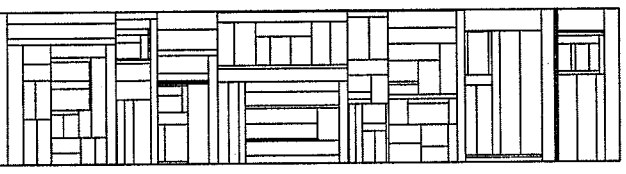
SYSTEM DESCRIPTION AS A STARTING-POINT FOR TEACHING SIMULA

Lars Mathiassen

DAIMI PB-66
January 1977

Institute of Mathematics University of Aarhus
DEPARTMENT OF COMPUTER SCIENCE
Ny Munkegade - 8000 Aarhus C - Denmark
Phone 06-12 83 55



Abstract

In the first part of the paper it is shown by an example how system description can be used as a starting-point for teaching SIMULA to beginners. The system description techniques used are based upon the philosophy developed in connection with the DELTA language. In the example used, the teaching process is divided into three phases. In each phase a different language tool is used. In the first phase we use a very informal system description tool. In the second phase we use a more formalized system description tool, while still allowing a certain amount of informal language. Finally in the third phase a totally formalized system description tool - in this case the programming language SIMULA - is used.

In the second part of the paper it is argued why system description - as shown in the example - should be used as a starting-point for teaching programming, and, having chosen this line of approach to the teaching process, why SIMULA is a good choice of programming language. Various requirements are imposed upon the planning and teaching of a programming course. These requirements are primarily derived from the connection between production, qualification and education. Secondly they are derived from the educational context of the programming course and from the knowledge and methods that have been developed in different areas of computer science. It is argued that these requirements can be met in the planning and teaching of a programming course, when using system description as a starting-point.

This paper was given as a talk at the fourth SIMULA-users conference, 8th - 10th of September 1976 at Leeuwenhorst Congres Center in Holland.

CONTENTS

1. INTRODUCTION

1.1 about the subject

The title of this talk is "System description as a starting-point for teaching SIMULA".

I am not going to discuss system description, the DELTA project (DELTA75) or SIMULA in any detail. Instead I am going to emphasize the problems in connection with the teaching of programming. I shall give a concrete proposal on how programming should be taught to coming edp-professionals. The main idea behind this proposal is, not to start the programming course by introducing a programming language, but instead by introducing less formalized language tools. In this way it becomes easier already in the beginning of the programming course to emphasize the teaching of reasonable working habits and attitudes. Using this approach, the student is not confused by all the peculiarities of a specific programming language.

In my proposal on how programming should be taught to coming edp-professionals I use system description as developed in the DELTA project and SIMULA as tools - thus the title of the talk.

1.2 background

Before I go on describing the proposal to you, I shall give you the background - or my main sources of inspiration - for making this proposal.

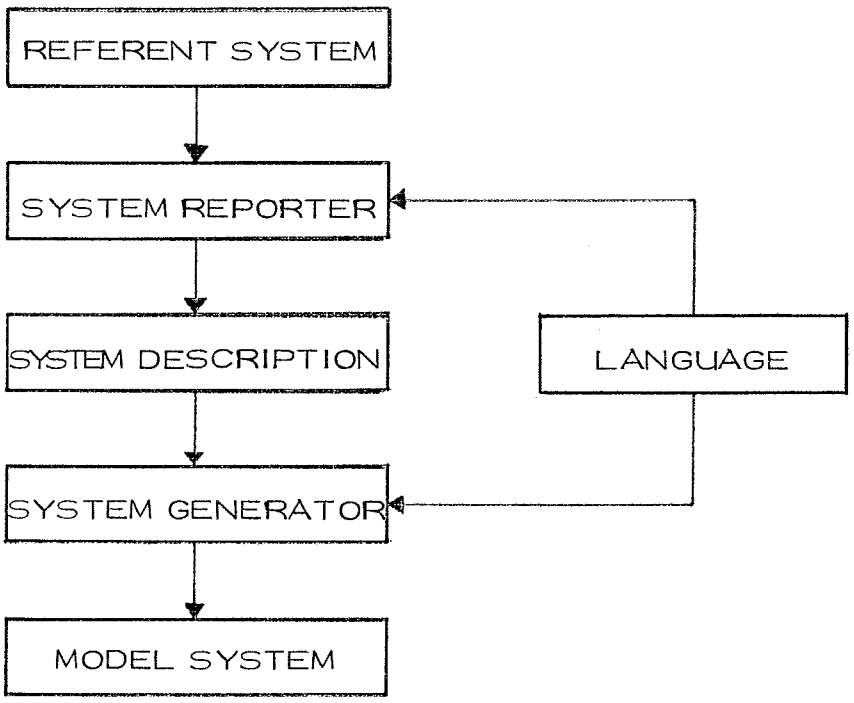
On one hand I have been inspired by some practical teaching experiences. In the introductory programming course at the Computer Science Department at the University of Aarhus in Denmark a very simple algorithmic language have been used as an introduction to programming (Mathiassen76a). This language forces the student to use certain control-structures (if - then -, if - then - else, while - do -) when describing algorithms. Except for this restriction the descriptions can be given in a natural language. Experience has proved that the students get more reasonable working habits and attitudes when using this approach as opposed to the more traditional approach in which a specific programming language is introduced already in the beginning of the programming course. The approach used has, however, one serious drawback. Using this very simple algorithmic

language the student gets accustomed to thinking of the structuring and description of algorithms as the one important aspect of programming. In a programming course the structuring and description of data should be stressed as well.

