Des mots aux bits : quelques considérations en marge de culture et technique du point de vue de l'ingénierie des logiciels

Ivan Maffezzini, Institut Trempet, Université du Québec à Montréal, Maffezzini.Ivan@Uqam.ca
Alice Premiana, Institut Trempet, Université du Québec à Montréal,
URL : http://www.trempet.uqam.ca

« *La technique seule est absolument universalisable parce que ce qui, de l'homme, résonne en elle, est si primitif, si près des conditions de la vie, que tout homme les possède en soi.* »
(Gilbert Simondon, *Les limites du progrès humain*)

RÉSUMÉ. Culture, science et technique forment une triade où la technique fait souvent figure de parent pauvre. Le génie logiciel nous semble occuper une place particulière parmi les techniques et c'est à partir de cette place particulière que nous essaierons d'indiquer quelques pistes de réflexion. Nous montrerons comment le langage ordinaire est au centre du travail de l'ingénieur du logiciel et comment cette centralité est constitutive de toute automatisation du réel. Cette caractéristique unie à l'ubiquité de l'informatique dans les processus d'automatisation permettra de montrer comment le travail d'automatisation déborde dans des champs propres à la « culture ». La parole en génie logiciel acquiert un pouvoir presque magique, ce qui, loin de vouloir dire que les systèmes informatiques comprennent le langage naturel, implique que les humains doivent distiller le langage. Cette distillation ne permet pas de trouver une « essence » mais un noyau dur qui alimente les machines. Comment enseigner l'autonomie de la parole avec ses restes non « machinisables » et, en même temps, la précision algorithmique qui seule peut permettre de résoudre un « problème » ? C'est là la question.

Mots clés : Génie logiciel, Culture, Technique, Machine, Automatisation, Langage.

1 AVANT PROPOS

Le génie logiciel (GL) nous semble occuper une place particulière parmi les professions modernes à haut contenu technique. Cette place particulière est due d'une part au fait que l'informatique est omniprésente (et donc qu'elle n'a pas de place propre), et de l'autre au fait que, dans le GL, on ne manipule pratiquement que des mots. Mais, ne pas avoir de place fixe et manipuler des mots, n'est-ce pas le propre de la culture que, de façon assez cavalière, trop souvent, les « humanistes » opposent à la technique ?

Nous n'aurions jamais osé aborder une telle thématique si une considération de Gilbert Simondon ne nous avait pas donné l'élan initial [1] : « *La culture doit être contemporaine des techniques, se reformer et reprendre son contenu d'étape en étape. Si la culture est seulement traditionnelle, elle est fausse, parce qu'elle comporte implicitement et spontanément une représentation régulatrice des techniques d'une certaine époque ; et elle apporte faussement cette représentation régulatrice dans un monde auquel elle ne peut s'appliquer*. »

Nous ne sommes pas assez naïfs pour imaginer régler en quelques lignes le conflit millénaire qui oppose les arts libéraux aux arts mécaniques, ni assez prétentieux pour vouloir pondre un essai d'épistémologie qui impose de nouveaux canons d'analyse. Nous aimerions plus humblement indiquer une piste ou deux de réflexion pour apporter de l'eau au moulin d'une technologie[1] qui, sans être arrogante, ne craint pas de se confronter à quelques grands thèmes philosophiques

2 INTRODUCTION

Le syntagme « génie logiciel » est plus problématique qu'il n'en a l'air, surtout si l'on considère la guerre de tranchée que, depuis plus de trente ans, se livrent les ingénieurs du logiciel et les informaticiens. Même si nous avons d'énormes doutes sur le domaine du GL tel qu'il est presque universellement accepté et défini, de manière fort démocratique, en [2], nous nous reconnaissons complètement dans la définition de [3] qui est pourtant à base de [2] : « *L'application d'une approche systématique, maîtrisée, quantifiable au développement, à l'exploitation et à la maintenance du*

[1] Dans cette communication nous employons « technologie » selon l'acception primitive de « science des technique » et non selon l'acception, plus répandue, d'« ensemble de techniques ».

