Chapitre 1: Introduction au Génie Logiciel

Il a été identifié pour ce cours les objectifs suivants :

- La sensibilisation au rôle du logiciel dans notre quotidien
- L'impact de la qualité du logiciel
- Le processus de développement des logiciels

I. Le Logiciel

Avant de donner une définition au terme logiciel, il sans doute nécessaire d'identifier où se trouve le logiciel ? Le logiciel est quasiment présent « partout » dans beaucoup de choses de notre quotidien. Ils dévient même parfois inimaginables de faire certaines tâches aujourd'hui sans le logiciel. Ainsi, on retrouve les logiciels dans la bureautique, les téléphones, les voyages aériens (avion, voiture), les maisons (smart house), la scolarité, la recherche scientifique, les loisirs,… Par conséquent, notre vie dépend très fortement de **la qualité des logiciels** qui la gèrent.

a) Impacts Positifs du logiciel

Grâce aux logiciels, notre quotidien a été amélioré de plusieurs manières :

- Le logiciel accélère les traitements
- Le logiciel ne « se lasse » pas
- Le logiciel résout des problèmes complexes rapidement
- Le logiciel dispose d'une capacité de calcul, de stockage et de traitement incroyables
- Le logiciel a introduit de nouveaux loisirs (jeux vidéo)

Exemples : Paiement électronique, Achat sur internet, Recherche d'information, Logiciels métier (ERP, tableurs, traitement de texte), Bibliothèques en ligne, …

b) Impacts d'un logiciel de Mauvaise Qualité

Plusieurs désastres plus ou moins importants peuvent ou ont été causés par des **« erreurs »** dans des logiciels :

- **La paranoïa de bug de l'an 2000** : à l'époque, la plupart des logiciels traitaient les dates avec deux chiffres. Quand l'an 2000 fut sur le point d'arriver, personne ne pouvait vraiment prédire ce qui allait se passer. Finalement, plus de peur que de mal.
- **Le bug du Mariner-1 en 1962** : Une fusée spatiale a dérouté de sa trajectoire à cause d'une formule mathématique qui a été mal transcrite en code source.
- **Therac-25 accélérateur médical (1985)** : La machine était destinée à soigner des malades. À cause d'un bug sur le déclenchement des radiations, au moins cinq personnes ont trouvé la mort.
- **En 1983, la troisième guerre mondiale a failli éclater** : En pleine guerre froide, un logiciel de surveillance soviétique a détecté de faux missiles balistiques envoyés des USA.

- **1991, pendant la guerre du golfe** : Un missile américain tue 22 soldats américains au lieu d'intercepter un missile ennemi. Cause : une erreur de fonction d'arrondi,
- **1996 Cash de la fusée Ariane 5 – Vol 501** : Un module convertissait des réels 64 bits en des entiers signés 16 bits ce qui a cause un fonctionnement anormal des moteurs. La fusée s'est désintégrée après 40 secondes de vol.
- **2000 Panama, machine médicale traitement de cancer**. Le logiciel a permis de dessiner cinq zones protégées alors que la machine ne tolérait que quatre. Résultat : des tissus sains ont été irradiés et huit personnes au moins décédées.

c) Qu'est-ce qu'un logiciel ?

Un **logiciel** est un ensemble d'informations relatives à des traitements effectués automatiquement par un appareil informatique. Plus formellement, c'est un ensemble des programmes, procédés et règles, et éventuellement de la documentation, relatifs au fonctionnement d'un ensemble de traitement de données. Une approche minimaliste de ce concept voudrait qu'on perçoive comme étant programme informatique. Mais force est de constater que professionnellement, le logiciel réunit à la fois les **exécutables** (binaires), la **documentation** et les **produits connexes**.

d) Nature du logiciel

Les logiciels possèdent certaines spécificités qui permettent de les distinguer des autres produits normaux :

- Le logiciel est ***intangible (non palpable)***. Il est difficile d'estimer l'effort de développement.
- Le logiciel est un produit **Développé**. D'autres produits sont manufacturés (Transformation des matières premières en produits finis à échelle industrielle).
- Le processus de développement est difficile à automatiser car entièrement basé sur les humains.
- Le logiciel résulte d'un **assemblage difficile** par **réutilisation**. Un produit manufacturé s'assemble facilement par composants.
- Le logiciel est maintenu par l'équipe de développement elle-même. La maintenance produit manufacturé se fait par Remplacement des composants.
- Un logiciel ne **s'use pas**. Il se détériore à mesure que des changements sont effectués (*software aging*) en raison de l'introduction d'erreurs ou par une complexification indue.
- La **qualité** d'un logiciel **n'est pas apparente.** Même des informaticiens **peu qualifié** peuvent arriver à **bricoler** quelque chose qui **semble** fonctionner car
- Un logiciel **semble** facile à modifier. La tentation est forte d'effectuer des changements rapides sans vraiment en mesurer la portée.