On the other hand I have been inspired by the DELTA project. First of all this project provides me with a "world picture": the concept of a system is defined in such a way that no part of the world is a system in itself. You can, however, choose to regard any part of the world as a system. Doing so you will find no inherent structure of the considered system, but you can choose to regard the system as consisting of a set of components each characterized by a data structure and an action pattern involving itself and other components.

Secondly the DELTA project provides me with a system description tool. And thirdly it provides me with a set of working habits and attitudes.

In the following I will use a simple communication model from the DELTA project.



A system reporter communicates information about a given referent system by constructing a system description of the referent system in a certain language. A system generator receives the system description and uses it to generate a model system.

To illustrate this: in a moment I (the system reporter) will describe to you a programming course (the referent system) in, I hope, understandable English (the language). You (the system generator) listen to my talk (the system description) and use it to generate your own model of the programming course (the model system).

2. ONE WAY OF TEACHING PROGRAMMING

In the following I shall describe one way of teaching programming. I am going to describe the educational context of the programming course and the programming course itself.

The following approach to the teaching of programming applies not only to the chosen educational context, but has a much wider range of application.

My reason for describing not only the programming course itself, but also the educational context of the course is twofold. Firstly I hope in this way to make the description more concrete. Secondly it is my belief that the problems in connection with the planning and teaching of a programming course cannot be analysed or discussed as an isolated phenomenon, but should be seen in relation to the educational and occupational context of the programming course.

I must stress that the following is not a description of a programming course as it is taught. It is a description of the programming course as it should in my opinion be taught in the educational context, with which we will be concerned here.

2.1 description of the educational context

We are going to look at the training of edp-assistants in Denmark. When the students begin this training theyhave passed high-school or equivalent exams, and after having finished the education they have qualified to a job as a programmer or an operator.

The training takes one year of full-time study.

These are the main objectives:

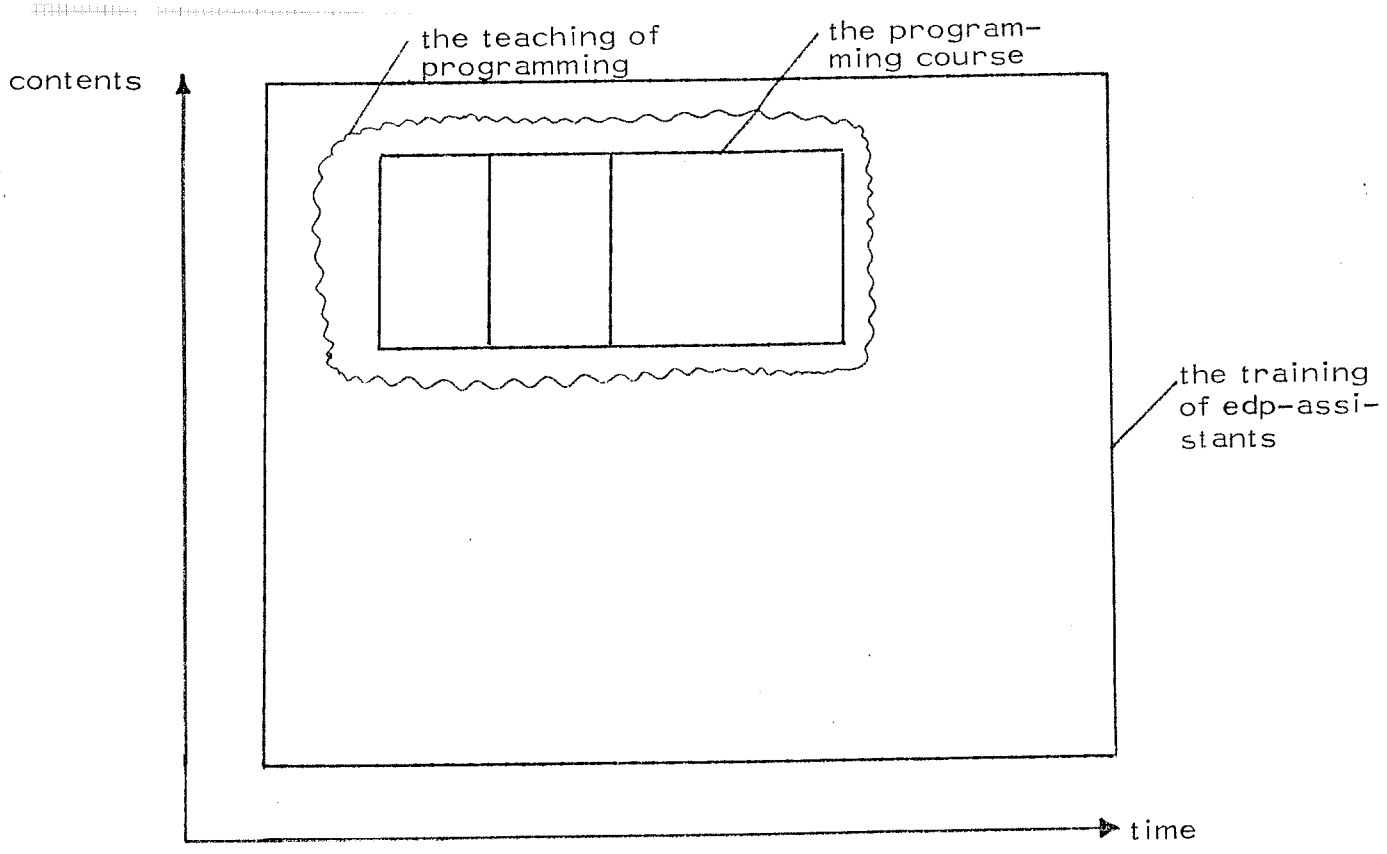
- the edp-assistant must be able to take part in the different phases of an edp-project.
- the edp-assistant must have a broad knowledge making it possible to continue her education.