logiciel (…) ». C'est surtout ce « maîtrisé » appliqué à l'approche, c'est-à-dire à la démarche pour approcher le but, et non au but lui-même, qui indique qu'il s'agit de génie plutôt que d'informatique[2]. C'est ce « maîtrisé » aussi qui, parmi les technophobes, fait faire gorge chaude de la technique. N'est-il pas vrai, nous disent-ils, que les techniciens, en maîtrisant les méthodes, ne maîtrisent rien ? que, si les buts sont hors de contrôle, la maîtrise de la démarche pour les atteindre n'est qu'illusion ? Pire encore, que la maîtrise de l'approche favorise une sous culture qui, dans le meilleur des cas, mélange les moyens et les buts mais qui, normalement, prend les uns pour les autres ? Il faut ajouter que, pour bien des technophobes, le point crucial n'est sans doute pas que le génie ne contrôle que les méthodes — cela, à la limite, les rassure — mais que les buts soient complètement hors de contrôle : entre les mains d'un hasard technicien qui exploite toute les possibilités pour créer de nouveaux outils ou de nouvelles machines qui n'ont d'autres buts que le « progrès » technique.

La maîtrise des approches est commune à tous les génies ; ce que nous devrions plutôt considérer, c'est ce sur quoi l'approche s'applique : le développement et la maintenance des *logiciels*.

Qu'est-ce que le logiciel quand on le regarde du point de vue de l'ingénieur ? Si l'on considère la définition de [3] il s'agit de : *programmes, procédures et documentation associée et* [de] *données qui concernent le fonctionnement d'un système informatique*. Dans cette définition, il est important de d'insister sur *documentation*, non pas pour affirmer une énième fois que la documentation est importante (dans quel domaine technique ne l'est-elle pas ?) mais parce que, dans le GL, les artefacts qui composent la documentation sont plus concrets que l'artefact final — au moins en tant qu'artefact qui passe entre les mains et, surtout, sous les yeux des ingénieurs du logiciel. Cet aspect de « concrétude » de la documentation, uni à l'ubiquité du logiciel et, surtout, à la centralité de la parole nous semble être suffisant pour caractériser le GL et permettre d'avoir un point de vue différent sur les rapports entre culture et technique.

La parole en GL acquiert un pouvoir presque magique, ce qui, loin de vouloir dire que les systèmes informatiques comprennent le langage naturel, implique que les humains doivent distiller le langage. Cette distillation ne permet pas de trouver une « essence » mais un noyau dur qui alimente les machines. Quelle est l'implication de tout cela dans l'enseignement ? Comment enseigner l'autonomie de la parole avec ses restes non « machinisables » et, en même temps, la précision algorithmique qui seule peut permettre de résoudre un « problème » ? Est-il suffisant d'introduire un cours d'épistémologie ou de « science et société » pour que les ingénieurs du logiciel puissent situer leur métier dans la société ? Difficile d'essayer de répondre à ce genre de questions sans se questionner sur le travail en GL et sans tâcher de réfléchir sur une façon de le rendre plus à l'écoute du réel qu'il veut/doit plier. C'est en regardant d'un peu plus près ce qu'est le GL que l'on pourra voir s'il a une contribution quelconque à donner dans le débat qui oppose culture et technique. Mais cette contribution ne pourra être décelée qu'en analysant les caractéristiques qui le rendent si spécial parmi les techniques modernes. Analyse à laquelle nous espérons donner une petite contribution.

Pour finir, quelques mots pour nous situer dans le débat entre technophobes et technophiles. Il est évident qu'en tant qu'ingénieurs du logiciel la technophobie ne nous guette guère mais, surtout, en tant qu'ingénieurs du logiciel, nous ne voyons aucun danger dans la maîtrise des approches ; au contraire, nous y retrouvons une exigence minimale d'efficacité[3]. Et c'est aussi en tant qu'ingénieurs que nous sommes conscients que la maîtrise (même celle des approches !) n'est toujours que partielle et que nous sommes loin de croire que la quantification soit une panacée comme les « hommes de culture » pensent que les techniciens le pensent. Affirmer ne pas être technophobe n'implique aucune technophilie. Ce qui n'est pas toujours évident dans un débat où technophobie et technophilie polarisent les intervenants. S'il était imaginable de tracer une ligne qui unit (sic!) technophobie et technophilie, nous nous situerons sans doute quelque part après la moitié, du côté de la technophilie

