Giving up a hierarchical plan in a design activity

Willemien Visser

INRIA

UNITÉ DE RECHERCHE
INRIA-ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél.:(1) 39 63 55 11

Rapports de Recherche

N° 814

GIVING UP A HIERARCHICAL PLAN IN A DESIGN ACTIVITY

Willemien VISSER

MARS 1988

RR814

GIVING UP A HIERARCHICAL PLAN IN A DESIGN ACTIVITY

ABANDON D'UN PLAN HIERARCHIQUE DANS UNE ACTIVITE DE CONCEPTION

Willemien VISSER

This text is an english, slightly revised version of a paper, presented in french, at the COGNITIVA87 conference (Abandon d'un plan hiérarchique dans une activité de conception. Actes du colloque scientifique COGNITIVA 87 (Tome 1). Paris: Cesta, 1987)

RESUME

Des observations ont été conduites sur la conception d'une machine-outil.

L'étude se distingue de précédentes études empiriques notamment sur les points suivants:

* Nous avons observé en temps réel des concepteurs pendant leur activité de travail habituel.

* Nous avons étudié plusieurs étapes dans le processus de conception d'un projet réel complexe, impliquant différents opérateurs.

Les résultats présentés sont centrés sur l'éventuel caractère hiérarchique de l'activité de conception.

1. Chaque opérateur décrit son activité sous la forme d'un plan structuré hiérarchiquement. Il décompose son activité en composants de niveau différent qui seront traités dans un certain ordre. Ce plan ne traduit cependant pas l'activité réelle de l'opérateur. Celle-ci est plutôt opportuniste. La hiérarchie du plan est abandonnée. Des retours en arrière et des anticipations complètent les composants de conception, mais y apportent également des modifications qui vont de la correction d'erreur à la remise en cause de décisions.

Nos résultats confirment ceux d'études antérieures conduites sur des tâches de conception simplifiées. Ils comportent cependant aussi un aspect nouveau: nous montrons que ces traitements qui caractérisent la conception ne s'appliquent pas seulement à l'activité de chaque opérateur dans son étape de conception, mais au processus de conception dans son ensemble. Ainsi, les spécifications établies antérieurement, par exemple, sont complétées et modifiées également.

2. Nous avons analysé les facteurs qui conduisent à ces abandons du plan structuré hiérarchiquement. Les principaux processus psychologiques qui en ressortent sont présentés. Il s'agit de processus bien connus dans d'autres contextes de résolution de problème ou de diagnostic.

D'une part, ces résultats permettent alors de préciser dans quelle mesure l'activité de conception se rapproche et se différencie de ces autres activités.

D'autre part, la connaissance de ces processus et de leurs conditions spécifiques de déclenchement nous fournit des renseignements précieux pour la modélisation de l'activité de conception.

Ces résultats sont illustrés, à l'aide d'exemples.

MOTS CLES

Activité de conception; Processus de conception; Planification; Organisation opportuniste de l'activité; Observations de l'activité; Etude de cas; Etude de terrain; Machine-outil automatisée

SUMMARY

An observational study was conducted on the design of a machine tool installation. The notable differences between this study and previous empirical research concern the following points :

* We carried out real-time observations on designers during their real work activity.

* We studied several stages of the design process of a real, complex project involving three different operators.

The results presented focus on the possible hierarchical nature of design activity.

1. Each operator describes his activity in the form of a hierarchically structured plan. He breaks his activity down into components of different levels which must be dealt with in a specific order. This plan, however, does not represent the operator's real activity which tends, in fact, to be opportunistic in nature.

The hierarchy of the plan is given up as (a) components are abandoned before they are completed, (b) addenda and modifications are made not only on components that have already been dealt with, but also on those that have yet to be dealt with. These modifications range from correcting errors to challenging design decisions.

Although our results concur with those previously obtained from laboratory studies carried out on simplified design tasks, they do introduce a new aspect in that they not only apply to each individual operator's activity in his stage of the design, but cover the design process as a whole. Thus, for instance, specifications that were established during one stage, can be completed and modified by another operator at a later stage.

2. We analyzed the factors that led to these deviations from a hierarchically structured plan. The main psychological processes involved are presented. They concern processes that are well-known to be used in other problem-solving or diagnostic contexts.

Thus, these results firstly make it possible to specify the degree to which design activity is similar to, or different from, these other activites.