As it is now, the training is organized in different subjects: machine architecture, operating systems, programming language 1, programming language 2, programming methodology, computers and society etc. It is my belief that, instead, the training should be orientated towards existing and actual problems, and that the training should be controlled by its active participants. Therefore, I shall here assume that the training as a whole is orientated towards one common problem or subject selected by its active participants: edp as a tool for rationalisation. Approximately half of the time, the students work on projects in relation to this subject. The other half of the time the students attend courses, some of which are planned beforehand and some of which are planned as the training goes on.

2.2 description of the programming course

The teaching of programming plays an important role in the training of edp-assistants. According to the description of the education the teaching of programming should occupy approximately 1/5 of the time spent. In this context the programming course is one of the courses planned beforehand.

The programming course and its relation to the rest of the education can be illustrated in the following way



The students perform different activities in parallel. At the same time as the students take part in the programming course, they attend other courses and they work on projects in relation to the selected common subject: edp as a tool for rationalisation. As will be shown in the following it is possible to use this common subject, and the experiences gained by the students working with this subject, as a starting point in the programming course - and in other courses. In this way the programming course becomes an integrated part of the education, and hence the teaching of programming will not be restricted to the programming course itself.

The programming course is divided into three phases. In each phase, the use of the computer system at hand plays an important role. Furthermore, different language tools are used, as follows. In the first phase we use a very informal system description tool. In the second phase we use a more formalized system description tool, while still allowing a certain amount of informal language. Finally, in the third phase a totally formalized system description tool - in this case the programming language SIMULA - is used.

In the following I shall describe each phase of the programming course in more detail. I am going to focus on how different language tools are used. For each phase I am going to show examples of system descriptions as they could have been made by students participating in the course.

In all of the examples the referent system is part of a hospital in Aarhus. In this hospital a big edp-based information system is in use. You may think of the information system as one of the systems investigated by the students as an example of the use of "edp as a tool for rationalisation".

The examples of system descriptions have actually been constructed by me in cooperation with one of our graduate students. The descriptions are based on visits to the hospital and on interviews with different staff members of the hospital.

2.2.1 phase one

In the first phase we use a very informal system description tool. This tool forces the student to describe a system as a collection of components,