3 AUTOMATISATION : VUE D'ENSEMBLE

Le GL est important du point de vue technique et culturel (et économique, il va sans dire) parce qu'il est un constituant fondamental de l'automatisation. Nous considérons *automatisation* dans une acception très générale, comme le processus (ou le résultat du processus) qui permet aux éléments physiques du monde de s'influencer mutuellement sans que l'intervention humaine soit essentielle. Cette *non importance* de l'intervention humaine ne signifie pas que l'automatisation se fasse sans l'humain mais que l'intervention humaine a eu lieu avant, lors de la conception et la réalisation du système. L'homme a donné ses instructions (au sens propre lorsqu'il s'agit d'automatisation informatisée) aux machines afin qu'elles échangent entre elles et avec le monde extérieur sans besoin de son intervention en temps réel.

[2] Pour ne pas embrouiller une terminologie déjà assez embrouillée, nous ne considérons pas que l'on emploie aussi l'expression « génie informatique ».

[3] Avec le mot *efficacité* nous donnons à ceux qui en ont envie le bâton pour nous faire battre.

Ce qui ne veut pas dire que d'autres êtres humains — les utilisateurs — n'interviennent pas en interagissant avec les machines.

L'automatisation qui nous intéresse est celle qui contient du logiciel, ce qui, au stade actuel de l'évolution de la technique, implique pratiquement toutes les automatisations.

La figure 1, tirée de [4], montre un cycle simplifié de l'automatisation telle que nous l'envisageons.

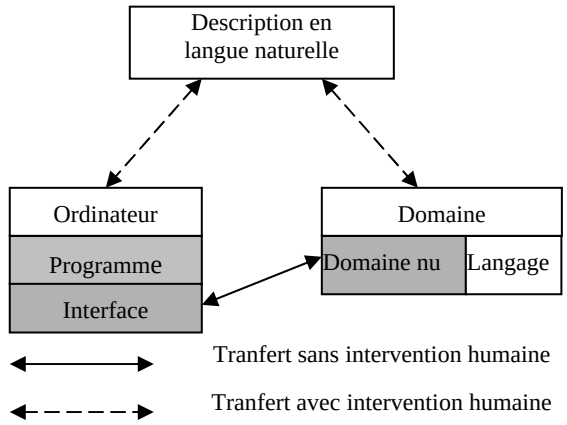

Fig.1 : Cycle d'automatisation

Le *monde* est « *la réalité physique enveloppée par le langage naturel* » et le *domaine* « *une partie du monde, un point de vue langagier sur la réalité* ». Puisqu'un domaine implique le langage comme élément d'observation, tout domaine est un domaine de connaissance.

Aidé par des échanges avec les utilisateurs et les clients, s'appuyant sur des études, analysant le vieux système… l'ingénieur du logiciel et/ou l'expert du domaine décrivent les exigences d'automatisation et les caractéristiques à automatiser dans une *Description en langue naturelle*. Même si cette description contient souvent (pratiquement toujours) des schémas et des dessins, cela ne la transforme pas en une description formelle. À partir de la *description en langue naturelle*, il y aura les étapes « normales » pour construire un programme informatique : conception[4], codage et tests. Le programme, une fois compilé et lié aux bibliothèques, à l'aide des interfaces physiques de l'ordinateur, échangera les données avec le domaine nu —le domaine sans le langage.

Nous avons déjà parlé de l'ubiquité de l'informatique et, en effet, ceci est un constat empirique facile à faire, mais nous aimerions le transformer, comme nous l'avons fait en [4], en un principe :

« *Tout domaine contient au moins une partie automatisable.*

Ce principe dit que quelle que soit la façon de diviser le monde, quelle que soit la dimension du domaine, il contiendra toujours des éléments qui peuvent être mis en relation avec des éléments d'une machine, et cela de manière telle que le « tout » fonctionne avec un certain degré d'autonomie. »

Ce principe a comme conséquence que : « *Le génie logiciel n'est pas un génie comme les autres en raison du nombre illimité de domaines pour lesquels on peut produire des logiciels et de l'extrême facilité à générer des copies des exécutables.* » [4] Ce qui permet de libérer le génie logiciel de l'analogie facile, mais à notre avis contre-productive, avec les génies traditionnels : l'ingénieur du logiciel. comme le commun des mortels, exploite surtout ses capacités langagières qui lui permettent de se mouvoir dans le monde en s'adaptant aux exigences changeantes des domaines.

