Chapitre I : Etude des Concepts ; Logiciel et Génie Logiciel.

I. Introduction sur l'importance du Logiciel

Plusieurs aspects socio-professionnels ne peuvent être imaginés aujourd'hui sans logiciel. C'est le cas entre autres des domaines : de l'Industrie, du transport, du commerce, des communications, de la médecine, des finances ou des loisirs. Cette omniprésence du logiciel fait que l'organisation et le fonctionnement des sociétés dépendent fortement de sa ***qualité***.

Les erreurs logicielles peuvent causer de vrais désastres, économiques ou sanitaires. On cite entre autres : l'explosion de la première fusée **Ariane 5 en 1996**, qui a coûté plus d'un demi-milliard de dollars, à cause du **dépassement de capacité d'une variable lors d'un calcul**.

Dans un registre différent, on relève la grande peur du « bug de l'an 2000 » et de son coût astronomique, estimé entre 300 et 5000 milliards de dollars.

L'importance d'une approche méthodologique dans le développement des logiciels est apparue avec la crise de l'industrie du logiciel à la fin des années 1970 engendrée par :

- ✓ l'augmentation des coûts de production,
- ✓ la difficulté d'évolution,
- ✓ la non fiabilité,
- ✓ le non-respect des spécifications,
- ✓ le non-respect des délais.

II. Définitions des Concepts

A) Quelques Rappels

1. Un Système

Un système est un ensemble d'*éléments* interagissant entre eux suivant un certains nombres de principes et de *règles* dans le but de réaliser un *objectif*.

2. **La frontière d'un système** est le critère d'appartenance au système.
3. **L'environnement** est la partie du monde extérieur au système.

Un système est souvent hiérarchisé à l'aide de ***sous-systèmes***.

Un système complexe se caractérise par :

- ✓ sa dimension, qui nécessite la collaboration de plusieurs personnes pour sa réalisation ;
- ✓ son évolutivité.

Toute organisation (Administration, Entreprise, collectivité sociale..) peut être modélisée comme constituée de trois sous-systèmes :

- ✓ Le système de pilotage qui décide des actions et politiques à mettre en œuvre,
- ✓ Le système opérant qui produit et transforme les Flux primaires en Sorties valorisées,
- ✓ Le système d'Information qui gère les informations de toutes les formes et de toutes natures.

4. **Un système d'information** : est l'ensemble des éléments participants à la gestion, au stockage, au traitement, au transport et à la diffusion de l'information au sein de l'organisation.
5. **Un système informatique** : est l'ensemble des équipements destinés au traitement automatique de l'information permettant d'acquérir, de stocker, de traiter et de communiquer des données.

B) Généralités sur La Notion de Logiciel

1. Définition (Logiciel)

Un logiciel est un ensemble d'entités nécessaires au fonctionnement d'un processus de traitement automatique de l'information.

Parmi ces entités, on trouve par exemple :

- ✓ des programmes (en format code source ou exécutables) ;
- ✓ des documentations d'utilisation ;
- ✓ des informations de configuration.

Un logiciel est un système d'information automatisé.

Un système d'information automatisé est l'ensemble des moyens et des méthodes qui se rapporte au traitement automatisé des données. Il constitue la partie logicielle du système informatique.

Au Niveau structural

Un logiciel est en général un sous-système d'un système englobant.

Il peut interagir avec des clients, qui peuvent être :

- ✓ des opérateurs humains (utilisateurs, administrateurs, . . .) ;
- ✓ d'autres logiciels ;
- ✓ des contrôleurs matériels.

Il réalise une spécification : son comportement vérifie un ensemble de critères qui régissent ses interactions avec son environnement.

Au niveau Conceptuel

« Le logiciel est l'ensemble des programmes, procédés et règles, et éventuellement de la documentation, relatifs au fonctionnement d'un ensemble de traitements de l'information' » (*arrêté du 22 déc. 1981*).

Un logiciel pourra donc être considéré comme un ensemble de programmes informatiques (codes sources, éditables, exécutables) mais également les données qu'ils utilisent et les différents documents se rapportant à ces programmes et nécessaires à leur installation, utilisation, développement et maintenance : spécifications, schémas conceptuels, jeux de tests, mode d'emploi, etc.

2. La Notion de Programme informatique

C'est un ensemble d'instructions, rédigé dans un langage de programmation, permettant à un système informatique d'exécuter une tâche donnée.

3. La diversité du logiciel

Il existe une grande variété de logiciels et de nombreuses manières de les classer.

