Concurrent use of different techniques for gathering data on the programming activity

Willemien Visser, A. Morais

INRIA

UNITÉ DE RECHERCHE
INRIA-ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
BP105
78153 Le Chesnay Cedex
France
Tél. (1) 39 63 55 11

Rapports de Recherche

N° 939

Programme 8

CONCURRENT USE OF DIFFERENT TECHNIQUES FOR GATHERING DATA ON THE PROGRAMMING ACTIVITY

Willemien VISSER
Alexandre MORAIS

Décembre 1988

RR_0939

Concurrent use of different techniques for gathering data on the programming activity*

Willemien VISSER & Alexandre MORAIS°

November 1988

---

* This text is an English version of a paper published in "Psychologie Française".

° When completing this study, the second author was supported by an INRIA grant.

Concurrent use of different techniques for gathering data on the programming activity

Willemien VISSER & Alexandre MORAIS

L'utilisation concurrente de différentes méthodes de recueil de données pour l'étude de l'activité de programmation

Willemien VISSER & Alexandre MORAIS

November 1988

PROGRAMME 8

Abstract. Discusses and advocates the concurrent use of several knowledge acquisition methods for eliciting different types of expertise. Illustrations in the domain of programming are presented.
INTERVIEWS are of particular use in the first stage of a domain study, providing general information on the organisation of the activity and topics to be investigated with other techniques.
ANALYZING THE RESULT OF THE ACTIVITY, knowledge the expert possesses may be identified, but the way it is used remains hypothetical.
Data on this knowledge use may be gathered OBSERVING THE EXPERT IN REAL TIME DURING HIS DAILY ACTIVITY, but, as this method is very expensive, it can only be applied to a few experts, and thus requires, in general, independent validation of its results.
OBSERVATION IN A CONTROLLED SITUATION may cover many subjects, but on a limited number of factors and, generally, in a rather limited context.
Using these methods concurrently however cancels out the disadvantages of a particular method and makes it possible to benefit from the advantages of them all.

Keywords: Expertise, Expert knowledge elicitation, Data gathering techniques, Programming activity, Knowledge representation

Résumé. L'utilisation concurrente de différentes méthodes de recueil de données est proposée, chacune étant appropriée pour recueillir un type particulier de données. L'exposé de son application au recueil de connaissances en programmation sert d'illustration.
Dans une première étape d'étude d'un domaine nouveau, des ENTRETIENS permettent d'obtenir des informations générales sur l'organisation de l'activité et des thèmes d'étude à approfondir à l'aide d'autres méthodes.
L'ANALYSE DU PRODUIT DE L'ACTIVITÉ permet une étude détaillée des connaissances que possède l'opérateur, mais ne fournit que des hypothèses sur la façon dont il les utilise.
L'OBSERVATION EN TEMPS RÉEL EN SITUATION DE TRAVAIL donne accès à cette utilisation des connaissances, mais -pour des raisons de coût, ne permettant pas l'étude de beaucoup d'opérateurs- nécessite, en général, une validation indépendante des résultats.
L'OBSERVATION EN SITUATION CONTROLÉE permet d'étudier beaucoup de sujets, mais sur un nombre de facteurs restreint et, en général, dans un contexte plutôt limité.
L'utilisation concurrente de ces méthodes permet alors de neutraliser les inconvénients de chacune prise individuellement, tout en tirant profit des avantages de chacune.

Mots clefs: Expertise, Recueil de connaissances, Techniques de recueil de données, Activité de programmation, Représentation de connaissances

Contents

INTRODUCTION

If the study of expertise did not begin as a consequence of expert systems development, it has nevertheless gained in importance since the appearance of this new application. Still rare however are the methods which may be used for this study, as much in psychology (but see Visser & Falzon, 1988) as in computer science (but see Kidd, 1987).

The concurrent use of different techniques for expertise elicitation is proposed in this paper, each individual technique being appropriate for gathering a particular type of data. Underlying these proposed techniques are classical psychological data gathering techniques which have been adapted and transformed to a greater or lesser degree (especially for a results validation combined with new data gathering, see §2.2).

The presentation of this concurrent use of techniques is illustrated by its application on the study of the programming activity, and on industrial programmable controllers (IPC)[1].

The following methods have been used in this study:
- semi-directed interviews with programmers about their work
- analysis of programs, that is, the final product of the activity
- observations on a programmer during his daily activity in a work context
- controlled observations on novices in a problem solving context of program design.

Each method will be presented starting with the reason why it was chosen, that is, with the goal to be attained. Next to the techniques used for gathering and analyzing the data, some results obtained with each of these methods will be presented in order to illustrate them.

The conclusion of this paper will be that, as each of the methods presented has its specific use due to the types of data it brings to light, they should all be used concurrently.

The study of the activity of programming has a double goal:
- modeling this activity
- constructing assistance tools.

Focusing therefore on the knowledge used by programmers, three aspects of the activity are distinguished corresponding to three types of knowledge:
- knowledge the programmer has in the domain of the application and of the computer device: programming knowledge
- the strategies the programmer implements during his activity and which call for this first type of knowledge
- the planning of his activity by the programmer.

---

[1] An IPC is a computer specialized in the control of automatic industrial processes.

1. FIRST APPROACH BY THE RESEARCHER TO THE STUDY OBJECT

1.1 INTERVIEWS: FIRST DATA ON THE MAIN ASPECTS OF THE ACTIVITY

When the psychologist approaches a new study object, he may gather the first information about it by (a) reviewing the (psychological) literature on the domain or on connected ones; and (b) interviewing people working in the domain itself.

Concerning the first possibility, in the IPC domain there are
- introduction books to IPCs which are for the most part aimed at (future) IPC users who generally have no computer education and in particular present information about technical aspects of IPCs (see Fray & Hazard, 1983);
- literature on the psychology of programming (see Gilmore, Green, Hoc & Samurçay, in press). This provides little information, on the one hand, on professional programming and, on the other hand, on the specificity of IPC programming (but see Visser, 1987).

The interview method has been chosen to explore the domain under study, especially for grasping the major stages and the main aspects of the programming activity.

The method must be used with circumspection. Like the retrospective verbalization method to which it is similar, it risks inducing the interviewee to structure his activity a posteriori. The plan presented by the programmer as describing his activity, for example, does not cover the "real" activity as it has been observed (see Visser, 1988).

1.2 METHOD

1.2.1 Semi-directed interviews with pre-established questions

The interviews are conducted with the programmers individually. A guide is provided by a list of more or less open questions concerning the major stages and the main aspects of the programmer's activity.

For example, "What does your work consist of?", "How do you handle a programming project? Where do you start, for example?", and "What are your work tools, which documents, and what other material do you use?".

Although these questions guide the interviews, the programmer's answers to questions may lead to other, unforeseen questions. Remarks the programmer makes about subjects on which the interviewer has not touched are taken into consideration and the subject may be followed up.

1.2.2 Qualitative analysis of the answers

The analysis of the data obtained from the interviews is mainly qualitative, as quantification of this type of data is rarely possible.

1.3 EXAMPLE OF RESULTS

This method has been used with six programmers. With regard to the different aspects of the activity that have been distinguished, the interviewing method has provided information on the global organization of the programming activity and the different types of information the programmers use.

1.3.1 Different representation formalisms for different types of knowledge

An interesting result from a methodological point of view comes from the observation that the programmer, during the interviews, uses different expressions according to the type of knowledge he refers to.

Knowledge about the global organization of the activity is described, in general, in terms of action sequences, such as "I start with the work stations, then I move on to the transfer unit, and afterwards the grip function; I keep the general commands for the end".

Knowledge about the coding into the program are described, in general, in terms of conditional actions, for example "If a movement is controlled by a distributor with three positions, its corresponding instruction must have four branches."

Our hypothesis is that these different types of knowledge do not have the same representation form, and that different formalisms are more or less appropriate for their expression (see also Visser & Falzon, 1988). A "schema" formalism (see Rumelhart, 1978), for example, would suit knowledge of the first type; "production rules" (see Davis & King, 1977) would be more appropriate for the second type.

2. STUDY OF THE RESULT OF THE ACTIVITY: GATHERING PROGRAMMERS' KNOWLEDGE

The analysis of the result of the programmer's activity, that is, of programs that he has written, gives access to knowledge he possesses; to study the way he uses it, "real time" observations on the programmer are required (see §3).

The techniques presented in this section have been used in different ways and for different goals. They serve data gathering (§2.1) as well as validating the obtained results (§2.2).

2.1 ANALYZING PROGRAMMERS' COMMENTS ON PROGRAMS: DATA GATHERING

By asking the author of a program to comment on its construction, the aim is to collect, next to programming knowledge, information about the programmer's representation of the strategies he uses and the way he organizes his activity.

2.1.1 Method

2.1.1.1 Programmers' comments on program modules they have written

The programmer is asked to choose a program module and to explain, for each instruction, why and how he has written it the way it figures in the program. Asking "why", data should be obtained about the way the programmer organized his program, whereas the "how" question is supposed to provide information about the procedures he has followed (see Graesser, 1978).

First analysis stage: Spotting the knowledge. To begin with, regularities and repeated references to the same type of knowledge are looked for in the programmer's statements.

So, for example, the two statements
" And then I put [in the command instruction for Rotation to the Right (RR)] /RL[1] as a security"
and
(in french) "/DSP (DesSerrage Pièce) est l'inverse en barre [de SP, Serrage Pièce][2]",
are interpreted as translating two instances of use of the same knowledge schema "Securities to be introduced into a command instruction."

Second analysis stage: Formalizing the collected knowledge. The example cited above has led to the formulation of the following production rule:
"To command an operation, one has to introduce, as a security, the negation of the bit of command of the inverse operation."

2.1.2 Examples of results[3]

2.1.2.1 Spotted knowledge

In particular, programming knowledge has been found. Some data about strategies has also been obtained however.

Example of a decomposition strategy. Having to command an action, the programmer decomposes it in phases, Advance and Return. In an Advance phase, he puts together the operations leading to the goal of the process (the "work" that is done, for example, conveying a piece). In a Return phase, he puts together those operations which are necessary for preparing the process to execute the first ones (their "prerequisites").

Examples of programming knowledge. Two types of knowledge are distinguished here:
- semantic knowledge, covering the information used when programming automatic installations;
- syntactic knowledge, covering the way this information has to be coded into the program.
  In each of these two categories, general and specific knowledge are distinguished.

General semantic knowledge holds for all applications covered by the programmer, that is, in the present case, machine-finishing (fr. *usinage*), assembling, and handling.

Among the specific semantic knowledge units, different sets can be distinguished, for example, those which concern only the application covered in the commented program module, that is a conveying process.

General syntactic knowledge concerns the programming language which is used. For example, this language forbids the programmer to use more than once in an instruction of a program the same result variable corresponding to

---

[1] No-Rotation to the Left.

[2] Translation and explanation:"/RGP (Release Grip Piece) is the opposite [of RP, Grip Piece, the operation commanded by the instruction] preceded by the negation sign".

[3] For more details, see Visser, 1985.

an operation. This leads the programmer to collect, for each operation, all its occurrences in one and the same instruction defining the operation.

Among the specific syntactic knowledge units, some translate standards of the firm which employs the programmer; others are idiosyncratic to a particular programmer.

2.1.2.2 Formalized knowledge

The knowledge referred to by the programmer concerns, in general, conditional actions and their linguistic expression has often been in conditionals. So, they suit the production rule formalism.

Example of a specific syntactic knowledge unit:
IF a combination of information units is used more than once in the program,
THEN make an intermediary bit out of it.

2.2 ANALYZING PROGRAMS: VALIDATING OBTAINED RESULTS AND GATHERING NEW DATA

The results obtained by analyzing the comments on programs might only be valid for the program which has been commented on, the "source program". In this section some techniques are presented for validating these results. The application of these techniques allows, otherwise, new data to be gathered.

Internal and external validation. The validation is made to examine if
1. The programmer has actually used the knowledge to which he refers in his comments to the source program.
2. The programmer has used this knowledge for the construction of other programs.

2.2.1 Judgment of the rules by the programmer

A first evaluation of the rules consists of asking the programmer to indicate, for all rules that have been formulated,
- in which cases they applied and in which cases they did not (generality test);
- which aspects had not been covered (completeness test).

The results of this evaluation may lead to rules being modified.

2.2.2 Manual and automatic simulation

The construction of a program, from rules (translating knowledge) and specifications, can be simulated
- manually: taking, as data, the rules formulated from the program comments and the specifications for a program, the experimenter, as an interpreter, selects the rules whose conditions are satisfied and makes them fire to produce the actions building up program;
- automatically: in collaboration with a computer scientist, an expert system has been written (in Prolog). The knowledge base of the system is constituted of the rules formulated from the program comments. The system is interactive: the user provides the program specifications to answer the questions of the system.

2.2.3 Confrontation of a program constructed by simulation and a reference

The reference necessary to evaluate the result of a program construction simulation can be
- an existing program (see Table 1);
- the mental representation that a programmer constructs from the program specifications used for the construction of the program. Programmers are then asked to judge the appropriateness of the program constructed by simulation (method of judges) (see also §2.2.4).

*** Reference: existing program ***

- Data: rules + specifications of the installation to be controlled
- For each rule, examine: are its conditions satisfied on the installation?
  If not: go to next rule
  If yes: generate the action part of the rule
- For the rules whose conditions are satisfied several times in the program:
  Compare its action with the one expressed in the program:

1. If all generated actions ≠ those in the program:
   reject the rule (but ask the programmer a question about it)

2. If all generated actions = those in the program:
   keep the rule

3. If some generated actions = those in the program
   but other generated actions ≠ those in the program:
   modify the rule's conditions to take into account all actions in the program

*** Reference: rules ***

- Data: rules + specifications of the installation to be controlled
- For each rule, examine: are its conditions satisfied on the installation?
  If not: go to next rule
  If yes: generate the action part of the rule
- If generated action ≠ existing action (that is, in the program):
  replace existing action with action generated by the rule (that is, "homogenization" of program according to the rules)

Table 1. Procedure for comparing a program constructed by manual simulation with an existing program

2.2.4 Programmers' evaluation of homogenized programs

The author of a program which has been homogenized (see Table 1) is asked if the homogenized version is equivalent to the one he has written, that is, if it performs the same functions. The finding that the programmer accepts as being equivalent instructions that have been modified, shows that there must be conditions that govern the choice between two, or more, ways of coding a same function.

3. REAL TIME STUDY OF THE PROGRAMMING ACTIVITY

3.1 OBSERVATIONS ON PROGRAMMERS IN WORK SITUATION: USE OF KNOWLEDGE AND STRATEGIES

The different simulations which have been conducted (see §2.2.2) suggest that there are decision criteria which intervene in programming and which can not been deduced by the experimenter from the finished program, nor explicited by the programmer in retrospective verbalization such as it is used in interviews or comments on programs.

That is why, for studying the real activity, observations are made on programmers in their work situation.

Other reasons lead to the choice of this method. The analysis of the result of the activity, commented on or not by the programmer, provides data on knowledge the programmer possesses; observations on programmers at work is the best way to obtain data on the way this knowledge is used, that is, on the programming activity[1].

3.1.1 Method

3.1.1.1 Observations + simultaneous verbalization: note taking and document collection

This method has been used for conducting full time observations on an IPC programmer during the construction of a program (four weeks). He was asked to proceed as normally, but to verbalize, as much as possible, his thoughts about what he was doing (see Ericsson & Simon, 1984).

Notes were collected concerning:
- all the programmer's comments and writings;
- the order in which he produced the different documents, and how he gradually built them up;
- the changes he made;
- the information sources consulted;
- events which were judged to be indicators of the subject meeting with difficulties: interruptions, reprocessing of elements already handled, construction of rough drafts and intermediary schemes.

In addition, all documents produced by the programmer during his work were collected

---

[1] In time constrained situations, this method often can not been used. Visser & Falzon (1988) present methods which may be used in such cases.

- the diagrams and schemas he constructed for himself during analysis and problem solving;
- the different versions of these documents and of the program;
- the rough drafts of (parts of) them.

3.1.1.2 Qualitative analysis of notes: Searching for knowledge and strategies used and difficulties encountered

Notes are examined in the construction order of the program, using, as references, the documents collected. The analysis focuses on the aspects of the activity which have been distinguished:
- if the programmer refers to knowledge units he uses, these are collected together with their triggering conditions, the context of their use, and the way the programmer refers to them;
- the search for strategies focuses especially on the way the programmer solves the problem of (a) analyzing the specifications which constitute his problem statement, and (b) organizing the program and its construction. With this aim, the notes are examined to respond to questions like "Which information sources are used and how?", "How does the programmer organize the information in the program?";
- elements of information about the way the programmer plans his activity are the order in which he works, the decomposition he makes in his problem analysis and the program construction, recurrences in his actions, statements translating planning, and the way he follows them or not subsequently.

The indicators presented above (see §3.1.1.1) are used to analyze the difficulties the programmer encounters and the way he resolves them.

3.1.2 Examples of results

3.1.2.1 Example of a general strategy used for constructing the program: Use of analogies

The strategy in question is often used by the programmer, and at various levels of problem solving. The analogies which are taken advantage of may be between
- structures, for example, to organize the program inspired by another one[1];
- functions, for example, to analyze the operation of a machine station using that of another one.

3.1.2.2 Knowledge invoked in a compiled way

The knowledge units collected when analyzing finished programs (see §§2.1-2.2) have a degree of detail to which the observed programmer often does not refer. The observations show that, in general, he seems to invoke his knowledge as a whole, as if it had been compiled (see Anderson, 1986). Only when high level compiled knowledge units are not appropriate, does he activate elements of them. To judge their degree of appropriateness, he uses, for example, knowledge in the application domain: a categorization of machine units is used to decide on the opportunity to use a module of the example program[2].

---

[1] The programmer often turns to the listing of a program he has written in the past.

3.2 OBSERVATIONS ON PROGRAMMERS IN A CONTROLLED SITUATION: PRECISE DATA ON SOME SPECIFIC ASPECTS OF THE ACTIVITY

Certain types of data obtained with the methods presented above need to be validated (especially the data coming from a single subject). The controlled study of one or more factors allows such a validation to be made. It can be also used of course for studying factors one knows already, by other means, or one supposes, to influence the activity. Finally, it can serve to discover factors in a well fixed context. From this last viewpoint, it has been used to explore the programming activity of novice programmers in a context of simple problem solving. The focus was on the search for factors influencing an aspect of the activity which anterior studies had shown to be important, but problematic, that is, the functional analysis of problems.

3.2.1 Method

3.2.1.1 Observations + simultaneous verbalization: note taking and document collection

3.2.1.2 Qualitative analysis of notes: Searching for knowledge and strategies used and difficulties encountered

The two methods are the same as those used for observations in a natural work situation (see §3.1.1).

3.2.1.3 Quantitative error analysis

This analysis provides
- an evaluation of the difficulties encountered by the novices, conducted in parallel to the qualitative analysis;
- comparisons between different parts of the functional analysis with regard to their degree of difficulty;
- comparisons between the subjects with regard to their degree of expertise.

3.2.2 Example of results[1]

3.2.2.1 Adaptation of the learned specification method: Different status of goals and prerequisites

The subjects modify the method they learned in order to adapt it to their representation of the functioning they have to analyze, especially by processing the aspects of the process directly related to its goal before its prerequisites.

The actions leading directly to the goal are considered right from the first resolution stage. The actions preparing the process to attain its goals (the prerequisites of the first ones) are handled in a late processing stage or even completely omitted.

---

[1] See Morais & Visser, 1987, for more detail.

4. CONCLUSION

As the data gathered differ depending on which of the above methods is used, it is necessary to apply these methods concurrently in order to gather the data that each one separately cannot provide. Indeed, the four methods have been used for the following reasons:

Interviews may not allow precise data on the activity to be obtained, but this method is useful, in the first stage of a study on a new domain, to collect
- general information about the organization of the activity;
- study objects to be developed with the other methods.

The analysis of the result of the activity allows a detailed study to be made of the knowledge the programmer possesses, but only gives hypothetical indications about its use.

Real time observations on a programmer in his natural work situation gives access to this use of knowledge. It is however a costly method and it does not allow many subjects to be studied. In general, an independent validation of the results is necessary.

Observations on programmers in a controlled situation makes it possible to study many subjects, but only on a limited number of factors, and, in general, in a rather limited context. The data gathered on the factors which influence the activity are not exhaustive, but the role of those which have been covered appears clearly.

Thus the concurrent use of the presented methods makes it possible to cancel out the inconveniences of each one taken separately, while benefitting from the advantages of each one.

REFERENCES

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 Publishers, 1986.

Davis, R., & King, J. J. An overview of production systems. In E. Elcock & D. Mitchie (Eds.), Machine intelligence 8. Chichester, England: Ellis Horwood, 1977.

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

Fray, M., & Hazard, C. Les automatismes par la logique programmée. Paris: Nathan, 1983.

Graesser, A. C. How to catch a fish: the memory and representation of common procedures. Discourse Processes, 1978, 1, 72-89.

Gilmore, Green, Hoc & Samurçay, in press

Kidd, A. L. (Ed.). Knowledge acquisition for expert systems. A practical handbook. New York: Plenum, 1987.

Morais, A. & Visser, W. Programmation d'automates industriels: adaptation par des débutants d'une méthode de spécification de procédures automatisées. Psychologie Française, N° Spécial *Les langages informatiques dans l'enseignement*, 1987, 32, 253-260.

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.

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.

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

Visser, W., & Falzon, P. Eliciting expert knowledge in a design activity: some methodological issues (Research Report N° 906). Rocquencourt: INRIA, 1988.

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