Ce n'est pas un hasard si, dans la figure 1, la *description en langue naturelle* est au centre : elle est au centre de la transformation que les humains opèrent avant de laisser les machines et le domaine nu échanger entre eux. Qu'il s'agisse d'une langue naturelle et non d'une langue formelle, est d'une importance capitale, même si cela n'implique pas que, dans le passage du domaine au code du programme, on ne puisse pas avoir des langues formelles. Dans notre approche, s'il y a une formalisation, elle se situe dans la partie gauche entre la *Description en langue naturelle* et l'ordinateur. On pourrait ici nous accuser de pétition de principe : pour démontrer l'importance des mots dans le GL, on met au centre une description fondée sur le mots du langage du domaine[5]. Il est vrai que nous n'avons pas démontré que le langage naturel est au centre, mais nous croyons que la démonstration n'est pas nécessaire (et en cela nous nous situons du côté des « humanistes ») : si le domaine est enveloppé par le langage, il n'est pas possible de passer du domaine à un langage formel sans passer par une description en langue naturel qui limite les possibilités inscrites dans le domaine[6]. La description pourrait être dans la tête d'un individu, ou seulement sonore, mais elle reste une description ; le fait de la coucher sur papier ou sur écran est un besoin en vue de maîtriser l'approche.

On a une description en langue naturelle même quand on construit un pont, un avion, une fenêtre… Pourquoi

[4] Dire, comme on le fait d'habitude, que l'on conçoit un logiciel est un abus de langage si l'on accepte la définition de « logiciel » de IEEE que nous avons rapportée ci-dessus.

[5] Le langage du domaine est un sous-ensemble du langage naturel avec des extensions souvent jargonnantes.

[6] Inutile de dire que même dans le domaine le plus formel qui existe, celui des mathématiques, la langue naturelle est fondamentale pour comprendre et faire comprendre.

donc cette insistance à propos du logiciel ? Parce que, à cause de la flexibilité des programmes et de l'illusion que l'on peut pratiquement tout faire avec le logiciel, il est nécessaire de comprendre le domaine et les exigences d'automatisation en les « fixant ». Et les humains disposent d'un « mécanisme » de compréhension et de fixage très puissant qui est la langue. Tout cela est moins vrai quand on construit une voiture ou une fenêtre parce que, entre autres, les voitures et les fenêtres, on peut les toucher (et elles peuvent nous toucher) tandis que les bits ne nous touchent pas, sinon indirectement quand ils sont manipulés par des ordinateurs.