- Une première différenciation oppose les ***logiciels « génériques »***, ou progiciels, qui sont vendus en grand nombre comme des produits de consommation, aux ***logiciels « spécifiques »*** qui sont développés pour un contexte et un client particulier en fonction de ses besoins généralement présentés dans un cahier de Charges.
- Une seconde différenciation repose sur la nature de la dépendance entre les logiciels et leurs environnements. Elle distingue trois classes :
  - → les ***logiciels « autonomes »*** (*standalone*), comme les traitements de texte, les logiciels de calcul ou les jeux,
  - → les ***logiciels qui gèrent des processus*** (*process support*), aussi bien industriels que de gestion,

→ ***les logiciels « embarqués »*** dans des systèmes (*embedded*), comme dans les transports, la téléphonie ou les satellites.

4. Les Caractéristiques du logiciel

- ✓ Un objet immatériel,
- ✓ Ne s'use pas,
- ✓ Le logiciel est facile à reproduire.

Ses fonctionnalités complexes sont difficiles à borner au départ et généralement remises en cause.

5. Les principaux facteurs de qualité d'un logiciel :

Les principaux facteurs qualitatifs d'un logiciel sont les suivants :

a) ***La Validité*** : (correction, justesse, conformité) est la capacité que possède un produit logiciel à remplir exactement ses fonctions, définies par **le cahier des charges** et les spécifications. Adéquation entre :

- Le besoin effectif de l'utilisateur,
- Les fonctions offertes par le logiciel.

b) ***La Fiabilité ou Robustesse*** : la robustesse (fiabilité, sureté) est la capacité qu'offrent des systèmes logiciels à réagir de manière appropriée à la présence de conditions anormales (i.e. rien de catastrophique ne peut survenir, même en dehors des conditions d'utilisation prévues).

c) ***La Facilité d'utilisation*** : la facilité d'utilisation est la facilité avec laquelle des utilisateurs présentant des formations et des compétences différentes peuvent apprendre à utiliser les produits logiciels et s'en servir pour résoudre des problèmes. Elle recouvre également la facilité d'installation, d'opération et de contrôle.

d) ***Compatibilité*** : la compatibilité est la facilité avec laquelle des éléments logiciels peuvent être combinés à d'autres (un logiciel doit pouvoir interagir en synergie avec d'autres logiciels).

e) ***L'Efficacité*** : l'efficacité est la capacité d'un système logiciel à utiliser le minimum de ressources matérielles, que ce soit le temps machine, l'espace occupé en mémoire externe et interne, ou la bande passante des moyens de communication (les logiciels doivent satisfaire aux contraintes de temps d'exécution).

f) ***La Portabilité*** : la portabilité est la facilité avec laquelle des produits logiciels peuvent être transférés d'un environnement logiciel ou matériel à l'autre (un même logiciel doit pouvoir fonctionner sur plusieurs machines).

g) ***La Réutilisabilité*** : la réutilisabilité est la capacité des éléments logiciels à servir à la construction de plusieurs applications différentes (80 % du code est du " tout venant " qu'on retrouve à peu près partout, 20 % du code est spécifique).

h) ***La Maintenabilité*** : la maintenabilité est le degré de facilité de la maintenance d'un produit logiciel.

i) ***L'Extensibilité*** : l'extensibilité est la facilité d'adaptation des produits logiciels aux changements de spécifications.

j) ***L'Intégrité*** : aptitude d'un logiciel à protéger son code et ses données contre des accès non autorisé.

k) ***Ponctualité*** : la ponctualité est la capacité d'un système logiciel à être livré au moment souhaité par ses utilisateurs, ou avant.

En définitive, un logiciel bien conçu est un logiciel :

- correct (valide), fiable (robuste), avec un code réutilisable, compatible avec d'autres logiciels, efficace, portable, facile à utiliser, maintenable, ponctuel et extensible.

6. La complexité : de conception, de Mise en Œuvre et d'Exploitation du logiciel

Techniquement, le terme « logiciel » désigne un riche ensemble d'artefacts incluant :

- ✓ des codes sources et des exécutables,
- ✓ des programmes et des données de test,
- ✓ des fichiers de configuration,
- ✓ des documentations « externes » à destination des utilisateurs,
- ✓ des documentations « internes » à destination de l'équipe de développement, etc.

La Complexité du développement des Logiciels étant également liée au matériel, on note également une évolution des systèmes matériels :

Après l'architecture de **Von Neumann**, à la base de toutes les architectures modernes, les systèmes matériels évoluent également rapidement dans le temps, pour permettre de développer des solutions de plus en plus performantes ; avec au centre le **processeur** (entité capable d'interpréter et exécuter un traitement ou processus).