e) Vieillissement de logiciel (*software aging)*

David Lorge Parnas en 1994 affirme « *Les programmes, comme les gens, vieillissent. Nous ne pouvons pas empêcher le vieillissement, mais nous pouvons en comprendre les causes, prendre des mesures pour en limiter les effets, inverser temporairement certains des dommages causés et préparer le jour où le logiciel n'est plus viable* ». Parnas en 1994 identifie deux types distincts de vieillissement du logiciel. Le premier est causé par l'incapacité des **propriétaires du produit** à le modifier pour répondre aux besoins changeants; la seconde est le résultat **des changements** qui sont faits. Ce «coup double» peut entraîner une baisse rapide de la valeur d'un produit logiciel. Les raisons à la base du vieillissement du logiciel peuvent être les suivantes :

- Maintenance (e.g., bug fixes)
- Érosion architecturale (l'écart entre l'architecture planifiée et réelle d'un système logiciel tel qu'observé dans sa mise en œuvre)
- inflexibilité dès le début
- documentation insuffisante ou inconsistante
- Pression des délais
- duplication de code
- manque de modularité
- complexité croissante
- ….

f) Classification de Logiciels

Plusieurs classifications de logiciels existent. La plus célèbre est celle du NAPCS (North American Product Classification System) qui regroupe les logiciels en deux classes :

- Les **Logiciels Système** (Drivers, Système d'exploitation, Outils de développement, SGBD, Logiciels réseau, …)
- Les **Applications** (Bureautique, Loisirs, Métier, Archiva, …)

Une autre approche de classification distingue :

- Les logiciels **sur mesure** (*custom*) : Pour un client spécifique
- Les logiciels **Génériques** (*generic*) : Vendu sur le marché
    - e.g. un tableur (spreadsheet), un outil de base de données (database), un outil de traitement de texte (word processor)
- Les logiciels **Embarqués** (*embedded*) : Scellé dans du matériel électronique (machine à laver, télévision, lecteur DVD, ...) et difficile à modifier
- Les Logiciels en **temps réel** (*real-time*) : systèmes de contrôle et de surveillance

- Logiciel de traitement de données (*data processing*). Ces logiciels sont utilisés pour l'administration des affaires, Fiabilité des résultats, Sécurité dans l'accès aux données

g) Software et Hardware

Dans cette partie nous attirons l'attention sur le lien étroit qui existe entre Le « hardware » et le « software ». En effet, le « **hardware** » a besoin du « software » pour être piloté et le « **software** » a besoin du hardware pour être exécuté. Il a été remarqué que, l'évolution phénoménale des capacités des logiciels est intimement liée à l'évolution du hardware et aussi d'autres facteurs :

- Amélioration de la puissance du processeur
- Amélioration des capacités de stockages
- Changement des dispositifs d'entrée ou de sortie (Ecran tactile, stylo optique, kinect, …etc.)
- Augmentation de la mobilité et des unités mobiles (Smartphones, tablettes, notebooks,…etc.,)

II. Le développement de logiciels

Le développement est la transformation d'une **idée** ou d'un **besoin** en un **logiciel fonctionnel.** L'idée est produite par un client (**utilisateur**) et développée par un **fournisseur.** Dans certains cas, le client et le fournisseur peuvent être la même entité.

Du point de vue du client (le commanditaire du logiciel), un logiciel sera qualifié de bon s'il est :

- Peu couteux,
- Fait ce qu'on lui demande de faire
- Respecte des critères de qualité,
- Livré dans les délais.

Le fournisseur caractérisa quant à lui un bon logiciel comme celui qui :

- Minimise le coût
- Minimise les délais
- Maximise les profits
- Fait ce qu'on attend de lui
- Respecte les exigences de qualité

Ces contraintes renseignent à suffisance que, les projets de développement des logiciels font intervenir plusieurs parties prenantes (stakeholders) de compétences différentes. On ne saurait donc ramener le développement du logiciel à la seule activité de programmation (le codage) qui n'est rien d'autre qu'une activité dans l'ensemble d'activités qui le compose.

Pour mener donc un projet de développement d'un logiciel, on besoin d'une **équipe**, d'un **plan de communication**, des **outils** et d'un **procédé**.

