# L'activité de comparaison de représentations dans la mise au point de programmes

Willemien Visser

▶ **To cite this version:**

Willemien Visser. L'activité de comparaison de représentations dans la mise au point de programmes. Le travail humain, 1988, Numéro Spécial "Psychologie ergonomique de la programmation informatique", 51 (4), pp.351-362. hal-00642568

**HAL Id: hal-00642568**
**https://inria.hal.science/hal-00642568**

Submitted on 18 Nov 2011

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Ce texte est une version de l'auteure de l'article suivant:
Visser, W. (1988). L'activité de comparaison de représentations dans la mise au point de programmes. *Le Travail Humain, Numéro Spécial "Psychologie ergonomique de la programmation informatique"*, *51*(4), 351-362.

# L'ACTIVITE DE COMPARAISON DE REPRESENTATIONS DANS LA MISE AU POINT DE PROGRAMMES

# COMPARING REPRESENTATIONS IN PROGRAM TESTING AND DEBUGGING

Willemien VISSER

Projet de Psychologie Ergonomique pour l'Informatique - INRIA (bât. 9) - Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 Le Chesnay Cedex

**Résumé**. L'article présente une étude de cas, conduite dans un contexte industriel de conception d'installations automatisées. Il s'agit de la mise au point d'un programme de commande d'une machine-outil automatisée. Un opérateur (programmeur-metteur au point) a été observé pendant toute la durée de son travail (cinq semaines), qui consistait dans la mise au point du programme, qui avait été auparavant écrit par un collègue programmeur. Nous nous sommes centrée sur un processus central dans la mise au point: l'activité de comparaison de couples de représentations, une référence et une source. Pour chacune des deux, différents types de représentation sont construits et comparés consécutivement, si nécessaire, avant la détection d'une erreur et/ou de sa source. Ces constructions et comparaisons consécutives nécessitent un traitement cognitif de plus en plus complexe. Selon le niveau dans cette progression, un diagnostic de la source de l'erreur est conduite ou non pour expliquer, en termes de variables de programme, l'écart entre le comportement observé du système (la source) et son comportement correct (la référence).
**Mots clés**. mise au point de programme, conception de programme, observations de l'activité, étude de cas, étude de terrain, machine-outil automatisée, construction de représentations, diagnostic

**Summary**. This article presents a case and field study on the debugging of a program that was to control an automatic machine tool installation. A programmer was observed full time for five weeks in his daily work, debugging the program written previously by a colleague. We chose to focus on a central process in debugging: the activity of comparison of pairs of representations, a reference and a target. For both, different types of representatations are constructed and compared consecutively, if necessary, before an error and/or its source are detected. These consecutive constructions and comparisons require increasingly complex information processing. Depending on the level in this progression, a diagnosis of the cause of the error is conducted to explain, in terms of program variables, the discrepancy between the observed (target) and correct (reference) system behaviour.
**Key words**. debugging, programming, observational study, field study, case study, automatic machine tool installations, construction of representations, diagnosis

Dans l'étude de la conception de programmes, les activités psychologiques auxquelles fait appel l'étape de mise au point ont reçu moins d'attention que celles déployées dans des étapes antérieures. Les revues de question en témoignent, ou bien en présentant relativement peu ou pas d'études (cf. Hoc, 1985[1]; Pennington, 1982), ou bien en soulignant la maigre quantité de connaissances que nous possédons sur ce sujet (cf. Curtis, 1984; Sheil, 1981). Ce manque d'attention ne peut se justifier par l'absence d'intérêt

[1] Les études sur "La détection et la correction des erreurs dans les programmes" citées par l'auteur sont des "travaux guidés par des cadres de référence informatiques" "sans référence aux mécanismes cognitifs sous-jacents".

Ce texte est une version de l'auteure de l'article suivant:
Visser, W. (1988). L'activité de comparaison de représentations dans la mise au point de programmes. *Le Travail Humain, Numéro Spécial "Psychologie ergonomique de la programmation informatique"*, *51*(4), 351-362.

ni d'ordre économique, ni d'ordre psychologique. Selon les études, les auteurs estiment que la mise au point d'un programme prend 25% du temps requis pour la conception d'un logiciel ou qu'elle occupe trois fois plus de temps que le codage (études citées par Gould, 1975). Les processus cognitifs auxquels le metteur au point a recours pendant son activité sont du ressort d'aussi bien la compréhension que la résolution de problème. Ils vont de la construction de représentations, via le diagnostic, à la solution du problème.
Dans cet article, nous présentons les grandes lignes de l'activité psychologique sous-jacente à la mise au point d'un programme d'un type particulier. Il s'agit d'un programme de commande d'une installation automatisée, commandée par un ordinateur (par un "automate programmable industriel")[2].
Le premier intérêt de cette étude réside donc dans l'apport de données sur un aspect de la programmation peu exploré. Le second intérêt est dû à ce nous avons fait des observations dans un contexte de travail professionnel:

(a) le problème dont nous étudions la résolution est réel et complexe (plutôt qu'artificiel et simplifié comme le sont les problèmes étudiés dans la plupart des recherches menées jusqu'ici sur la programmation);
(b) (b) les observations ont été conduites "en temps réel", pendant toute la durée de la mise au point d'un programme, sur le lieu de travail de l'opérateur.