Néanmoins, on note qu'au fur et à mesure que les coûts du matériel diminuent, les coûts des programmes informatiques par contre augmentent.

La diminution des coûts du matériel (hardware) a conduit à associer l'ordinateur à de plus en plus de produits. Les coûts des logiciels connaissent une très forte croissance, dominant actuellement le coût du matériel, et peuvent représenter plus de 80 % du coût total d'un système informatique.

a) A la fin des années 70, on relève des difficultés pour concevoir des logiciels (plusieurs tentatives de logiciels vont échouer) ; du fait que :

→ Les besoins et contraintes de l'utilisateur ne sont pas respectés (pas de facilité d'emploi, interfaces homme/machine inexistantes).

→ Les coûts ne sont pas maîtrisés.

En outre,

→ On relève régulièrement des retards dans la livraison des produits (non maîtrise du temps).

→ On note une grande rigidité dans les solutions mises en place.

→ Quelque fois on arrive à des incohérences pendant l'utilisation.

→ On est aussi confronté au manque de performance et de robustesse (fiabilité).

→ Les codes sont difficiles à maintenir.

→ On arrive parfois à des logiciels non utilisables.

b) En 1983, on se rend compte que :

→ 43% des logiciels sont livrés et non utilisés,

→ 28% payés et non livrés,

→ 19% utilisés tels quels,

→ 3% utilisés sans modification (échec de tentative),

→ 2% utilisés avec modifications (réussite de tentative),

→ Les coûts de maintenance (lorsque cela se fait) dépassent ceux du développement,

→ Le coût d'une erreur parfois dépasse le coût du système,

→ Les pannes causées par le « **bug de l'an 2000** » ont coûté environ 175 milliards de dollars aux entreprises du monde, (Journal **Le Monde** – 23 oct. 2001).

c) D'une étude sur 487 sites de toutes sortes de développement de logiciels, il ressort que :

→ 70% du coût du logiciel est consacré à sa maintenance,
→ 42% des modifications sont dues à des demandes de l'utilisateur,
→ 30% des exigences de spécification changent entre la 1ère édition d'une documentation et la première sortie du produit,
→ 17% à des changements de format des données,
→ 12% problèmes d'urgence résolus à la hâte,
→ 9% déboguages de programmes,
→ 6% à des changements de matériel,
→ 6% problèmes de documentation,
→ 4% améliorations de l'efficacité,
→ 3% autres…

Ces différents problèmes vont conduire à la création du Génie Logiciel.

7. Les Raisons expliquant les conséquences négatives

→ Il n'y a pas que les erreurs de programmation,
→ Projet non réaliste,
→ Management de projet de mauvaise qualité,
→ Mauvaise estimation des ressources,
→ Mauvaise définition des besoins du système,
→ Mauvais reporting du statut du projet,
→ Risques non gérés,
→ Mauvaise communication entre les clients, développeurs, utilisateurs,
→ Inaptitude à gérer la complexité du projet,
→ Mauvaise méthodologie de conception,
→ Mauvais outils de développement,
→ Mauvaise méthodologie de test,
→ Mauvaise couverture des tests,
→ Processus de développement inapproprié.

8. Les Domaines d'application du logiciel

Système • Compilateurs (A-0 System) • Editeurs de textes • Gestion de fichiers **Temps réels (Real-time)** • Contrôle de machine **Affaires (Data processing)** • SGBD (Oracle) • ERP (SAP)	**Embarqué (Embedded)** *Les logiciels embarqués s'exécutent dans du matériel électronique isolé.* • Programme de contrôle des Processus • Autocontrôle **Scientifique** • Simulation • Conception assistée par ordinateur • Calcul numérique intensif **Bureautique** **Intelligence artificielle** • Système expert

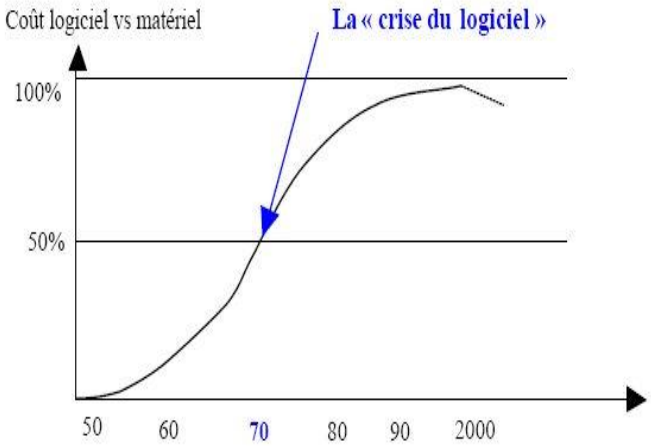
En conclusion, le logiciel est aujourd'hui présent partout, sa taille et sa complexité augmentent de façon exponentielle, les exigences de qualité sont de plus en plus sévères. La crise du logiciel apparue dans les

