Travailler avec un atelier de génie logiciel

Yves Constantinidis

On définit formellement le génie logiciel *(software engineering)* comme une approche systématique du développement, de l'exploitation, de la maintenance et de la mise à la retraite d'un logiciel. Le génie logiciel fait donc appel à une démarche globale, aussi bien dans le temps (bien avant la naissance et jusqu'à la retraite) que dans les divers métiers des personnes qui transforment une idée (par exemple : automatiser la gestion du restaurant d'entreprise) en un produit (le logiciel de gestion du restaurant d'entreprise). Cette démarche est systématique, c'est-à-dire qu'elle est codifiée et peut être répétée. Transformer une idée en produit, cela s'appelle développement de logiciel. Ce processus fait actuellement très souvent appel à l'intuition et au tâtonnement, de sorte qu'il est difficile de parler de génie logiciel au sens strict du terme.

Quant aux outils qui accompagnent ce processus, on les appelle tantôt outils de développement, tantôt outils, tantôt ateliers de génie logiciel. Comme nous le verrons, ces outils ont eux-mêmes été, très souvent, développés par tâtonnements, par un processus qui s'apparente plus à du bricolage de haut niveau qu'à une approche formelle et systématique. Pour cette raison, les outils qui méritent pleinement leur titre d'ateliers de génie logiciel sont l'exception plutôt que la règle.

LCN, volume 1, n° 3-2000, pages 15 à 26

L'existant

L'offre actuelle : plusieurs centaines d'outils

Etant donnée l'extrême variété d'outils développement sur le marché, il est quasiment impossible de les dénombrer avec précision. Cependant, les différentes études menées permettent de dénombrer, toutes spécialités confondues, plusieurs centaines d'outils se positionnant comme des outils de génie logiciel.

Qui les utilise ?

Les premiers outils de génie logiciel, au sens strict du terme, étaient utilisés pour la conception et la réalisation d'applications scientifiques ou d'applications pour lesquelles la sécurité était prépondérante (nucléaire, espace, aéronautique). Actuellement, outils et ateliers sont utilisés très largement, mais avec un sens plus large (et parfois abusif) donné au terme de génie logiciel. L'industrie représente encore plus du tiers des ventes d'outils de réalisation, talonnée par l'administration (25 %).

Pourquoi sont-ils choisis ?

Les bénéfices escomptés sont importants, mais les critères éliminatoires sont nombreux. Les critères de choix des ateliers indiqués par les utilisateurs sont révélateurs : pérennité du fournisseur, richesse de l'atelier et support des méthodes, simplicité et convivialité de l'atelier, formation des utilisateurs et réactivité du support client. La pérennité, la notoriété et la santé financière du fournisseur sont des critères déterminants, car l'atelier est une arme stratégique pour l'entreprise. La simplicité et la convivialité des outils sont également des critères importants ; ceux qui ont connu les outils de la génération précédente se méfient de leur lourdeur et de la difficulté de leur mise en œuvre.

Comment sont-ils utilisés ?

Une étude menée par Adeli en 1996 sur les outils de conception donne quatre raisons principales d'utilisation de ces outils [ADE 96] :

– l'aide à la conception pour la modélisation et la documentation,

– l'aide à la réalisation (l'outil de conception alimentant un outil de réalisation),

– l'aide à l'amélioration de la communication,

– l'aide à la capitalisation, la maîtrise et la réutilisation.

Les outils de réalisation répondent à des critères différents et sont plus choisis pour la productivité que pour la qualité. Cependant, les aspects conception et communication sont également considérés comme importants.

Répondent-ils au besoin ?

Les outils et ateliers apportent des services incontestables mais également quelques désillusions. Sélectionner, mettre en place et gérer au quotidien un environnement de développement est une tâche difficile nécessitant du savoir-faire. Il s'agit d'un véritable projet d'entreprise. Chaque étape doit être anticipée, planifiée, et élaborée avec soin :

– compte tenu de l'enjeu, le choix des outils constituant un atelier de génie logiciel demande souvent une étude de plusieurs jours, voire de quelques semaines ;

– les outils doivent être installés, configurés, testés et éventuellement adaptés à des besoins spécifiques ; comme tout logiciel, ils doivent être maintenus ;

– les personnes qui l'utilisent doivent recevoir la formation adéquate ; la courbe d'apprentissage est d'autant plus longue que l'outil est complexe ;

– l'arrêt de son utilisation pose des problèmes cruciaux, et son remplacement également.

Qu'est-ce qu'un atelier de génie logiciel ?

