More or less following a plan during design: opportunistic deviations in specification

Willemien Visser

MORE OR LESS FOLLOWING A PLAN DURING DESIGN: OPPORTUNISTIC DEVIATIONS IN SPECIFICATION*

Willemien VISSER

Institut National de Recherche en Informatique et en Automatique
Rocquencourt, B.P. 105, 78153 Le Chesnay Cedex, France

July 1989 (third version)

Abstract. An observational study was conducted on a mechanical engineer throughout his task of defining the functional specifications for the computerized control part of an automated machine tool installation. The engineer described his activity as following a hierarchically structured plan. The actual activity is in fact opportunistically organized. The engineer follows his plan as long as it is cognitively cost-effective. As soon as other actions are more interesting, he abandons his plan to proceed to these actions. This paper analyzes when and how these alternative-to-the-plan actions come up. Quantitative results are presented with regard to the degree of plan deviation, the design components and the definitional aspects which are most concerned by these deviations, and the deviation patterns. Qualitative results concern their nature. An explanatory framework for plan deviation is proposed in the context of a blackboard model. Plan deviation is supposed to occur if the control, according to certain selection criteria, selects an alternative-to-the-planned-action proposal rather than the planned-action proposal. Implications of these results for assistance tools are discussed briefly.

Keywords. Design activity, Specification, Planning, Opportunistic organization of activity, Control, Real-time observational study, Field study, Protocol analysis, Automatic machine tool installation.

* * * * *

Il existe une version française raccourcie de cet article:
Visser, W. Comment le concepteur organise réellement son activité. In K. Zreik & B. Trousse (Eds.), Organisation de la conception. Paris: EUROPIA, 1994.

Résumé. On montre que la façon dont un concepteur décrit son activité (un plan structuré hiérarchiquement) ne correspond pas à l'organisation de l'activité réelle, qui est "opportuniste". Le concepteur adopte, parmi les actions qui se proposent à lui, celle qui "coûte" le moins en termes cognitifs. Ainsi il suit son plan tant que celui-ci est "intéressant" de ce point de vue, mais dès que se présentent des "opportunités" d'actions plus intéressantes, le concepteur dévie de son plan pour procéder à ces actions. On présente les cinq types de données dont la prise en considération et le traitement conduisent à la proposition d'actions "intéressantes". Il s'agit, d'une part, de connaissances qu'a le concepteur, d'autre part, de représentations mentales qu'il construit à partir de ces connaissances et d'informations qui lui parviennent de l'extérieur: l'état du projet, les remarques et suggestions de collègues, les ajouts et modifications aux spécifications de départ fournies par le client.

Mots-clés. Activité de conception, Spécification fonctionnelle, Planification de l'activité, Organisation de l'activité, Organisation opportuniste, Modèle "blackboard" (tableau noir), Contrôle, Observation de l'activité, Observation en temps réel, Étude de terrain, Analyse de protocoles, Machine-outil automatisée.

---

* The author wishes to thank André Bisseret, Pierre Falzon and Françoise Détienne for helpful comments on earlier versions of this paper.

Summary

1. INTRODUCTION

This paper presents the most important results obtained in a study on the cognitive aspects of specification. The study was conducted on a mechanical engineer in charge of writing functional specifications for the computerized control part of an automated machine tool installation.

The installation has two parts: an operative part - which does the work of tooling connecting rods - and a control part, which consists of a programmable controller (that is, a computer specialized in controlling industrial processes). It is the control part which makes the installation an *automatic* machine tool installation.

The specifications produced by the engineer were to define the functioning of the operative part so that a software designer could design the program for the control part.

1.1.1 Relevance of the study

The presented study may be considered relevant for several reasons:

The study of specification: a new contribution to the study of design

The design *stage* examined (the specification stage) differs from those studied in other empirical design studies which focussed on later design stages.

> Some examples of these studies conducted in different domains are the following: *architectural design*: Eastman (1969), one of the first empirical design studies; *software design*: Adelson & Soloway (1985), Guindon, Krasner & Curtis (1987), Malhotra, Thomas, Carroll, & Miller (1980), Hoc (1988a), Jeffries, Turner, Polson & Atwood (1981) and Ratcliff & Siddiqi (1985); *mechanical design*: Ullman, Staufer & Dietterich (1987) and Whitefield (1986); *computational geometry algorithm design*: Kant (1985); *traffic signal setting*: Bisseret, Figeac-Létang & Falzon (1988); *text composition*: Hayes & Flower (1980).

With regard to the *cognitive activity* studied, however, the difference between our study and those mentioned above is relative. Design consists in transforming a problem representation into another: it always starts with "requirements" and produces "specifications."
Differences of two types characterize these two problem representations:
• domain: the representation domains of both representations differ;
• level: the first representation is at a more abstract level than the second (that is, it is less complete, less precise and less concrete than the second).
In software design, a specifications representation using concepts in the application domain is to be transformed into computer specifications represented in the software domain. In mechanical design, going from the client's requirements to the working drawings, three consecutive representation stages may be distinguished:
• representation of the goal to be attained by the artifact to be designed (or its function);
• representation of the mechanical operations which will allow to attain this goal (to realize this function);
• representation of the machines or tools which are going to materialize these operations.
For each couple of consecutive representations, the first defines the requirements for the second.

The designer in our study was observed in the "specification stage." He wrote specifications for another designer, in this case, a software designer. In other design studies, the observed designers are supposed to produce a "final design," that is, specifications not for other designers, but for the persons in charge of implementing the result of the design activity (in software design the programmer doing the coding, in mechanical design the workshop operators building the artifact designed).

> N.B. The engineer observed in our study wrote functional specifications for the control part of an installation, that is, he defined the functioning of the operative part of the installation. It was a mechanical engineer, not a software engineer. He was to write *functional* specifications, that is, he had to specify *what* the program should do, not

how it should be done. This last aspect was to be specified by a software designer. The non-familiarity of the observed mechanical engineer with the software domain may have at least two consequences for the specifications he writes, a positive one and a negative one:
• the specifications do not risk to contain already particular implementation decisions, but
• they may contain requirements impossible to implement.
These two points would need examination to settle.

The importance of the specification stage in design

There are at least two reasons why the study of design in the specification stage is important:
• It is a very early design stage, and the activity during this stage has consequences for all the design activity performed afterwards (see, for example, Letovsky, Pinto, Lampert, & Soloway, 1987, who show the importance of "Design Reconstruction" during code inspection).
• As the specification stage defines the starting point for the following stages, its outcome - and, therefore, the specification stage itself - is of crucial importance to the resulting design.
These points hold even if design does not consist of independent, consecutive stages, as empirical studies of design activity tend to show (contrary to the presentation of design in prescriptive studies) (see Visser, 1988a).

1.1.2 Originality of the study

Next to examining an important design stage not yet studied, the original aspects of this study are its methodology and the cognitive activity on which it focuses.

Study of a professional designer in his daily work environment, during his work on a real, industrial design project

Most empirical design studies conducted to date have been carried out in an artificially limited context, generally the psychological laboratory. Even if some studies used professional designers (novices and/or experts), these studies generally concerned design tasks which were simplified and somewhat limited compared to real design projects. Moreover, the subjects in these studies were working individually. In reality, design is characterized by people working together and to some extent depending on work done by colleagues (upstream and downstream) and on the information provided by these colleagues.
In order to model real design activity, however, empirical studies of this activity must also be conducted on designers working on real, and therefore complex, projects. Compared with laboratory studies, this type of study is, without a doubt, much more time-consuming, and the data obtained may be more difficult to process, due to the number of factors that cannot be eliminated during observation, as may be done in an experimentally controlled study.

Real-time, continuous observation during the entire specification process

The engineer was observed throughout the duration of his work (three weeks), until he had completed the writing of the specifications.

Context of the study: a longitudinal design study

The study on the engineer during specification was the first of three studies which together constituted a longitudinal study carried out on a project to design a computer-controlled machine tool installation
Various stages in this design process could have been observed. As the study presented here was carried out in the framework of research on programming, the stages observed were those that were considered the most relevant from this point of view. Besides the stage which concerned programming as such, the stages in the design process that

immediately preceded and followed it were also observed. The three studies were thus conducted on:

- the writing of the functional specifications for the control part of the installation (the study presented in this paper). This work was done by a mechanical engineer, and the functional schema that he produced constitutes the main document used to provide the specifications for designing the program to control the installation;
- the design and writing of this program by a specialist in electronics and automation (the "programmer"[1]) (see Visser, 1987, for a detailed presentation);
- the debugging and testing of this program by another specialist in electronics and automation (see Visser, 1988a, for some results).

Focus of the analysis developed in this paper: opportunistic plan-deviation actions

The study presented in this paper focuses on the way the engineer organizes his specification activity. The engineer describes his activity in the form of a hierarchically structured plan. He breaks his activity down into components of different levels to be dealt with in a specific order. This plan, however, does not represent the actual activity which is, in fact, opportunistically organized. The engineer follows his plan as long as it is interesting from a viewpoint of cognitive cost. As soon as other actions are more interesting in this respect, he abandons his plan to proceed to these actions. This paper analyzes when and how these alternative-to-the-plan actions arise.

The model adopted is a blackboard model, as has been proposed and used in other design studies (cf. Bisseret, Figeac-Létang & Falzon, 1988; Visser, 1988b; Whitefield, 1986). To formulate an explanatory framework for plan deviation, this study develops the control component of the model. Plan deviation is proposed to occur if the control selects an alternative-to-the-planned-action proposal rather than the planned-action proposal. Several processes leading to proposing plan-deviation actions are described. Two main criteria the control uses in selecting among these action proposals are introduced.

Organization of the paper. Section 2 describes the method used to collect and analyze data on the specification activity. Section 3 presents the main results, distinguishing between quantitative results as to the degree of plan deviation observed and qualitative results concerning the processes leading to plan-deviation actions. Section 4 concludes with a short discussion of the implications of these results for design assistance tools.

2. METHOD

2.1 AN OBSERVATIONAL STUDY USING SIMULTANEOUS VERBALIZATION

During a period of three weeks, full time observations were conducted in a machine tool factory on a mechanical engineer involved in a specification task. The engineer's normal daily activities were observed without any intervention, other than to ask him to verbalize his thoughts during his problem-solving specification activity as much as possible (see Ericsson & Simon, 1984; Newell & Simon, 1972).

2.2 THE OBSERVED TASK

The engineer observed had the task of specifying the functions of the operative part of an automatic machine tool installation (described below). The functional schema he produced was intended to specify these functions for the programmer of the control part of the installation. The specifications were to be produced in the form of a particular type of functional schema (described below).

---

[1] He is not called a "software designer" anymore in this text, because
- he is called a "programmeur" (in French) by his colleagues and by himself.
- he is called a "programmer" in our other articles.