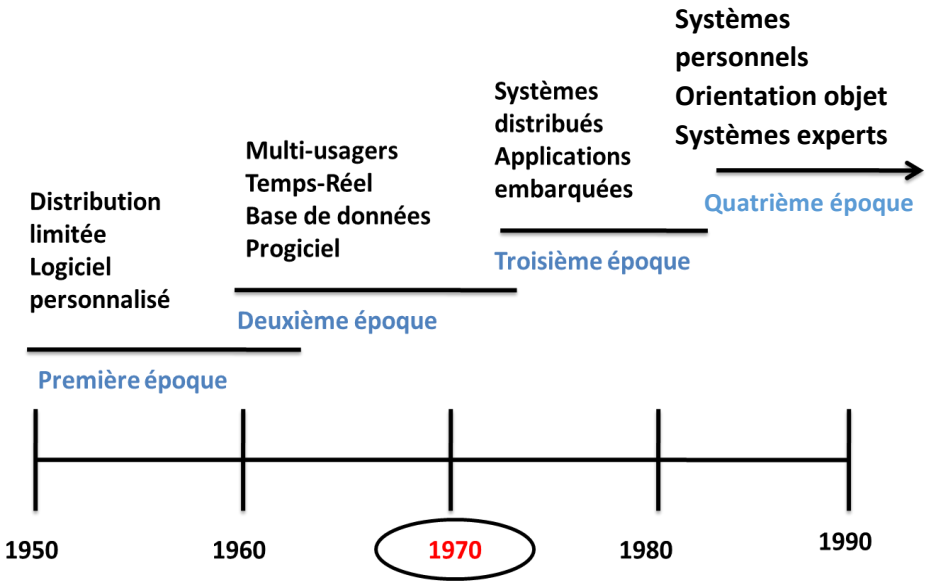
années 70 n'est toujours pas logiquement résolue même si de grands progrès ont été réalisés dans de nombreux domaines.

C) LA CRISE DU LOGICIEL

La crise du Logiciel, mise en évidence au début des années 70, se caractérise par l'**absence de maîtrise des projets,** au niveau des coûts et des délais, la **mauvaise qualité des produits:** les produits ne répondent pas aux besoins définis et des erreurs résiduelles persistent dans le produit final un stock important de projets en attente faute d'une gestion rigoureuse.

La croissance explosive de la vitesse des ordinateurs et la baisse de leurs coûts engendrent une demande pour des logiciels complexes et augmentent les attentes des usagers.





1. Quelques défaillances relevées dans la Mise en Œuvre de Logiciels

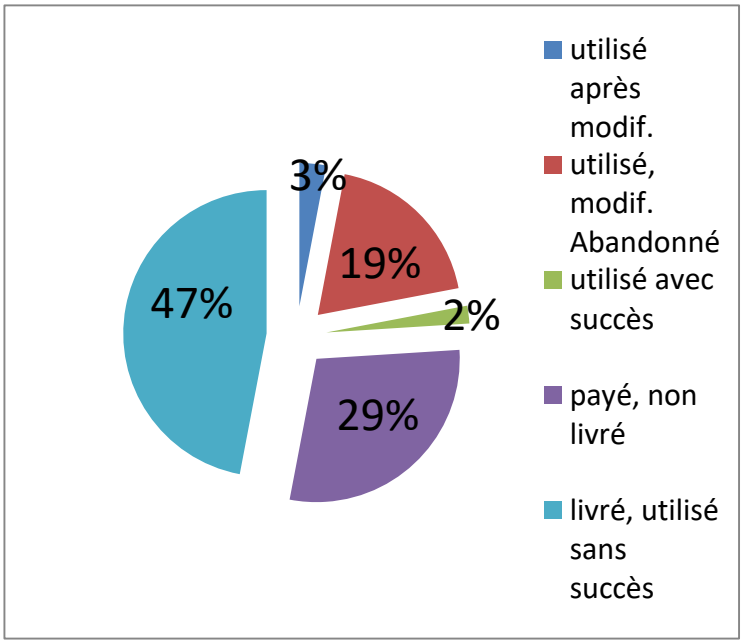
Les exemples de défaillances dues à un développement mal mené sont nombreux :