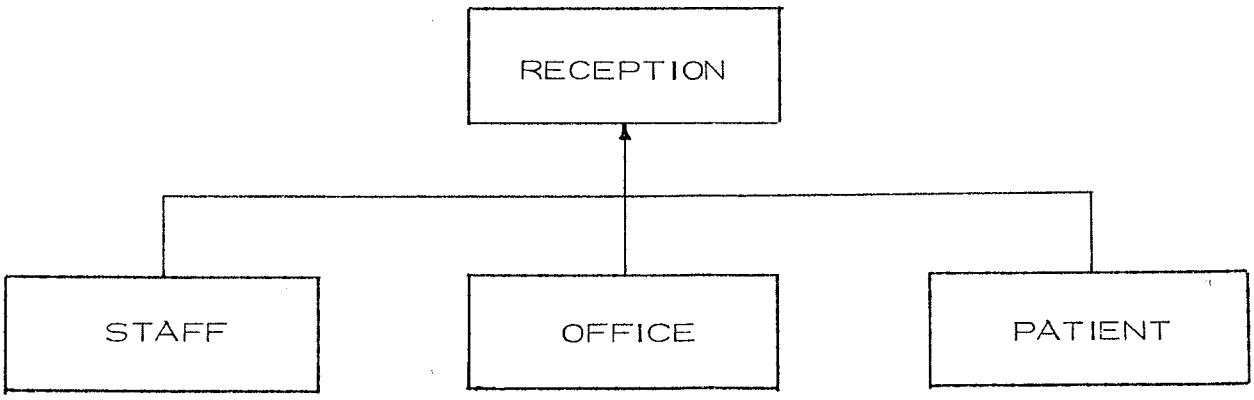
each characterized by a data structure and an action pattern. Except for this formalization of the structure of a system description, the descriptions can be given in a natural language.

In the following example the system reporter is a student attending the programming course. The student has been asked to describe the reception (the referent system) of the hospital. The description should be based on visits to the hospital. The student is asked to make a description that can be read by the other students of the course (the system generator) with the purpose of giving them an impression of the reception as a place of work.

This is a first not very detailed description:

RECEPTION: SYSTEM
BEGIN this system consists of a staff, an office and possibly one or more patients. The patients arive at the office. Among other jobs the staff admits patients to the hospital.
END RECEPTION SYSTEM

To give a more detailed description, the student chooses to decompose the system in the following manner:



Now representing each of the components, i.e. the staff, the office and the patients, as an object, a more formal description can be given:

```
RECEPTION: SYSTEM
    BEGIN
        STAFF: OBJECT;
        OFFICE: OBJECT;
        PATIENT: OBJECT;
    END RECEPTION SYSTEM
```

Describing the data structure and the action pattern of each component it is now possible to give a more detailed description of the reception:

```
RECEPTION: SYSTEM
BEGIN   this system consists of
    a STAFF: OBJECT
        BEGIN
            DATASTRUCTURE  this object consists of 12 secreta-
                ries working in three shifts. All of the secretaries
                are women, 4 of them are engaged on full-time, the
                rest on part-time basis. They feel that the working
                conditions are bad: there is too little room in the
                OFFICE and there is too much noise from the diffe-
                rent technical installations. They have only had one
                hour of on-the-job training on how to use the new
                terminal in the OFFICE;

            ACTIONPATTERN  the secretaties' job is to do all the
                paperwork in connection with the arrival of new
                PATIENTs. In the early morning the secretaries
                admit new PATIENTs to the hospital. During the
                day they also do a lot of typewriting in connection
                with the keeping of the case records, and they re-
                cieve PATIENTs for the emergency room. They
                also answer  a lot of inquiries - via telephone,
                via pneumatic dispatch and from different persons
                coming  to the OFFICE.
        END STAFF OBJECT;
```

an OFFICE: OBJECT
BEGIN
DATASTRUCTURE the floor is 3 x 5 meters in area.
There is a permanent opening to the corridor. Normally there is much noise. This object contains
3 typewriters,
3 dictaphones,
1 terminal,
1 lineprinter,
1 stamping machine,
1 telephone,
1 pneumatic dispatch tupe conveyor,
and some other office furniture.
END OFFICE OBJECT;

and possibly one or more
PATIENT: OBJECTs
BEGIN
DATASTRUCTURE A reference from a doctor or an acute disease.

ACTIONPATTERN this object applies to the STAFF and answers the questions put by the STAFF. Afterwards she goes to a waiting room or to one of the other sections of the hospital.
END PATIENT OBJECT;
END RECEPTION SYSTEM

As you probably have already noticed, these descriptions are not written in the imperative mood . The student writes "... In the early morning the secretaries admit new ..." instead of writing "Admit new ...". It is my belief that this is the natural way for a non-computer-specialist to describe systems. In this first phase the student only has to describe the structure of the referent system as a collection of components and to seperate the description of a single component in a description of the data structure and the action pattern of the component.

In this first phase the students use the computer system at hand. They use the program library but they do not write programs themselves. One of the programs they use is a toy information system (Studynotes76):

The students can go to one of the terminals of the computer system at hand and play the role of a secretary. The secretaries, working in the reception of the hospital, use a terminal to communicate with the real information system. This information system contains among other things a record of information for each patient in the hospital.

Sitting at the terminal the students can do various transactions such as:

```
REGISTERPATIENT
FINDPATIENT: personal-id-of-patient
CORRECTPATIENTINFORMATION
      .
      .
      .
END
```

In this way the students learn to handle the terminal and at the same time they get a better understanding of part of a secretary's job. (We will return to this toy information system in phase three).