Que doit-on attendre d'un outil ?

Les outils de génie logiciel ne font pas de miracles. Ils aident ou assistent les différents acteurs sur le projet. Analysons leurs fonctionnalités de base :

– aide au dessin, pour les spécifications des schémas de conception (données et programmes), respectant des normes préétablies (par exemple Merise ou UML) ;

– aide à l'écriture des programmes : mise en forme du texte et correction des erreurs de syntaxe ;

– gestion de la cohérence des entités traitées par l'outil : par exemple contrôle de la cohérence d'un modèle de données ;

– stockage et restitution des objets : modèles de données, traitement, programmes, jeux d'essais, documentation, etc. ;

– automatisation des transformations : génération de code machine à partir de code source, génération d'ordres de création de tables relationnelles à partir des modèles de données ;

– information de l'utilisateur sur les activités de développement en cours ;
– analyse d'impact : prévision des conséquences d'une modification.

Qu'est-ce qu'un atelier ?

On peut donc définir un AGL comme un ensemble cohérent d'outils logiciels qui aident une organisation à analyser, concevoir, réaliser, maintenir et documenter ses logiciels, en accord avec ses besoins, sa politique, sa stratégie, son histoire et ses normes de qualité [CON 98]. Un atelier est donc un ensemble d'outils intercommunicants, grâce auquel plusieurs personnes, de disciplines et d'origines différentes, s'attellent à un projet commun.

Coordination, coopération et cohérence

Un atelier n'est pas un simple empilement d'outils, déconnectés les uns des autres. Il permet aux outils de communiquer entre eux, au travers d'une structure d'accueil sur laquelle différents outils viennent se brancher, ou d'un référentiel, zone de stockage de tous les objets manipulés et garant de la cohérence entre ces objets. De ce fait, l'atelier joue le rôle de secrétaire du projet, permettant à des personnes, même géographiquement dispersées, de travailler en équipe.

Un AGL n'est pas une usine

L'analogie d'un AGL avec une usine à logiciel *(software factory)* est impropre, car le processus de construction du logiciel est très différent de celui d'un bâtiment ou d'une automobile. Un logiciel n'est pas monté exemplaire par exemplaire, comme un objet technique matériel. La division du travail à la façon d'une usine serait contre-productive, car la construction du logiciel est essentiellement une activité de conception, faisant largement appel à l'autonomie et à l'initiative individuelle.

Un triple enjeu

Pour comprendre un outil de développement, il est important de l'examiner sous trois aspects :

– l'aspect méthodologique exprime la capacité de l'outil à décrire une situation, à établir des spécifications fonctionnelles, à favoriser l'application de procédures ;

– l'aspect technologique concerne l'architecture technique, l'infrastructure sur laquelle l'application cible va s'appuyer ;

– l'aspect lié à l'utilisation de l'outil concerne sa capacité à coopérer avec ses utilisateurs, ainsi qu'avec d'autres outils.

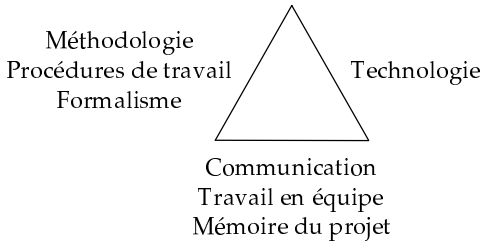

Figure 1. Les trois facettes d'un outil

C'est avec ces trois points de vue à l'esprit que l'on pourra donner à l'outil la place qu'il mérite dans l'entreprise. Or ces trois aspects sont souvent antagonistes. Les prendre en compte en permanence et simultanément, c'est effectuer des arbitrages.

Couverture des outils du marché

L'AGL idéal, parfait selon ces trois axes, est une utopie. Inversement, les outils totalement inadaptés au monde de l'entreprise sont beaucoup plus nombreux qu'on ne l'imagine. La figure 2 illustre la situation, en fonction de ces trois facettes et montre le danger de se polariser sur une seule d'entre elles.

Ceux qui ne couvrent qu'un axe sur les trois relèvent du gadget (il ne tient compte ni des méthodes en vigueur ni des contraintes technologiques), du fantasme méthodologique (faits pour et par des méthodologues, de plus en plus rares de nos jours) ou du délire technologique (développés à la hâte pour répondre à une mode à un changement technologie soudain). Il faut les éviter à tout prix.

