Chapitre 1

Introduction

Ce qui est à fermer, il faut d'abord l'ouvrir,
D'abord consolider ce qui est à fléchir.
Lao-tzeu, *Tao-tê-king*

1.1. Quelques définitions

Avant de tenter de définir ce qu'est un outil de génie logiciel, un atelier de génie logiciel ou un environnement de développement, donnons quelques définitions qui vont nous faciliter la tâche par la suite.

Logiciel

On appelle *logiciel* la somme des programmes, procédures, règles de gestion, en rapport avec le fonctionnement d'un ordinateur, ainsi que la documentation et les données qui leur sont associées (définition IEEE).

Cette définition très formelle permet déjà de dégager une idée importante. Un logiciel n'est pas simplement un programme qu'on installe sur un ordinateur, ni une suite de lignes de code qui vont se transformer en programme. La documentation du logiciel, les structures de données, et même les spécifications du logiciel font partie du logiciel.

Génie logiciel

On appelle *génie logiciel* (en anglais *Software Engineering*) l'approche systématique du développement, de l'exploitation, de la maintenance et de la mise à la retraite d'un logiciel (définition IEEE).

Cette deuxième définition met en lumière une dimension importante et trop souvent négligée, qui permet déjà de comprendre la raison de la grande variété d'outils abordée dans cet ouvrage. Le génie logiciel accompagne le logiciel tout au long de sa vie, pour ainsi dire « du berceau au tombeau ». Cet aspect est systématiquement négligé par la plupart des auteurs, qui se contentent de parler de la conception et du développement.

Un autre terme important de cette définition est le mot *systématique*. Confrontés à cette définition, la plupart des outils dits de génie logiciel, n'en sont pas. Bien que la plupart d'entre ces outils soient compatibles avec une approche systématique, rien ne permet de garantir qu'ils forcent, ou même qu'ils encouragent, l'application d'une approche systématique.

Développement de logiciel

On appelle *développement de logiciel* le processus par lequel les besoins des utilisateurs sont transformés en spécifications, les spécifications en conception, la conception en code, et le code testé, documenté et validé en vue d'un usage opérationnel (définition IEEE).

Le développement est donc bien plus que la conception et la réalisation de logiciel. Il commence bien avant cette phase et se termine bien après. Cette définition du développement sera utile, car, tout en étant très globale, elle ne fait pas appel à la notion de rigueur, d'approche systématique, que l'on trouve dans le génie logiciel.

Pour cette même raison, il est à la fois plus exact est plus simple de parler d'outils et d'ateliers de développement, plutôt que d'outils de génie logiciel et d'AGL [1]. Dans cet ouvrage, nous allons décrire et étudier les outils qui aident ou assistent au développement de logiciel, au sens large du terme.

[1] Pour nous conformer aux usages, nous parlerons indifféremment d'outils (ou d'ateliers) de développement et d'outils (et d'ateliers) de génie logiciel.

1.2. Les trois facettes d'un outil

Le développement de logiciel peut être examiné sous trois angles : technologique, méthodologique, et humain :

– un logiciel ne tourne pas seul. Il s'appuie sur une infrastructure : un ordinateur bien sûr, un système d'exploitation, et de plus en plus souvent, une infrastructure de communication qu'il faut prendre en compte. De plus, il utilise des briques logicielles préexistantes : un SGBD, par exemple ;

– un logiciel est construit en respectant certaines méthodes, plus ou moins formalisées : méthodes de représentation (de modélisation des données, des traitements) méthodes de développement (étapes du cycle de vie) de tests, etc. ;

– enfin, on l'oublie trop souvent, un logiciel est créé par des êtres humains, non par des machines. D'où d'innombrables conflits, interprétations erronées, malentendus, erreurs d'inattention. Un outil doit tenir compte de la culture, de l'histoire, de ceux qui vont l'utiliser.

De manière analogue, pour comprendre l'outil qui va servir à développer ce logiciel, il est important de l'observer sous trois aspects :

– l'aspect technologique : comment prend-il en compte les technologies sur lesquelles s'appuieront les logiciels que l'on veut construire ? A-t-il le potentiel d'évolution pour prendre en compte les technologies du futur ?

– l'aspect méthodologique : est-il compatible avec les pratiques de l'entreprise ou du département ? Encourage-t-il les méthodes mises en place ?

– un troisième aspect, intrinsèque à l'outil : comment celui-ci est-il constitué en tant que logiciel ? Est-il ergonomique ? Comment dialogue-t-il avec ses utilisateurs ? Comment communique-t-il avec d'autres outils ?