2.2.1 Physical layout of the operative part

The operative part of the installation is composed of four "stations." A central turntable rotates to take the rods from and to the loading-unloading station and to position them in front of each of the other stations. The rods are brought to and from the loading-unloading station by a system of conveyors (see Figure 1).

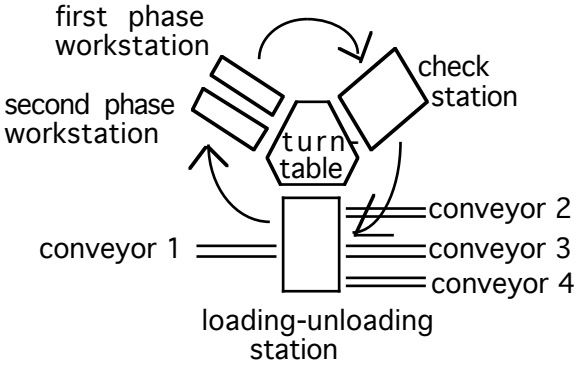

Figure 1. Spatial organization of the operative part of the machine tool installation

Key: ↘ Rotational direction of the turntable

For mechanical reasons, the Second Phase Workstation precedes the First Phase Workstation in the rotational direction of the turntable. The connecting rods to be made on this installation are, however, first shaped on the First Phase Workstation and then finished on the Second Phase Workstation. We will see that this arrangement has consequences for the specification activity.

2.2.2 The engineer's informational starting point for specification: functional requirements, mechanical specifications and an example of functional specifications

In writing his specifications, the engineer starts with two types of information about the operative part of the installation: functional requirements and mechanical specifications. He also uses the functional specifications document which had been previously written by a colleague for an analogous machine tool installation.

Functional requirements. These are mainly the client's requirements, but include a first, global analysis of them which has been made by mechanical design colleagues, in the so-called "Pre-Study" stage. These designers specifically listed the operations which they judged necessary for the required functions. This analysis is reflected in one of the mechanical specifications documents, that is, in the "Tool Plates" for the two workstations. Next to a rough drawing of each station, these plates list the operations, their duration, and the technical specifications for the motors and the tools which control them (rotation speed, power, and advance distance/rotation).
These tool plates will be consulted, modified and completed by the engineer.

Mechanical specifications. They specify:
- Machine tool mechanics (stations, station tools, motors which govern tool forward/backward movements, physical tool devices, such as jacks and detectors);
- Actual capabilities of these mechanics: stations and station tools may move forward and backward, faster or slower, etc.

Functional specifications for an analogous machine tool installation (the "example"). Before he starts his specification activity, the engineer looks for an example of specifications which are analogous to the specifications which he is to write. Thus, he selects the sequences

schema (see below) of an analogous machine tool installation which has been previously written by a colleague.

The engineer's use of these documents. The first two types of documents specify the mechanical *possibilities* of the operative part - and some design hypotheses, in the form of possibly required operations. In specifying the functioning of the operative part, the engineer has to make *choices and decisions* as to which of the various possibilities will be implemented and how this will be done.

For example, the connecting rods to be made on the machine tool installation have two ends. Both have to be tooled, but they do not undergo the same operations. Each workstation has specific tools with separate control units. Thus, the engineer has to decide which operations will be performed on which end of each rod and how these operations will be articulated (for example, will they take place simultaneously or successively?).

The engineer sometimes refers to the example when he has a problem, in order to study the solution it proposes.

2.2.3 Specification formalism: a functional schema

Several representational formalisms are used in industry for specifying sequential processes. In the industrial plant where the observed engineer works, the formalism that had been used until this study was the "sequences schema." This installation was the first to be specified using the "Grafcet" formalism, which is supposed to allow less ambiguity (see below).
The observed engineer decided, however, to proceed in two stages:
• to specify installation functioning using his usual formalism, that is, a "sequences schema;"
• to "translate" this schema into a Grafcet.
Both stages were observed. This paper presents only the first stage, except for some global comparisons with the second. The Grafcet construction has been presented elsewhere (see Morais, 1987).

As the engineer uses a sequences schema for his specification activity, this representational formalism will be described with the detail required to understand the results. Only the most important characteristics of the Grafcet formalism will be presented.

> N.B. It was the engineer's intention to use a sequences schema for all specifications and to "translate" the results into a Grafcet. During this "translation" stage, however, he took additional design decisions (and made numerous modifications in the specifications as reflected by the sequences schema). For example, he defined a function for which the mechanical requirements had not yet been set by the client when the engineer was constructing the sequences schema (the conveyor function).

Sequences schema

This formalism breaks installation functioning down into three levels: global installation functioning, individual installation functions, and function operations. Materially, installation functioning is represented on "Cycle Plates," which are special paper forms. These plates are called "cycles" by the designers, but they use the term "cycle" also to refer to the functioning represented on a plate. Each function is accomplished by a sequence of operations, together constituting the corresponding "Function Cycle." Installation functioning is cyclic on the level as well of the functions, as of the function operations. In this paper, "cycle" will be used to refer to functioning ("General Cycle" for the global installation functioning; "Function Cycle" for the functioning of individual installation functions), "cycle plate" to its representation.

Design components. Installation functioning is broken down into what in this text are called "components"[2]. At the highest level, it is broken down into functions, together constituting the "General Cycle" (see Figure 2).

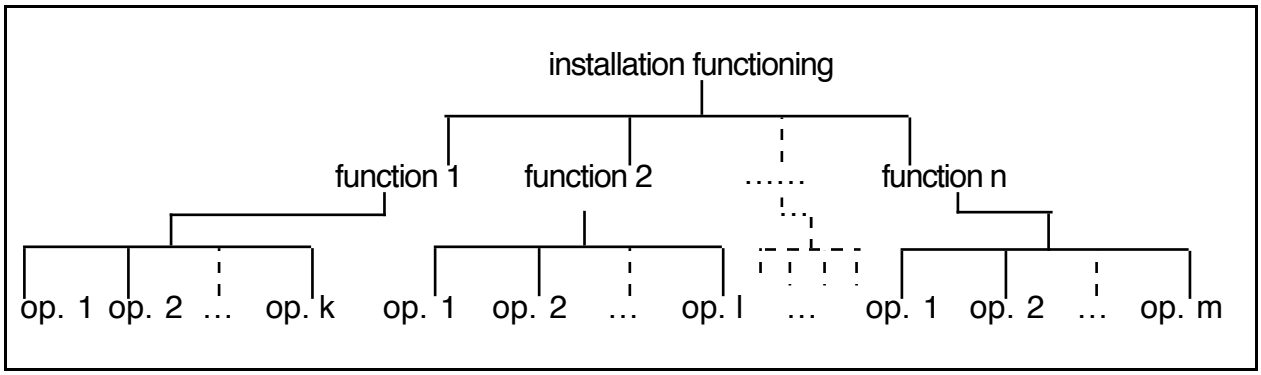

Figure 2. Breakdown of installation functioning

Thus, there are two types of cycles, and corresponding cycle plates. There is one General Cycle Plate, globally defining the different installation functions. There are six functions: first phase tooling (shaping the rods), second phase tooling (finishing the rods), checking, loading-unloading the rods to and from the installation, conveying between the different stations of the installation (implemented by the central turntable), and conveying to and from the installation via the loading-unloading station (implemented by the system of four conveyors). For each function (except for the conveyors, for which the client's requirements had not yet been set), there is a Function Cycle Plate defining the operations fulfilling this function (see Figure 3).

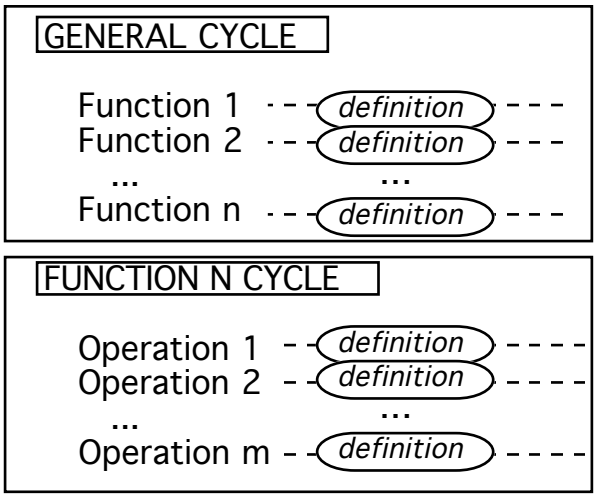

Figure 3. Schematic representation of the two types of cycle plates: the General Cycle Plate and the Function Cycle Plates

Component descriptors. Each function, as well as its operations, is defined by way of several "descriptors" (for example, duration, or starting and ending conditions). The temporal articulation between functions (in the General Cycle) and operations (in each function cycle) is represented graphically. This will be illustrated in the example of a Function Cycle Plate, the Turntable Cycle Plate (see Figure 4).

---

[2] "Components" is a term introduced to refer, without distinction, to "Functions" and "Operations" which are the terms used by the engineer.

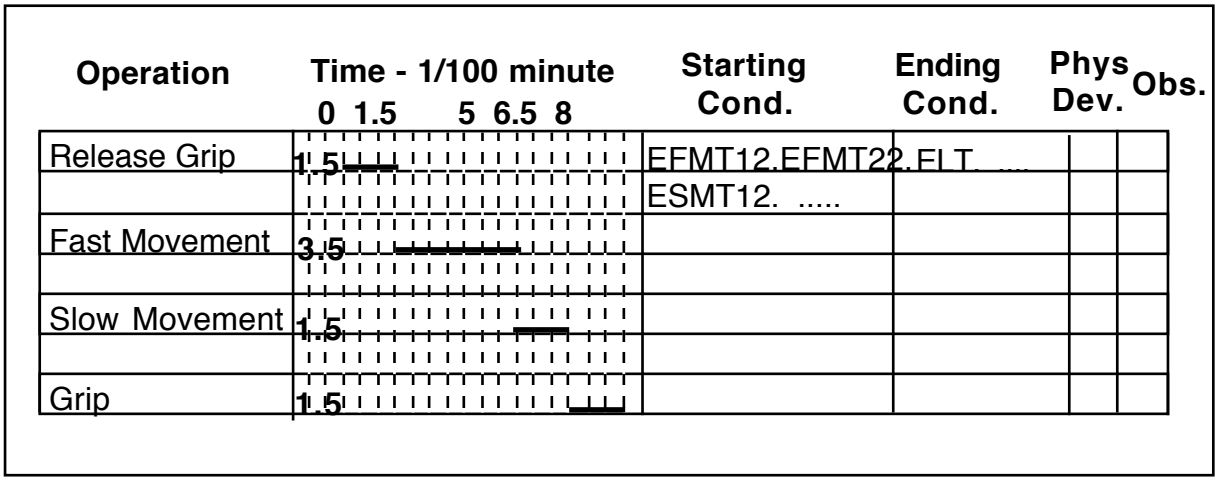

Figure 4. Example of a Function Cycle Plate: the Turntable Cycle Plate (partial)