III. Le Génie Logiciel

a) Echecs des projets de développement de logiciels

Les études menées par le **Standish Group** démontrent que la proportion de projets qui sont considérés comme des succès (autrement dit, respectant les 3 C) reste faible : entre 25 % et 30 %. Cela signifie que trois projets sur quatre sont des échecs complets ou partiels : les projets sont abandonnés en cours de route ou aboutissent, mais au prix de dépassements importants, ou offrent moins de fonctionnalités que prévu. Dans ce registre, 28 % de Projets sont annulés et 46 % de projets sont un échec.

b) Crise du logiciel

L'apparition de la crise du logiciel dans les années 60 provenant d'un décalage entre les progrès matériels d'une part et logiciels d'autre part a permis lors de la conférence sur « ***Software Engineering »*** d'identifier les causes d'échecs des projets de développement de logiciels. Alors que les ordinateurs devenaient de plus en plus puissants et de moins en moins coûteux, la construction de logiciels restait dans le domaine de l'artisanat et du folklore, où chacun y allait de sa petite recette. Il a était identifié que :

- la construction de logiciels coutait très cher et les budgets prévus étaient largement dépassés (200 millions de dollars pour fabriquer OS-360),
- les logiciels ne satisfont pas les attentes des clients (les clients et les développeurs ne se comprenaient pas)
- les délais n'étaient pas respectés (2 ans de retard pour les premiers compilateurs PL/1, Algol 68, ADA),
- les logiciels n'étaient pas évolutifs (parfois écrits en assembleur pour un type de machine) ce qui les rendait très rapidement obsolètes,
- les logiciels avaient des performances approximatives ou tout simplement des temps de réponse trop lents (Univac, le système de réservation pour United Air Lines au début des années 75 n'a jamais servi car les temps de réponse étaient trop longs !),
- une fiabilité aléatoire entrant des graves erreurs de fonctionnement (la sonde américaine qui devait aller sur Vénus s'est perdue, à cause d'une mauvaise instruction... plus récemment, la trajectoire de Ariane 5 a été modifiée à cause d'un débordement de capacité),
- La maintenance trop chère car trop difficile
- et une convivialité discutable (des interfaces homme/machine inexistantes).

Ces limites ou manquements sont à l'origine de l'apparition du **génie logiciel**.

c) Qu'est ce que le génie logiciel ?

Le terme "**génie logiciel**" a été introduit en 1968 (*P. Naur, B. Randall (Eds.). Software Engineering: A Report on a Conference Sponsored by the NATO Science Committee, NATO, 1969)* pour reconnaître le fait que les principes du **génie** peuvent s'appliquer au développement du logiciel. Le ***génie*** est une pratique régulée par une corporation professionnelle qui veille à la protection du public, repose sur l'application de principes scientifiques et économiques et dont le fonctionnement est régit par des pratiques conformes à une éthique et une déontologique établies.

Plusieurs définitions ont été proposées par la communauté scientifique pour qualifier le terme génie logiciel. Le terme **génie logiciel** (en anglais ***software engineering***) désigne l'ensemble des méthodes, des techniques et d'outils concourant à la production d'un logiciel, au-delà de la seule activité de programmation.

L'IEEE définit le **génie logiciel** comme l'application d'approches **systématiques**, **rigoureuses**, **quantifiables** pour le développement, la mise en œuvre et la maintenance d'un logiciel, c'est à dire **l'application de l'ingénierie** au domaine du logiciel.

Nous définissons le **génie du logiciel** comme étant le processus visant la résolution de problèmes posés par un client par le **développement systématique** et **l'évolution de systèmes logiciels** de grande taille et de **haute qualité** en **respectant les contraintes** de coûts, de temps, et autres.

L'objectif du génie logiciel est :

- …***la résolution de problèmes posés par un client (de manière satisfaisante)***…Il s'agit là du but essentiel du génie logiciel. Rappelons que, dans certains cas, la solution peut être de ne rien développer, si un produit satisfaisant existe déjà (ex : logiciel générique). Il est important de se focaliser dans la résolution du problème du client et non dans l'ajout des options non requises par le client qui ne solutionne en rien le problème.

- …***le développement systématique et l'évolutif***… il s'agit ici de l'application de techniques bien maîtrisées reconnues et standardisées (e.g. IEEE ou ISO) de façon organisée et disciplinée prédisposées à faire ***évoluer*** un logiciel existant.

- …***systèmes logiciels de grande taille et de haute qualité***… Un logiciel de grande taille est un logiciel qui ne peut être compris par une seule personne et qui exige un travail en équipe et une bonne coordination. Un des défis principaux est d'arriver à subdiviser le travail à accomplir tout en s'assurant que chacune de ces parties fonctionneront harmonieusement ensemble et que le produit final doit répondre aux critères de qualité bien établis.

- …***en respectant les contraintes de coûts, de temps, et autres.*** Il est essentiel de souligner ici que les ressources sont généralement limitées, que le bénéfice résultant doit être supérieur aux coûts, que la productivité de l'équipe doit demeurer

concurrentielle et qu'une mauvaise estimation des coûts et de la durée du projet peut mener à l'échec du projet

d) Quatre P du génie logiciel

Pour donner un aperçu bref et concis du génie logiciel, on le résume souvent en quatre **P :**

- **P**ersonnel : Qui produit le logiciel?
- **P**rocessus : Comment le logiciel est-il produit?
- **P**rojet : La production réelle du logiciel
- **P**roduit : Tous les objets fabriqués pendant la production : code source, exécutables, documentation, modèles de conception, cahier de charges, résultats de tests, mesures de productivité, …

e) Parties prenantes dans le génie du logiciel (*stakeholders*)

Les parties prenantes d'un projet de développement de logiciel peuvent être de 04 catégories :

- Utilisateurs (*users*). Ceux qui se servent du logiciel
- Clients (*customers*). Ceux qui paient pour le logiciel
- Développeurs (*developers*). Ceux qui conçoivent le logiciel
- Gestionnaires (*managers*). Ceux qui supervisent la production du logiciel

Tous ces rôles peuvent être remplis par la même personne

f) Risques et difficultés en génie du logiciel

Plusieurs difficultés caractérisent le génie logiciel :

- Les clients arrivent difficilement à décrire leurs besoins de façon assez claire pour les fournisseurs (développeurs);
- Les besoins sont en constantes évolution ainsi que l'environnement;
- Les personnes techniques et non techniques ne parlent pas le même langage;
- La difficulté de gérer le projet et les personnes
- L'incertitude concernant la technologie, les exigences, les compétences;
- La complexité et la quantité des éléments à tenir en compte;
- Les risques politiques

g) Projets de génie logiciel

La plus part des projets consiste généralement à :

- Faire évoluer ou à maintenir un logiciel existant dont on a hérité de la responsabilité
- Développer à partir de zéro (Concevoir un nouveau produit). Il s'agit là de la minorité des projets entrepris
- Faire la *restructuration* (***re-engineering, refactoring***). Il s'agit des changements apportés à la structure interne du programme, sans changer le comportement externe du programme.

h) Activités connexes aux projets de génie logiciel

Les projets de génie logiciel font appels à plusieurs activités dont :

- **Gestion du projet**. Choisir un **modèle/processus** et des techniques (itératif, incrémental, prototypage) de développement qui permet de diviser le projet en différentes activités, de décider ce qui est inclus dans chaque activité et de décider sur la séquence ou la chronologie d'activités.
- **Définition et spécification des exigences**. Définir les exigences fonctionnelles et non fonctionnelles du système à développer qui permette d'établir le cahier de charges.
- **Conception (*design*).** Décider comment la technologie disponible sera utilisée pour répondre aux besoins. Ce qui inclut la détermination de ce qui sera réalisé par le logiciel (***software***) et par le matériel (**hardware**), la mise au point de l'***architecture*** du système, la définition des sous-systèmes et de leurs interactions, la conception des interfaces des utilisateurs et des bases de données.
- **Modélisation (en UML).** Créer des représentations du logiciel et de son domaine d'application. Ce qui inclut la modélisation de son utilisation (***use case modelling***), la modélisation de sa structure (***structural modelling***) à travers les classes et objets, la modélisation de sa dynamique et de son comportement *(dynamic and behavioural modelling)* à travers les diagrammes d'activités, d'états-transitions, etc.
- **Programmation (en Java).** La production du code source de l'application.
- **Déploiement (*deployment*).** La distribution et installation du logiciel dans son environnement d'exploitation.
- **Assurance de qualité.** Tester le logiciel pour évaluer sa la qualité fonctionnelle et structurelle.

i) Distribution des coûts au long du cycle de vie

En 1975, Boehm propose une grille de répartition des coûts suivant les phases du projet.

Type du système	**Coûts pour chaque phase**		
	Besoins/conception	Implémentation	Test
Système de contrôle	46%	20%	34%
Système de l'aéronautique	34%	20%	46%
Système d'exploitation	33%	17%	50%
Systèmes scientifiques	44%	26%	30%
Système de gestion	44%	28%	28%

IV. Conclusion

Il a était présenté dans ce chapitre les concepts et les raisons d'être du génie logiciel. Dans le prochain chapitre il sera question d'aborder la notion de cycles de vie du logiciel.