Secondly, knowledge about these processes and the specific conditions that trigger them can provide us with valuable information for modeling the design activity.

The results are illustrated by examples.

KEY-WORDS

Design activity; Design process; Planning; Opportunistic organization of activity; Observational study; Field study; Protocol analysis; Automated machine tool installations

1. THEORETICAL MODELS AND REAL DESIGN ACTIVITY 1
2. METHOD 3
3. RESULTS 3
3.1 The plan of the activity as described by the operator 3
3.2 The real activity as it was observed 4
3.2.1 Returning to components within each of the stages 4
3.2.2 Returning to components from a previous stage in the process 6
3.3 Psychological processes that lead to the deviation from the hierachically structured plan 7
3.3.1 Processes leading to the completion of a component 7
3.3.2 Processes leading to the completion and/or the modification of a component 9
3.3.3 Processes leading to the modification of a component 10
4. CONCLUSION 11
REFERENCES 12

1. THEORETICAL MODELS AND REAL DESIGN ACTIVITY

When the heads of design departments are asked to describe the design process in their company, they generally provide a list of clearly separated, consecutive phases in each of which one or more different persons intervene.

Table 1 presents the description we received from the head of the design department in a software development company.

Table 1. The theoretical phases of the design process in a software development company

specification of requirements
preliminary design
detailed design
coding
testing
integration
reception

Each phase should lead to the production of one or more documents. Those produced by the people working in one phase constitute a state of the project which, for their colleagues in the following phase, should no longer be open to question.

The design process covered by this kind of description appears similar to the process advocated or laid down by design methodologies. Thus, in MERISE, a French method for designing and developing information systems, the life cycle (from the initial design to the maintenance) of such systems "is characterized by several phases that are necessarily situated on a temporal dimension" (Tardieu, Rochfeld & Colletti, 1986, p.38). Moreover, "the use of abstraction levels is absolutely necessary in order to ... consider only one class of problem at each level" (id., p.41). For a critical review of such methods, see Carroll & Rosson, 1985; Robert, 1979).

However, design activity, as it has been observed in empirical studies on software design, is quite different from the one presented in these normative or descriptive theoretical models. It appears to be non-hierarchical (processing is neither completely top-down, nor totally bottom-up), it proceeds in a cyclical manner, introducing new goals as it goes along (see the reviews cited above).

The empirical studies on which these conclusions are based however were all carried out in an artificially limited context which was nearly always the

psychological laboratory. Even if some 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 whereas, in reality, design is characterized by people working together with a greater or lesser dependence on the work done and information provided by colleagues. So in order to model real design activity, empirical studies of this activity must be conducted on designers working on real, and therefore complex projects.

Compared with laboratory studies, this kind of study is however much more time consuming and the data obtained can be more difficult to process due to the number of factors that cannot be eliminated, as is done in a laboratory study, during the observation.

This paper presents a longitudinal study that was carried out on a project to design a computer-controlled automated industrial plant.

Given the existing division of labour, we were able to study several consecutive steps in this design process. As in all undertakings of this type, each step is theoretically supposed to be successive and independent of the others. Now, following on from the studies referred to above, we aimed to find out how far the individual adapts to the hierarchical process that is imposed on him by the division of labour, and/or how far he oversteps the boundaries between these stages.

As the study presented here was carried out in the framework of research on programming (see Visser, 1985, 1987), we observed the steps that we considered the most relevant from this point of view. Besides the step which concerned programming as such, we also observed the stages in the design process that preceded and followed it immediately. These three steps were thus :

1. the construction of a functional representation of the operating part of the installation. This work was done by a mechanical engineer, and the scheme that he produced constitutes the main document used to provide the specifications for designing the program to control the installation;

2. the writing of this program by a specialist in electronics and automation (the "programmer");

3. the debugging and testing of this program by another specialist in electronics and automation.

The notable differences between previous empirical studies and our study are that

* we carried out real-time observations on designers in the course of their usual work

* we studied several steps in the design process for a real, complex project involving several operators.

2. METHOD

For a total of thirteen weeks, we carried out observations in a factory which produces machine tools going to be controlled by programmable controllers.

Each of the operators was observed throughout the duration of his work, and we asked them to proceed as usual, with one difference : we requested them to "think aloud", that is to verbalize as much as possible their thoughts about what they were doing.

