Pourquoi un AGL n'est pas une usine

À quoi ressemblerait l'AGL idéal ? À quoi ressemblera l'AGL de l'avenir ?
*Dans son récent ouvrage, **Outils de construction du logiciel**, paru aux éditions Hermès, Yves Constantinidis s'attaque à quelques mythes bien ancrés. Celui de l'AGL considéré comme une « usine à logiciels », est l'un des plus répandus. Dans cet extrait, nous allons voir pourquoi les AGL ne sont pas, et ne seront probablement jamais, des usines à logiciels (software factories) comme on le dit souvent.*

Le mythe de l'usine à logiciels

On compare souvent l'AGL idéal à une usine. D'après cette vision, le logiciel est produit par une équipe d'ingénieurs spécialisés, un peu comme une automobile est construite par une équipe d'ouvriers spécialisés.

- Avec l'aide de l'AGL, le logiciel est d'abord spécifié, dans le respect de la définition des besoins préalablement stockée dans le référentiel de l'AGL.
- Une deuxième équipe de concepteurs prend alors en charge les spécifications détaillées et le découpage du logiciel en modules.
- Les spécifications détaillées de ces modules sont alors prises en charge par des programmeurs.
- Puis chaque module est codé par un spécialiste d'un domaine technique ou d'un domaine métier : interface graphique, accès à la base de données, algorithmes plus ou moins sophistiqués, modules spécialisés.
- Après la phase de programmation, les modules sont assemblés par une autre équipe, comme les pièces d'une automobile sur une chaîne de montage.
- La totalité des fonctionnalités est testée par une équipe d'informaticiens-testeurs, suivant des jeux de tests préalablement conçus et stockés dans le référentiel de l'atelier.
- Le logiciel ainsi construit peut alors entrer en production et passe sous la responsabilité de l'équipe de maintenance.
- Chaque modification, suite à une demande d'amélioration ou à la détection d'une erreur, est répertoriée et enregistrée dans le référentiel de l'outil, garantissant la traçabilité de chaque intervention.

Pour les cas, heureusement de plus en plus rares, de logiciels qui n'auraient pas été conçus, développés et maintenus selon les mêmes principes et avec les mêmes outils, et pour lesquels toutes les étapes de production n'auraient pas été tracées et stockées dans un référentiel, l'AGL idéal dispose d'un ou de plusieurs outils de « rétroingénierie » ou « rétroconception », permettant de reconstituer *a posteriori* les divers documents manquants, en remontant en sens inverse les étapes de spécifications détaillées, de spécifications globales, ou de définition des besoins.

Conçu et réalisé avec un tel atelier, un logiciel a la même finition qu'une automobile sortant d'une usine d'automobiles, et il est aussi sûr d'utilisation qu'un pont construit selon les règles du génie civil.

La réalité

Cette vision de l'AGL idéal, quelque peu caricaturée ici, est décrite par nombre de « méthodologues » et propagée par la presse spécialisée, soutenue dans ses efforts par un nombre impressionnant de fournisseurs d'outils de développement.

Si de tels AGL-usines n'ont encore jamais vu le jour, ce n'est pas faute de disposer de moyens pour concevoir, développer et commercialiser des outils et des référentiels suffisamment puissants. Ce n'est pas non plus par manque de méthodologies adéquates.

L'AGL-usine est tout simplement un mythe, et ceci pour trois raisons : (1) L'activité de développement de logiciel n'est pas une activité de production. (2) Le développement de logiciel, au sens large du terme, ne suit pas un processus linéaire. (3) Dans le monde du développement de logiciel, une trop grande spécialisation nuit à la productivité.

Revenons sur ces trois affirmations :

- *l'activité de développement n'est pas une activité de production.*
  Contrairement au matériel, le logiciel n'a pas à être fabriqué, au sens industriel du terme. Il passe presque directement de la conception à l'emballage. Les tests, la validation, le contrôle qualité, ne concernent quasiment pas la fabrication (qui consiste en réalité à dupliquer des disquettes, à graver un CD-Rom ou à télécharger des fichiers) mais le développement. Les tests, comme chacun le sait, se font au niveau du développement. Il est donc abusif de parler d'usine là où il s'agit en fait de bureaux d'études ;