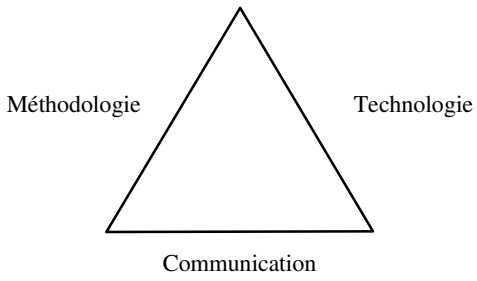

Figure 1.1. *Les trois facettes d'un outil*

1.3. Un outil de communication

Un outil de génie logiciel est un logiciel qui opère une jonction entre méthodologie et technologie. Par exemple, un outil de conception partira de modèles conceptuels, en entrée (aspect méthodologique), pour générer des ordres de création de tables relationnelles (cible technique). Mais il va également permettre à des concepteurs de communiquer entre eux, au travers de schémas conceptuels, soit générés par l'outil, soit en consultant directement l'outil.

De même, un outil de réalisation partira d'un langage de haut niveau pour générer un langage plus proche de la machine. Mais le but d'un langage de haut niveau (qu'il soit de troisième ou de quatrième génération, à supposer que ce terme ait un sens) n'est pas seulement de faire communiquer une personne avec une machine, mais également de servir de langue de dialogue entre plusieurs personnes. L'aspect communication correspond à une demande, souvent implicite, de la part des organisations utilisatrices d'outils.

1.4. Un choix stratégique

On choisit souvent un outil pour résoudre les problèmes du moment, ou pour un projet particulier. N'oublions cependant pas qu'il y a des outils à portée locale (l'équivalent de la pince ou du tournevis en mécanique ou en électricité) et des outils à l'échelle de l'entreprise (analogues à une machine-outil, voire un atelier complet).

D'autre part, acheter un outil, c'est introduire dans l'entreprise une nouvelle norme. S'il est vrai qu'aujourd'hui, un outil a une durée de vie de cinq ans au plus, son introduction va nécessairement entraîner l'achat de la version suivante, et de celle d'après, et ainsi de suite… de façon à rester compatible et à ne pas devoir former les équipes à nouveau.

Se débarrasser d'un outil est une tâche beaucoup plus difficile que de l'acquérir. Que deviendront les modèles, les programmes existants ? Faudra-t-il les faire migrer, les convertir ? Avec quelle perte d'information ? Acheter un outil, c'est donc s'engager pour vingt ans, même si l'outil lui-même a une vie beaucoup plus courte.

Le choix d'un outil ou d'un ensemble d'outils de développement est donc un choix stratégique. Avant de l'entreprendre, il est important que les différents outils sur le marché soient étudiés à fond, de même que le contexte technologique, méthodologique et humain de son utilisation.

1.5. Structure de l'ouvrage

Les chapitres de cet ouvrage sont ordonnés selon le cycle de vie d' un outil ou d' un environnement de développement dans une entreprise, la gestion d' un tel outil étant vue comme un projet à part entière. Ils suivent donc à peu près les étapes de ce projet, depuis l' étude d' opportunité jusqu' à la décision d' évolution vers un autre environnement. On y trouvera donc, dans l' ordre naturel, l' analyse de l' existant (chapitre 2), la définition des besoins (chapitre 4), la description des différentes pièces (chapitre 5), les éléments de leur intégration (chapitres 7 et 11), et enfin les éléments permettant de valider l' ensemble (chapitre 15). Y ont été intercalés des chapitres concernant des aspects particuliers comme le langage ou l' évolution des outils.

Le chapitre 2 dresse un panorama de l'existant, en présentant quelques chiffres sur l' offre actuelle et sur l' utilisation des outils, en particulier dans les grandes entreprises et administrations. Le chapitre 3 apporte des précisions sur le langage, élément extrêmement important du développement de logiciel, et trop souvent négligé. Le chapitre 4 définit les besoins en outillage des différents utilisateurs. Le chapitre 5 détaille les différents types d' outils, qui constituent autant de briques d' un atelier.

Le chapitre 6 apporte une vision plus commerciale que technique du monde des outils. Comme tous les produits, les AGL ont une naissance, une vie et une mort. Cependant, un AGL est le cœur du système d' information, et sa disparition est trop lourde de conséquences pour qu' une organisation puisse se permettre de voir un atelier disparaître sans autre forme de procès. Savoir comment les outils voient le jour, comment ils évoluent et comment ils disparaissent permet d' anticiper la trajectoire, d' avant leur naissance jusqu' après leur mise au rebut, et de prendre les bonnes décisions d' acquisition ou de remplacement.

De même qu' une maison n' est pas un simple empilement de briques et de poutres, il ne suffit pas d' acquérir des outils pour avoir un atelier de génie logiciel à sa disposition. Comme tout logiciel, un AGL a une architecture, à ceci près qu' il existe peu d' ateliers « prêts à utiliser » et que, très souvent, l' architecture doit être définie par l' acquéreur autant que par le (ou les) fournisseur(s). Le chapitre 7 définit ce qu' est un atelier et précise ce qui le différencie d' un simple outil.