*Recherches antérieures sur la mise au point: l'importance de la comparaison de représentations*.
Même s'ils divergent sur d'autres points, la plupart des auteurs qui ont étudié l'activité de mise au point ("*debugging*") s'accordent pour considérer, comme un processus central dans celle-ci, la comparaison entre des représentations (Détienne, 1986; Gugerty et Olson, 1986; Michard, 1975; Kessler et Anderson, 1986; Vessey, 1985). Le couple de représentations comparées peut couvrir (a) ce que le programme fait et ce qu'il est censé faire ou (b) les résultats du programme à corriger et ceux du programme "correct". Il s'agit donc toujours d'une comparaison entre deux représentations de l'opérateur: une référence (sa représentation du programme-cible) et une source (sa représentation du programme qu'il doit mettre au point).
Il s'agit d'un résultat que nous avons retrouvé dans notre étude de la mise au point d'un programme de commande d'une installation automatisée. Nous présenterons ci-dessous une analyse détaillée des différentes représentations construites et des comparaisons conduites sur elles.

*Spécificité du type de programmation étudié*. La différence avec un programme "classique" qui nous intéresse ici concerne les entrées et sorties. Les entrées d'un programme de commande traduisent, par l'intermédiaire des états de capteurs sur le procédé commandé, l'état de celui-ci; les sorties sont traduites en actions sur le procédé (les "fonctions" étudiées dans cet article). L'analogue des résultats fournis par un programme "classique" quand on le fait tourner avec un certain jeu de données d'essai consiste ainsi dans le comportement de l'installation commandée dans une certaine configuration. Si le programme ne répond pas aux spécifications, certaines actions ne se feront pas ou se feront d'une façon différente que spécifiée. C'est, entre autres (cf. ci-dessous), à partir de ce comportement de l'installation que le metteur au point vérifiera le programme.

[2] Une description détaillée de cette installation peut être trouvée dans Visser, 1986.

Ce texte est une version de l'auteure de l'article suivant:
Visser, W. (1988). L'activité de comparaison de représentations dans la mise au point de programmes. *Le Travail Humain, Numéro Spécial "Psychologie ergonomique de la programmation informatique"*, *51*(4), 351-362.

**Méthode**

Nous avons observé, dans l'atelier de l'usine où l'installation avait été construite[3], un metteur au point, sans interruption, du début de son intervention sur le projet jusqu'à un moment très avancé de cette intervention[4] (cinq semaines).

L'étude avait les caractéristiques suivantes:

Sujet. L'opérateur observé était un électricien-automaticien, ayant une expérience professionnelle de plus de trois ans.

Tâche. La mise au point étudiée devait conduire à ce que l'installation commandée par le programme fonctionne en accord avec ses spécifications. La tâche du metteur au point concernait, à côté du programme de commande, tous les autres aspects électricité et automatismes de l'installation. Nous ne nous en occupons pas ici.

Le programme à mettre au point avait été construit par un autre électricien-automaticien (cf. Visser, 1987b).

Procédure. Nous avons demandé au metteur au point de procéder comme d'habitude, à une exception près: nous l'avons prié d'énoncer, le plus possible, à haute voix, ses pensées concernant son travail.

Recueil des données. Pour des raisons pratiques, nous avons jugé inexploitables des données que nous aurions pu recueillir en faisant, pendant les cinq semaines d'observation, un enregistrement complet de l'activité de l'opérateur (caméra + magnéthophone). C'est pourquoi nous avons pris des *notes* au sujet des aspects de l'activité de l'opérateur qui nous ont paru pertinents par rapport à notre centre d'intérêt dans cette étude, les processus de traitement de l'information. Ces notes concernaient

- les sources d'information consultées (listing du programme, documents de spécification, comportement de l'installation mise au point)
- tout ce que l'opérateur écrivait (les annotations qu'il apportait sur le listing du programme, le code qu'il écrivait, les notes qu'il prenait, les croquis, brouillons et schémas qu'il faisait);
- toutes ses interventions sur l'installation (notamment sur ses capteurs et actionneurs);
- tout ce que l'opérateur disait au sujet de ces trois types d'actions;
- l'ordre de ces actions;
- les arrêts, pauses et autres événements conçus par nous comme des indicateurs de rencontre par l'opérateur de difficultés: silences, reprises d'éléments déjà traités, écriture de brouillons, construction de diagrammes et de schémas intermédiaires.

A côté des notes, nous avons recueilli, notamment dans le but de disposer, lors de l'analyse des données, d'un support d'interprétation de nos notes, toutes les *traces matérielles* de l'activité de l'opérateur:

- les différentes versions de programmes modifiés;
- les brouillons, diagrammes et schémas que l'opérateur s'est construits, en général, comme aides à la compréhension du programme.