Figure 4 is a partial representation of a Function Cycle Plate showing only its elements that are relevant to explain how an operation is defined. Specifically, it shows the representational formalism used for the different aspects of an operation's definition:

- Identifier. The operation "identifier" is written in the column "Operation;"
- Duration. Its "duration" is represented in two ways: (a) in the form of a number in the "Time" column, written on the left of (b) its analogical representation, a line of a length which is proportional to its duration. The duration of the Fast Movement operation, for example, is 3.5 time units (1/100 minute), represented by a line of 7 time-representation units.
- Temporal articulation between operations. This articulation is represented by the relationship between the lines representing the different operation durations. The duration lines of two operations which take place simultaneously will be parallel; the duration line of an operation which follows another one will start where the duration line of the other operation ended. Thus, the Fast Movement operation, for example, follows the Release Grip operation.
- Starting and Ending conditions. Represented by boolean expressions, they consist of mnemonics - that is, the abbreviated names - of the control detectors which must be activated or deactivated for the operation to start or to end. For example, the starting conditions for Release Grip are: End of Fast Movement in Table12 (EFMT12) AND End of Fast Movement in Table22 (EFMT22) AND End of Slow Movement in Table12 (ESMT12), etc.
- Physical device. This column contains the mnemonic for the motor or jack that activates or deactivates the operation.
- Observations. May or may not be present for an operation. The engineer used this column for several things: ending or starting conditions for which he did not know the physical form; safety conditions which are generally different from the starting conditions of one particular operation, because they apply to all the operations of a function[3]; conditions under which a machine operator has to adjust or fix something, which generally involves stopping the installation[4].

Grafcet

This formalism provides a graphical representation of the sequential progress of the process, showing the sequence of actions and their enabling conditions (corresponding

3 The sequences schema formalism poses a representation problem in this regard. It does not provide the possibility of representing conditions applying to more than one operation other than by repeating these conditions for all the operations involved. The Grafcet formalism allows this kind of information to be represented.

4 Here, the only possibility is to note the information in the Observations column. Once again, the Grafcet formalism allows this kind of information to be represented in a more appropriate manner.

partially[5] to the starting and ending conditions used in the sequences schema). For each function, the graphical illustration is accompanied by documents providing a written description of (a) the safety conditions for each action; (b) the physical starting point of the cycle; (c) the procedure to be followed in the event of a mechanical or process control problem (see Morais, 1985).

2.3 THE OBSERVED MECHANICAL ENGINEER

The observed engineer had more than ten years of professional experience in the machine tool factory. The mechanical installation studied - type tooling - was not his specialty, which was assembling. Tooling and assembling are two applications in machine-tool manufacturing. Most mechanical engineers know both, but are specialized in one of the two. We will see that certain results may be attributed to this characteristic of the subject.

2.4 DATA COLLECTION

Notes were taken on the engineer's actions; all documents which he produced during his work were collected.

Notes. These concerned:
- The engineer's problem-solving actions (disclosed by his verbalization), his other remarks and comments;
- The order in which he produced the different documents, and how he gradually built them up;
- The changes he made;
- The information sources he consulted;
- The events considered by the observer to be indicators of the subject's encountering difficulties.

Documents collected.
- The different versions of the sequences schema;
- The diagrams and schemas the engineer constructed for himself during problem solving.

2.5 DATA ANALYSIS

Specification-action units. Specification involves a problem-solving activity. Among the notes concerning the engineer's actions, those which contribute to specification of installation functioning are isolated and constitute the reasoning chain examined. This reasoning chain has been broken down into problem-resolution steps: one or more steps that concern the same aspect of a design component constitute a specification-action unit.

Example. Verifying the duration of a component constitutes one action unit.
Defining a component with regard to its identifier and its duration constitutes two action units.

Component processings. A sequence of specification-action units concerning one or more aspects of the same design component (function or operation) is considered to be one component processing.

Example1. While defining a component consecutively with regard to its identifier and its duration constitutes two action units, it constitutes one component processing.

Example2. Verifying the duration of a component constitutes one action unit. It is considered to be one component processing unit, if immediately afterwards the engineer goes on to another component. If he defines also the ending conditions of the component - without any interruption between the two action units - this ending conditions definition belongs to the same component processing as the duration verification.

Cycle-definition sessions. A sequence of component processings involving the same cycle is called a cycle-definition session.

Figure 5 presents schematically the relationships between these three units of analysis.

---

5 The "ending conditions" of an operation are not simply the "starting conditions" of the following operation, but some of the "ending conditions" of an operation are generally among the "starting conditions" of the following operation.

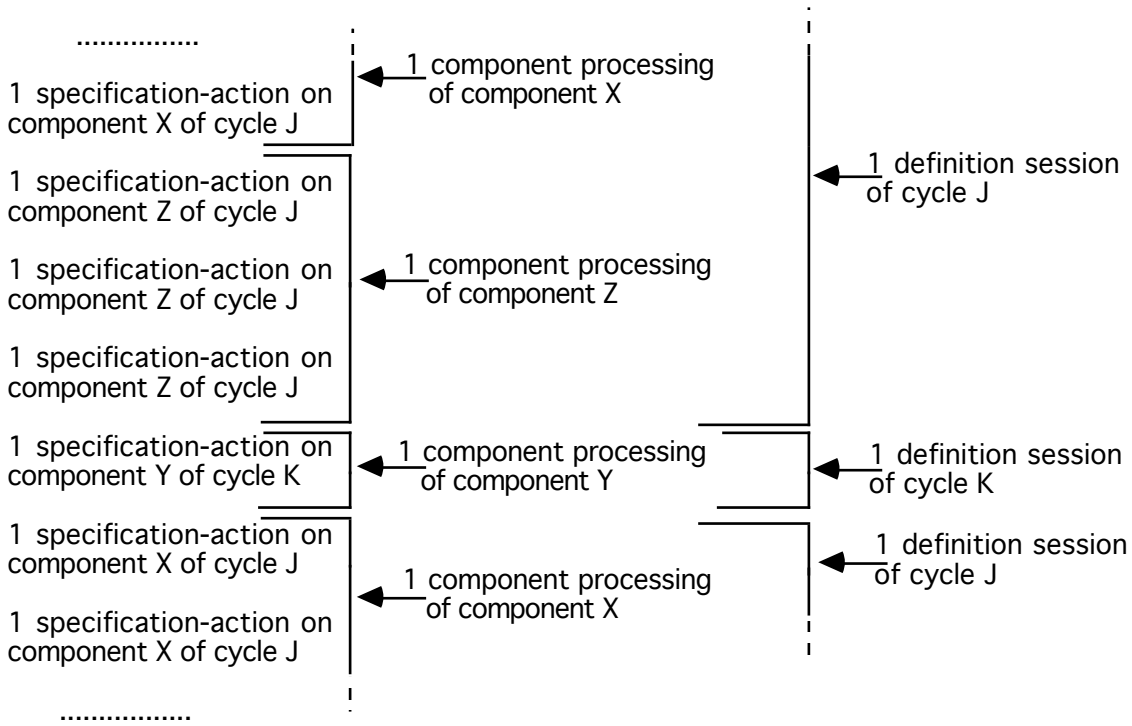

Figure 5. Relation between specification-action units, component processings and definition sessions

As the reader of this text may want to consult the definition of some terms used in this paper, Table 1 (see p. 10) collects them in a glossary. Some terms defined here will only be introduced later on in the text.

3. RESULTS

Preliminary methodological remark. Fluent and continuous verbalization. Simultaneous verbalization - as was asked of the observed subject - is not always easy, or even possible, for a person involved in a problem-solving activity. The engineer did not find it difficult, or troubling, to "think out loud." The observer never had to remind him to keep to verbalize, as is generally required in protocol studies, where subjects may be seen to become lost in thought, or to resume their normal, silent mode of thinking (see Ericsson & Simon, 1984).

> N.B. In the observational study conducted on the programmer designing the program from the specifications made by the engineer, the same simultaneous verbalization technique was used (see Visser, 1987). The programmer, however, verbalized rather little while writing the program. The hypothesis was formulated that many of the programmer's actions during program construction, especially during coding, are automated (as a consequence of his experience in the field) and that verbalizing would require a "decompilation" of these automated procedures (see Anderson, 1986). Encouraging the programmer to verbalize more might lead him to make the knowledge sources underlying these procedures explicit, but such verbalization would not express the actual activity the programmer performs in writing his program. The engineer's continuous verbalizing may, with some caution, be interpreted as his specifying being an actual problem-solving activity, with only few automated components.

Table 1. Glossary of terms

component (design-)	an operation or a function
component processing	a sequence of specification-action units concerning one or more aspects of the same component
cycle	installation functioning. There are two types of cycles: - the General Cycle refers to global installation functioning; - Function Cycles refer to the functioning of individual installation functions.
cycle plate	paper form ("plate") for representing a cycle
cycle-definition session	a sequence of component processings involving the same cycle
function	component of the General Cycle. Each function is defined globally on the General Cycle Plate; it is defined completely on its individual Function Cycle Plate by way of its components, the operations.
operation	component of a function cycle. Each operation is defined on the corresponding Function Cycle Plate by way of its descriptors.
prerequisite operation	particular component of a Work cycle. Prerequisites to the work operations, the prerequisite operations allow for satisfying the conditions required for the work operations to take place.
specification-action unit	one or more problem-resolution steps (in the reasoning chain leading to the specification of the installation) that concern the same aspect of a design component
Work function	particular component of the General Cycle, corresponding to a tooling function. There are two work functions: First Phase (shaping) and Second Phase (finishing).
Work cycle	particular function cycle corresponding to the Work functions. There are two work cycles: First Phase (shaping) and Second Phase (finishing).
work operation	particular component of a work cycle. Doing the tooling, work operations allow directly to achieve the goal of the machine tooling process.

3.1 GLOBAL ORGANIZATION OF THE SPECIFICATION ACTIVITY

At the level of the activity organization, the most important and interesting global result of this study is that
• the engineer describes[6] his activity as following a hierarchically structured plan[7];
but

[6] The engineer was requested to describe his activity before and after performing it.
[7] We call the plan "hierarchically structured" and not "hierarchical," to avoid confusion with "hierarchical planning" as described by Sacerdoti (1974).

• his actual activity, that is, the activity such as it was observed, is opportunistically organized: the engineer follows his plan only as long as he does not perceive more opportune actions (see Hayes-Roth & Hayes-Roth, 1979).