2.2.2 phase two

In the second phase we use a more formalized system description tool, while still allowing a certain amount of informal language. This new system description tool can be regarded as an extension of the tool used in the first phase. This second phase includes the addition (to the "phase one" tool) of facilities for datastructuring and for describing interaction between components, as well as procedures and some control structures.

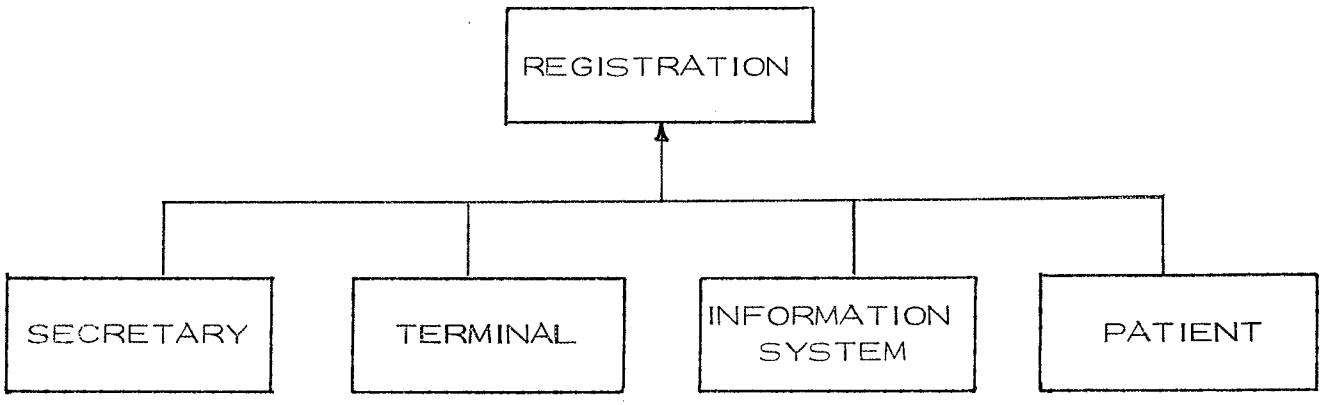
In the following example of the use of this more formalized system description tool, the system reporter is again a student attending the course. This student has been asked to describe a registration of a patient (the referent system).

The description should be based on visits to the hospital and on interviews with secretaries. The student is asked to make a description that may later be used as the basis for the construction of teaching material to new secretaries explaining them how the terminal and the information system is used when registrating new patients. So, the system generator is the person (maybe the student herself) who is going to construct the teaching material.

This is a first not very detailed description:

REGISTRATION: SYSTEM
BEGIN A secretary interviews a patient and transmits the obtained information to the informationsystem via a terminal.
END REGISTRATION SYSTEM

To give a more detailed description, the student chooses to decompose the system in the following manner:



Now, representing each of the components as an object, a more formal description can be given:

```
REGISTRATION: SYSTEM
    BEGIN
        SECRETARY: OBJECT;
        PATIENT: OBJECT;
        TERMINAL: OBJECT;
        INFORMATIONSYSTEM: OBJECT;
    END REGISTRATION SYSTEM
```

The registration may then be described in greater detail by describing the data structure and the action pattern of each component.