Analyse des données. Nous exposons dans cet article les résultats d'une première analyse globale des données recueillies. Cette analyse a porté sur

- l'organisation de l'activité (comment l'opérateur découpe-t-il sa mise au point du programme?);

[3] Et où, parallèlement à sa mise au point par le metteur au point, l'installation était mise au point par les mécaniciens.

[4] Après cinq semaines de travail, la mise au point du programme étudié ne pouvant continuer à cause de problèmes mécaniques, le metteur au point a commencé la mise au point d'un autre programme. De façon intermittente, nous avons suivi le reste de ses interventions sur le programme dont la mise au point nous intéressait.

Ce texte est une version de l'auteure de l'article suivant:
Visser, W. (1988). L'activité de comparaison de représentations dans la mise au point de programmes. *Le Travail Humain, Numéro Spécial "Psychologie ergonomique de la programmation informatique"*, *51*(4), 351-362.

- les stratégies utilisées (comment l'opérateur procède-t-il aux actions élémentaires de mise au point, c'est-à-dire aux vérifications auxquelles il estime devoir soumettre les données dont il dispose, le programme et l'installation?);
- les différents niveaux des traitements que l'opérateur applique sur ses données (étant donnée une unité de mise au point, est-ce que l'opérateur en décompose sa vérification et comment?).

**Résultats**

*Organisation hiérarchique de l'activité: des composants aux unités*. Le metteur au point organise son activité en découpant la mise au point de l'installation. Il découpe la machine dans ses unités physiques et chaque unité dans ses fonctions (et certains de leurs constituants, à savoir ceux qui ont été définis indépendamment de la fonction[5]).
Chaque unité physique a une fonction différente dans le procédé d'usinage: le transfert des pièces à usiner d'une unité à une autre, l'usinage lui-même (ébauche et finition), son contrôle et le (dé)chargement des pièces.
Sur chaque unité, différentes fonctions sont assurées: sur celle du transfert, par exemple, les principales fonctions sont: débridage, évolution rapide, évolution lente et bridage.
Le fonctionnement de la machine est découpé par le metteur au point dans ses différents modes (manuel, semi-automatique et automatique).
Au niveau des unités et des modes de fonctionnement, l'activité du metteur au point peut être résumée globalement de la façon suivante[6]:
- Pour chaque fonction, ses constituants sont vérifiés avant la fonction elle-même.
- Les différentes unités individuelles sont vérifiées, unité par unité, d'abord en manuel et semi-automatique[7], puis en automatique.
- Ensuite, la machine en entier est vérifiée, d'abord en semi-automatique, puis en automatique.

Le metteur au point procède donc de façon hiérarchique: d'abord sont traités les composants, puis les unités (de la machine ou d'une fonction) (mais cf. la note 6).
Dans cet article, nous décrirons l'activité du metteur au point au niveau des *fonctions dans les modes manuel et semi-automatique*.

*Choix des vérifications effectuées: selon un plan et de façon opportuniste*. Les vérifications des fonctions auxquelles procède le metteur au point dépendent partiellement d'un plan qu'il a en tête selon lequel les fonctions sont vérifiées dans des configurations satisfaisant certains critères. La plupart des vérifications ne sont faites cependant que suite à des problèmes rencontrés en vérifiant.

---

[5] Une fonction est définie par une combinaison booléenne de bits. Certaines groupes de bits qui rentrent dans la définition d'une fonction rentrent également dans la définition d'autres fonctions. Ces groupes de bits sont rassemblés dans des bits "intermédiaires" définis dans d'autres instructions que celle définissant la fonction. Les "Contrôles" de fins d'opération en constituent un exemple.

[6] Nous représentons dans cet article les grandes lignes de l'activité. De cette façon elle coïncide presque avec le plan qui rend compte de la description donnée par le metteur au point quand nous lui avons demandé de nous décrire son activité. Nous faisons ici donc abstraction des déviations de ce plan qui, comme nous l'avons exposé ailleurs (Visser, 1987a), rendent "opportuniste" l'activité réelle.

[7] Il s'agit de deux modes distincts dans le fonctionnement de l'installation. Etant donné l'imbrication des vérifications concernant l'un ou l'autre mode, ils ne font pas l'objet d'étapes distinctes dans l'activité du metteur au point (cf. la Figure 1).

Ce texte est une version de l'auteure de l'article suivant:
Visser, W. (1988). L'activité de comparaison de représentations dans la mise au point de programmes. *Le Travail Humain, Numéro Spécial "Psychologie ergonomique de la programmation informatique"*, *51*(4), 351-362.