4 OBJETS TECHNIQUES

En [1] Simondon travaille à faciliter la rencontre entre technique et culture en étudiant les objets techniques qu'il ne considère pas, comme cela semblerait plus naturel, surtout à des ingénieurs du logiciel, en les classifiant selon leur usage ou leur structure, mais en analysant leur genèse. « *L'emploi de la méthode génétique a précisément pour objet d'éviter l'usage d'une pensée classificatrice intervenant après la genèse pour répartir la totalité des objets en genres et en espèces convenant au discours*. (…) *L'être technique* (…) *ne peut être objet d'une connaissance adéquate que si cette dernière saisit en lui le sens temporel de son évolution*. » Et l'objet technique, pour Simondon c'est une automobile, un moteur, un amplificateur ou un diode. L'objet technique de Simondon est un « véritable » objet : c'est-à-dire quelque chose qui affecte les sens. Mais nous croyons de ne pas nous tromper en disant que, pour Simondon, les dessins du moteur, même si ce sont des objets et qu'ils sont « techniques », ne sont pas des « objets techniques ». Si l'on voulait suivre Simondon dans sa démarche si prometteuse, on devrait se demander si, parmi tous les artefacts du GL, il y a des objets techniques ou si ne sont objets techniques que les ordinateurs avec leurs programmes exécutables. Étant donné que tous les artefacts produits par l'ingénieur du logiciel sont des descriptions bien plus proches des dessins du moteur que du moteur, est-il possible, en GL, de parler d'objets dans le sens de Simondon (qui est aussi le sens commun) ? Considérons le logiciel pour la conduite des postes d'EDF (Énergie de France), par exemple et oublions les objets techniques ordinateurs et réseaux locaux qui permettent au logiciel d'exécuter les fonctions voulues[7]. Nous ne considérerons que trois artefacts : la *Description en langue naturelle*, la *Description de la conception* (l'équivalent strict des dessins du moteur) et le listage. Même si cela peut paraître étonnant, nous ne considérons pas le programme exécutable, d'une part parce qu'il s'agit d'un artefact généré par d'autres programmes et qui donc ne concerne l'ingénieur du logiciel que comme élément de validation de son travail, et d'autre part parce que le programme exécutable est un objet qui ne touche les sens que par l'intermédiaire de l'ordinateur. Cela n'implique pas que le programme exécutable soit sans importance, loin de là, mais son importance concerne surtout les utilisateurs, les vendeurs. Si l'on suivait l'évolution de ce type de logiciel, on pourrait certainement apprendre beaucoup sur la technique informatique, sur celle des postes électriques et des télécommunications. On verrait comment, pour rendre plus concret[8] l'objet disjoncteur, par exemple, il n'a pas suffit d'établir de nouvelles normes et d'intégrer des microprocesseurs, mais que chaque artefact ait subi des évolutions profondes. Dans la description en langue naturelle, on peut voir que l'importance des exigences non fonctionnelles a augmenté par rapport aux exigences fonctionnelles ; que l'on a introduit des diagrammes conceptuels à la place des organigrammes ; que l'on a fait une place plus grande aux rôles des utilisateurs… L'analyse de l'histoire d'un tel artefact permet d'entrer dans une culture technicienne et dans ses intersections avec la « culture » bien plus facilement qu'en faisant fonctionner un disjoncteur ou en l'observant fonctionner.

Mais il est encore plus intéressant de considérer l'augmentation de l'intérêt pour la traçabilité. Que dire du fait que la traçabilité[9] soit devenue si importante ? que c'est tout à fait normal ? que c'est vrai dans toutes les activités de génie ? Oui. Mais, dans le GL, la quantité des changements possibles est telle que, pour paraphraser un célèbre homme politique, la quantité devient qualité. En termes plus crus : il est faux d'affirmer qu'en GL cela se passe comme dans les autres génies. On peut constater cela aussi en considérant la plus grande variabilité des objets informatique dans un domaine donné. Dans le génie mécanique, par exemple, on peut étudier la genèse de l'objet moteur en prenant en considération des voitures Honda, Renault ou Ford, sans que les conclusions auxquelles nous arrivons changent beaucoup. Est-ce qu'en étudiant plusieurs applications de conduite des postes on arriverait aux mêmes conclusions ? Certainement pas : pour que les application se ressemblent comme se ressemblent les moteurs, il est

[7] Ici il serait possible d'ouvrir une parenthèse, pas si oiseuse, en se demandant s'il est préférable de dire que c'est le logiciel qui permet à l'ordinateur d'exécuter les fonctions ou s'il s'agit du contraire.

[8] Concret dans le sens de Simondon : c'est-à-dire un objet où « *le schème de fonctionnement incorpore les aspects marginaux ; les conséquences qui étaient sans intérêt ou nuisibles deviennent des chaînes de fonctionnement*. »

[9] La traçabilité indique le degré de facilité dans le suivi des impacts des changements dans un artefact sur les autres artefacts.

nécessaire d'imposer des standards externes. Si on laisse évoluer les applications « librement » il est presque certain que l'on se retrouve avec des applications qui se ressemblent, du point de vue de la structure et de l'évolution, comme un moteur Honda et un… Bonobo.

En radicalisant la position de Simondon et en considérant la description en langue naturelle comme un objet technique, on risquerait sans doute d'ouvrir des pistes qui ne mènent pas forcément nulle part.