After a short presentation of this plan, and the roles it may play, the rest of this chapter will concern the actual specification activity, and focus on plan deviation.

3.1.1 The activity plan as described by the mechanical engineer

The engineer described his activity in the form of a hierarchically structured plan containing four levels (the global functioning to be represented on the sequences schema, its cycles, their components and their descriptors) (see Table 2, p. 12). The engineer's control procedure for covering this tree-structure reflects top-down, depth-first planning.

For reasons given below, this plan does not represent the engineer's actual activity. The same is true for both of the other subjects observed on the design project: the plans they produced did not reflect their actual activity. In general terms, the subjects did not follow a systematic path through the tree representing the plan which they produced as describing their activity (see Visser, 1988a).

One plan: Two functions

A plan may guide the activity in at least two ways (see Hoc, 1988b).

Declarative plan: Structure of the result of the activity. By showing the structure which the result - or an intermediary state - of the activity must have, a "declarative plan" allows the states this activity must attain - or go through - to be anticipated.

Procedural plan: Structure of the activity. If a plan represents both the coordination between actions to be realized and elements of the control structure, it may really guide the activity and is considered to be a "procedural plan."

The plan produced by the engineer, that is, a hierarchical action structure accompanied by a procedure which covered it, reflects such a "procedural plan." The engineer may have thought that he was following it, but he was observed to deviate from it whenever more opportune actions, or more opportune local plans, were perceived. His plan certainly guided him, but only as long as such interesting (i.e., cognitively more economical) opportunities did not arise. When they did, it was immediately abandoned.

3.1.2 The actual activity as observed

The actual activity as it was observed will be presented from two points of view. First, the degree to which the engineer deviates from the plan will be quantified (§3.2). Then, the processes leading to these deviations will be presented in the context of a general blackboard model of the activity (§3.3).

Table 2. Representation of the engineer's description of his activity

+ Construct a sequences' schema
  + Define the General Cycle
    + Define its first function
      + Determine the first function
      + Determine its descriptors
        + Determine its identifier (id)
          Insert its value in the id column
        + Determine its duration (du)
          Insert its value in the du column
        + Determine its starting conditions (st)
          Insert their values in the st column
        + Determine its ending conditions (en)
          Insert their values in the en column
    + Define its second function
      + Determine the second function
      + Determine its descriptors
        …
      …
    + Define its nth function
      ...
  + Define the first Function Cycle
    + Define its first operation
      + Determine the first operation
      + Determine its descriptors
        + Determine its identifier (id)
          Insert its value in the id column
        + Determine its duration (du)
          Insert its value in the du column
        + Determine its starting conditions (st)
          Insert their values in the st column
        + Determine its ending conditions (en)
          Insert their values in the en column
        + Determine its physical device (ph)
          Insert its value in the ph column
    + Define its second operation
      + Determine the second operation
      + Determine its descriptors
        …
      …
    + Define its mth operation
      …
  + Define the second Function Cycle
    ...
    …
  + Define the nth Function Cycle
    ...

Key : + precedes a goal to be achieved
(Lines not preceded by + describe the actions carried out in order to achieve the immediately preceding goal)

... Replaces goals whose breakdown follows the breakdown presented previously in the schema for a similar goal

3.2 QUANTITATIVE RESULTS: DEGREE OF PLAN DEVIATION

Table 3 shows the abbreviations used in the presentation of the results.

Table 3. Abbreviations used

GC	General Cycle
Tt	Turntable Cycle
Ld	Loading Cycle
W1	First Phase Work Cycle
W2	Second Phase Work Cycle
Ch	Check Cycle
C	component
D	descriptor
O	operation
id	identifier
du	duration
st	starting conditions
en	ending conditions
ph	physical device

3.2.1 Highest level: Cycle definitions

Preliminary methodological remark. Missing data. The observations on the definition of the Check Cycle are incomplete: the engineer started it one evening after work. This function will be considered in this chapter only where qualitative data are sufficient, that is, for questions of definition order.

Cycle-definition interruption

The engineer does not complete the definition of a cycle before he starts another: only half (52%[8]) of the definition of each cycle is done without interruption (between 31% for W2 and 73% for GC).

On average, he defines a cycle in five cycle-definition sessions; that is, after he has started the definition of a cycle, and after he has abandoned it to define other cycles, he still interrupts the definition of these other cycles four times in order to come back to it. He mainly does so to complete, not modify, it. In this case, he mostly completes the information that specifies the start of the cycle (by way of the descriptor "starting conditions" for its first function or operation[9]).

Except for the Loading Cycle, the definition of each cycle is interrupted with approximately the same frequency: about once every four operation-processing units, that is, one cycle-definition session is made up, on average, of some four operation-processing units. The Loading Cycle is interrupted much less frequently than the other cycles: its cycle-definition sessions are made up, on average, of 11 operation-processing units. Explanations for those results will be presented below.

---

[8] All percentages presented below are averages. If spread is important, the extreme values are also given.

[9] This is not "correct," but the sequences schema does not at all allow this information to be represented. Once again, the Grafcet formalism does.

Only for the General Cycle is most of the definition done during the first cycle-definition session (73%). For the other cycles, most of the definition is accomplished during later sessions (the third, fourth or sixth session).

The following questions arise regarding the different cycles:
• When do these sessions take place?
and
• What proportion of total definition is done during these sessions?

This information is given in Figure 6, which shows all successive cycle-definition sessions. In this figure
• The x-axis presents the successive cycle-definition sessions throughout the three weeks specification-writing duration.
• A cycle is considered to be defined at 100% when all cycle components have received all descriptors on which the engineer defines them. We will see below that many components were not defined on all five descriptors.

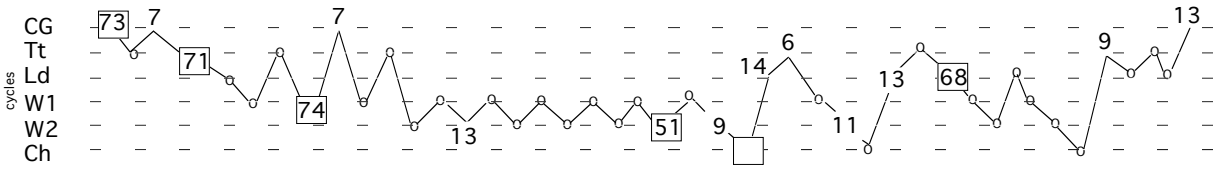

Figure 6. Successive sessions of cycle definition

Key: ° Corresponds to a cycle-definition session during which ≤5% of the cycle is defined.

x (where x > 5) Corresponds to a cycle-definition session during which >5% of the cycle is defined, with x indicating this percentage.

□ Corresponds to the cycle-definition session during which the greatest contribution to the definition is made, with the definition percentage corresponding to this contribution.

N.B. The empty case for the Check Cycle definition session reflects
• the hypothesis that the greatest contribution to the definition of this cycle was made during the first definition session
• the fact that this session was not observed

• For those sessions during which more than 5% of the cycle is defined, the percentage of cycle definition is given.
• The session during which the quantitatively greatest contribution to the definition was made is indicated for each cycle.

Two aspects of the cycle-definition interruptions were examined:
• Transition patterns: what is the nature of the other cycle for which a cycle's definition is interrupted? Is there a regularity? If yes, what type of relation exists between the two cycles?
• Deviation patterns: what is the nature of the interruption? How many components are processed during an interruption? Is only one other than the plan component processed, and does the engineer come back to the interrupted cycle definition immediately afterwards? Or does one deviation lead to another?

Figure 6 shows that only the two Work Cycles have a rather systematic transition pattern: for each of them, about two-thirds of the interruptions are made in order to define the other. The lowest-level presentation will detail the reasons for these movements.
With regard to deviation patterns, the Work Cycles are also the only cycles for which an interrupted definition is, in most cases (77%), resumed immediately afterwards.

Function definition order in the General Cycle Plate vs. definition order of the function cycles

The order in which the different functions are defined in the General Cycle Plate was not changed afterwards - contrary to what occurred with the operations on the function cycle plates. The engineer planned to define the individual function cycles in the same order as he defined the corresponding functions in the General Cycle Plate. In reality, he defined them afterwards in another order (see Table 4).

Table 4. Comparison of the orders in which functions are defined in the General Cycle Plate and in which function cycles are defined

Function definition order on GC	Function cycle definition order
Tt	Tt
Ld	W 1
W 2	W 2
W 1	Ch
Ch	Ld

The observed differences deserve comment on two points:
• the criteria used for ordering the functions
• the moment of definition of the Loading Cycle

Other criteria used for ordering the functions in the General Cycle Plate than for ordering the function cycles. The functions in the General Cycle Plate followed the order in which the turntable rotates to position the parts in front of each of the corresponding stations on the installation (see Figure 1). The engineer started with the Loading-Unloading Station - which is generally the point of reference for machine tool installations.

On all requirements and mechanical specification documents, as well as on construction drawings, the stations are also numbered according to the rotation order:
• the Loading-Unloading Station is numbered 01
• the unit consisting of the two workstations together is numbered 02:
    • the Second Phase Workstation is numbered 021
    • the First Phase Workstation is numbered 023
• the Check Station is numbered 03
However, this rotation order does not correspond to the order in which the parts are tooled, in which the First Phase (shaping) precedes the Second Phase (finishing).

The tooling order - that is, the functional order of the installation - is the order that the engineer followed in defining the individual function cycles. This explains why the Work Cycles are defined in the reverse order from their definition order on the General Cycle Plate. On the General Cycle Plate they were only globally defined. For a detailed, precise definition, what is relevant is function, not the rotation order of the turntable or the numbering of the stations.

Time of definition of the Loading Cycle: postponement for reasons of cognitive cost. Figure 6 seems to show that definition of the Loading Cycle is initiated after Turntable Cycle definition. However, during this first cycle-"definition" session for the Loading Cycle, the engineer does no actual definition. He examines the Loading Cycle on the example sequences schema and does not understand how it has been broken down[10], which leads him to decide to postpone the Loading Cycle definition. Thus, the engineer defines the Loading Cycle only after having defined the Check Cycle.

[10] In the example, the Loading Cycle is divided into two parts: "General" and "Detail."

3.2.2 Intermediate level: Global component definitions

Half of the function and operation definitions (55%) were finished during the first processing unit, that is, before the engineer started the definition of another component.

With General Cycle functions, an important part of each function was defined during its first processing: on average, 86% of the descriptors used for function definition were attributed. However,
- as will be detailed below, on the General Cycle Plate the engineer defined only one or two descriptors of the functions (id, or id and du);
- only 67% of the descriptors used are set once and for all, the rest were later modified;
- during this first component processing, 63% of the functions were defined completely, that is, on all (one or two) descriptors on which the engineer defined them.