- la **sonde Mariner vers Vénus** s'est perdue dans l'espace à cause d'une erreur de programme FORTRAN ;
- La Destruction de la **sonde Mariner 1 le 27/07/1962** pour cause d'une erreur d'un caractère dans un programme Fortran (coût – 80 millions de dollars)
- en 1981, le premier lancement de la **navette spatiale européenne** a été retardé de deux jours à cause d'un problème logiciel. La navette a d'ailleurs été lancée sans que l'on ait localisé exactement le problème (mais les symptômes étaient bien délimités) ;
- la **SNCF** a rencontré des difficultés importantes (et coûteuses pour le contribuable français) pour la mise en service du système Socrate.
- L'explosion **d'Ariane 5, le 4 juin 1996**, qui a coûté près d'un demi-milliard de dollars (non assuré !), est due à une faute logicielle d'une composante dont le fonctionnement n'était pas indispensable durant le vol ;

- **TAURUS**, un projet d'informatisation de la bourse londonienne : définitivement abandonné après 4 années de travail et 100 millions de £ de pertes,
- La **mission VENUS** : passage à 500000 km au lieu de 5000 km à cause du remplacement d'une virgule par un point,
- **Le faux départ de la première navette Colombus** : manque de synchronisation entre calculateurs assurant la redondance (un délai modifié de 50 ms à 80 ms entraînant une chance sur 67 d'annulation par erreur de la procédure de tir),
- **Processeur Pentium** en 1994, **Bug dans la table de valeurs** utilisée par l'algorithme de division,
- **PlayStation Network**, avril 2011
  - Des millions de données personnelles et bancaires piratées,
  - Pertes financières de plusieurs milliards de dollars,
  - Vulnérabilité du réseau connue mais conséquences mal évaluées.
- **Outil de chiffrement OpenSSL**, mars 2014
  - 500 000 serveurs web concernés par la faille,
  - Vulnérabilité permettant de lire une portion de la mémoire d'un serveur distant.
- Le **22/12/2001 750 000 terminaux** de paiement ne répondent plus ce qui entraine de longues files d'attente causées par :
  - La Saturation des serveurs de la **société Atos** chargés des autorisations de paiements dépassant **600F**. Habituellement quelques dizaines de secondes, ce jour-là : 30mns.
  - Des clients abandonnent leurs chariots pleins. On a chiffré son préjudice à 2 millions d'euros.

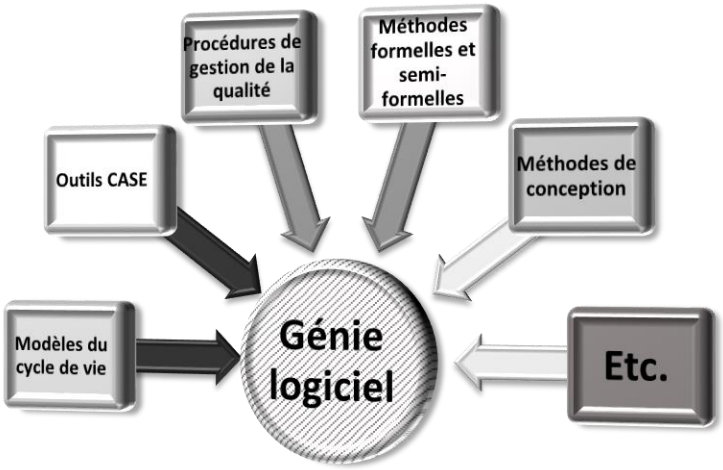
2. Les Principales Raisons des bugs :

- Erreurs humaines,
- Taille et complexité des logiciels,
- Taille des équipes de conception/développement,
- Manque de méthodes de conception,
- Négligence de la phase d'analyse des besoins du client,
- Négligence et manque de méthodes et d'outils des phases de validation/vérification.
- Difficulté de maîtrise **des coûts** (200 millions de dollars pour fabriquer OS-360),
- Difficulté de maîtrise **des délais** de réalisation (2 ans de retard pour les premiers compilateurs PL/1, Algol 68, ADA).



Solution

En octobre 1968, lors d'une conférence de l'OTAN
à Garmisch-Partenkirchen en Allemagne



III. Le Génie Logiciel ou L'art de Produire du Logiciel

A) Introduction et Origines du Terme.

La production de logiciel tout comme celle de n'importe quel autre bien nécessite la mise en œuvre de **méthodes, techniques et outils** dépassant largement le cadre de la seule programmation, regroupés sous le l'appellation générale de **Génie Logiciel**. Le terme est apparu pour la première fois lors d'une conférence de **l'OTAN à Garmisch en 1969**, a commencé à se répandre dans les **années 80**, notamment avec l'arrivée sur le marché d'Ateliers de Génie Logiciel (AGL). Il faut attendre le milieu des années 70 pour que le terme émerge : la première conférence sur le génie logiciel a lieu en 1973 et la revue **IEEE Transactions on Software Engineering** existe depuis 1975.

