.