Avant de poser quelques questions sur l'enseignement, nous allons faire un détour par la spécialisation qui est ce qui s'oppose le plus à l'ubiquité et par l'étonnement qui ne se trouve pas qu'aux sources de la philosophie.

5 SPÉCIALISATION

La spécialisation est une caractéristique fondamentale de la science, que les formules percutantes du genre : « l'hommes de science connaît tout sur rien » et « l'homme de lettres ne connaît rien sur tout » soulignent de façon assez paradoxale. Le travail de l'ingénieur du logiciel, dans la phase qui permet de préparer la description en langue naturelle, comme celui de l'homme de lettres, demande de ne « connaître rien sur tout ». Ce qui implique qu'il doit « connaître » surtout le langage naturel et, à l'intérieur de celui-ci, déployer les capacités non spécialisées propres à l'être humain. Les connaissances de l'informatique, pendant cette phase, ne représentent qu'un poids et un défaut, et créent souvent des produits de piètre utilisation et d'utilité douteuse, comme tout gestionnaire de projet a sans doute eu souvent l'occasion de le constater.

Si le « ne rien connaître » de l'homme cultivé peut conduire à des joutes verbales interminables (ce qui constitue, en un certain sens, le sel des rapports humains), les joutes des ingénieurs du logiciel doivent conduire à une machine qui interagit avec des éléments concrets du monde. Il faut donc que la spécialisation refasse surface, pas forcement dans la même personne physique, mais dans le travail d'automatisation. Voilà donc comment « tout sur rien » et « rien sur tout » dans leurs interactions projectuelles, dans leur devenir, permettent de créer quelque chose de « solide » : une machine qui exécute des tâches concrètes en interagissant avec un réel qui lui est, de prime abord, complètement extérieur. Une machine qui, par la suite, sera elle-même un élément du réel à respecter et par rapport auquel il faudra, à l'aide du langage[10], construire une nouvelle machine

Cette dynamique entre spécialiste et homme-de-langage est une autre ouverture à notre avis assez caractéristique du génie logiciel permettant de penser une nouvelle culture technicienne qui, loin de mépriser le bavardage de ceux qui n'ont pas de professions scientifiques ou techniques, en assimile les « jeux de langage »[11].

6 ÉTONNEMENT

Cette considération de Proust que « *l'étonnement nous est souvent donné de rencontrer des abstractions réalisées* » s'applique parfaitement au génie logiciel quand la machine que l'on vient de programmer interagit avec le réel comme nous l'avions « abstraitement » pensé.

C'est cet étonnement qui nous permet un premier lien avec une technologie et donc une pensée de la technique. Il s'agit de quelque chose qui s'apparente à l'étonnement dont parle Platon. C'est l'étonnement qui, comme pour Théétète, est le prélude au « vertige » devant l'abîme que les mots creusent. Le vertige de découvrir ce qui est au-delà des mots, ce qui est au-delà de la mécanique, ce qui unit déterminisme machinique et liberté des paroles. Il a beau avoir trente ans que vous écrivez des programmes, coté étonnement, rien ne change. Quand, après des mois de discussions et d'écriture, vous branchez l'ordinateur à un alternateur et, qu'après les tests de routine des LEDs, « tout » fonctionne pendant quelques minutes, avant que la satisfaction ne s'installe, vous êtes envahis par une bouffée d'étonnement. Vous n'en croyez pas vos yeux. La machine réagit. La machine ordinateur et la machine alternateur se « parlent ». Comment se fait-il que, seulement avec des mots, l'on puisse agir sur la réalité physique ? la plier à son vouloir ? Que les mots soient des « outils » très puissants pour agir sur les humains n'a rien d'étonnant : c'est la banalité du monde ; mais que ces mêmes mots puissent modifier le comportement d'objets sans vie relève de la magie. Rien que des mots et l'alternateur démarre selon les règles établies par EDF, rien que des mots et le disjoncteur se ferme et s'ouvre trois fois avant d'attendre que le défaut disparaisse. Rien que des mots. Rien que des mots ? Quels mots ? Comment les agencer ? De quoi ont-ils besoin ? Voici ce que l'étonnement nous indique. Voici une porte ouverte vers l'extérieur de la technique que la technique contribue à former.