With function cycle operations, 68% of the descriptors used for their definition were attributed during their first processing (between 41% for Tt and 81% for Ld). However,
- only 55% of the descriptors are set once and for all (between 25% for Tt and 74% for Ld), the rest were later modified;
- during this first component processing, only 57% of the operations were defined completely (between 0% for Tt and 67% for Ch).

Temporal relationships among the operations of a function cycle

Using the same criterion as for the definition order of the function cycles, the engineer plans to represent the components of a function cycle - that is, its operations - in their functional order. Generally, he also actually defines and represents them right from the start in this order, but some operations are initially forgotten, and processed only afterwards in order to define them. The engineer inserts these operations then at their appropriate position on their function cycle plate. This holds especially in the case of the Work Cycles.

- Turntable Cycle. Its four operations are introduced immediately in their functioning order.

- Loading Cycle. In 32 operations introduced "under our observation"[11], three changes were observed:
  - For two operations, the engineer changed his mind about their temporal order relative to the other operations. For both of them, this decision was made immediately after he introduced them, that is, during their first processing or during the first processing of the next operation (which he then decided should take place before the one already introduced).
  - A group of two simultaneous operations was forgotten until much later. Although they actually take place in 13th position, they were only remembered in 30th and 31st positions.

- Work Cycles. It is on these two cycles that important omissions and changes regarding the temporal articulation between operations occurred.
  Out of a total of 26 (2 times 13), the engineer forgot 10 operations during the first pass.

Two remarks before detailing and commenting on these results:
- As mentioned in §2.2.2, for the definition of these cycles, the engineer used two requirements analysis documents, the tool plates. His most important point of reference for their definition was, however, his knowledge of the required functioning and his own functional analysis, based on the client's requirements. The tool plates were used for retrieving physical information about the operations they describe (duration, etc.).
- There are three types of Work Cycle operations[12]:

---

[11] The other 20 loading operations (constituting the second half of the cycle) were defined one evening, after work.

[12] Four other operations on both workstations are not considered here. They are, for example, cleaning and oiling operations taking place during the whole cycles.

Station movements (forward and backward), during which no work (tooling) is performed. The forward movement positions the station in tooling departure position. The backward movement positions the station in a mechanical safety position[13]. There are four of these movements, one forward movement and one backward movement on each station.
Work operations are the operations doing the work (tooling) on the installation. Thus, they directly allow to achieve the goal of the machine tooling process. There are five of them.
Prerequisite operations - that is, prerequisites to the work operations - are operations that allow for satisfying the conditions for the work operations to take place (for example, positioning a tool, or setting a stop to establish the position where a work operation must end). There are 17 of them.

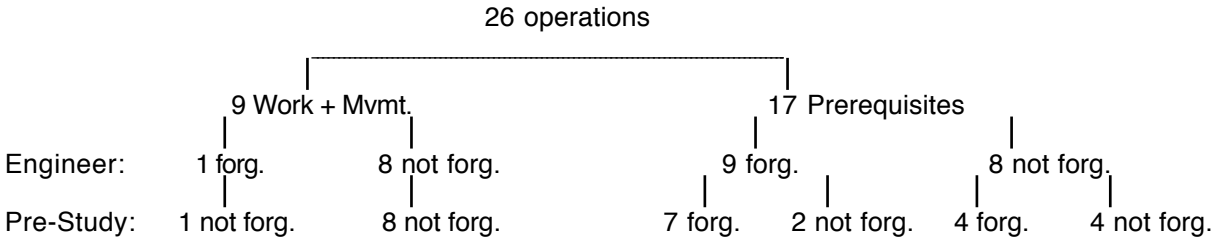

Figure 7. Distribution among Prerequisites and Work operations + Movements of operations forgotten by the engineer during his first pass and/or by his colleagues during their requirements analysis (the Pre-Study stage)

Operations forgotten are Prerequisites. This holds true for nine out of the ten operations which were forgotten by the engineer[14] (see Figure 7). The only other operation which he forgot was a very unusual work operation which takes place during the backward movement of the station. In machine tool installations, work operations nearly always take place during the forward movement of the station.
The engineer often only discovered that he had forgotten a prerequisite operation when he processed the operation of which it is a prerequisite, an opposite operation, or a similar operation - that is, an operation of which the representation may activate the representation of the forgotten one by way of the relationship between the two representations in memory (see §3.3.1 below).

N.B. The programmer did not actually forget the work operation which takes place during the backward movement of the station, but he defined it as taking place during the forward movement - an error which was subsequently corrected by the person doing the debugging. This result was interpreted as due to *Schema-guided information processing* (see Visser, 1987). The programmer violated the specifications for this "atypical" operation by defining it as a "prototypical" operation. If he did read the specifications - he was observed to skim through them only once rather rapidly - his expectations, based on prototypical schema slot values, were probably so strong that he did not take into account the values which were explicitly given (in the specification document). If he wrote the relevant part of the program without reading its specifications, the schema he instantiated may have provided him preferentially with this prototypical value (see also Détienne, to be published).

Operations forgotten were not specified before. Out of the nine prerequisite operations which the engineer forgot, seven had neither been planned during the Pre-Study (the preceding design stage, see §2.2.2); that is, they do not appear on the tool plates. Four other operations had not been planned during the Pre-Study, but were not forgotten by

[13] In which, for example, the Turntable may rotate without damaging the stations, or tools may be changed.
[14] "Forgotten," that is, forgotten during the first pass and introduced afterwards.

the engineer; they were also all Prerequisites (see below, §3.3.1, for a comment on the specific role of prerequisites in information processing).

Modifications during construction of the Grafcet. The engineer's intended strategy was to do the specification in a sequences schema formalism, and then to "just translate" the result into a Grafcet. During this "translation," however, he still made numerous modifications. One important modification type concerned the temporal articulation of operations. For the First Phase Work Cycle, he moved one operation to later in the cycle. The Second Phase Work Cycle was considerably modified in this respect: for 5 out of 13 operations, the temporal occurrence was changed, and 3 of these operations were Prerequisites.

3.2.3 Lowest level: Detailed component definitions

Descriptors used for definition

Components are defined with five descriptors:
id identifier
du duration
st starting conditions
en ending conditions
ph physical device

They may also be further defined by observations in a special "Observations" column (larger than could be represented in Figure 4). This column is sometimes used for information which is difficult - or even impossible - to represent on a sequences schema (see Notes 3 and 4).

For all but 5 of the 70 components, when all 5 of the descriptors were used, the engineer attributed them in this order, which is in complete accordance with his plan. However, he generally did not attribute them during the first processing of the corresponding component, and once they had been attributed, he often changed them.

As a matter of fact, even if the specifications are supposed to define all components using these five descriptors, the sequences schema produced by the engineer did not specify all components completely. Two remarks may be formulated:
- We may consider that specifications are always imprecise and incomplete and that this result is "normal;"
- The engineer may not consider the specifications in the sequences schema to be complete and may plan to complete them during Grafcet construction.
  This is contrary to what he says after finishing the construction of the sequences schema. It could be, however, that the engineer's verification criteria are less severe than usual for this non-final specification formalism, which is to be converted into another specification formalism.

General Cycle. General Cycle functions are not defined by all five descriptors: for half of the functions, two descriptors are used (id and du), for the other half only one (du).

Function cycles. Less than half of the operations in the function cycles are defined by five descriptors (41%). 27% are defined by only one (id) or two (generally id and du), 32% by three or four.
With regard to the type of descriptor attributed to the operations, the results are as follows:
- All operations are assigned an identifier.
- Duration, and starting and ending conditions are attributed in most cases (71%).
- Half of the operations are also defined by their physical device (48%).
- Observations are made for 9% of the operations.

Reprocessing an operation: quantitative differences from cycle to cycle

The number of times an operation is processed differs between the function cycles (see Table 5).

Table 5. Percentage of operations per function cycle processed one or more times (from 1 to 11)

	----- number of processings per operation -----												
	1	2	3	4	5	6	7	8	10	11	tot op	tot pr	av % pr/op
Tt	0	0	0	0	25	25	0	25	0	25	4	30	7.50
W1	41	6	18	12	6	12	0	0	6	0	17	53	3.12
W2	29	29	12	12	6	6	6	0	0	0	17	47	2.76
Ld	47	44	9	0	0	0	0	0	0	0	32	52	1.63
tot	39	29	12	6	4	6	1	1	1	1	70	182	2.60

Key:
- tot op — total number of operations
- tot pr — total number of processings
- av % pr/op — average percentage of processings per operation

Two results are of special interest:
- The number of processings per operation declines as definition progresses from Turntable Cycle to Loading Cycle definition;
- Loading Cycle operations are processed the fewest number of times.

Decline in the number of component processings per function cycle. The Turntable operations are reprocessed more often than others: from five to eleven component processings vs. some three processings for the other function cycles. The decline in the number of processings per operation follows the order of cycle definition. This result can not be attributed to the definition of the machine functioning becoming more and more constrained as specification progresses, because - except for a very few interactions (see below, §3.3.1) - each station functions independently of the others. The most plausible explanation for this result seems to be a learning effect.

The few component processings per Loading Cycle operation. Proportionally, Loading Cycle operations are processed the fewest times (an average of 1.63). This result would appear to be related to a result presented earlier: the Loading Cycle definition being interrupted less frequently than the other cycle definitions - only once every 11 component processings instead of every 4. The following explanation may be proposed. The Loading Cycle is for the engineer by far the easiest one to define, because loading is the type of function he knows best from his experience, which mostly involves assembling installations.

The fact that the first processing of this function only leads to postponing its definition does not contradict this explanation. The postponement is done at a stage of the specification process in which the engineer is still using the example sequences schema: his difficulty in understanding the Loading Cycle reference in this example is due to the representation used for the function, not to the functioning represented.

Reprocessing an operation: completing and/or modifying it

- An operation may be reprocessed in order to complete its definition by adding new information (descriptor or observation); 40% of operation reprocessing is of this type (between 21% for W1 and 54% for W2);

- An operation may be reprocessed in order to modify its definition by changing a descriptor or observation; 44% of operation reprocessing is of this type (between 29% for W2 and 59% for W1).
- During the same operation-reprocessing procedure, the engineer may complete and modify the definition of the operation; 10% of operation reprocessing is of this type.
- Finally, 6% of operation reprocessing do not contribute to the definition of the operation.

The contrasting results for the two Work Cycles with regard to reprocessing the definition of an operation, that is, for completing it (W1 < W2) or for modifying it (W1 > W2), may be interpreted as follows.
The engineer starts the definition of the Work Cycles with the First Phase. On the one hand, when he defines the Second Phase, which is analogous to the First Phase in various respects, the already defined aspects of the First Phase provide him with a reference. Thus, the definition of the Second Phase is often initiated by processing on the First Phase. That is, many processings of the Second Phase are for completing components, adding new aspects analogous to the ones just defined on the First Phase. On the other hand, defining the Second Phase leads the engineer to discover errors and omissions in the First Phase and, thus, to reprocess the First Phase in order to modify it.