Toutes les fonctions[8] des unités sont vérifiées par le metteur au point, mais non pas dans toutes les configurations d'états que l'installation peut prendre. Ceci n'est d'ailleurs pas possible: notamment les conséquences des défauts (mécaniques et électriques) qui peuvent se présenter pendant l'exploitation de l'installation ne peuvent pas toutes être évaluées, tous les défauts ne pouvant pas être provoqués par le metteur au point.
Pour chaque fonction, le metteur au point doit donc choisir les configurations dans lesquelles il la vérifiera. Certaines configurations sont vérifiées systématiquement. Ainsi toutes les fonctions sont vérifiées commandées dans ce qui constitue leur position de départ en fonctionnement "normal". Pour la plupart des "familles" de configurations cependant, seulement un échantillon en est vérifié. Pour cela, le metteur au point semble utiliser, d'une part, un critère de sélection, dans lequel rentrent des facteurs de caractère divers, comme "danger en cas de mauvais fonctionnement" ou "fréquence avec laquelle des problèmes ont été rencontrés dans cette configuration sur d'autres installations". D'autre part, beaucoup de vérifications ne se font qu'à la suite d'un problème surgi lors d'une vérification précédente sur l'installation sous traitement (aspect de l'activité qui la rend "opportuniste", c'est-à-dire, plutôt que par un plan préétabli, l'activité est guidée par le résultat des actions et l' information reçue pendant l'activité ).

*Deux types de vérification: examen sur papier et essai sur machine*. Chaque fonction qui est effectivement vérifiée subit, de la part du metteur au point, deux types de traitements. Le premier est un examen qui se fait uniquement sur le listing du programme, c'est-à-dire sans prise d'information sur l'installation. Le second se fait à partir d'un essai de la commande de la fonction sur l'installation. Nous décrirons ci-dessous ces deux types de vérification.

*Examen programme: première comparaison de représentations*. Nous présentons cet examen en commentant la Figure 1 (page suivante).
Un examen de fonction commence toujours par une comparaison que le metteur au point établit entre deux représentations, celle du programme-source reçu du programmeur et celle du programme-cible qu'il s'est construite à partir des spécifications [1][9]. Si les deux sont équivalentes[10], la comparaison est vite conclue (le temps d'un survol des yeux) et la fonction est essayée [2]. Quand les deux sont

---

[8] Et ceux de leurs constituants que nous avons indiqués. La distinction entre fonctions et constituants n'étant pas pertinente pour notre exposé, nous ne les distinguerons pas dans ce qui suit.
[9] Les numéros entre crochets renvoient aux numéros des branches de l'organigramme.
[10] Toute instruction est constituée d'une combinaison booléenne de bits définissant une variable résultat, c'est-à-dire une fonction ou un bit intermédiaire (cf. note 6). L'ordre des éléments n'a pas d'influence sur la valeur de la variable résultat, mais il peut être important pour des raisons de durée d'inspection des instructions. Mise à part cette réserve, l'ordre des bits dans une instruction n'est pas pertinent et deux instructions qui diffèrent en surface peuvent donc être équivalentes par rapport à la valeur de la variable résultat. Ceci fait que le metteur au point doit faire une lecture au-delà de la surface de l'instruction.
Nous présentons deux exemples simplifiés d'équivalence entre instructions:

| | | |
|---|---|---|
| Les instructions | | E14 + B13 = A13 |
| et | | B13 + E14 = A13 |
| sont équivalentes. | | |
| Les deux ensembles d'instructions | (1) | E15.E16 = B13 |
| | | E21+ B13 = A32 |
| et | | |
| | (2) | E15.E16 + E21 = A32 |
| sont également équivalents. | | |

Ce texte est une version de l'auteure de l'article suivant:
Visser, W. (1988). L'activité de comparaison de représentations dans la mise au point de programmes. *Le Travail Humain, Numéro Spécial "Psychologie ergonomique de la programmation informatique"*, *51*(4), 351-362.