Faute de trouver l'outil idéal, qui respecte les trois conditions du succès, il est toujours possible de faire des compromis :

– outils plutôt orientés technologie : certains outils privilégient la génération et la production au détriment de la démarche formalisée. C'est le cas de nombreux outils de réalisation. Ce peut être un bon choix, surtout pour des projets de taille moyenne ;

– outils plutôt orientés méthodologie : la plupart des outils connus sur le marché comme des outils de conception entrent dans cette catégorie ;

– ponts, référentiels et structures d'accueil : un pont permet de coupler deux outils, une structure d'accueil est une sorte de plate-forme sur laquelle viennent se brancher plusieurs outils.

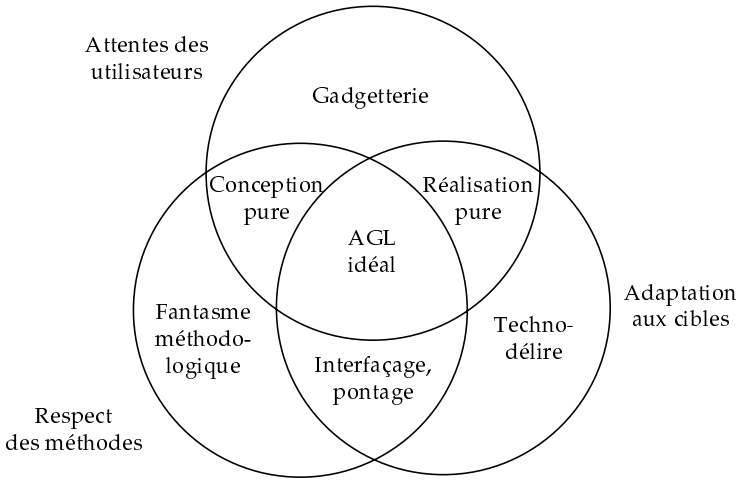

Figure 2. Adéquation des outils aux besoins des entreprises

Les utilisateurs des outils

Une équipe de projet est un peu comme une troupe de théâtre. Un acteur peut jouer plusieurs rôles. Chaque outil doit correspondre au besoin d'un rôle (un métier, une fonction) et non à celui d'un acteur. Nous allons énumérer les principaux rôles joués tout au long du projet, ainsi que les outils qui peuvent les accompagner.

Le chef de projet

Le chef de projet a un rôle précisément défini : il doit mener le projet à bien, en respectant les coûts, les délais et la qualité du produit final. C'est à la fois un chef d'orchestre (un coordinateur) et un chef d'entreprise (stratège, gestionnaire et tacticien).

Ses responsabilités au quotidien sont très variées :

– établir et défendre le budget du projet, en faisant l'adéquation coût/ressources ;

– planifier le projet, répartir les tâches dans le temps selon les contributeurs ;

– constituer l'équipe de projet ;

– anticiper les difficultés et y remédier ;

– documenter l'avancement du projet ;

– apporter du savoir-faire pour les projets suivants (capitaliser) ;

– dialoguer et négocier avec le client ;

– savoir à tout moment qui fait quoi et quand.

Les outils de gestion de projet ne peuvent évidemment assister le chef de projet dans toutes ces tâches, mais ils devraient l'aider dans les tâches de gestion et de planification. Or, les outils de gestion de projet actuellement sur le marché sont, paradoxalement, peu adaptés aux projets informatiques, car ceux-ci comportent un certain nombre de particularités, difficiles à prendre en compte. Un logiciel est un produit extrêmement complexe, modifié en permanence, même après sa mise en production. L'estimation des charges et des délais est beaucoup moins fiable que dans les autres industries. Les phases du projet (analyse, conception, réalisation) se chevauchent souvent. Les intervenants sont amenés à changer assez souvent de poste ou de métier, alors même que chacun d'entre eux est détenteur d'un savoir-faire important.

L'architecte de l'application et le concepteur

L'architecte définit et modélise une application ou un groupe d'applications dans sa globalité. Il a besoin d'outils lui permettant de modéliser les flots de données, entre des applications ou à l'intérieur d'une même application, ainsi que les processus de l'entreprise à un haut niveau d'abstraction.

Le concepteur modélise les traitements, les données, les flux de données, en général selon une méthode formalisée. Il recherche des outils qui lui permettent de se débarrasser de ses tâches les plus répétitives (en particulier le dessin, la mise en page, la génération de documentation).

L'objectif des outils de conception est de modéliser tout ou partie du système d'information d'une entreprise (données, traitements, flux de données…) et d'en documenter les règles de gestion et d'organisation.

Les outils de conception sont tous régis par les mêmes principes de base, mais ils diffèrent sensiblement dans leurs fonctions. Les outils les plus avancés sont de véritables bases de données documentaires du projet. Ceci explique des rapports de prix de un à dix.