B) Généralités sur Concept de Génie Logiciel

1. Définitions

a) Le **Génie Logiciel** est défini le **30 décembre 1983** comme étant « l'ensemble des activités de conception et de mise en œuvre des produits et des procédures tendant à rationnaliser la production du logiciel et son suivi ».

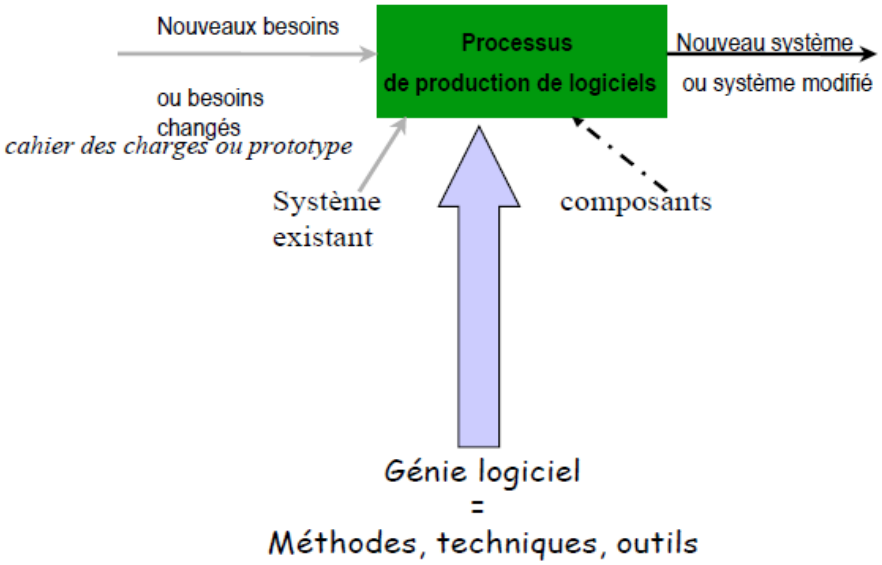
b) Ensemble de moyens mis en œuvre pour la construction de logiciels.

c) Le processus visant la résolution de problèmes posés par un client par le développement et l'évolution de systèmes logiciels de **grande taille** et de haute **qualité** en respectant les contraintes de **coûts** et de **temps.**

d) On appelle **Génie Logiciel** l'application de méthodes scientifiques au développement de théories, méthodes, techniques, langages et outils favorisant la production de logiciels de qualité.

e) Définition par l'IEEE du terme « Génie Logiciel »
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.

f) Le génie logiciel (anglais software engineering) est une science de génie industriel qui étudie les méthodes de travail et les bonnes pratiques des ingénieurs qui développent des logiciels. [Wikipedia]



La production de logiciel sous contrôle du génie logiciel

2. Les Objectifs du Génie Logiciel

Le GL se préoccupe des *procédés de fabrication des logiciels* de façon à s'assurer que les quatre critères suivants soient satisfaits :

- Le système qui est fabriqué en réponse aux *besoins* (*exigences*) des utilisateurs (correction fonctionnelle).
- La ***qualité*** correspond au contrat de service initial. La qualité du logiciel est une notion multiforme qui recouvre notamment :
  - ✓ la ***validité*** : aptitude d'un logiciel à réaliser exactement les tâches définies par sa spécification,
  - ✓ la ***fiabilité*** : aptitude d'un logiciel à assurer de manière continue le service attendu,
  - ✓ la ***robustesse*** : aptitude d'un logiciel à fonctionner même dans des conditions anormales,
  - ✓ l'***extensibilité*** : facilité d'adaptation d'un logiciel aux changements de spécification,
  - ✓ la ***réutilisabilité*** : aptitude d'un logiciel à être réutilisé en tout ou partie,
  - ✓ la ***compatibilité*** : aptitude des logiciels à pouvoir être combinés les uns aux autres,
  - ✓ l'***efficacité*** : aptitude d'un logiciel à bien utiliser les ressources matérielles (mémoire, CPU...),
  - ✓ la ***portabilité*** : facilité à être porté sur de nouveaux environnements matériels et/ou logiciels,
  - ✓ la ***traçabilité*** : capacité à identifier et/ou suivre un élément du cahier des charges lié à un composant d'un logiciel,
  - ✓ la ***vérifiabilité*** : facilité de préparation des procédures de recette et de certification,
  - ✓ l'***intégrité*** : aptitude d'un logiciel à protéger ses différents composants contre des accès ou des modifications non autorisés,
  - ✓ la ***facilité d'utilisation***, *d'entretien*, etc.
- Les *coûts* restent dans les limites prévues au départ.
- Les *délais* restent dans les limites prévues au départ.