In order to obtain data on the operators' activity, we noted :

- all that the operator said and wrote down;

- the order in which the different documents were produced, how they were composed, and their subsequent modifications;

- the nature of any information used;

- the events which we considered to be indications of the operator meeting with difficulty.

The operator who had to test and debug the program not only produced documents, but also worked on the physical installation, and we made notes about these actions.

As well as making these notes, we also gathered all material traces of the operators' activity :

- the different versions of the documents that they produced;

- the rough drafts of the documents;

- the diagrams and schemes that the operators made for themselves.

3. RESULTS

From all the results we obtained, we present here those which focus on a particular aspect of the design activity, namely its possible hierarchical nature.

We shall see that, although the operator describes his activity as being hierarchically structured, his real activity does not, in fact, follow the plan he describes. We present below the main psychological processes that lead to this deviation from this plan.

3.1 The plan of the activity as described by the operator

We asked each of the operators to describe his activity. The programmer was requested to describe his activity before carrying it out, and the two other operators were asked to describe their activity both beforehand and afterwards.

In each case the operator described his activity in the form of a hierarchically structured plan[1] containing a greater or lesser number of levels.

We present here, as an example, the plan given by the mechanical engineer (see Table 2, p. 5).

The succession of goals and actions is presented in the order of the description given by the operator. The planning he describes is top-down and depth-first.

For the reasons given below, this plan does not represent the mechanical engineer's real activity. The same is true for both of the other operators : the plans they produce do not conform to their real activity. In general terms, the operators do not follow a systematic path through the tree representing the plan of their activity as they describe it

3.2 The real activity as it was observed

3.2.1 Returning to components within each of the stages

The plan described by the operators enables us to break down the problem that he faces. We shall call each element of this breakdown a "component."

For the mechanical engineer, a component is thus a cycle or an element of a cycle (a function or an operation).

For the programmer, a component may be a module of the program, a sub-module, an instruction or a branch of an instruction.

For the operator who must test the program and its functioning, the components are the "units" of the installation in different modes of functioning (manual, semi-automatic, or automatic). For this operator, a "unit" can range from an operation, the machine's workstations, to the machine as a whole.

We observed that each operator returns several times to each of the different components before considering them to be completed. He returns to them after varying intervals of time, during which he deals with other components.

Thus, a component is not dealt with as a whole, nor is it complete when work on the next component begins (the elements of order we refer to come from those given by each operator in his plan).

These two aspects of the operator's deviating from the hierarchically structured plan make it possible to identify two reasons for his returning to a component : either to build it up gradually by adding new elements, or else to modify it by changing one or more of its elements.

---

[1] This plan is called "hierarchically structured" and not "hierarchical", in order to avoid confusion with the "hierarchical planning" as described by Sacerdoti (1974).

Table 2. Schematic representation of the description of his activity as given by the mechanical engineer

+ Construct a functional representation
  + Describe the General Cycle
    + Determine its first function
      + Describe the first function
        + Determine its descriptors
          + Determine its identifier (id)
            Insert its id in its column
          + Determine its duration (du)
            Insert its du in its column
        + Determine its starting conditions (st)
          Insert its st in their column
        + Determine its ending conditions (en)
          Insert its en in their column
    + Determine its second function
      + Describe the second function
        + Determine its descriptors
          ...
    + Determine its nth function
      + Describe its nth function
        ...
  + Describe the Cycle of the first function
    + Determine its first operation
      + Describe the first operation
        + Determine its descriptors
          + Determine its identifier (id)
            Insert its id in its column
          + Determine its duration (du)
            Insert its du in its column
          + Determine its starting conditions (st)
            Insert its st in their column
          + Determine its ending conditions (en)
            Insert its en in their column
          + Determine its physical device (ph)
            Insert its ph in its column
    + Determine its second operation
      + Describe the second operation
        + Determine its descriptors
          ...
    + Determine its mth operation
      + Describe its mth operation
        ...
  + Describe the Cycle of the second function
    ...
  + Describe the Cycle of the nth function
    ...

Key : + precedes a goal to be realized
(the lines not preceded by + describe the actions carried out in order to realize the immediately preceding goal)

... replaces goals whose breakdown follows that given above for a similar goal

Returning to a component in order to complete it. At first, generally, a component is only dealt with partially, and it is not until later, after the operator has returned to it several times, that it is completed. For example, the mechanical engineer returns, on average, five times to one cycle after he has begun to describe another cycle. He does so mainly to complete the information that specifies the start of the cycle (the descriptor "starting conditions", see Table 2).