Le réalisateur

Le réalisateur a pour responsabilité de livrer, dans un temps imparti, une partie d'application, selon des spécifications plus ou moins précises (comme nous l'avions écrit précédemment, les rôles entre les acteurs sont souvent flous).

Les outils de réalisation facilitent la tâche du programmeur ou remplacent certaines de ses activités. L'archétype de l'outil de réalisation est bien entendu le compilateur, c'est-à-dire un logiciel qui traduit un programme écrit dans un langage de programmation en langage machine. Cependant, les applications actuelles ont souvent une interface graphique, et les outils de réalisation intègrent un module de dessin de l'interface homme-machine, en liaison avec le corps de l'application. Le réalisateur a également besoin d'un outil d'aide à la mise au point *(debugger)* permettant de détecter et de corriger les erreurs de programmation.

Ces différents outils sont actuellement vendus sous forme intégrée. On peut donc déjà parler, à ce niveau, d'environnements de programmation.

Les équipes de test

Plusieurs personnes ou plusieurs équipes se partagent les activités de test, qui occupent une partie importante du cycle de vie du logiciel. On distingue trois niveaux de test principaux :

– unitaires (contrôle d'un composant de l'application) : ils consistent à vérifier la conformité d'un module à ses spécifications internes et sont souvent effectués par les développeurs eux-mêmes ;

– d'intégration (contrôle d'un sous-système) : le but ici est de vérifier les interfaces entre sous-systèmes. Les aspects performances sont également pris en compte à ce niveau ;

– de validation, contrôle du fonctionnement global de l'application et de la conformité de l'application aux spécifications externes. Elle peut revêtir un caractère contractuel.

S'y ajoutent les tests de non-régression, répétés à chaque sortie d'une nouvelle livraison. Ils permettent de vérifier, suite à des corrections ou des modifications, que les fonctionnalités n'ont pas été altérées et que les performances ne se sont pas dégradées suite à ces modifications.

On trouve donc sur le marché :

– des outils de « rejeu », permettant de répéter automatiquement les mêmes jeux de tests avec des paramètres différents (endurance) ;

– des simulateurs de charge : utilisés pour tester le comportement d'une application face à un grand nombre d'utilisateurs, de façon à prévoir les performances et le comportement ;

– des générateurs de jeux d'essais : ils fabriquent automatiquement les jeux d'essais qui viendront alimenter les outils précédents.

Le responsable de la gestion de configuration du logiciel

La gestion de configuration du logiciel est l'ensemble des activités qui permettent la décomposition d'un logiciel en éléments (articles de configuration), l'identification précise de ces éléments, le contrôle des évolutions de ces éléments et de l'ensemble, ainsi que le contrôle de la livraison des versions du logiciel.

Ces activités ne sont pas propres au développement de logiciel, mais leur application au logiciel comporte quelques particularités :

– le logiciel peut être facilement modifié ou détruit ;

– un même article (fichier source, image d'écran, type de donnée) peut être manipulé par plusieurs personnes ;

– une ou plusieurs personnes peuvent modifier des versions légèrement différentes d'un même article ;

– un logiciel est un produit complexe.

La gestion de configuration du logiciel est une activité transversale. Elle concerne l'ensemble des acteurs du projet. Sur un projet d'une certaine importance, c'est un métier à part entière.

La gestion de configuration du logiciel ne peut se faire correctement que si elle est bien outillée : les outils doivent être à la fois rigoureux pour l'organisation et peu contraignants pour les développeurs, ce qui représente un défi pour les fournisseurs de tels outils.

Les outils de gestion des versions et de la configuration doivent aussi gérer les changements, c'est-à-dire les demandes de modifications tout au long de la vie d'un projet. Ils doivent permettre de pister les erreurs et leur correction ainsi que les demandes d'amélioration et leur prise en compte. La gestion des changements et la gestion de la configuration sont donc deux activités interdépendantes.

La maintenance

La maintenance fait partie intégrante du processus de développement. Les besoins en outils de l'équipe de maintenance sont au moins identiques à

ceux de l'équipe de développement. Les équipes de maintenance doivent disposer de moyens sophistiqués d'analyse qui leur permettent de se replonger dans les programmes des autres. La gestion de configuration joue ici un rôle important.

Les outils de rétroingénierie (rétroconception ou *reverse engineering*) qui permettent de reconstituer les spécifications d'un programme, et les outils d'analyse de code, font également partie de la panoplie des équipes de maintenance.