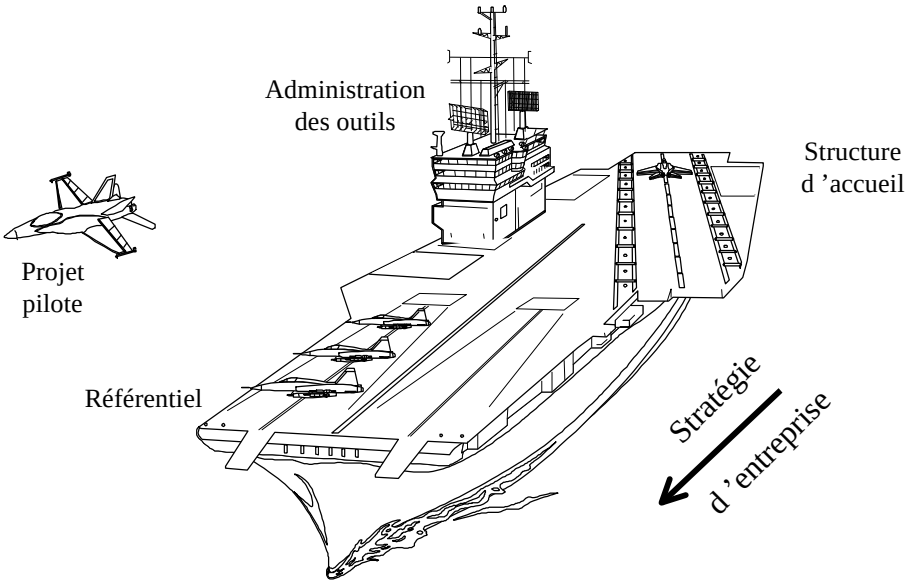
- *le développement du logiciel ne suit pas un processus linéaire.*
  Le modèle de cycle de développement en cascade (waterfall model) n'est qu'une approximation de la réalité, et il a été remis en cause à de nombreuses reprises. Le cycle en spirale est lui-même un modèle qui, bien que se rapprochant plus de la réalité, peut être sujet à controverse. En réalité, un grand nombre d'activités de développement sont conduites, non seulement en parallèle, mais de façon totalement désynchronisée. Et la généralisation du développement par objets, où des groupes de modules (clusters) sont construits de manière quasi indépendante par de petites équipes, et souvent par un développeur isolé, ne va faire qu'accélérer cette tendance. Le modèle de l'usine, et même du bureau d'études traditionnel, est inapplicable à la construction du logiciel ;

- *dans le monde du développement de logiciels, une trop grande spécialisation nuit à la productivité.*
  Le métier de programmeur qui se contente de coder en Cobol des spécifications hyperdétaillées écrites en français a presque disparu. Ceci n'est pas seulement dû à une inflation des termes, par ailleurs très réelle. C'est le métier lui-même qui appartient au passé. La division « verticale » du travail augmentait la productivité des usines de montage du temps de la Ford T. Dans une équipe d'informaticiens, le seuil à partir duquel cette division du travail devient contre-productive est vite atteint.

En effet, la difficulté de construire un système cohérent ne réside pas dans la conception des briques élémentaires, mais dans l'architecture de l'ensemble et dans la constitution d'interfaces stables entre les différents modules. Ces trois arguments suffisent à prouver que l'usine ne peut, ni ne doit, servir de modèle aux ateliers de génie logiciel.

L’avenir : le porte-avions

Si l’AGL idéal n’est pas une usine, à quoi peut-on alors le comparer ? Peut-on trouver une analogie avec un secteur d’activité existant ? C’est l’aéronautique qui nous apportera une analogie : l’AGL de l’avenir ressemblera de moins en moins à une usine et de plus en plus à un porte-avions.



Il est nécessaire, pour un chef de projet, de disposer d’une large part d’initiative, de suffisamment d’autonomie, tout en se conformant à l’orientation générale qui lui est imposée par l’entreprise. En d’autres termes, il doit être seul maître à bord de son projet, après avoir reçu une mission clairement définie qu’il s’est engagé d’accomplir.

Le porte-avions est un référentiel muni de points d’accès permettant l’échange de modèles avec les autres projets. Chaque projet reçoit un plan de vol précis et travaille sur un objectif précis avec une équipe de taille réduite. Une fois l’objectif traité, l’avion (le projet) revient à sa base (le référentiel), sachant néanmoins que cette base peut elle-même avoir évolué.

Chaque projet apporte des informations pertinentes au référentiel global, c’est-à-dire uniquement des informations pouvant présenter un intérêt pour les autres projets ou pour l’ensemble de l’entreprise. Ces informations peuvent être retravaillées au niveau du référentiel global avant d’être mises à la disposition des autres projets. En cours de « vol », le chef de projet peut recevoir des directives nouvelles tout en conservant une totale autonomie opérationnelle.

L’AGL idéal est donc un ensemble hétérogène d’outils, aussi différents les uns des autres que peuvent l’être un hélicoptère de combat d’un avion de chasse, mais parfaitement communicants. Ils disposent chacun d’une large autonomie, tout en pouvant opérer sur un objectif global bien défini. ▲

Yves Constandinidis

Extrait de l’ouvrage *Outils de construction du logiciel*, Hermès, 1998.