For the construction of certain instructions, the programmer returns to it some ten times, each time making additions and/or modifications.

An example are the "Auxiliaries" *Forward* and *Return* for the workstations (i.e. the instructions that bring together the conditions necessary for the Forward or Return movements in the different modes of the installation's functioning).

Returning to a component in order to modify it. Such modifications can vary from correcting an error in an element, to challenging a decision that was previously taken by the operator about a component. (The challenging of a decision previously taken by another operator is discussed below in §3.2.2.)

Example : Firstly the programmer defines a grip function for each workstation - as is the case for most installations of this type (see §3.3.3 *Reading guided by the use of a schema with prototypical values for certain variables*).

Later, when he returns to these specifications, he discovers that there is only one grip function for both stations. Therefore he takes out the two sub-modules for the grip function for each of the modules covering a workstation, and from one of these he creates a single module for the grip function on both workstations.

3.2.2 Returning to components from a previous stage in the process

As this study covered several stages in the design process, we were able to observe that each operator not only returns to components from his own stage in the process, but also to components from previous stages which are supposed to have been completed by other operators.

This can be done for the same two reasons which we saw before.

Returning to a component in order to complete it. An example of this is the introduction by the mechanical engineer, into the specifications that he makes, of operations that have not yet been specified. Only prerequisite operations had been forgotten in previous stages : each of the operations added by the mechanical engineer (11 out of a total of 26) constitutes a prerequisite for an operation already specified.

The mechanical engineer often only notices a prerequisite operation when dealing with the operation of which it is a prerequisite, an opposite operation, or a similar operation (see §3.3.2 below). In other words, in his stage, he neither deals with these operations at the moment when, according to his plan, he should have done.

Returning to a component in order to modify it. An interesting example of returning to a previous step in the design process for making a modification is the following : the programmer considers that a certain function as specified by the mechanical engineer is impossible to realize given the electronic means planned for the installation. Therefore he modifies the specifications accordingly : he not only programs the solution, but also modifies the functional representation constructed by the mechanical engineer.

3.3 Psychological processes that lead to the deviation from the hierachically structured plan

Using examples, we have presented four types of deviation from the hierarchically structured plan that the operators use to describe their activity. Although these deviations make the plan of the real activity opportunistic in so far as the activity is guided by circumstances, this does not mean that it is haphazard.

We analyzed the factors which led the operators to leave one component in order to return to another, and we present below the main psychological processes involved.

We distinguish processes leading to completion of a component, processes leading to modification of a component, and processes used in both contexts.

3.3.1 Processes leading to the completion of a component

The economic use of available means. Among the external and internal sources of information, as well as his own cognitive resources, which the operator has, in principle, at his disposal, not all are equally accessible. The economic use of available means leads to :

- delay in dealing with (the elements of) a component

  * when the operator does not have the source of information that he needs at hand (or when he has on hand the source of information that is useful for dealing with another component - see the example below concerning *Dealing with components when guided by the structure of the source of information available*). Thus the programmer skips certain sub-modules whose treatment requires information from the mechanical engineer, and he makes up a list of questions for the mechanical engineer. He may leave a blank in an instruction, considering that the information to be inserted may be found later at a more appropriate time. Thus, he is aware that he will have to fill in a variable, and in such cases he generally knows certain of the variable's characteristics which he may sometimes mention above the blank in the form of a comment.

  * when the operator considers that dealing with the component will be easier after having dealt with other components (when, for instance, he knows more about the interactions with the components in question).

Thus the mechanical engineer at first skips the *Loading cycle* as he considers it too difficult to deal with at the time when, according to his plan, he should describe it.

- dealing with components guided by the structure of the source of information available. Example : When dealing with a certain operation, the mechanical engineer takes a document which provides the elements that allow him to determine the first two descriptors for this operation (its "identifier" and its "duration"). This document also gives the elements which make it possible to determine both of these descriptors for two other operations in this cycle. The operator therefore takes advantage of this information to determine these two descriptors for all three operations. If he had followed his plan, he would only have determined them for the first of these operations before moving on to the next one, after which he would have determined the others by some other means.

We also include in this category, the drifting made by the third operator (who must test and debug the program) when he decides which test to carry out according to the structure of the problems that he faces rather than according to the structure of his plan.