3. La Règle du CQFD

→ Les **Coûts** restent dans les limites prévues au départ.

→ La **Qualité** correspond au contrat de service initial.
→ Le système répond aux besoins des utilisateurs et réalise les **Fonctionnalités** demandées.
→ Les **Délais** restent dans les limites prévues au départ.

4. Les principales Tâches du génie Logiciel

Le génie logiciel englobe les tâches suivantes :

- ✓ **Spécifications** : capture des besoins, cahier des charges, spécifications fonctionnelles et techniques,
- ✓ **Conception** : analyse, choix de la modélisation, dentition de l'architecture, dentition des modules et interfaces, dentition des algorithmes,
- ✓ **Implantation** : choix d'implantations, codage du logiciel dans un langage cible,
- ✓ **Intégration** : assemblage des différentes parties du logiciel,
- ✓ **Documentation** : manuel d'utilisation, aide en ligne,
- ✓ **Vérification** : tests fonctionnels, tests de la fiabilité, tests de la sureté,
- ✓ **Validation** : recette du logiciel, conformité aux exigences du Cahier Des Charges,
- ✓ **Déploiement** : livraison, installation, formation,
- ✓ **Maintenance** : corrections, évolutions.

5. Les protagonistes du génie logiciel (les quatre P)

La production de logiciels est une tâche extrêmement complexe qui pose de nombreux challenges :

a) La Coordination de nombreuses ***Personnes*** (organisées en équipes),
b) L'Objectif premier = construire un ***Produit*** (qui répond aux besoins du/des client(s)).
c) L'Effort de développement doit être organisé en ***Projet*** (avec un planning rigoureux pour assurer le succès).
d) Pour développer le Produit avec succès, les activités des personnes doivent être organisées suivant un ***Processus*** ordonné bien défini.

6. Les Principes fondamentaux du génie Logiciel

De manière générale on récence sept principes fondamentaux (proposés par **Carlo Ghezzi**):

Principes fondamentaux	Caractéristiques	Explications
Rigueur	principale source d'erreurs humaine, s'assurer par tous les moyens que ce qu'on écrit est bien ce qu'on veut dire et que ça correspond à ce qu'on a promis (outils, revue de code...)	Le niveau maximum de rigueur est la *formalité*, c'est-à-dire le cas où les descriptions et les validations s'appuient sur des notations et lois mathématiques.
Abstraction	extraire des concepts généraux sur lesquels raisonner, puis instancier les solutions sur les cas particuliers	L'abstraction consiste à ne considérer que les aspects jugés importants d'un système à un moment donné, en ignorant les autres aspects non significatifs.
Décomposition en sous-problèmes	traiter chaque aspect séparément, chaque sous-problème plus simple que problème global	consiste à considérer séparément différents aspects d'un problème afin d'en maîtriser la complexité. C'est un aspect de la stratégie générale du « diviser pour régner ».
Modularité	partition du logiciel en modules interagissant, remplissant une fonction et ayant une	La modularité permet de considérer séparément le contenu du module et les

	interface cachant l'implantation aux autres modules	relations entre modules (ce qui rejoint l'idée de séparation des questions). Elle facilite également la réutilisation de composants bien délimités
Construction incrémentale	construction pas à pas, intégration progressive	Un procédé incrémental atteint son but par étapes en s'en approchant de plus en plus ; chaque résultat est construit en étendant le précédent.
Généricité	proposer des solutions plus générales que le problème pour pouvoir les réutiliser et les adapter à d'autres cas	Il est parfois avantageux de remplacer la résolution d'un problème spécifique par la résolution d'un problème plus général. Cette solution générique (paramétrable ou adaptable) pourra être réutilisée plus facilement.
Anticipation des évolutions	liée à la généricité et à la modularité, prévoir les ajouts/modifications possibles de fonctionnalités	La caractéristique essentielle du logiciel, par rapport à d'autres produits, est qu'il est presque toujours soumis à des changements continuels (corrections d'imperfections et évolutions en fonction des besoins qui changent).

De façon transversale et en appui des sept principes cités on note :

- ***La Documentation*** : essentielle pour le suivi de projet et la communication au sein de l'équipe de projet
- ***La Standardisation/normalisation*** : aide à la communication pour le développement, la maintenance et la réutilisation.