3.3 QUALITATIVE RESULTS: PROCESSES LEADING TO PROPOSING PLAN-DEVIATION ACTIONS

To account for plan deviation during specification, a model of the activity is necessary (see Visser, 1988b, where some first elements were presented, especially with regard to control). The observed deviations did not follow another plan - one the engineer was not conscious of, for example. They are not systematic, but depend - like the occurrence of planned actions - on the data which the engineer has at the time: specifically, the state of the sequences schema in progress, his representation of this schema and his knowledge, and the information which he has at his disposal and which he receives. This amounts to giving the activity's organization an opportunistic character.

Control. The component in a model of an activity on which the type of activity organization depends is the control. This is the system component which accounts for deviating from a plan - and for resuming it. To contribute to the construction of a model, and especially to formulate the control knowledge used by the engineer, a qualitative analysis is made of the data, especially with regard to the nature of the *transitions between the different component processings*. The engineer's remarks are very important in this respect, but the analysis goes of course beyond the data. Having a type of model in mind, the researcher may infer - from component processing sequences and transitions, as well as the remarks made by the engineer - the knowledge used in specifying and the way it is used.

A blackboard model of specification

The qualitative analysis of the observations confirms the hypothesis that the specification activity is organized opportunistically. The presence - and even the use - of a plan does not contradict this conclusion. The plan which guides the activity is used in an opportunistic way, that is, only when no more opportune actions arise (see Hayes-Roth & Hayes-Roth, 1979).

Blackboard models have been proposed for opportunistically organized activities. Their main components and control structure will be presented briefly (for a general presentation of the model, see Nii, 1986; for three applications of the model, see Hayes-Roth & Hayes-Roth, 1979, for errand planning; see Whitefield, 1986, for mechanical design; see Bisseret, Figeac-Létang, & Falzon, 1988, for the activity of traffic signal setting).

Main components of the model. A blackboard model has two basic components, (a) the knowledge sources partitioning the knowledge used for solving the problem, and (b) the blackboard data structure, the database holding the problem-solving state (the problem

solver's working memory). In general, a separate control component exists in systems based on the model.

Control structure. The different actions are articulated according to the following iterative sequence:

i. An action modifies the state of the blackboard
ii. One or several knowledge sources make action proposals because they are able to contribute to the resolution of the problem as it is defined by the state of the blackboard
iii. Of the proposed actions, the control selects one action to be executed, depending on
  • the state of the blackboard (see i.)
  • the knowledge sources having made proposals (see ii.)
  • the control knowledge (especially the selection criteria used, see below)
iv. Back to i.

Selection of a plan-deviation action[15] rather than the planned action

For reasons given above (see section "Control"), this study develops the control component. Deviations from the engineer's plan are considered to be actions which (a) were proposed as an alternative to the planned action and (b) were "preferred" by the control over the planned action (both for reasons analyzed below).

To decide which action is going to be selected for execution, the control evaluates each proposed action according to at least two criteria. Thus, when there are several competing actions - that is, when one or various actions are proposed that compete with the planned action - these actions are compared by the control.

The control may also decide, as a result of evaluation, to not select an isolated planned action - that is, the planned action which is the only action proposed. In that case, another action must be selected (see §3.3.6).

First action selection criterion: Cognitive cost. The most important criterion in selecting an action is its relative cognitive cost. For each action proposed, this cognitive cost is determined.

In evaluating the cognitive cost of an action, the control considers such not necessarily independent factors as:

• **the availability of a "schema" for executing the action**. A schema is a memory data structure defining a generic concept on a number of variables and providing, for each of these variables, a certain number of values, generally including one default value (see Rumelhart, 1978). Executing an action for which such a memory representation is available may cost relatively little if all variables relevant for execution have default values.

  The engineer's plan may be considered to be a schema of this nature. This is why its use is so profitable from the viewpoint of cognitive cost. But deviating from this plan may sometimes be still more profitable (from the viewpoint of cognitive cost) for various reasons which will be presented below (see §3.3.1-§3.3.6).

• **the availability of information**. On the one hand, an action may be interesting provided that the information required for executing it is available, that is, accessible without much effort. If this is not the case, the cognitive cost of accessing this information affects the cognitive cost of the action - and thus, its interest.
  On the other hand, an action may become interesting because information available allows for its execution (see §3.3.1-§3.3.4).

---

[15] Shorthand for "an alternative-to-the-planned-action proposal leading to a plan deviation if selected."

• **the relative difficulty of the action**. This factor still asks for definition. Yet it is mentioned because it functions as an explanation heading for various observed deviations. Next to the postponement of the definition of the Loading Cycle (described in §3.2.1), it may be illustrated by the following examples.

Example1. In general, when determining the value of a descriptor, the engineer knows which type of information is required and in which type of information source he may find it.
The duration of operations, for example, may be determined in various ways, from *retrieving it* from a document (such as a tool plate) to *calculating it* from its constituents (motor speed and advance distance/rotation). *Retrieving a value* is the easiest way to determine it, *calculating it* is rather difficult.
A still more difficult action seems to be an action for which the values of the required constituents (such as motor speed and advance distance/rotation for calculating a duration) are not given (in an information source), and thus must be determined by the engineer.

Example2. If duration values are given - in a tool plate for example - they are generally given for the individual operations. However, the engineer's design colleagues made only a very global requirements analysis, and sometimes several operations as defined by the engineer were assembled into one global operation by these colleagues. In that case, the engineer will have to divide the corresponding global duration into the different individual operations. Rules or heuristics have been identified for this division. They will not be presented. What matters here is that such a division action seems to be rather difficult.

N.B.1 The high cognitive cost of the planned action - even if it does not compete with any other proposed action - may lead the control to skip over it (see below §3.3.6).

N.B.2 The cognitive cost of a deviation is a combination of the cost of the corresponding plan-deviation action **and** the cost of plan resumption.

Second action selection criterion: Importance. The second criterion is "importance." Actions differ on this point. Their importance depends on:

• **the importance of the type of action**

Example. As we know the engineer's plan, actions may be identified as plan-deviation actions. Thus, when the engineer, discovering a hitherto forgotten operation, proceeds - in spite of his plan - to the definition of this operation, this action is identified as a plan-deviation action. The nature of the action (here "fixing an omission") and of the component concerned (here, an "operation") + any remarks by the engineer are used to explain the deviation. For example, one of the (control) knowledge elements formulated to account for this (and other) observations, is the following:

Fixing the omission of an operation which has been forgotten is an important action.

• **the importance of the object concerned by the action**

Example. "Verifying" is an important action if it concerns "durations," but not if it concerns "identifiers." The engineer frequently deviates from his plan in order to verify the duration of an operation[16], but never to verify its identifier.

Resulting selection of an action. Other criteria may intervene. They have not been identified. Neither has the way in which the two criteria are combined. If an action proposed as an alternative to the planned action is important and/or if its cognitive cost is relatively low compared to that of the planned action, this action is selected for execution.

Distance between the component abandoned and the component the deviation leads to

For the purpose of presenting the different processes leading to plan deviation, three types of plan-deviation actions may be distinguished, based on the distance between the component concerned by the current processing and the component concerned by the plan-deviation action (hereafter referred to as a "deviation action").

Inter-component intra-cycle deviations. Most deviations stay inside the cycle currently being defined, going from one of its components to another.

[16] To verify the duration of an operation, the engineer recalculates it from the information elements which his mechanical design colleagues used for calculating them, that is, the technical specifications of the motors or tools concerned (their rotation speed and their advance distance/rotation).

Inter-cycle deviations. Several deviations go from a component in one cycle to an - often related - component in another cycle. The cycle to which the deviation leads is, generally, a previously - albeit incompletely and/or imperfectly - defined cycle. These deviations are those shown in Figure 6.

Intra-component deviations. Some deviations go from the descriptor of a component to another descriptor of this component. They constitute deviation actions in so far as they go against the planned descriptor order for the definition of the component.

* * *

After this introduction, presenting the conceptual context developed for plan deviation, the different processes underlying plan-deviation proposals will be described.

The processes which have been identified as leading to deviation actions are only a first condition for a plan deviation to occur. They lead to *proposing* actions. The second condition is that the action which has been proposed be *selected*. For this selection, the control uses the criteria presented above. That is, the processes presented below lead to alternative-to-the-planned-action proposals. Only if the proposed action is selected will they have led, albeit indirectly, to a plan deviation.

The processes presented are inferred mostly from observed actual deviations. Processes leading to proposing actions which were not selected afterwards, or processes which could have led to proposing actions, but which did not, were simply not observed: the only possible way of identifying them - other than by observing them - would have been for the engineer to have verbalized them. Thus, the processes presented led in fact, with some rare exceptions, to deviation actions.

Most processes are probably not specific to one type of deviation. However, most deviations go from a function cycle operation to another operation of the same function cycle - that is, they are inter-component intra-cycle deviations -, so most processes have been identified as being this type of deviation, and will be illustrated by examples of it.

3.3.1 A component's mental representation activates another component's representation

The processing of the mental representation of a component Cx in order to define Cx may lead to activate in memory the representation of another component, Cy.

This activation may have several causes, one of which is the type of relationship between the components Cx and Cy[17] (see below). Aside from relationship between the components, the relationship between the types of processing performed, or to be performed, on the components plays also a role: Cy may be activated by the processing of Cx as

- "also requiring definition"
  Example. The engineer starting to define the Second Phase Work function in the General Cycle Plate deviates from his plan by not finishing this definition in order to define - also only partially - the analogous First Phase Work function (see below).

- "having been erroneously defined" or "having been incompletely defined"
  Example.The engineer defining the Fast Advance Movement in the Second Phase Work Cycle interrupts the planned sequence of actions in order to verify the corresponding Fast Advance Movement on the First Phase Work Cycle (already defined, if only partially). He thinks that he has forgotten a starting condition for this First Phase Movement - "starting conditions" being the aspect of the Second Phase Movement that he was defining when he abandoned his plan.

[17] "Relationships between components" is used here as shorthand for "Relationships between mental representations of components."

If the control selects the corresponding deviation-action proposal for defining or fixing Cy, it is because, if the corresponding processing of Cy is done *now*, advantage may be taken of the processing just done on Cx with regard to cognitive cost.

Four different types of relationships among components leading to this type of deviation have been identified:

Analogy. When this type of activation was observed, it involved in nearly all cases work components (either the two Work Functions in the General Cycle or two analogous operations in or between the Work Cycles).