L'avenir de la programmation

On croyait naguère que les outils de génie logiciel allaient enfin nous débarrasser de cette tâche à la fois pénible, coûteuse et risquée qu'est la programmation. Il suffirait alors d'exprimer des besoins, dans un langage presque naturel pour qu'un outil nous la traduise en une application, opérationnelle et sans erreurs. C'était là une utopie. Mais une utopie stimulante, qui a permis d'entrevoir les outils futurs et de faire progresser les outils actuels.

Le présent

Pour entrevoir l'avenir, observons les outils et langages de programmation d'aujourd'hui.

Les langages de quatrième génération. Le concept est fuyant, mais correspond néanmoins à une réalité. Un L4G est un outil qui génère automatiquement l'interface utilisateur (la partie visible de l'application), ou permet de la construire interactivement grâce à la programmation événementielle.

La programmation événementielle. Les outils permettant ce type de programmation, rendus populaires par Visual Basic, sont très répandus, y compris dans le grand public. L'idée consiste à associer des événements (clic, double clic, frappe au clavier, etc.) à des objets (bouton, menu, champ de saisie) et des morceaux de programme à ces événements. Les applications sont vite écrites, quoique souvent difficiles à maintenir.

La programmation visuelle. Certains outils génèrent automatiquement l'interface homme-machine à partir du modèle de données ou de traitement d'une application. Cette technique permet, dans l'idéal, d'obtenir une interface utilisateur cohérente avec le modèle de données ou de traitements.

L'avenir : un continuum entre conception et programmation

Les outils de l'avenir vont briser les barrières entre programmation graphique et programmation littérale, entre formalisation et communication interpersonnelle, conception et réalisation, en conciliant sécurité et créativité.

Entre graphique et littéral, entre conception et réalisation. En d'autres termes, il doit être possible de passer d'un modèle conceptuel graphique à un code écrit dans un langage de programmation, et inversement. Les produits actuels sont expérimentaux ou au stade de la recherche.

Entre formalisation et communication interpersonnelle. Pour passer sans transition de la conception à la réalisation, il est nécessaire d'utiliser le même langage pour formaliser un problème et pour communiquer autour de ce problème.

Apporter des vues multiples. De même qu'un bâtiment peut être représenté par plusieurs schémas selon différents points de vue, une application doit pouvoir être vue sous plusieurs angles. Cette possibilité est déjà souvent mise en œuvre sur les outils de conception, mais moins sur les outils de réalisation. La figure 3 donne une vision synoptique de l'outil de réalisation de demain.

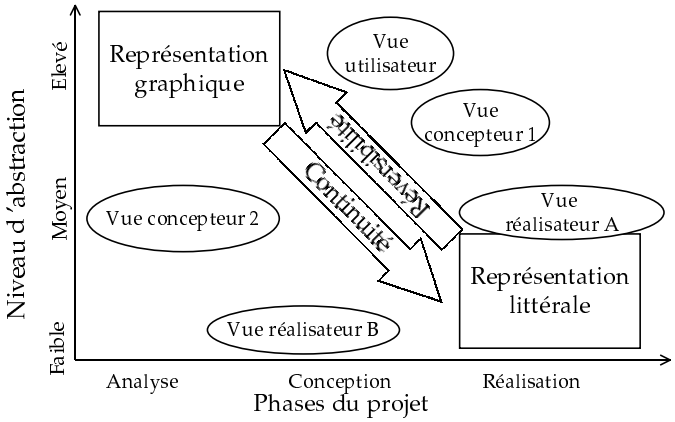

Figure 3. L'atelier de l'avenir

Bibliographie

[ADE 96] Adeli, *AGLoscope 1996*, Presses de l'Adeli, 1996.

[BRU 96] BRUCKHAUS T., MADHAVJI N.H., JANSSEN I., HENSHAW J., « The Impact of Tools on Software Productivity », *IEEE Software*, September 1996, p. 29-38.

[CON 98] CONSTANTINIDIS Y., *Outils de construction du logiciel,* Hermès, 1998.

[LEJ 97] LEJEUNE J.-M., « Le mythe des outils de gestion de projet », *L'informatique professionnelle*, n° 152, mars 1997.

[LEW 91] LEWIS T.G., *CASE: Computer-Aided Software Engineering*, Van Nostrand Reinhold, 1991.

[PRI 95] PRINTZ J., *Le génie logiciel*, « Que sais-je ? », PUF, 1995.

[SPU 92] SPURR K., LAYZEL P., *CASE, Current practice, future prospects*, Wiley, 1992.