The formalizations used should be easy to understand except perhaps the colon-semicolon-parenthesis. Using the colon-semicolon-notation it is possible, in the same description, to give descriptions of the same aspect of the referent system at different levels of detail. When reading the system description the colon of the colon-semicolon-parenthesis should be read as an "i.e.".

```
REGISTRATION: SYSTEM
BEGIN    this system consists of
    a TERMINAL: OBJECT
        BEGIN
            DATASTRUCTURE  this object consists of
                a KEYBOARD and a SCREEN:

                    KEYBOARD:
                        CHARACTERBUTTONs
                        SENDMESSAGEBUTTON
                        CURSORMANIPULATIONBUTTONs;

                    SCREEN:
                        CURSOR: denotes the position, where
                                next character will be written;
                        PICTURE: 24 lines of 80 characters;;;

                this object can execute the tasks described by

                    RECIEVE PICTURE FROM INFORMATIONSYSTEM:
                    TASK BEGIN
                        when information is sent from the INFORMA-
                        TIONSYSTEM show this information on the
                        PICTURE of the SCREEN.
                    END TASK;

                    SEND PICTURE TO INFORMATIONSYSTEM:
                    TASK BEGIN
                        when the SENDMESSAGEBUTTON of the
                        KEYBOARD is activated send the infor-
                        mation on the PICTURE of the SCREEN
                        to the INFORMATIONSYSTEM.
                    END TASK;

                    WRITE CHARACTER ON SCREEN:
                    TASK BEGIN
                        when a CHARACTERBUTTON of the KEY-
                        BOARD is activated show the character de-
                        noted on the button at the CURSOR-position
                        on the SCREEN. Move the CURSOR one po-
                        sition forward.
                    END TASK;
```

```
                    MOVE CURSOR:
                    TASK BEGIN
                        when a CURSORMANIPULATIONBUTTON
                        is activated move the CURSOR of the
                        SCREEN as denoted on the button.
                    END TASK;

            ACTIONPATTERN
                WHILE power on REPEAT reacting on impulses:
                    (* IF CHARACTERBUTTON is activated
                       THEN WRITE CHARACTER ON SCREEN
                       IF SENDMESSAGEBUTTON is activated
                       THEN (* SEND PICTURE TO INFORMATIONSYSTEM
                               WHILE no answers WAIT
                            *)
                       IF CURSORMANIPULATIONBUTTON is activated
                       THEN MOVE CURSOR
                    *)
        END TERMINAL OBJECT;


an INFORMATIONSYSTEM: OBJECT
    BEGIN
        DATASTRUCTURE  this object contains information about
            the PATIENTs. It is connected to a number of TER-
            MINALs that are situated at different places in the hos-
            pital. In periods it is down. In these periods it is im-
            possible for the TERMINALs to communicate with this
            object.

        ACTIONPATTERN
            REPEAT reacting on impulses from the TERMINALs:
            (* WHILE no impulses from the TERMINALs WAIT
               receive picture
               do proper operations
               prepare answer to TERMINAL
               ask TERMINAL to RECEIVE PICTURE
                 FROM INFORMATIONSYSTEM
            *);
    END INFORMATIONSYSTEM OBJECT;
```

```
a SECRETARY: OBJECT
    BEGIN
        DATASTRUCTURE  Is sitting on the chair at the TERMINAL,
                facing the PATIENT.

                This object can execute the tasks described by

                        TYPE ANSWER:
                        TASK BEGIN
                                type individual characters of the answer
                                by activating appropriate CHARACTER-
                                BUTTONs of the TERMINAL. Activate the
                                CHARACTERBUTTON of the TERMINAL
                                corresponding to carriage-return.
                        END TASK;

                        REGISTRATE PATIENT:
                        TASK BEGIN
                                ask PATIENT about  personal-id.
                                TYPE ANSWER
                                WHILE INFORMATIONSYSTEM
                                        hasn't answered with standard
                                        information WAIT
                                WHILE more unanswered questions on
                                        the SCREEN of the TERMINAL REPEAT
                                    (* ask PATIENT next question
                                         TYPE ANSWER
                                    *)
                                activate SENDMESSAGEBUTTON of the
                                        TERMINAL
                                explain PATIENT where to go
                        END TASK;
```

```
                    CARRY OUT OTHER FUNCTIONS:
                    TASK BEGIN
                        carry out other functions than registration
                    END TASK;

            ACTIONPATTERN
                WHILE it is still working hours REPEAT
                    (* WHILE no PATIENTs to registrate
                        CARRY OUT OTHER FUNCTIONS
                       REGISTRATE PATIENT
                    *)
        END SECRETARY OBJECT;
```

and

```
a PATIENT: OBJECT
    BEGIN
        DATASTRUCTURE  Is standing in the doorway of the
            reception, facing the SECRETARY.

        ACTIONPATTERN
                WHILE more questions from SECRETARY REPEAT
                  try to answer question
                leave the reception and go to the destination
                  explained by the SECRETARY.

    END PATIENT OBJECT;
END REGISTRATION SYSTEM
```

In this second phase the students use the computer system at hand. As in the first phase they use the program library but they do not program on their own.

2.2.3 phase three

In the third phase a totally formalized system description tool - in this case the programming language SIMULA - is used. At this point of the teaching process we benefit from the knowledge and experience already gained by the student. At this point the student should have reasonable working habits and attitudes, she should know how to handle formal language tools, and she should be able to handle a certain part of the computer system at hand.

When introducing SIMULA we start by introducing those concepts which - at least in this context - are the easiest to understand, and which in any case from a programmer's point of view are the most important ones: the class concept, the concept of a procedure and the control structures of the language. Expressions and the assignment statement, elements of the programming language that are difficult for the student to understand, are introduced later.

It is not necessary to introduce all the facilities of SIMULA before asking the students to write their first program. Using the toy information system from the first phase as a programming language the students can write simple programs solving realistic problems.

In the following example (of a system description using SIMULA as a system description tool) a student has solved the problem of constructing a list of all the personal identifications from each of the patient records stored in the information system.

```
begin
    external class informationsystem;
    informationsystem (informationfile)
    begin
        reset;
        while more patients
          do begin
              next patient;
              outint (thispersonalid, 11);
            end;
    end;
end program;
```

'reset', 'more patients' etc. are defined in the class 'informationsystem', and can - from the student's point of view - be regarded as parts of the programming language used. 'informationfile', containing the patient records, can be thought of as a file, constructed by the student in the first phase of the programming course when using the toy information system as an interactive system.