Example. The engineer defining a Tool Compensation operation in the Second Phase Work Cycle returns to the First Phase Work Cycle he has just defined, in order to complete the corresponding Tool Compensation operation. Completion consists of adding the starting conditions analogous to those which he just defined for the Second Phase Tool Compensation operation.

> N.B. The engineer dealt with the First Phase Work Cycle before the Second Phase Work Cycle and often went back to the First Phase from the Second for reasons of analogy.
> The programmer worked in the opposite order (partly because he was guided by an "example" program which presented the corresponding modules in that order). Thus, the programmer wrote sub-modules for the First Phase Workstation taking advantage of the analogies with the corresponding Second Phase sub-module, which he had already written (see Visser, 1987).

Prerequisites. Several times the engineer discovered an operation which he and/or his colleagues had forgotten - generally a prerequisite operation (see §3.2.2) - when he was defining the work operation for which it was a prerequisite.

Example. The definition of the Tooling-by-a-Backward Movement Work operation[18] leads the engineer to discover that he has forgotten two of its prerequisite operations: the Tool Retract Movement and the Tool Compensation.

> N.B. The specific role of prerequisites in information processing was also noted in a study on control engineering students making a functional analysis in a programmable controller program design task. These students modified the analysis method they had learned (based on the Grafcet formalism) in order to adapt it to their mental representation of the functioning they had to analyze. In particular, they did this by processing the aspects of the process directly related to its goal (≈ Work operations) before the process prerequisites (≈ the Prerequisites to the Work operations).
>
> Actions leading directly to the goal were considered right from the first problem-solving stage. Their prerequisites (that is, actions allowing for satisfying the conditions required for these first operations to take place) were handled in a late processing stage, or even completely omitted (see Morais & Visser, 1985).

Interaction. This type of relationship may be based, as in the example above, on a relationship between two physical units in the installation (for example, the turntable interacts with each of the three other stations, since these must be retracted before it can turn). The engineer started to define the Turntable Cycle - and defined it partially - before the other function cycles. When he went back to the Turntable Cycle while defining the other function cycles, it was generally to add one or more descriptors related to the interaction between the two cycles. He generally added ending conditions of the function cycle operation in question to the starting conditions of the first Turntable operation[19], but he also fixed any related errors which he had discovered.

---

[18] See the § Operations forgotten are Prerequisites in section 3.2.2.

[19] See notes 5 and 9.

Example. While defining two Plier Advance operations in the Loading Cycle, the engineer discovers that he has forgotten to include the corresponding Plier Return operation ending conditions among the starting conditions of the first Turntable operation, Release Grip[20].

At the operation level, another type of interaction plays a role. The starting conditions for an operation always comprise one or more of the ending conditions of the preceding operation(s). These two-directional relationships lead to two different deviation patterns. On the one hand, the processing of the starting conditions for an operation has been observed to lead to the discovery - and fixing - of omissions or errors in the ending conditions of the preceding operation. On the other hand, the processing of the ending conditions for an operation can lead the engineer to jump ahead to the following operation, in order to define its starting conditions.

Example. The definition of the starting conditions of the Turntable Fast Movement is interrupted in order to complete the ending conditions of the Turntable Release Grip operation. Later, after several other component processings, the engineer interrupts a Release Grip operation processing in order to reflect a modification of this operation's ending conditions in the Fast Movement starting conditions.

Opposites. When dealing with an operation, the engineer sometimes discovers that he has omitted its opposite operation.

Example. While defining a Tool Return Movement, the engineer discovers that he has forgotten the corresponding Tool Advance Movement.

3.3.2 Defining aspects of a component leads to a local plan for defining the same aspects of other components

The preceding presentation of deviation strategies focused on the relationship between component representations. In this section, an "analogous" strategy leading to plan deviation is presented. The relationship is between the descriptors used to define components. The components to which the deviation leads have no relationship with the currently defined component other than belonging to the same cycle. The strategy consists of processing one or more aspects of a component that are related to the aspects being defined for the current component. It may lead to plan-deviation actions, and leads frequently to several consecutive ones.

Example. This strategy, which is responsible for numerous deviation actions, is the most important source of deviation in the Loading Cycle. The engineer starts the definition of this cycle by defining consecutively the "identifier" and "starting conditions" descriptors of each one the first five operations. After processing some operations according to the plan, for four consecutive operations he defines only the "identifier," "starting conditions," and "ending conditions" descriptors. He then completes several actions as specified by the plan, before defining only the "identifier" and "duration" descriptors for four other operations.
When he actually defines the Loading Cycle - not during the first, failed definition session - this definition is the first and only[21] for which the engineer does not refer to the "example" sequences schema, and for which his most important information source is his knowledge. This could explain the observed strategy. That is, if the engineer were relying mostly on his knowledge, i.e., on his memory, processing would proceed by way of "definition-aspect" correspondence: having defined some aspects of an operation, the engineer would define the corresponding aspects of other operations. But if this way of processing is very "spontaneous" or "natural" for the engineer, what about his plan, in which the processing unit is not an "definition-aspectual" grouping, but the component (in the present case, the operation)?

This type of plan deviation was also observed as a consequence - or a corollary - of taking advantage of available information (see below, §3.3.4). Using an information source for the definition of some aspects of a component leads the engineer to formulate a local plan for using this information source to define the same aspects of other components. This plan is formulated because the information source provides the information needed to define these aspects of several operations, but no others, and it cannot be used to finish the definition of the component currently being processed.

---

[20] The Pliers take the parts and transfer them to the Conveyors.
These conditions are starting conditions for the Turntable, not for its first operation. See Note 9.

[21] The Conveyor Cycle will also be defined without the use of another Conveyor Cycle representation, but only during Grafcet construction.

Example1. This data-driven definition strategy of moving from the information in the consulted information source to the components that may be defined by it may have led the engineer to define the two work functions of the General Cycle consecutively by first taking advantage of his working memory to define the "duration" descriptor for both and then using the tool plates to define the "identifier" descriptor for both.

Example2. While in the plan deviation in Example1 a relationship of analogy between both functions may also have played a role, this possibility is absent in the following deviation.
For the first Turntable Cycle component processings, the engineer used a document representing the operations involved and their duration. Thus, the engineer first defined all the Turntable operation "identifier" and "duration" descriptors before looking elsewhere for the information needed to define the other descriptors of each operation.

3.3.3 Processing information from different points of view

Information used to define a component may be processed from a point of view that makes it useful in defining another component.

For example, information used for functional definition may be considered from its mechanical (physical) point of view.

Example1. In defining the starting conditions of the first operation of the Turntable Cycle, the engineer consults a mechanical specifications document that included information on the station's electrical detectors (the activation of which may constitute operation starting - or ending - conditions). In processing this information, the engineer comes to question the mechanical safety of the Turntable: he is not sure that the Turntable will not damage other stations during its rotational movement. So he interrupts his cycle definition - he even interrupts his functional specification activity - to verify the mechanical specifications, and thus to proceed to a completely different task, that is, mechanical design.

Example2. In looking for the ending conditions of the loading function, the engineer discovers that not all of the detectors required to detect the ending conditions (that is, that the operation is finished) are specified in the mechanical specifications document. This leads him to conclude that the operation is not completely controlled on the installation as designed, and he interrupts his functional specifications writing task, that is, he abandons the current definition processing of the loading function in order to complete the mechanical design task.

Another example of this type of information processing is the interpretation of information which allows one component descriptor to be defined as also useful in defining another descriptor. In the observed definition activity, this occurred only for the starting and ending condition descriptors (see §3.3.1, section Interaction).

Example. Having retrieved the information needed to define the ending conditions of an On operation, the engineer also frequently uses it to define the starting conditions of On+1 (or vice versa).

3.3.4 Taking advantage of available information

Acting according to a plan is a concept-driven activity. Starting from a goal imposed by the plan, the engineer will look for the information needed to achieve this goal. However, most deviations observed stem from data-driven processing, such as starting from information which the engineer has at his disposal and which allows a goal not imposed by the plan to be achieved.

In addition to information being there because it is being used for the current component definition (see above regarding possible deviational use in this case), it may also be available because the engineer is "presented" with the relevant information source. Some examples of such "information presentation" are the following:

- the client communicates modified requirements to the engineer
  Example. There is no example of the client changing the requirements during construction of the sequences schema. A few were observed during construction of the Grafcet (see Morais, 1987).
- a colleague presents him with new information, especially in relation to the mechanical specifications
  Example. When the engineer consults a colleague about the use of three detectors - shown in a mechanical specifications document - in obtaining the ending conditions of an operation, he learns that one of the three is no

longer needed. So the engineer uses only the other two. This leads him to verify the operations in which these detectors had already been used in some aspect of the definition and to fix the errors identified[22].

- a colleague, specialist in another domain, comments on the current problem solution state attained by the engineer

  Example. At a rather advanced stage of the functional specification, the engineer learns from a design colleague - an electrical engineer - that, if one uses a particular type of detector, one must not only control the presence of the operation for which it is used (as the engineer had done) BUT ALSO control for the absence of the corresponding opposite operation (something the engineer had not done). This leads him to verify the previously defined operations controlled by this type of detector and to add the missing ending conditions.

- the information source the engineer consults comprises modifications which "stick out a mile"

  Example. Most information sources used by the engineer are also used by colleagues. Thus, modifications may have been made from one consultation to another. The tool plates, which are worked on and consulted by various persons, are a case in point. In addition, the engineer consults only a rather restricted area of them, in which the information is very important for his task, and the engineer knows some of the values inside out.
  In consulting the tool plates for the definition of the General Cycle, the engineer notices that the durations of the work functions have been changed. These are "obvious" modifications for him, because they concern this group of values which he knows inside out. The engineer interrupts his current processing in order to redefine the two work functions to reflect these new values.

3.3.5 Drifting (during a "difficult" specification action)

"Drifting" - that is involuntary[23] attention switching to a processing other than the current one - was observed to occur especially when the engineer was involved in a "difficult" action, that is, in general, trying to determine the value of one of the definitional aspects of a component (see §3.3, section "Selection of a plan-deviation action rather than the planned action").

Example. The engineer, involved in his definition of the Turntable Fast Movement - the ending conditions of which he finds hard to define - is observed to interrupt and to define the ending conditions for the following operation, remarking: "There, I know what to put."
Without this remark, the observed deviation could have been explained as due to "Postponing an action which costs too much - if executed now" (a process presented below, §3.3.6).

One might think that during a "difficult" definition action, all attention would be focused on the current problem. Perhaps drifting, under these conditions, may be explained by the possibility that, in looking in various directions for possible solution elements, the engineer comes upon information which is "obviously" applicable to another component definition (see above, §3.3.4). An hypothesis inspired by the observations is that these drifting-caused deviations occur especially during information retrieval for problem solving that is not guided by strict information searching rules.