It is impossible for this operator to check the installation's behavior in all its possible states. Certain tests are carried out as a matter of course, but many are only made when a problem arises during the testing of another state.

Thus the order in which tests are made largely depends on the problems which the operator meets, rather than on the order that he had previously given in his plan.

Example : In his plan the operator says that he will check the *Forward command* well before testing the movements of the tools. At a certain moment he tests the *Forward command* for the first phase workstation. It does not work. As his hypothesis (after examining different instructions) is that a certain tool is responsible, he begins by checking this tool.

This example is somewhat simplified as drifting may lead the operator to test four or five components (which he would not otherwise have done), each test being imposed by the problems encountered during the preceding one.

Dealing with a later component in advance (we could have equally included this process in the previous category, see the example given above). Example : The programmer "thinks of" a similar component (which according to his plan should be dealt with later), or a component which he thinks he might forget if he does not deal with it at once, and he therefore leaves the component he is working on in order to deal with the component he thought of.

Another type of anticipation (but which does not concern an element of his plan) is the programmer's use of variables which combine a certain amount of information (intermediate bits) before their definition has been made.

3.3.2 Processes leading to the completion and/or the modification of a component

The identification of a certain type of relationship between the component being dealt with and the component the operator returns to. This relationship may be :

- a relation of analogy or correspondence (certain modalities of this process may be included in the category of *Economic use of available means*). The installation has two workstations. When the operator is dealing with a component for one of these stations, he often returns to, or anticipates the design of the corresponding component in the other station.

The mechanical engineer deals with the first phase workstation cycle before the second phase workstation cycle, and he often returns, for reasons of analogy, from the second to the first phase. The programmer works in the opposite direction (partly because he is guided by an "example" program which gives the corresponding modules in that order). Thus the programmer writes the sub-module for the grip function on the first phase workstation using as an example the sub-module for the second phase workstation, which he has already written.

- a relation of interaction. This may be based, as in the example above, on a relationship between two physical units in the installation (the turntable interacts with each of the three other stations as these must be retracted before it can turn). The mechanical engineer partially describes the turntable cycle before the cycles for the other units. When dealing with the cycles of these stations, he often goes back in order to introduce information on the turntable cycle.

- a relation of opposites. The mechanical engineer discovers omissions in operations when dealing with their opposite operation. The operator who must debug the program checks to see if an operation that he has already tested manually can also be made in the opposite direction.

- a relation of prerequisites. The mechanical engineer discovers an omission (made by himself and his colleagues) in an operation when dealing with that operation of which it is a prerequisite. We have already stated the specific role of prerequisites in information processing in another study which we carried out on student automaticians (see Morais & Visser, 1985).

The testing of components already dealt with sometimes leads both the mechanical engineer and the programmer to return to an earlier step (the testing of the program by the third operator is presented separately below). The mechanical engineer, for instance, may interrupt work on one work cycle in order to check the calculations of the operations' durations whose values are given in the specification documents that he has received. The programmer uses intermediate bits before having defined them. Later, once he has defined them, he returns to check that he has used them correctly in the instructions (see above *Dealing with a later component in advance*).

The operator who must test and debug the program proceeds from mental checks (by reading the instructions) to checks made by observing the functioning (during physical tests on the machine which involve generally pushing a button on a control panel and observing the installation's subsequent behavior).

Simulation is a strategy that may be used by the programmer when, from time to time, he interrupts his writing of the components to check what he has already written. It is frequently employed by the operator who must test and debug the program.

This operator simulates the installation's functioning when, for example, he reads a (sub-)module of the program for the first time. This can help to detect errors or omissions (see Détienne, 1984).

An interesting example, from the point of view of this article, is that of the the last operator's modifying the specifications of the manual mode such as they were written by the programmer in his program. Thus the modifies the program after simulating the installation's future use both mentally (by reading the program) and through the action of making the machine work. He modifies the specifications in this way in order to give the installation the most flexible (manual) use possible (see below *Changing the criteria used*).

The formulation and testing of hypotheses is a basic strategy used by the third operator to check the functions. We consider each physical test on the machine (see above) to be the testing of a hypothesis. Even if it is not explicit (either to us or to himself), the operator has a hypothesis on the installation's behavior, which leads to the test being made. If the test leads to the expected behavior, the operator generally moves on to the next point, i.e. the testing of the next function or operation. If not, he begins to diagnose the problem he faces.