Les chapitres 8, 9 et 10 concernent la conception et les outils de conception. En effet, développer du logiciel, c' est avant tout concevoir. La programmation elle-même n' étant qu' une phase, détaillée à l'extrême, de la conception. Le chapitre consacré à l' orientation objet n' a pas pour vocation de revenir une fois de plus sur ce sujet tant de fois rebattu, mais d' expliciter les challenges auxquels les outils de

conception et de programmation par objets sont confrontés et les différentes perspectives qui sont ouvertes aux organisations par cette technologie logicielle.

Les cinq chapitres suivants sont consacrés au management, au sens large du terme, c'est-à-dire à la gestion du développement et à la communication entre les différents acteurs concernés, qu' ils soient programmeurs ou non, informaticiens ou non. En effet, les aspects humains et managériaux du développement, pourtant si importants, sont systématiquement sous-estimés, tant par les décideurs que par les utilisateurs mêmes de l' atelier.

Le chapitre 11 parle du référentiel, cet outil qui, dans l' idéal, constitue le pivot autour duquel s' articulent les différents outils de l' atelier, en même temps qu' il est le point de contact entre le monde du développement et les autres activités du système d' information. Sa technologie peut être extrêmement complexe. Cependant, c' est principalement sous l' angle de son utilisation qu' il sera abordé dans ce chapitre, en montrant le poids qu' il peut avoir dans la stratégie d' entreprise ou dans la politique d' une administration.

Les chapitres 12, 13 et 14 concernent le choix d' un AGL et la manière de le mettre en place dans une organisation. Ils seront utiles à toute personne concernée par le développement de logiciel, même celles qui ne sont pas directement concernées par le choix. Le chapitre 12 indique les trois conditions préalables au choix d' un outil ou d' un atelier. Une fois ces conditions remplies, le chapitre 13 permet de comprendre la démarche qui conduit au meilleur choix des outils et à une mise en place sans heurts de l' environnement de développement. On énonce au chapitre 14 cinq règles simples à respecter lors du choix ; elles permettent d' éviter cinq erreurs lourdes de conséquences, fréquemment commises, et faciles à éviter.

Le chapitre 15 présente deux études de cas de choix et de mise en place d' outils. Bien que les noms des outils et de leurs acquéreurs aient été dissimulés, il s' agit de cas réels. Les schémas utilisés sont les mêmes que ceux qui ont été soumis aux décideurs pour les aider à prendre une décision efficace en toute connaissance de cause. Le symbolisme utilisé a été mis au point par l' auteur, qui les utilise lors de ses missions de conseil pour communiquer des résultats ou des recommandations sous forme synthétique à des non-spécialistes. Il semble, en effet, important que tous les acteurs puissent appréhender un environnement de développement dans sa globalité, en ayant un niveau identique d' information, avant de prendre une décision.

Le chapitre 16 tente un peu de prospective, en abordant ce qui paraît être l' avenir de la programmation et du développement de logiciel. Enfin, le chapitre 17 clôt (du moins provisoirement) le débat « construire ou acheter » en montrant à quel point les deux démarches ne sont pas antinomiques et tendent effectivement à converger.

1.6. Guide de lecture

Bien entendu, l' ouvrage gagne à être lu dans sa totalité en suivant l' ordre des chapitres. Mais à l' heure où tous parlent de développement incrémental il serait excessif de demander à un lecteur, souvent pressé de surcroît, une lecture linéaire.

Bien que ce livre s' adresse à tout public possédant un bagage informatique de base, certains chapitres sont plus techniques, d' autres concernent en premier lieu la stratégie d' entreprise. Enfin, certains chapitres sont destinés au consultant et à toute personne faisant appel aux services d' un consultant pour le choix, l' acquisition et la mise en place d' un AGL.

Le manager très pressé pourra lire rapidement les chapitres 4, 5 et 11, s'appesantir sur les chapitres 12, 13 et 14, et jeter un regard sur les études de cas du chapitre 15, avant de passer directement à la conclusion. Le chapitre 6, bien que non indispensable sur le plan opérationnel, pourrait également l' intéresser.

Un étudiant en génie logiciel lira en priorité les chapitres 3, 4 et 5, ainsi que le chapitre 11. Les chapitres 8 et 9 lui seront également utiles. Les chapitres 7 et 15 lui permettront de prendre le recul qui lui manque parfois, et le chapitre 6 lui donnera une ouverture sur un monde qui lui est encore souvent inconnu.

Enfin, un consultant en systèmes d' information, même spécialiste des outils, découvrira dans les chapitres 12 à 15 un point de vue sans doute nouveau et une démarche complémentaire à la sienne.