Example. In the above example, the engineer was observed to "think and hesitate" ("What shall I put there? ... I don't know"), rather than to wonder which information was required to define the ending conditions and which information source(s) should be consulted.

3.3.6 Postponing actions which cost too much - if executed now

The cognitive cost of each action is evaluated before (possible) execution. Frequently, an important alternative action is "cheaper" than the planned action, and therefore selected. This is how plan deviation generally occurs. But sometimes a planned action is skipped because it costs too much, not compared to another action, but to the action itself if executed later. Not only is another action then selected, but a local plan is formed for reproposing the postponed action as soon as the conditions leading to its postponement no longer prevail.

Example. The definition of the Loading Cycle was postponed, because the engineer did not understand the way it had been broken down in the example he used for the specification of the installation (see §3.2.1).

---

[22] It is not appropriate here to comment on the communication problems - especially the absence of communication - among the various persons involved in the same design project. We only point it out.

[23] One may also think that it is "not yet explained" attention switching.

A planned action may also cost too much because the required information is unavailable (see §3.3, section "Selection of a plan-deviation action rather than the planned action"). This may be the case, for example, because

- the client has not yet made a decision required to define the component in question (and the engineer thinks that he cannot make this decision)
  Example. This is the reason why the engineer did not define the Conveyor Cycles until he was constructing the Grafcet.
- the information source which has the relevant information (mostly a colleague) is absent
  Example. The definition of the ending conditions of an operation controlled by a hydraulic control detector is skipped because the hydraulic specialist is absent.

* * *

The preceding section - presenting processes leading to alternative-to-the planned-action proposals - focused on the control component in a blackboard system. The structure of the blackboard and the knowledge sources were only alluded to. To study these points, the structure of the engineer's knowledge representation would need to be examined. Some possible directions have been sketched, as, for example, in the observation that deviations may go from component to component by way of relationships

- between their respective representations (analogy, prerequisites, etc.)
- between the types of definitional processing they may undergo (definition of their descriptors).

Such results suggest representational units with links existing between them along several dimensions, but this point needs to be examined more specifically.

4. CONCLUSION

The results presented show the activity involved in specification writing to be an opportunistically organized activity. It is a typical design activity. For example,

- the activity does not follow a pre-existing plan;
- the problem to be solved is not completely and immutably defined from the outset: an important part of the engineer's activity consists of constraining this problem;
- the solution to be obtained is not unique: the "final" specifications arrived at by the engineer may be modified, not because they are "incorrect," but because an alternative design may be chosen by another designer, using other criteria.
  Example. The person in charge of debugging and testing the control part of the program modified the manual operation mode specifications in order to give the installation "a more flexible manual use."

The opportunistic organization of the activity was the result on which this paper focused. The engineer had a hierarchically structured plan for his activity, but he used it in an opportunistic way. He used it only as long as it was profitable from the point of cognitive cost. If more cognitive economical actions arose, he abandoned it.

Other design studies noted the opportunistic aspects of the design activity (see Bisseret, Figeac-Létang & Falzon, 1988; Guindon, Krasner & Curtis, 1987; Kant, 1985; Ullman, Staufer & Dietterich, 1987; Visser, 1988a, 1988b; Whitefield, 1986). The idea developed in this study is that, at the control level of his activity, an expert designer may use a plan for guiding his activity, but that deviations of this plan occur if the control selects an alternative-to-the-planned-action proposal rather than the planned-action proposal. Several processes leading to these proposals were described, and the two main criteria the control uses in selecting among them were introduced.

Implications for design assistance tools (CAD). Most design studies are conducted not only to model the activity, but also consider as an important goal the specification of assistance tools (see Adelson & Soloway, 1985; Guindon & Curtis, 1988; Ullman, Staufer & Dietterich, 1978). This study would lead, in this regard, to the following conclusion. If the design activity is opportunistically organized, a system which supposes - and therefore imposes - a hierarchically structured design process will at the very least constrain the designer and will probably even handicap him (see Visser & Hoc, to be published). Hoc (1988a), in an evaluation study of a programming environment supporting top-down processing, showed that professional programmers - trained in the underlying structured,

top-down programming method - experienced difficulties due to the processing imposed by the environment, and generated non-optimal solutions.

The results presented here constitute a strong argument for tools that allow the problem solution to be abandoned at a certain level, in order to process solution elements at another level, and to possibly later resume the solution state at the abandoned level. However, such tools should not *impose* such a resumption either. The engineer observed had - and used - a hierarchically structured plan, to which he returned after deviation actions. Not all designers necessarily refer to such a plan - nor do all design activities, especially if their results have a less constrained structure than a sequences schema.

5. REFERENCES

Adelson, B. & Soloway, E. The role of domain experience in software design. IEEE Transactions on Software Engineering, 1985, SE-11, 1351-1360.

Anderson, J. R. Knowledge compilation: the general learning mechanism. In R. S. Michalski, J. G. Carbonell & T. M. Mitchell (Eds.), Machine learning. An artificial intelligence approach (Vol. II). Los Altos, Calif.: Morgan Kaufmann, 1986.

Bisseret, A., Figeac-Létang, C., & Falzon, P. Modeling opportunistic reasonings: the cognitive activity of traffic signal setting technicians (Research Report N° 898). Rocquencourt: INRIA, 1988.

Détienne, F. Program understanding and knowledge organization: the influence of acquired schemata. In P. Falzon (Ed.), Cognitive ergonomics: learning and designing HCI. London: Academic Press, to be published.

Eastman, C. M. Cognitive processes and ill-defined problems: A case study from design. Proceedings of the First Joint International Conference on Artificial Intelligence. Washington, D.C., 1969.

Ericsson, K. A., & Simon, H. A. Protocol analysis. Verbal reports as data. Cambridge, Mass.: MIT Press, 1984.

Guindon, R., & Curtis, B. Control of cognitive processes during software design: What tools are needed? In E. Soloway, D. Frye, & S. S. Sheppard (Eds.), CHI'88 Conference Proceedings. Reading, MA: Addison Wesley,1988.

Guindon, R., Krasner, H., & Curtis, B. Breakdowns and processes during the early activities of software design by professionals. In G. Olson, S. Sheppard & E. Soloway (Eds.), Empirical Studies of Programmers: Second Workshop. Norwood, N.J.: Ablex, 1987.

Hayes, J. R., & Flower, L. S. Identifying the organization of writing processes. In L. W. Gregg, & E. R. Steinberg (Eds.), Cognitive processes in writing. Hillsdale, N.J.: Erlbaum, 1980.

Hayes-Roth, B. & Hayes-Roth, F. A cognitive model of planning. Cognitive Science, 1979, 3, 275-310.

Hoc, J. M. Towards effective computer aids to planning in computer programming. Theoretical concern and empirical evidence drawn from assessment of a prototype. In G. C. van der Veer, T. R. G. Green, J. M. Hoc, & D. Murray (Eds.), Working with computers: theory versus outcomes. London: Academic Press, 1988a.

Hoc, J.M. Cognitive psychology of planning. London: Academic Press, 1988b.

Jeffries, R., Turner, A. A., Polson, P. G. & Atwood, M. E. The processes involved in designing software. In J.R. Anderson (Ed.), Cognitive skills and their acquisition. Hillsdale, N.J.: Erlbaum, 1981.

Kant, E. Understanding and automating algorithm design. IEEE Transactions on Software Engineering, 1985, SE-11, 1361-1374.

Letovsky, S., Pinto, J., Lampert, R., & Soloway, E. A cognitive analysis of a code inspection. In G. Olson, S. Sheppard & E. Soloway (Eds.), Empirical Studies of Programmers: Second Workshop. Norwood, N.J.: Ablex, 1987.

Malhotra, A., Thomas, J. C., Carroll, J. M. & Miller, L. A. Cognitive processes in design. International Journal of Man-Machine Studies, 1980, 12, 119-140.

Morais, A. Hierarchical planification in programming. In I. D. Brown, R. Goldsmith, K. Coombes, & M. A. Sinclair (Eds.), Ergonomics International 85. Proceedings of the Ninth Congress of the International Ergonomics Association, Bournemouth, England, 2-6 September 1985. London: Taylor & Francis, 1985.

Morais, A. Acquisition d'un outil de spécification (GRAFCET) pour décrire un procédé automatisé (Rapport de Recherche INRIA n°631). Rocquencourt: INRIA, 1987.

Morais, A., & Visser, W. Etude exploratoire de la programmation d'automates industriels chez des élèves de l'enseignement technique (Rapport de Recherche INRIA n°404). Rocquencourt: INRIA, 1985.

Newell, A., & Simon, H. A. Human problem solving. Englewood Cliffs, N.J.: Prentice-Hall, 1972.

Nii, H. P. Blackboard systems: the blackboard model of problem solving and the evolution of blackboard architectures. Part One. The AI Magazine, 1986, Summer, 38-53.

Ratcliff, B., & Siddiqi, J. I. A. An empirical investigation into problem decomposition strategies used in program design. International Journal of Man-Machine Studies, 1985, 22, 77-90.

Rumelhart, D. Schemata: the building blocks of cognition. In R. Spiro, B. Bruce, & W. Brewer (Eds.), Theoretical issues in reading comprehension. Hillsdale, N.J.: Erlbaum, 1978.

Sacerdoti, E. D. Planning in a hierarchy of abstraction spaces. Artificial Intelligence, 1974, 5, 115-135.

Ullman, D., Staufer, L. A., & Dietterich, T. G. Toward expert CAD. Computers in Mechanical Engineering, 1987, 6, 56-70.

Visser, W. Strategies in programming programmable controllers: a field study on a professional programmer. In G. Olson, S. Sheppard & E. Soloway (Eds.), Empirical Studies of Programmers: Second Workshop. Norwood, N.J.: Ablex, 1987.

Visser, W. Giving up a hierarchical plan in a design activity (Research Report N° 814). Rocquencourt: INRIA, 1988a.

Visser, W. Towards modelling the activity of design: an observational study on a specification stage. Proceedings of the IFAC/IFIP/IEA/IFORS Conference Man-Machine Systems. Analysis, Design and Evaluation (Oulu, Finland, 14-16 June 1988). IFAC, 1988b.

Visser, W., & Hoc, J.M. Expert software design strategies. In J.M. Hoc, T. Green, R. Samurçay & D. Gilmore (Eds.), Psychology of programming (pp. 235-250). London: Academic Press, 1990.

Whitefield, A. An analysis and comparison of knowledge use in designing with and without CAD. In A. Smith (Ed.), Knowledge engineering and computer modelling in CAD. Proceedings of CAD86. Seventh International Conference on the Computer as a Design Tool. London: Butterworths, 1986.