3.3.3 Processes leading to the modification of a component

Changing the criteria used leads the operator to question decisions that have been taken previously either by himself or by his colleagues.

Modifying the duration of an operation by the mechanical engineer is often an example of this. This modification may concern the design as defined by the mechanical engineer himself, or by another operator (see the example given above in *The testing of components already dealt with*). In both cases, the mechanical engineer may prefer, for example, to attribute a "safer" and therefore longer duration to the operation that has already been dealt with. A longer value is "safer" for the following reason.

A major constraint in this kind of automated machine tool design projects is the maximum duration of the cycles. For the mechanical engineer, an important criterion for the duration that he attributes to the operations is thus the total duration of the cycle in question. He regularly checks that this limit is not exceeded, by adding together the durations of the operations that make up the cycle he is dealing with. A longer cycle is therefore "safer" as, if with longer values the operations do not exceed the time limit for the

cycle, the operator can be virtually certain that they will hold good when the machine is produced.

Homogenizing the structure of several components is to a large extent applied by the programmer. He aims to create a uniform structure at several levels of the program :

- the order of instructions in the modules;

- the order of the branches in the instructions;

- the order of the bits in the branches of the instructions;

- the numbers of certain bits which cover the variables found on several stations (if, for example, he attributes B613, B614 and B615 to three variables on one station, he will attribute B713, B714 and B715 to identical variables on another station, and B813, B814 and B815 on a third one).

To achieve this, he regularly returns to the instructions that have already been written :

- to make them homogeneous with the instruction that he is writing;

- to make the instruction that he is writing homogeneous with them.

Moreover, the programmer tries to make the structure of the program that he is writing as homogeneous as possible with the program that he is using as an example. This sometimes leads him to reorganize his program to quite a large extent.

Reading guided by the use of a schema that has prototypical values for certain variables may explain why the programmer, through his program, makes certain changes in the specifications. Thus on machine tools, the work operations are generally made during the *Forward movement*. On one of the stations in the installation that we studied, the last work operation is made during the *Return movement*. The programmer, however, gives this operation the same conditions as those that are made during the *Forward movement*.

4. CONCLUSION

Firstly, the results obtained concern the operators' giving up of the hierarchically structured plan that they consider to be a representation of, or a guide to, their activity. This result is also interesting in that it confirms, in a study made on complex tasks, the results of other studies carried out in the laboratory on simplified tasks. It therefore reinforces the conclusion that design is an activity which - even if it is often guided by a hierarchically structured plan which breaks the problem or its solution down into hierarchically organized components - involves many opportunistic, non-hierarchical aspects (see Hayes-Roth & Hayes-Roth, 1979).

However, our results do not simply confirm those obtained in previous studies; we also show that this opportunistic nature does not only characterize the activity of each individual operator during his stage of the process, but that it is also a feature of the design process as a whole.

We believe, however, that the most important contribution our study makes is the analysis of the factors which lead the operators to give up their hierarchically structured plan. The psychological processes we describe are well-known to be at work in other diagnostic or problem-solving contexts.

On the one hand, these results thus make it possible to specify the extent to which design activity resembles or differs from these other activites.

On the other hand, knowledge of these processes and the specific conditions which trigger them, provides valuable information for the modeling of design activity.

REFERENCES

Carroll, J. M. & Rosson, M. B. Usability specifications as a tool in iterative development. In H. Rex Hartson (Ed.), Advances in human-computer interaction (Vol. 1). Norwood, N.J.: Ablex, 1985.

Détienne, F. Analyse exploratoire de l'activité de compréhension de programmes informatiques. Proceedings AFCET du Séminaire "Approches quantitatives en Génie Logiciel", Sophia Antipolis, France, 7-8 juin 1984.

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

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

Robert, J.-M. Les résultats des recherches sur le processus de conception, les comportements et les processus cognitifs mis en jeu (Rapport INRIA EC 7912 R01). Rocquencourt: INRIA, 1979.

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

Tardieu, H., Rochfeld, A. & Colletti, R. La méthode MERISE, Tome 1: principes et outils (éd. revue et corrigée). Paris: Les Editions d'Organisation, 1986.

Visser, W. Modélisation de l'activité de programmation de systèmes de commande. Actes du colloque COGNITIVA 85 (tome 2). Paris: Cesta, 1985.

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.

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique