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

Willemien Visser, Alexandre Morais

Pre-print de
Visser, W., & Morais, A. (1988). L'utilisation concurrente de différentes méthodes de recueil de données pour l'étude de l'activité de programmation. *Psychologie Française, No. spécial "Psychologie de l'expertise*", 33, 127-132.

L'UTILISATION CONCURRENTE DE DIFFERENTES METHODES DE RECUEIL DE DONNEES POUR L'ETUDE DE L'ACTIVITE DE PROGRAMMATION

Willemien VISSER & Alexandre MORAIS[1]

RESUME. 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 contrôlée 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-CLES. Méthodes de recueil de données, Expertise, Activité de programmation, Entretiens, Observation de l'activité, Observation en temps réel, Représentation des connaissances.

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 have especial utility 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, one may identify knowledge the expert possesses, 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, this method being very expensive, it can only be applied to a few experts, and thus requires, in general, independant 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 constrained context.
Using these methods concurrently, however, one cancels out the inconveniences of each one taken apart, benefitting of the qualities of each one.
KEYWORDS. Data-collection methods; Expertise; Software programming activity; Interviews; Observing the activity; Real-time observation; Knowledge representation

INTRODUCTION

Si l'étude de l'expertise n'est pas née suite aux essais de construction de systèmes experts, elle a pris quand même un essor depuis que cette application est apparue. Les références concernant des méthodes pour cette étude sont cependant encore rares, aussi bien en psychologie qu'en informatique (mais v. Kidd, 1987).
L'utilisation concurrente de différentes méthodes de recueil d'expertise est proposée ici, chacune étant appropriée pour recueillir un type particulier de données. Il s'agit de méthodes classiques en psychologie que nous avons adaptées et transformées plus ou moins (notamment pour la validation des résultats accompagnée du recueil de nouvelles données, v. §2.2).
L'exposé de cette utilisation concurrente de méthodes est illustré par son application à l'étude de l'activité de programmation, telle qu'elle s'exerce sur des automates programmables industriels (API[2]).

Les méthodes suivantes ont été utilisées:
- entretiens semi-dirigés avec des programmeurs au sujet de leur travail
- analyse de programmes, c'est-à-dire du produit final de l'activité

[1] Ce travail a été réalisé alors qu'Alexandre Morais était boursier de thèse à l'INRIA, où Willemien Visser l'a encadré.
[2] Un API est un ordinateur spécialisé dans la commande de procédés automatisés.

Pre-print de
Visser, W., & Morais, A. (1988). L'utilisation concurrente de différentes méthodes de recueil de données pour l'étude de l'activité de programmation. *Psychologie Française, No. spécial "Psychologie de l'expertise*", 33, 127-132.

- observation d'un programmeur pendant son activité dans une situation de travail
- observation contrôlée de débutants dans une situation de résolution d'un problème de conception de programme.

Chaque méthode sera présentée à partir du but poursuivi en la choisissant. A côté d'un exposé des techniques de recueil et d'analyse des données utilisées, quelques résultats obtenus avec chacune des méthodes serviront d'illustration.
En anticipant sur la conclusion, nous considérons que chacune des méthodes a son utilité spécifique, étant donné les types de données qu'elle permet et qu'elle ne permet pas de recueillir, et que, pour cela, il faut en faire une utilisation concurrente.
En étudiant l'activité de programmation, nous avons un double objectif:
- modéliser cette activité
- construire des outils d'aide.

Nos études étant alors centrées sur les connaissances utilisées par le programmeur, nous distinguons, dans l'activité de celui-ci, trois aspects qui correspondent à trois types de connaissances:
- les connaissances du programmeur dans le domaine d'application et dans celui du dispositif informatique: les connaissances de programmation
- les stratégies déployées lors de son activité et qui font appel à ce premier type de connaissances
- la planification de son activité par le programmeur.

1. PRISE DE CONNAISSANCE PAR LE CHERCHEUR DE L'OBJET D'ETUDE

1.1 ENTRETIENS: PREMIÈRES DONNÉES SUR LES GRANDES LIGNES DE L'ACTIVITÉ

Lorsque le psychologue aborde un nouvel objet d'étude, la récolte des premières informations peut se faire (a) par un examen de la littérature (psychologique) concernant le domaine dont relève son objet d'étude ou concernant des domaines annexes; et (b) auprès de personnes exerçant l'activité concernée. Une forme que peut prendre cette dernière approche est celle des entretiens.
Quant au premier point, on dispose de
- livres de présentation d'API qui, étant destinés surtout à des utilisateurs (futurs) d'API dépourvus, en général, de formation informatique, comportent notamment des informations sur les aspects techniques et informatiques des API (Fray & Hazard, 1983);
- la littérature sur la psychologie de la programmation (v. Hoc & Visser, 1988). Elle fournit peu d'informations, d'une part, sur la programmation professionnelle, d'autre part, sur la nature spécifique de la programmation d'API (mais v. Visser, 1987b).

La méthode des entretiens a été choisie pour débroussailler le terrain d'étude, notamment pour nous faire une idée des étapes de la programmation et des grandes lignes du travail.
Il s'agit d'une méthode à utiliser avec prudence. Proche de la verbalisation rétrospective (Ericsson & Simon, 1984), elle comporte le danger que l'opérateur structure son activité a posteriori. Ainsi nous avons constaté, par exemple, que le plan que l'opérateur dresse pour décrire son activité ne couvre pas l'activité réelle telle qu'elle a été observée (Visser, 1987a).

1.2 MÉTHODE

1.2.1 Entretiens semi-dirigés à partir de questions pré-établies

Les entretiens sont conduits avec les opérateurs individuellement. Ils sont guidés par une liste de questions plus ou moins ouvertes, concernant les grandes lignes du travail. Par exemple, "En quoi consiste votre travail?", "Comment traitez-vous un projet, par où commencez-vous par exemple?" et "Quels sont les outils de travail, les documents, le matériel que vous utilisez?".
Ayant ces questions comme fil conducteur des entretiens, nous nous laissons cependant également orienter par les réponses des programmeurs. Leurs remarques à propos de sujets non-abordés par l'intervieweur sont prises en compte et éventuellement approfondies.

1.2.2 Analyse qualitative des réponses

L'analyse des données recueillies à l'aide des entretiens est surtout qualitative, la quantification de telles données étant rarement possible.

1.3 EXEMPLE DE RÉSULTATS

Nous avons utilisé cette méthode avec six programmeurs. Quant aux différents aspects de l'activité que nous avons distingués, la méthode des entretiens a fourni des informations sur l'organisation globale de l'activité du programmeur et les différents types d'information qu'il utilise.

Pre-print de
Visser, W., & Morais, A. (1988). L'utilisation concurrente de différentes méthodes de recueil de données pour l'étude de l'activité de programmation. *Psychologie Française, No. spécial "Psychologie de l'expertise*", 33, 127-132.

1.3.1 Différents formats de représentation pour différents types de connaissances
Un résultat intéressant d'un point de vue méthodologique provient de notre constatation que le programmeur, dans les entretiens, s'exprime différemment selon le type de connaissances auxquelles il renvoie.
Les connaissances concernant l'organisation globale sont décrites, en général, en termes de suites d'actions, tels que "Je commence par les postes d'usinage, ensuite j'attaque l'unité de transfert, puis le bridage; je garde les commandes générales pour la fin".
Les connaissances concernant le codage sont décrites, en général, en termes d'actions conditionnelles, par exemple "Si un mouvement est commandé par un distributeur à trois positions, l'instruction de commande doit comporter quatre branches".
Nous faisons l'hypothèse que ces différents types de connaissances n'ont pas le même format de représentation et que différents formalismes sont plus ou moins appropriés pour les exprimer (v. aussi Visser & Falzon, dans ce même numéro). Si le premier type de connaissances, par exemple, se prête bien à un formalisme type "schéma" (Rumelhart, 1978), le second se traduit mieux sous la forme de "règles de production" (Davis & King, 1977).

2. ETUDE DU RESULTAT DE L'ACTIVITE: RECUEIL DE CONNAISSANCES DU PROGRAMMEUR

L'analyse du produit de l'activité du programmeur, c'est-à-dire de ses programmes, permet d'accéder aux connaissances que celui-ci possède; pour étudier l'utilisation qu'il en fait, l'observation du programmeur en temps réel s'impose (v. §3).
Nous avons utilisé cette méthode de différentes manières et à différentes fins. Elle nous a servi aussi bien pour le recueil de données (§2.1) que pour la validation des résultats obtenus (§2.2).

2.1 ANALYSE DE COMMENTAIRES DE PROGRAMMES: RECUEIL DE DONNEES
En demandant à l'auteur d'un programme de commenter la construction de celui-ci, nous comptons recueillir, à côté de connaissances de programmation, des informations sur la représentation que le programmeur a de l'organisation de son activité et des stratégies qu'il utilise.

2.1.1 Méthode
2.1.1.1 Commenter un module de programme écrit par le programmeur
Nous demandons au programmeur de choisir un module de programme et d'expliquer, à propos de chaque instruction, pourquoi et comment il l'a écrite telle qu'elle figure dans le programme. Par la question du "pourquoi" nous comptons obtenir des données sur la façon dont le programmeur a organisé son programme, tandis que la question du "comment" est censée nous renseigner sur les procédures suivies (Graesser, 1978).
Première étape d'analyse: Repérer les connaissances. Nous commençons par chercher à repérer, dans les énoncés, d'une part des régularités, d'autre part des références répétées à un même type de connaissances.
Ainsi, par exemple, les deux énoncés
"Et puis je mets [dans l'instruction de commande de la Rotation A Droite (RAD)] /RAG[3] comme sécurité"
et
"/DSP[4] est l'inverse en barre [de SP, Serrage Pièce, l'opération commandée dans l'instruction]"
traduisent à nos yeux l'utilisation, à deux occurrences différentes, d'un même schéma de connaissances "Sécurités à introduire dans une commande d'opération".
Seconde étape d'analyse: Formalisation des connaissances relevées. L'exemple cité ci-dessus a conduit à la formulation de la règle de production suivante:
"Pour commander une opération, il faut introduire, comme sécurité, la négation du bit de commande de l'opération inverse".

2.1.2 Exemples de résultats[5]
2.1.2.1 Connaissances repérées
Il s'agit surtout de connaissances de programmation. Nous avons cependant obtenu également quelques données sur des stratégies.
Exemple de stratégie de décomposition. Devant l'action d'un bras à commander, le programmeur décompose celle-ci en phases, avance et retour. Dans une phase d'avance, il réunit les opérations conduisant au but du procédé (le "travail" qui est

[3] NON-Rotation A Gauche.
[4] DesSerrage Pièce.
[5] Pour plus de détails, voir Visser, 1985.

Pre-print de
Visser, W., & Morais, A. (1988). L'utilisation concurrente de différentes méthodes de recueil de données pour l'étude de l'activité de programmation. *Psychologie Française, No. spécial "Psychologie de l'expertise*", 33, 127-132.

fait, ici le convoyage d'une pièce). Dans une phase de retour, il réunit celles qui sont nécessaires pour préparer le procédé à exécuter ces premières (leurs "prérequis").
Exemples de connaissances de programmation. Nous avons distingué
- des connaissances sémantiques, portant sur les informations utilisées lors de la programmation d'installations automatisées;
- des connaissances syntaxiques, portant sur le codage de ces informations.
Dans chacune de ces deux catégories, nous distinguons des connaissances générales et des connaissances spécifiques.
Les connaissances sémantiques générales sont valables pour toutes les applications couvertes par le programmeur, dans ce cas de l'usinage, de l'assemblage et de la manutention.
Parmi les connaissances sémantiques spécifiques, il y a différents ensembles, par exemple celles qui ne concernent que l'application programmée dans l'exemple commenté, à savoir un procédé de convoyage.
Les connaissances syntaxiques générales concernent le langage de programmation utilisé.
Exemple. Ce langage interdit d'utiliser le bit correspondant à un organe de commande plus d'une seule fois comme sortie d'une instruction dans le programme. Ceci conduit le programmeur à regrouper toutes les occurrences d'une opération dans une seule instruction définissant cette opération.
Parmi les connaissances syntaxiques spécifiques, il y a celles qui traduisent des standards de l'entreprise dans laquelle travaille le programmeur et celles qui sont propres au programmeur individuel.

2.1.2.2 Connaissances formalisées
Les connaissances évoquées par le programmeur concernent, en général, des actions conditionnelles et leur expression linguistique a été souvent de nature conditionnelle. Ainsi, elles se prêtent bien au formalisme des règles de production.
Exemple de connaissance syntaxique spécifique:
SI une combinaison d'informations est utilisée plus d'une fois dans le programme, ALORS il faut en faire un bit intermédiaire.

2.2 ANALYSE DE PROGRAMMES: VALIDATION DES RESULTATS OBTENUS et RECUEIL DE NOUVELLES DONNEES
Les résultats de l'analyse des commentaires risquent de ne valoir que pour le programme commenté, le "programme source". Dans ce qui suit, nous présentons un certain nombre de techniques pour valider ces résultats. L'application de ces techniques sert, par ailleurs, à recueillir de nouvelles données.
Validations interne et externe. En validant les règles, nous avons voulu examiner si
1. Le programmeur a utilisé effectivement les connaissances auxquelles il a fait appel lors de ses commentaires du programme source.
2. Le programmeur a utilisé ces connaissances lors de la construction d'autres programmes.
2.2.1 Jugement des règles par le programmeur
Une première évaluation des règles a consisté à demander au programmeur d'indiquer, pour toutes les règles formulées par nous,
- dans quels cas elles étaient applicables et dans quels cas elles ne l'étaient pas (test de généralité);
- quels aspects n'étaient pas pris en compte (test de complétude).
Les résultats de cette évaluation conduisent à d'éventuelles modifications de règles.

2.2.2 Simulations manuelle et automatique
Une simulation de la construction d'un programme, à partir de règles (traduisant des connaissances) et de spécifications, peut se faire
- manuellement: avec, comme entrées, les règles et les spécifications d'un programme, nous faisons tourner "à la main" les règles pour produire celui-ci, c'est-à-dire que nous remplissons le rôle d'un moteur d'inférences;
- automatiquement: en collaboration avec un informaticien, nous avons écrit, en PROLOG, un système expert. Ce système possède, en tant que connaissances, les règles formulées à partir des commentaires. Il travaille en interactif: l'utilisateur fournit les spécifications en réponse à des questions que le système pose.

2.2.3 Confrontation d'un programme construit par simulation et d'un élément de comparaison
L'élément de comparaison nécessaire pour évaluer le produit d'une simulation peut être
- la représentation qu'un programmeur se fait à partir des spécifications utilisées lors de la simulation. Dans ce cas, nous demandons à un ou plusieurs programmeurs de juger de l'adéquation du programme construit par simulation (méthode des

Pre-print de
Visser, W., & Morais, A. (1988). L'utilisation concurrente de différentes méthodes de recueil de données pour l'étude de l'activité de programmation. *Psychologie Française, No. spécial "Psychologie de l'expertise*", 33, 127-132.

juges) (v. aussi §2.2.4);
- un programme existant (v. Tableau 1).

2.2.4 Evaluation par le programmeur d'un programme homogénéisé
Nous demandons à l'auteur du programme existant si le programme homogénéisé (v. Tableau 1) est équivalent au sien, c'est-à-dire s'il remplit les mêmes fonctions. Il y a des instructions modifiées que le programmeur accepte comme équivalentes. Ceci montre qu'il doit y avoir des conditions qui gouvernent le choix entre deux modes de codage pour une même fonction.

* Programme existant comme point de référence *
- Matériel de départ: règles + spécifications de l'installation à commander
- Pour chacune des règles, se demander: ses conditions sont-elles remplies sur l'installation?
Si non: passer à la règle suivante
Si oui: générer la partie action de la règle
- Comparer l'action avec celle exprimée dans le programme pour les règles dont les conditions sont remplies à plusieurs reprises dans le programme:
1. SI toutes les actions générées ≠ celles du programme:
rejeter la règle (mais formuler une question à son sujet, à poser au programmeur)
2. SI toutes les actions générées = celles du programme:
maintenir la règle
3. SI certaines actions générées = celles du programme
tandis que d'autres actions générées ≠ celles du programme:
modifier les conditions de la règle pour que celle-ci puisse rendre compte de toutes les actions générées dans le programme

* Règles comme point de référence*
- Procéder de la même façon jusqu'à la génération d'action
- SI action générée (par la règle) ≠ action existante (dans le programme):
remplacer action existante par action générée par règle ("homogénéisation" du programme en accord avec les règles)

Tableau 1. Procédure de comparaison d'un programme construit par simulation manuelle et d'un programme existant

3. ETUDE DE L'ACTIVITE DE PROGRAMMATION EN TEMPS REEL

3.1 OBSERVATION DE PROGRAMMEURS EN SITUATION DE TRAVAIL: UTILISATION DES CONNAISSANCES ET STRATEGIES

Les différentes simulations que nous avons menées (v. §2.2.2) nous suggèrent qu'il y a des critères de décision qui interviennent lors de la programmation que nous ne pouvons ni inférer du programme fini, ni faire expliciter par le programmeur à l'aide de la verbalisation rétrospective telle qu'elle est utilisée dans les entretiens ou les commentaires de programmes.
C'est pourquoi, parallèlement à la simulation de l'écriture de programmes, nous utilisons, pour l'étude de l'activité réelle, l'observation de l'opérateur en situation de travail.
Il y a d'autres raisons qui conduisent au choix de cette méthode. L'analyse du produit de l'activité, commenté ou non par l'opérateur, fournit des données sur les connaissances que possède le programmeur; l'observation d'opérateurs au travail est la méthode indiquée pour obtenir des données sur leur utilisation[6], c'est-à-dire sur l'activité de programmation.

3.1.1 Méthode

3.1.1.1 Observations avec demande de verbalisation simultanée: prise de notes et recueil des traces

Nous avons utilisé cette méthode pour observer un programmeur d'API pendant toute la construction d'un programme (quatre semaines).
Nous lui avons demandé de procéder comme d'habitude, mais d'énoncer à haute voix ses pensées au sujet de son travail.

[6] Cette méthode ne peut cependant pas être utilisée toujours (v. Visser & Falzon, 1988).

Pre-print de
Visser, W., & Morais, A. (1988). L'utilisation concurrente de différentes méthodes de recueil de données pour l'étude de l'activité de programmation. *Psychologie Française, No. spécial "Psychologie de l'expertise*", 33, 127-132.

Pour le recueil des données, nous avons pris note de
- tout ce qu'il écrivait et disait
- l'ordre de l'écriture des instructions et leur construction progressive
- les modifications apportées
- les sources d'information consultées
- les événements que nous concevions comme des indicateurs de rencontre de difficultés: interruptions, reprises d'éléments déjà traités, écriture de brouillons, construction de schémas intermédiaires.

A côté des notes, nous avons recueilli les différentes versions et les brouillons du programme et les schémas que le programmeur s'est construits.

3.1.1.2 Analyse qualitative des notes: recherche des connaissances utilisées, des stratégies et des difficultés rencontrées
Nous examinons les notes dans l'ordre de l'écriture du programme, utilisant, comme points d'appui, les traces recueillies. L'analyse est centrée sur les aspects de l'activité que nous avons distingués:
- quand le programmeur fait référence à des connaissances utilisées, nous relevons celles-ci avec leurs conditions de déclenchement, le contexte d'utilisation et la façon dont il s'y réfère;
- quant aux stratégies, nous nous intéressons surtout à la façon dont le programmeur résout les problèmes de (a) l'analyse de l'énoncé du problème que constituent les spécifications reçues et (b) l'organisation du programme et sa construction. Pour ce faire, nous cherchons, dans nos notes, des réponses à des questions comme "Quelles sources d'information utilise-t-il et comment?", "Comment organise-t-il les informations dans le programme?";
- nous examinons la planification de l'activité du programmeur à partir de l'ordre dans lequel le programmeur travaille, des découpages qu'il apporte dans son analyse du problème et la construction du programme, des récurrences dans ses actions, des énoncés traduisant de la planification et la façon dont il réalise ou non ensuite celle-ci.

Nous analysons en outre les difficultés rencontrées par le programmeur et la façon dont il les résout à partir des indicateurs cités en §3.1.1.1.

3.1.2 Exemples de résultats
3.1.2.1 Exemple d'une stratégie générale utilisée pour la construction du programme: Utilisation d'analogies
Il s'agit d'une stratégie que le programmeur utilise beaucoup et à divers niveaux en exploitant l'analogie entre
- structures, par exemple pour organiser le programme à partir d'un autre[7];
- fonctions, par exemple pour analyser le fonctionnement d'un poste de la machine à partir de celui d'un autre.

3.1.2.2 Connaissances invoquées de façon compilée
Les connaissances relevées lors de l'analyse de programmes finis (§§2.1-2.2) ont un niveau de détail auquel le programmeur observé ne descend souvent pas. Nos observations montrent que, en général, il semble invoquer ses connaissances "en bloc", comme "compilées" (Anderson, 1986). C'est seulement quand l'adéquation à un niveau supérieur échoue qu'il active des composants de ces blocs. Pour juger du degré d'adéquation, il utilise, par exemple, des connaissances dans le domaine d'application: à partir d'une catégorisation des unités de la machine, il décide de l'opportunité d'utiliser un module du programme-exemple7.

3.2 OBSERVATION DE PROGRAMMEURS EN SITUATION CONTROLEE: DONNEES PRECISES SUR QUELQUES ASPECTS PARTICULIERS DE L'ACTIVITE
Certains types de données obtenues par les méthodes présentées jusqu'ici nécessitent d'être validées (notamment les données provenant du comportement d'un seul sujet). L'étude contrôlée d'un ou plusieurs facteurs permet ce type de validation. Elle se prête bien sûr également à l'étude de facteurs dont on sait déjà par ailleurs ou dont on suppose qu'ils influencent l'activité. Finalement, elle peut servir à découvrir des facteurs dans un contexte bien déterminé. Nous l'avons utilisée dans cette dernière optique pour explorer l'activité de programmation de programmeurs débutants dans un contexte de résolution de problèmes simples. Nous nous sommes centrés sur la recherche de facteurs qui influencent un aspect de l'activité que nos études antérieures avaient indiqué comme important, mais problématique: l'analyse fonctionnelle du problème.

3.2.1 Méthode
3.2.1.1 Observations avec demande de verbalisation simultanée: prise de notes et recueil des traces
3.2.1.2 Analyse qualitative des notes: recherche des connaissances utilisées, des stratégies et des difficultés rencontrées

[7] Le programmeur fait souvent appel au listing d'un programme écrit antérieurement

Pre-print de
Visser, W., & Morais, A. (1988). L'utilisation concurrente de différentes méthodes de recueil de données pour l'étude de l'activité de programmation. *Psychologie Française, No. spécial "Psychologie de l'expertise*", 33, 127-132.

Il s'agit des deux mêmes méthodes de recueil et d'analyse de données que celles utilisées lors de l'observation en situation de travail (v. §3.1.1).

3.2.1.3 Analyse quantitative des erreurs
Elle permet
- une évaluation des difficultés rencontrées par les débutants conduite en parallèle à l'analyse qualitative
- des comparaisons entre les degrés de difficulté des différentes parties de l'analyse
- des comparaisons entre les sujets plus ou moins débutants que nous avons observés.

3.2.2 Exemple de résultats[8]
3.2.2.1 Adaptation de la méthode de spécification apprise: statut différent des buts et des prérequis
Les élèves modifient cette méthode pour l'adapter à leur représentation du fonctionnement à analyser, notamment en traitant les aspects du procédé reliés directement au but avant ses prérequis.
Les actions conduisant directement au but sont prises en compte dès la première étape de résolution. Les actions qui préparent le procédé à remplir ses buts (les prérequis des premières) sont traitées dans une étape tardive du traitement ou oubliées complètement.

4. CONCLUSION

Selon que l'on utilise l'une ou l'autre des méthodes présentées ci-dessus, les données que l'on peut recueillir sont différentes. Il est alors nécessaire d'utiliser ces méthodes de manière concurrente afin de recueillir les données que chacune en particulier ne peut fournir. En effet, les quatre méthodes ont été utilisées pour les raisons suivantes:
Si les entretiens ne permettent pas d'obtenir des données précises sur l'activité, cette méthode est utile, dans une première étape d'étude d'un domaine nouveau, pour obtenir
- des informations générales sur l'organisation de l'activité
- des thèmes d'étude à approfondir avec d'autres méthodes.

L'analyse du produit de l'activité permet une étude détaillée des connaissances que possède l'opérateur, mais ne donne que des indications hypothétiques sur leur utilisation.
L'observation en temps réel en situation de travail donne accès à cette utilisation des connaissances. Elle est cependant très coûteuse et ne permet pas l'étude de beaucoup d'opérateurs. En général, une validation indépendante des résultats est nécessaire.
L'observation en situation contrôlée permet d'étudier beaucoup de sujets, mais sur un nombre de facteurs restreint et, en général, dans un contexte plutôt limité. Les données recueillies sur les facteurs qui influencent l'activité ne le sont pas de façon exhaustive, mais le rôle de ceux qui sont couverts apparaît clairement.
L'utilisation concurrente des méthodes présentées permet alors de neutraliser les inconvénients de chacune prise individuellement tout en tirant profit des avantages de chacune.

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.

Hoc, J. M., & Visser, W. (Eds.) L'approche cognitive de la programmation informatique. Le Travail Humain, Numéro Spécial "Psychologie ergonomique de la programmation informatique", 1988, 51(4), 289-296.

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.

[8] Pour plus de détails, voir Morais et Visser, 1987.

Pre-print de
Visser, W., & Morais, A. (1988). L'utilisation concurrente de différentes méthodes de recueil de données pour l'étude de l'activité de programmation. *Psychologie Française, No. spécial "Psychologie de l'expertise*", 33, 127-132.

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. Abandon d'un plan hiérarchique dans une activité de conception. Actes du colloque scientifique COGNITIVA 87 (Tome 1). Paris: Cesta, 1987a.

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, 1987b.

Visser, W., & Falzon, P. Recueil et analyse de l'expertise dans une activité de conception: questions de méthode. Psychologie Française, N° Spécial "Psychologie de l'Expertise", 1988, 33, 133-138.
Repris dans J. Leplat (Ed.). (1992), L'Analyse du travail en psychologie ergonomique (Vol. 1, pp. 389-401). Toulouse, France: Octarès.