7 ENSEIGNEMENT

Les difficultés que l'enseignement du GL rencontre sont reliées surtout à la nécessité de faire continuellement basculer les étudiants d'une approche technico-opératoire porteuse de résultats immédiats vers une approche du genre « jeux de langage » et

[10] À ne pas confondre avec ce que, à tort, on appelle langages de programmation et qui ne sont que des langues de programmation.

[11] Il est clair que le spécialiste aussi doit maîtriser le langage naturel, mais tandis que, dans la science ou dans les génies traditionnels, cette maîtrise est à l'arrière plan, dans le génie logiciel elle est à l'avant plan.

vice-versa. La critique simple et non articulée de la majorité des étudiants, « le génie logiciel, ce n'est que du bla-bla », est en effet une critique très sensée qui ne peut être balayée du revers de la main avec une attitude du genre : « ils n'ont pas envie de travailler » ou des considérations paternalistes : « un jour vous comprendrez que j'avais raison » ou des relents de mauvais élitisme : « ils arrivent à l'université sans formation de base ».

Dans le GL, il y a du « bla-bla » et du technique et les deux s'enchevêtrent, se supportent, se heurtent, se complètent. Et ce sont ces interactions complexes qui permettent (ou qui devraient permettre) de faire comprendre aux étudiants que le langage naturel (le « bla-bla »), quand il est maîtrisé (encore !) est un mécanisme d'appréhension du monde : un filet dont les mailles, plus ou moins larges, permettent d'emprisonner différents types de concepts.

Quoi de plus frustrant pour un enseignant que la situation où la majorité des étudiants ne veut pas le suivre lorsqu'il s'apprête à changer les dimensions des mailles pour prendre des poissons-concepts qui leur semblent sans intérêt et qui sont essentiels pour comprendre le domaine ?

Quoi de plus décevant que des étudiants qui savent tenir des discours sur tout mais sont incapables d'écrire la plus petite procédure informatique ?

Comment faut-il faire pour que le GL, technique à la lisière des techniques, ne soit pas perçu comme le règne de la confusion et de l'à peu près par des étudiants fascinés par la technique ? Il faudrait sans doute montrer que le « bla-bla » peut être structuré, clair et puissant comme un programme et que cela a été fait par bien des êtres humains depuis des millénaires.

Comment montrer aux étudiants hostiles au codage qu'écrire du bon code ne relève pas de la mécanique mais de la créativité et de la capacité de systématiser.

Valéry écrivit : « Je suis né plusieurs, je meurs un[12] », comme faire pour que les étudiants ne deviennent pas « un », ne meurent pas en esprit, mais restent, sinon plusieurs, au moins deux ou trois pour pouvoir suivre les différents aspects de l'automatisation ?

En tant qu'art libéral (dans la phase qui va du domaine à la description en langue naturelle), le GL doit sans doute employer les méthodes universitaires classiques (si quelque chose de ce genre-là existe encore) mais, en tant qu'art mécanique, il a besoin de l'exemple, de fonctionner dans les ateliers comme les apprentis peintres de la Renaissance. Seul l'apprentissage par imprégnation nous semble pouvoir former au métier, ce qui devrait faire réfléchir ceux qui, sensibles aux coûts, n'ont d'oreilles que pour le *e-learning*[13].

Comment donner aux ingénieurs du logiciel la déformation professionnelle qui seule permet de rendre automatiques certaines réactions de base sans que cette déformation ne rende inutilement figé le regard sur le domaine à automatiser ?

8 CONCLUSION

Dans la section précédente, nous avons posé des questions qui sont loin d'être rhétoriques et pour lesquelles nous n'avons que des bribes de réponses, mais cela ne nous empêche pas d'essayer de proposer, à titre de conclusion, quelques considérations plus optimistes.

1) Nous avons insisté sur le fait que la *Description en langue naturel* contient des schémas et des dessins. Pourquoi cette insistance ? Pour corriger le titre de l'artefact qui ne semble pas laisser d'espace à autre chose que les mots ? Sans doute, mais surtout parce que, si — comme il est écrit dans la première définition d'objet du *Trésor de la langue Française* — un objet est : « *Tout ce qui, animé ou inanimé, affecte les sens, principalement la vue* », alors la vue est plus fortement affectée par des rectangles qui représentent des concepts que par leurs noms (même s'ils sont écrits en caractères italiques ou gras). Ce qui ne fait que confirmer que les objets de la *Description en langue naturelle* sont « plus objets » que les objets du listages et, surtout, plus objets que les objets binaires incapables, eux, d'affecter les sens sinon en passant la commande au matériel. Les rectangles UML qui représentent les concepts et les lignes qui représentent les associations, par exemple, donnent un sens de solidité plus grand que les mots : les mots et les liens entre les mots dans les phrases sont perceptuellement « flous » tandis que les lignes qui renferment les concepts et qui les relient dans un diagramme sont, du point de vue perceptif, solides et précises.

2) Il nous semble fort probable que le GL aidera à revivifier les méthodes et les « cultures » d'un passé qui semblait être surtout un creuset de technophobie. Et si cette revivification fait un long détour à travers les techniques les plus modernes si chères aux technophiles, ce ne sera qu'une confirmation que, comme l'écrit Simondon : « *La culture doit être contemporaine des techniques, se reformer et reprendre son contenu d'étape en étape.*

3) Nous voyons dans la formation des ingénieurs du logiciel la nécessité, non artificiellement imposée par un élitisme gratuit, de fabriquer des personnes « cultivées », capables de créer des objets techniques et qui savent respecter ce nouveau monde où les objets,

[12] Je cite de mémoire.

[13] Qui, à notre avis, est bien plus difficile que le *e-teaching* !

indépendamment du fait qu'ils soient techniques ou non, se partagent l'espace et portent inscrites dans leur structure la mémoire du passé. Mais pour en arriver là, il ne suffit pas d'ajouter quelques cours de sciences humaines dans un curriculum à contenu scientifique et technique ou vice-versa. Il faut que les enseignants soient eux-mêmes capables de « bla-bla » et de technique, ce qui est loin d'être la règle. Que faire ? Attendre ? Attendre que les conditions « objectives » nous obligent à aborder différemment l'enseignement ? Sans doute, si l'on considère que les enseignants actuels sont souvent ou bien dans le champs du « tout sur rien » ou bien dans celui du « rien sur tout » et qu'ils sont incapables de satisfaire aux requêtes d'une automatisation imbriquée dans la culture et non asservie à des exigences économiques myopes.

4) Le fait que dans le GL il n'y ait pas de matière palpable favorise, hélas ! la croyance qu'il n'est pas un métier ; ne devrons-nous pas, en regardant l'évolution du travail, plutôt penser qu'il y aura toujours moins de métiers avec de la matière palpable ?

5) Le GL avec son « bla-bla » validé par le réel qui entoure la machine pourrait être le terrain d'essai des nouveaux rapports entre culture technicienne et la Culture. Il pourrait nous aider à accepter que le « bla-bla » sans fin dans la cité puisse ne pas porter des résultats hors du langage tandis que le « bla-bla » dans le GL peut avoir sa fin hors des paroles, dans le geste technique. D'une technique qui, un jour, pourrait conquérir sa place dans le monde des symboles duquel elle semble exclue depuis que l'homme est homme « *En effet, depuis le Paléolithique supérieur […] le monde des symboles (religieux, esthétiques, ou sociaux) a toujours hiérarchiquement prévalu sur le monde des techniques et la pyramide sociale s'est édifiée de manière ambiguë en donnant la prééminence aux fonctions symboliques sur la technologie, pourtant moteur de tout progrès*. » [5].

Bibliographie

1. Gilbert Simondon, *Du mode d'existence des objets techniques*, Aubier, 1989.

2. IEEE, Guide to the *SWEBOK, 2004 version*, http://www. swebok.org /

3. IEEE, Std 610.12-1990, IEEE standard Glossary of Software Engineering Terminology.

4 Ivan Maffezzini, Alice Premiana, Bernardo Ventimiglia, « *Prolégomènes à une critique du génie logiciel, Partie I : contextualisation* », *Génie Logiciel*, Septembre 2003, No 66.

5 André Leroi-Gourhan, *Le geste et la parole, Technique et langage*, Albin Michel, 1964.