The program shown is very simple. To illustrate that it is possible to solve rather complicated problems using this toy information system as a programming language, consider the following problem: Construct a list of all diagnoses occurring in the different patient records. The diagnoses should be listed alphabetically and each diagnoses should be followed by a list of all the personal identifications from each of the patient records in which the diagnosis occurred.(This problem is also known as the cross-reference problem.)

In this third phase, of course, the use of the computer system at hand plays an important role. The system descriptions constructed can - as opposed to the descriptions from the two earlier phases - be understood and executed by the computer.

3. WHY DO IT THIS WAY

In the following I shall argue why system description - as shown in the example - should be used as the starting-point for teaching programming, and, having chosen this line of approach to the teaching process, why SIMULA is a good choice of programming language.

Programming teaching cannot be discussed as an isolated phenomenon. Various requirements are imposed upon the process of planning and teaching a programming course. These requirements are primarily derived from the relationship between the qualifications that are needed in different jobs in the production-process and the qualifications that are given the students in the educational system. Secondly these requirements are derived from the educational context of the programming course, and from the knowledge and methods that have been developed in various fields of computer science.

Before arguing why programming should be taught as proposed, I shall list the main requirements that are imposed upon the process of planning and teaching a programming course. (A more detailed discussion can be found in Mathiassen76b).

3.1 requirements

Firstly we have some general requirements to the educational system as a whole:

- the education, of which the teaching of programming is a part, should be orientated towards existing and actual problems (as opposed to traditional education).
- the teaching process should be controlled by its active participants.

These general requirements are derived from certain didactical considerations. The basic question that is asked is: What qualifications should the education give its students?

Corresponding to different definitions on what didactics is all about you get different tools available to help you answer this basic question.

The tool I have chosen takes its starting-point on one hand in the development of the production process and on the other hand in some psychological-theoretical aspects. More precisely it analyses both the development of the forces of production, i.e. resources, machines, tools and human labour, and how different ways of teaching supports different needs on what qualifications the students should gain.

This didactical tool supplies you with some general didactical principles (the above mentioned requirements) on how to arrange the internal structure of the education as a whole. I shall not go further into a discussion of these didactical aspects, just mention that the two new university centres in Denmark - Aalborg University Center (AUC) and Roskilde University Center (RUC) - have been organized according to these general didactical principles.

Secondly we have some specific requirements to the teaching of programming:

- the context in which the programming tool is going to be used later should be the starting point for the teaching of programming.
- the teaching of programming should reflect that it is important to give the students reasonable working habits and attitudes. At the same time it is necessary that the student learns to master one or more totally formalized language tools.

The first of these requirements is derived from certain considerations on the educational context of the programming course. A programmer, a teacher and a chemist have different ways to use computers. The way you are going to teach programming should therefore reflect whether you are teaching coming edp-professionals, coming edp-teachers or coming edp-users.

The second of these requirements is derived from certain technological considerations. When planning a programming course you should take the results and methods from various fields of computer science into consideration. I believe, that the most important areas to be considered are: programming languages, programming methodology, program correctness and for-

mal semantics, systems work (dealing with the problems in connection with the use of edp) and finally what I should like to call: the computer as a tool.

In any programming course you must of course emphasize the teaching of technical **programming skills:** the structuring of data and algorithms, the representation of these in a formalized language, the running of a program on the computer etc. What is essential in relation to the teaching of these technical programming skills? Is it the teaching of a formalized language, is it the function of the computer or is it ways of thinking? Looking at the qualifications that are needed in the process of using edp as a tool, we get the following answer: it is important to teach reasonable working habits and attitudes, at the same time it is necessary to teach one or more formalized language tools.

In this last requirement you should be aware of one important contrast. I believe practical experiences has proved that you cannot start the programming course by introducing a formalized language at the same time hoping to give the students reasonable working habits and attitudes. When you start the programming course by introducing a programming language it will nearly automatically be the peculiarities and details of this language, the running of small programs written in this language and the technical problems caused hereby that will draw the attention of the students. And even if you emphasize the teaching of reasonable working habits and attitudes it will be difficult to motivate the students for them using the above-mentioned approach.

3.2 argumentation

In my proposal of how to teach programming a very simple system description tool is introduced in the beginning of the programming course. Using this approach it is possible to use problems and subjects from the training as a whole and the expectations of the participants as a starting-point when arranging the programming course. Exercises and examples can be constructed as the teaching process goes on and they can easily be related to the contents of the rest of the educational process. The point is that everything you want to consider as a system can be considered as such.

In the traditional approach to the teaching of programming algorithms and data are the fundamental concepts. However the concepts of algorithms and data are much more narrow than the concept of a system - you cannot choose to regard anything as an algorithm. Therefore, these concepts do not provide you with the same possibilities of using the problems and subjects from the training as a whole and the expectations of the participants as a starting-point. On the other hand, when using system description as a starting-point, it becomes possible to do so, and at the same time the concepts of algorithms and data can be introduced in a natural way: The data concept is closely related to the data structure of a component, and the concept of an algorithm is closely related to the action pattern of a component.

In the examples of system descriptions it is shown how referent systems can be chosen in close relation to the common subject of the education given. The referent systems - the reception and the registration - are both very closely related to the problems in connection with the use of edp in hospitals, which again, of course, is one aspect of the use of edp as a tool for rationalisation.

It is hence illustrated how the proposed approach to the teaching of programming supports the two general requirements to the educational process as a whole.

Now, the individual requirements to the teaching of programming should not be discussed isolated. These requirements are very closely related to one another, and it is precisely the combination of the requirements that should be the basis for the planning and teaching of a programming course.

Using the didactical principles of problem orientation and participant controllation when organizing the internal structure of the education it becomes easier to make the context in which the programming tools are going to be used later a starting-point of the programming course. In the examples of system descriptions it is shown how some aspects of the problems in connection with the use of edp becomes an integrated part of the teaching of programming.

It is hence illustrated how the context in which the edp-assistant is going to use the programming tools can be used as a starting-point in the programming course.

Concerning the last requirement to the teaching of programming you have probably noticed that I have not at all discussed what I understand by reasonable working habits and attitudes. Reasonable working habits and attitudes can for short be denoted as structured programming. In more detail they can be described by using different sets of concepts or methods: abstract vs concrete, buttom-up vs top-down, synthesis vs analysis and functional vs operational. Reasonable working habits and attitudes deal with the ability to handle the complex - the ability to make the complex simple by the use of structuring and systematics.

Reasonable working habits and attitudes can not be taught by showing the students "how to do", or by telling them what it is all about using these different concepts and methods. Reasonable working habits and attitudes can only be taught by making the students try. However in these experiments you can support the student by carefully selecting appropriate tools.

Programming is just a special case of system description. The working habits and attitudes needed in the process of describing systems are hence similar to those needed when programming.

The description tool used in the first phase of the proposal has been chosen with the purpose of supporting the teaching of reasonable working habits and attitudes. At the same time this tool, I believe, is so simple that teaching it should cause very few problems. In the next two phases more formalized system description tools are introduced. These tools support the student in the continued development of reasonable working habits and attitudes. In this way we also gradually stress the necessity of having totally formalized language tools.

3.3 further advantages

Using system description as a starting-point in the teaching of programming some further advantages are achieved.

Firstly it is possible in the training of coming edp-professionals to use the system description tool to describe the computer system. By constructing descriptions of the computer system - at different levels of abstraction and from different viewpoints - the student will gain a better understanding of the computer and its function.

Secondly, by using this approach in the training of coming edp-professionals, you have given the edp-professional a possibility of understanding her job in a wider context, and consequently given her better possibilities of influencing her own working conditions. By using the context in which the programming tool is going to be used later as a starting-point, and by representing programming, not as an isolated process, but as an integrated part of the whole process of using edp, you have among other things given the edp-professional a better background for counteracting the growing specialization in the edp-production.

Thirdly, it is possible to give the student a better understanding of the possible uses of programming tools. Using system description as a starting-point it is possible to discuss how different choices of language tool affect the range of referent systems that can reasonably be described.

3.4 the choice of SIMULA

Finally, I want to comment on the choice of SIMULA as the programming language taught. I will not discuss the different virtues of SIMULA in detail. Instead I shall give you the two main arguments for choosing SIMULA, when system description,as proposed, is used as the starting-point for teaching programming.

The first argument is that DELTA descriptions to be executed on computers lend themselves particularly well to SIMULA. The two languages are based upon the same philosophy, and they have got nearly the same syntax.

The second argument is that it is extremely easy for the teacher of the programming course to define special purpose languages. Using the class-concept it is possible to create contexts in which the students can solve realistic, but not necessarily difficult problems.

In some implementations of the SIMULA language it is even possible to have these classes precompiled.

4. CLOSING REMARK

It is my hope that I have given you some inspiration on how programming should be taught. As I said already in the beginning of the talk this proposal of how to teach programming has not been thoroughly tested. It is my belief that the only real criterion from which to judge a proposal like this is practical experience.

5. REFERENCES

DELTA75 System description and the DELTA language, DELTA report no. 4, E. Holbæk-Hansen, P. Håndlykken og K. Nygaard, Norsk Regnesentral 1975.

Mathiassen76a En datalære bog, L. Mathiassen, Gyldendal 1976 (in Danish).

Mathiassen76b Programming teaching and system description, L. Mathiassen, DAIMI PB-60, Århus Universitet 1976 (in Danish).

Studynotes76 A simple patient information system, studynotes, Department of Computer Science, Århus Universitet, Dat. 1- nr. 5-1976.
(in Danish. This material contains a description of a toy information system, which has been used in the introductory programming course at the department).