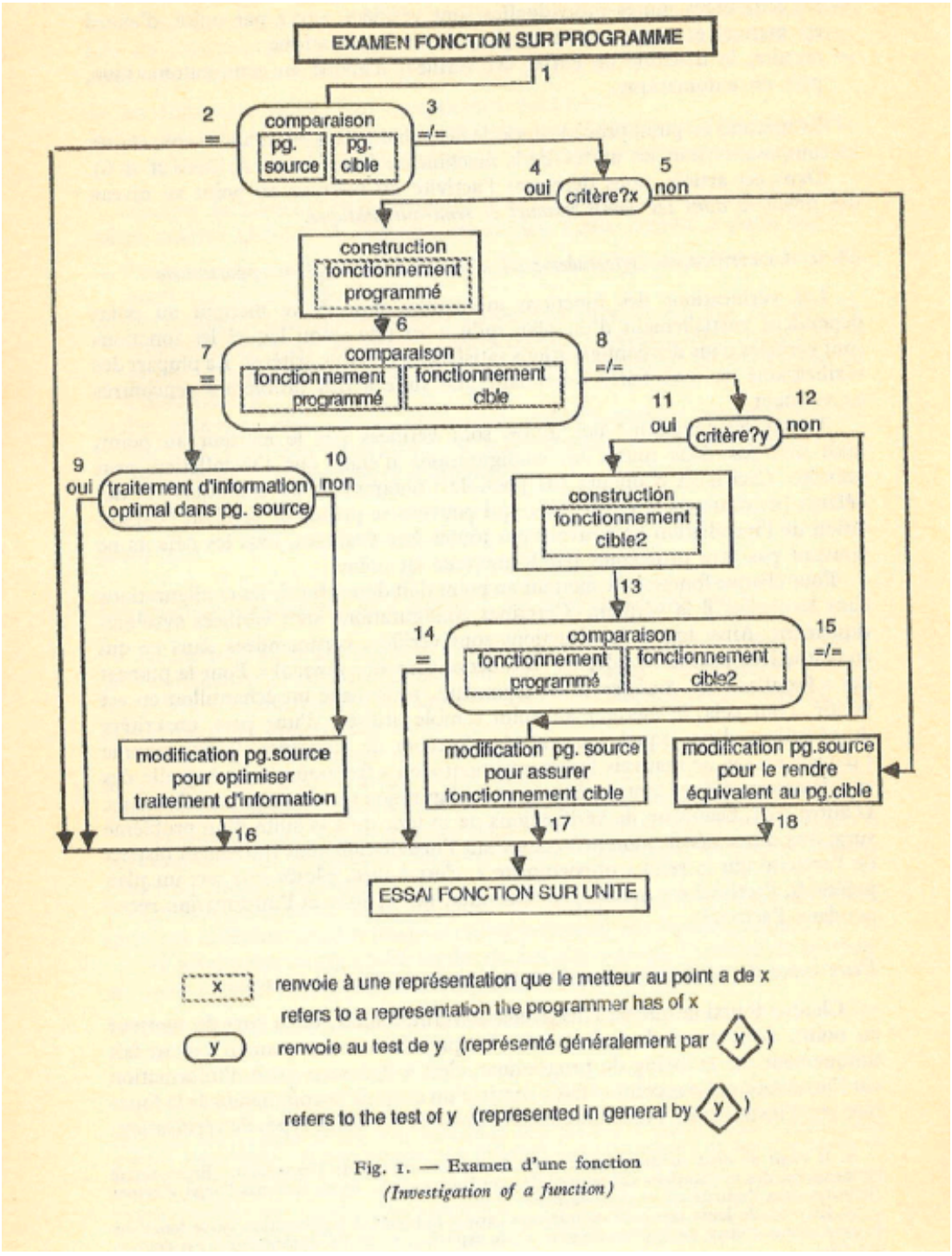


Fig. 1. — Examen d'une fonction
*(Investigation of a function)*

Figure 1. Examen d'une fonction
Figure 1. Investigation of a function

Ce texte est une version de l'auteure de l'article suivant:
Visser, W. (1988). L'activité de comparaison de représentations dans la mise au point de programmes. *Le Travail Humain, Numéro Spécial "Psychologie ergonomique de la programmation informatique"*, *51*(4), 351-362.

différentes [3], le metteur au point doit décider, en fonction d'un critère que nous ne connaissons pas, s'il modifie le programme-source pour le rendre équivalent au programme-cible [5] ou s'il procède à un examen plus approfondi du programme-source [4]. Dans ce dernier cas, il construit une représentation du fonctionnement tel qu'il est commandé par le programme-source et la compare avec celle du fonctionnement-cible qu'il s'est construite à partir des spécifications [6].Ces deux représentations de fonctionnement sont donc construites à partir, non pas de l'observation d'un comportement de l'installation, mais de la compréhension d'un programme (le programme-source) ou d'une autre représentation symbolique de fonctionnement (les spécifications). S'il s'agit du même fonctionnement [7], la différence constatée (lors de [3]) entre les deux représentations de programme provient de différences dans la façon de traiter l'information dans les programmes. Si le metteur au point estime non-optimale la façon dont le programmeur a fait ce traitement, il procède à une modification du programme-source [10]; sinon, il passe à l'essai de la fonction [9].
Si les deux représentations de fonctionnement diffèrent [8], le metteur au point ne procède pas systématiquement à une modification du programme-source [12]. Dans certains cas [11], il procède encore à une comparaison entre la représentation du fonctionnement programmé et une représentation, éventuellement nouvelle, du fonctionnement-cible qu'il construit à partir d'un nouvel examen des spécifications [13]. Selon les résultats de cette comparaison, il procède à une modification du programme-source [15] ou non [14].
Après chaque modification [16,17,18], le metteur au point passe à l'essai de la fonction.

*Essai machine: deuxième comparaison de représentations*. Après l'examen d'une fonction sur le programme, le metteur au point procède à son essai sur l'unité concernée de la machine.
Un essai est matérialisé, en général, par l'appui sur le bouton de commande de la fonction, l'observation du comportement de l'installation qui s'ensuit et la comparaison entre ce comportement et un comportement-cible, le comportement "attendu".
Chaque essai conduit en effet à un comportement de l'installation[11]. Nous distinguons deux types de comportement de l'installation lors d'un essai, le "comportement attendu" et des "comportements non-attendus". Nous les avons définis par rapport à la représentation que l'opérateur a du fonctionnement-cible de l'installation.

Représentation du fonctionnement-cible. Nous disposons d'éléments qui nous permettent de penser que l'opérateur possède une représentation de ce qu'il estime être le fonctionnement correct de l'installation, c'est-à-dire du fonctionnement qui répond aux spécifications telles qu'elles sont comprises par l'opérateur, le "fonctionnement-cible".
D'une part, nous avons des traces écrites de cette représentation. L'opérateur construit, pour chaque unité de la machine, des diagrammes qui représentent schématiquement le "fonctionnement-cible" de cette unité. Il annote ces schémas, en mentionnant les valeurs de bits "critiques", c'est-à-dire de bits qui changent de valeur lors du passage de l'unité d'un état à un autre.
D'autre part, l'opérateur fait des remarques qui traduisent sa représentation. Elles nous montrent comment, quand l'opérateur lance un essai, il fait, sur l'installation, des prises d'information orientées par cette représentation. Il s'agit de commentaires qui annoncent le comportement attendu, comme "Si l'on a fait des mouvements en manuel, il faut que l'on ne peut reprendre en automatique que si l'on est en cycle" (après quoi l'opérateur lance un essai qui consiste à faire quelques mouvements en manuel

[11] Le fait que "rien ne bouge" constitue bien sûr également un comportement.

Ce texte est une version de l'auteure de l'article suivant:
Visser, W. (1988). L'activité de comparaison de représentations dans la mise au point de programmes. *Le Travail Humain, Numéro Spécial "Psychologie ergonomique de la programmation informatique"*, *51*(4), 351-362.

qui font sortir du cycle, suivis par le bascul de manuel sur automatique) ou "Si j'appuie maintenant sur [le bouton Avance Programmée], ça devrait faire Avance Rapide"[12] ou, en lançant un essai, "Il faut qu'il s'arrête en FAL2" (c'est-à-dire l'une des deux positions de fin de l'opération d'Avance Lente).

Comportement attendu. L'opérateur a donc une représentation du comportement de l'installation auquel il s'attend si le programme est correct, le "comportement attendu". C'est pour cela que nous concevons chaque essai comme une mise à l'épreuve d'une hypothèse. Même s'il ne l'explicite pas toujours (devant nous ou devant lui-même), le metteur au point a une hypothèse au sujet du comportement de l'installation auquel doit conduire son essai si l'installation satisfait aux spécifications.
Un essai qui conduit à un comportement attendu est un "essai réussi".

Comportement non-attendu. Il s'agit de tout comportement autre que le comportement attendu par l'opérateur. L'essai qui y conduit est "non-réussi".
Il est rare qu'un essai réussisse du premier coup. Sur l'une des unités dont nous avons analysé en détail la mise au point, par exemple, ce n'est le cas qu'une seule fois: sur les quelques 30 fonctions vérifiées (en quelques 50 essais), une seule fonction (les Freins) a fonctionné correctement dès le premier essai.

A l'aide de la Figure 2 (page suivante), nous décrirons l'essai d'une fonction.

Si un essai conduit au comportement de l'installation attendu par le metteur au point, celui-ci passe, en général, à la vérification de la fonction suivante [1]. Sinon [2], il procède à un diagnostic sur l'origine du comportement non-attendu (Cx' dans la figure 2), c'est-à-dire il déterminera l'écart entre le comportement attendu et le comportement observé et il interprètera cet écart en termes des éléments de programme responsables. Nos observations nous font penser qu'il s'agit d'un raisonnement qualitatif basé sur des connaissances "profondes" du système de l'installation automatisée ("*from first principles*") plutôt que d'une approche qui s'appuie sur des règles de diagnostic, basées sur l' expérience du metteur au point (cf. Reiter, 1985).
Ce diagnostic le conduit, en général, à supposer que la responsabilité de Cx' se trouve
- dans une autre fonction qu'il n'a pas encore examinée [3]. Il procède alors à sa vérification [6].
- dans le comportement d'une autre fonction, Fz [4]. Il procède alors à un essai de cette fonction [7].
- dans la définition d'un constituant de Fx [4], qu'il (ré)examine alors [8].

Si l'essai de Fz conduit à la conclusion que le comportement non-attendu de Fx est dû à la fonction Fz, le metteur au point procèdera à une intervention sur le procédé [9]. Sinon, il refera un diagnostic [10], portant sur Cx' ou sur Cz'.
Une intervention sur le procédé pour remédier à Cx' est, en général, suivie d'un nouvel essai de Fx [11]. Parfois cependant, le metteur au point n'y procède plus [12]. Nos observations nous font penser que ce dernier test est absent quand il y a eu un grand nombre d'autres fonctions sur lesquelles ont porté les essais et éventuelles modifications consécutifs à l'essai d'une fonction (plus de quatre ou cinq).

[12] Il est utile de savoir que ce bouton correspond à différentes fonctions selon l'état du procédé.

Ce texte est une version de l'auteure de l'article suivant:
Visser, W. (1988). L'activité de comparaison de représentations dans la mise au point de programmes. *Le Travail Humain, Numéro Spécial "Psychologie ergonomique de la programmation informatique"*, *51*(4), 351-362.

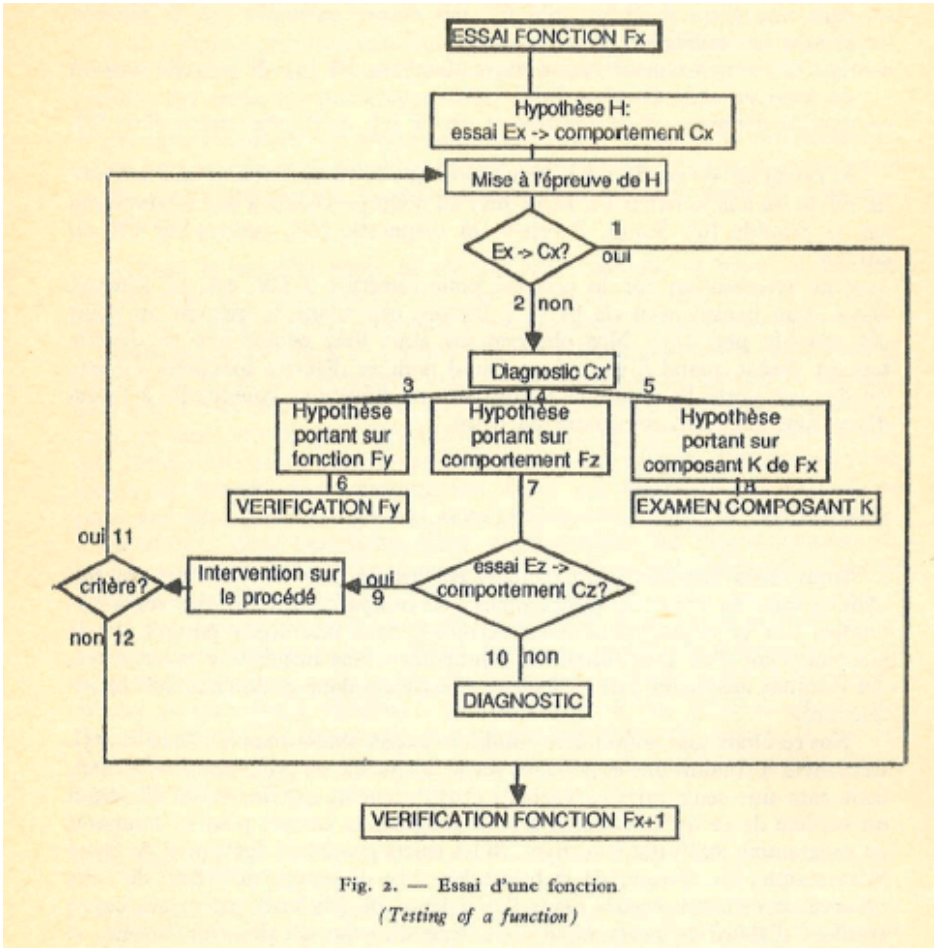


Fig. 2. — Essai d'une fonction
*(Testing of a function)*

Figure 2. Essai d'une fonction
Figure 2. Testing of a function

**Conclusion**

Nous nous sommes centrée, dans cet article, sur un processus central dans la mise au point de programmes: la comparaison entre des représentations. Sur ce point, nous avons retrouvé, dans une étude portant sur la mise au point d'un programme de commande d'une installation automatisée, des résultats analogues à ceux d'autres chercheurs dans ce domaine (cf. l'introduction).

Nos résultats sont notamment similaires à ceux obtenus par Michard (1975). Cet auteur a conduit une expérience sur le diagnostic de programmes FORTRAN, contenant une seule erreur. Michard (1975) conclut que les sujets élaborent un modèle de ce qu'ils pensent être le programme correct pour le comparer au programme qu'ils ont à corriger. Si les sujets procèdent également de façon hiérarchique, les niveaux de la hiérarchie sont de nature différente de ceux observés dans notre étude. Dans l'expérience de Michard (1975), les sujets vérifient d'abord le programme au niveau du plan d'ensemble, ensuite, si l'erreur n'est pas détectée, ils rentrent dans le détail du programme. Les

Ce texte est une version de l'auteure de l'article suivant:
Visser, W. (1988). L'activité de comparaison de représentations dans la mise au point de programmes. *Le Travail Humain, Numéro Spécial "Psychologie ergonomique de la programmation informatique"*, *51*(4), 351-362.

représentations sur lesquelles portent les comparaisons, qui se font à ces différents niveaux ,ont la même nature.
Dans notre étude, le metteur au point procède, pour chaque unité de vérification (dans cet article, les fonctions des unités de la machine), à aux moins deux, mais souvent même trois, comparaisons de représentations de nature différente: d'abord entre des représentations de programmes, ensuite entre des représentations de fonctionnements (les "résultats" des programmes). La constatation d'un écart peut conduire le metteur au point à différents types de modification, souvent portant sur le programme-source.
C'est que le metteur au point organise son activité de façon cognitivement économique: la construction de représentations plus coûteuses et la conduite de comparaisons plus complexes sont gardées pour la fin, pour le cas où la vérification n'a pas réussie par des moyens plus simples.
Nous illustrerons cette conclusion sous deux angles.

* Par rapport aux représentations construites, les modifications nécessitent de plus en plus de pas de raisonnement: la modification éventuelle du programme est basée sur une comparaison
- de représentations de programmes construites à partir de programmes;
- de représentations de fonctionnements construites à partir de programmes;
- de représentations de fonctionnements construites à partir de comportements.

* Dans la première étape de vérification, le metteur au point ne cherche pas la cause de l'écart observé. S'il est sûr de la non-équivalence des deux termes de la comparaison, il procède à une modification du programme-source.
Dans la seconde étape de vérification, une différence entre les deux termes, c'est-à-dire entre des représentations de fonctionnement, conduit le metteur au point à une recherche de la cause de la différence à laquelle il a conclu. Celle-ci peut être de plusieurs types et est, en général, d'une autre nature que l'entité observée: une absence de mouvement est, par exemple, souvent due à un élément de programmation. Dans son diagnostic, le metteur au point doit ainsi passer, en général, d'un système (ou format) de représentation à un autre: dans l'exemple, d'une représentation du fonctionnement en termes de composants physiques ou de fonctions à une autre en termes d'unités d'information qui y renvoyent (les bits ou les instructions du programme). Nous ne disposons actuellement pas d'éléments nous permettant de savoir comment ces passages opèrent. Nous supposons que l'un des facteurs qui les rendent difficiles est la différence de structure, aussi bien entre les représentations qu'entre leurs référents: un programme a une autre structure que l'installation qu'elle commande ou le fonctionnement de celle-ci et, sur ce point, les représentations que le metteur au point en a diffèreront sans doute également entre elles.

**Bibliographie**

Curtis, B. (1984) - Fifteen years of psychology in software engineering: individual differences and cognitive science. In Proceedings of the 7th International Conference on Software Engineering. Washington, DC: IEEE Computer Society.

Détienne, F. (1986) - La compréhension de programme informatiques par l'expert: un modèle en termes de schémas (Thèse de doctorat). Paris: Université Paris V.

Giboin, A. (1978) - Mémoire épisodique, mémoire sémantique et niveaux de traitement. L'Année Psychologique, 78, 203-232.

Ce texte est une version de l'auteure de l'article suivant:
Visser, W. (1988). L'activité de comparaison de représentations dans la mise au point de programmes. *Le Travail Humain, Numéro Spécial "Psychologie ergonomique de la programmation informatique"*, *51*(4), 351-362.

Gould, J. D., (1975) - Some psychological evidence on how people debug computer programs. International Journal of Man-Machine Studies, 7, 157-182.

Gugerty, L., & Olson, G. M. (1986) - Comprehension differences in debugging by skilled and novice programmers. In E. Soloway & S. Iyengar (Eds.), Empirical studies of programmers. Papers presented at the First Workshop on Empirical Studies of Programmers, June 5-6, 1986, Washington, DC. Norwood,N.J.: Ablex.

Hoc, J-M. (1985) - Psychologie cognitive et génie logiciel. Intellectica, 1, 21-36.

Kessler, C. M., & Anderson, J. R. (1986) - A model of novice debugging in LISP. In E. Soloway & S. Iyengar (Eds.), Empirical studies of programmers. Papers presented at the First Workshop on Empirical Studies of Programmers. June 5-6, 1986, Washington, DC. Norwood,N.J.: Ablex.

Michard, A. (1975) - Analyse du travail de diagnostic d'erreurs logiques dans un programme FORTRAN (Rapport INRIA CO7602-R48). Rocquencourt: INRIA.

Pennington, N. (1982) - Cognitive components of expertise in computer programming: a review of the litterature (Technical Report #46). Michigan: University of Michigan.

Reiter, R. (1985) - A theory of diagnosis from first principles (Technical report N°187/86). Toronto: University of Toronto, Dept. of Computer Science, December.

Sheil, B. A. (1981) - The psychological study of programming. ACM Computing Surveys, 13, 101-120.

Vessey, I. (1985) - Expertise in debugging computer programs: A process analysis. International Journal of Man-Machine Studies, 23, 459-494.

Visser, W. (1986) - Etude de deux étapes de la conception: écriture et mise au point d'un programme (Rapport interne). Rocquencourt: INRIA.

Visser, W. (1987a) - Abandon d'un plan hiérarchique dans une activité de conception - Giving up a hierarchical plan in a design activity. Actes du colloque scientifique COGNITIVA 87 (Tome 1). Paris: Cesta.

Visser, W. (1987b) - 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.