Remarque :

Il faut noter que les principes ci-dessus sont très *abstraits* et ne sont *pas utilisables directement*. Mais ils font partie du vocabulaire de base du génie logiciel. Ces principes ont un impact réel sur beaucoup d'aspects et constituent le type de connaissances le plus stable, dans un domaine où les outils, les méthodes et les techniques évoluent très vite.

7. Les Outils pour les activités du génie logiciel

Les activités de production du logiciel sont parfois complexes et nécessitent de la rigueur dans l'élaboration et exploitation des résultats fournis par chaque étape, raison pour laquelle on fait appel à des outils pour supporter les activités pendant tout le cycle de vie du logiciel. Ces outils sont appelés **CASE** (**Computer-Aided Software Engineering**). Les **CASE** sont classés selon leurs fonctions et phases où ils sont utilisés. On recense les essentiellement les classes de CASE suivantes :

- outils de planification (comme PERT, GANTT),
- outils d'édition de textes, d'images,
- outils de traçabilité,
- outils de gestion de configurations,

- outils de **prototypage**,
- outils d'aide à la conception,
- compilateurs et environnements de programmation,
- outils d'analyse de programmes et code,
- outils de tests,
- outils de déboguage,
- outils de documentation,
- outils de ré-ingénierie.

8. Les Domaines liés au Génie Logiciel

Selon **Swebok**, les domaines liés au génie logiciel :

- Les exigences du logiciel,
- La conception du logiciel,
- La construction du logiciel,
- Les tests logiciels,
- La maintenance du logiciel,
- La gestion de configuration du logiciel,
- L'ingénierie de la gestion logicielle,
- L'ingénierie des processus logiciels,
- L'ingénierie des outils et méthodes logicielles,
- L'assurance qualité du logiciel.

C) LE DÉVELOPPEMENT du LOGICIEL

Il s'agit de la mise en œuvre des outils, des méthodes et et des principes qui régissent et règlemente l'élaboration d'un logiciel.

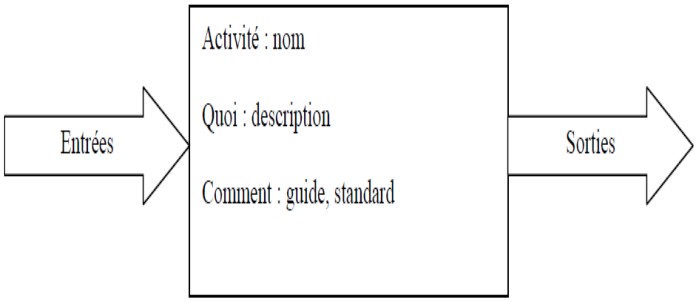
Comparé aux autres produits, le logiciel présente quelques particularités : le problème de production en série ne se pose pas.

- Le logiciel est un produit unique fait avec un *procédé de production* ou un *processus de développement*.

1. Les activités d'un processus de développement d'un Logiciel

Elles sont décrites en termes de :

a) Les Entrées de l'activité (matière première) ;
b) Les Sorties de l'activité (résultat) ;
c) Les Intervenants et rôles (qui ?) ;
d) La Description de l'activité (quoi – quel est le problème à traiter ?) ;
e) Les Standards, guides, « best practices » à appliquer (comment ?).



2. Les étapes de développement d'un logiciel :

a) *La Faisabilité* : (POURQUOI ?)

- Pourquoi développer le logiciel ?
- Y a-t-il de meilleures alternatives ?
- Comment procéder pour faire ce développement ?
- Y a-t-il un marché pour le logiciel ?
- Quels moyens faut-il mettre en œuvre ? A-t-on le budget, le personnel, la matériel nécessaires ?

Activité	Etude préalable
Description	Etudier la faisabilité du projet, ses contraintes techniques (coût, temps, qualité) et les alternatives possibles.
Entrées	Problème à résoudre, objectifs à atteindre.
Sorties	OUI (la décision est prise de réaliser le logiciel) ou NON (le projet est abandonné).

b) *Spécification :* (QUOI ?)

Activité	Spécifier
Description	Décrire ce que doit faire le logiciel (comportement en boite-noire). Décrire comment vérifier en boite-noire que le logiciel fait bien ce qui est exigé.
Entrées	Client qui à une idée de ce qu'il veut. Exigences, désir, besoins… concernant le système permettant de résoudre le problème.
Sorties	Cahier des charges du logiciel (ou spécification du logiciel). Des procédures de validation. Version provisoire des manuels d'utilisation et d'exploitation du logiciel.

c) *Conception* : (COMMENT ?)

Activité	Concevoir
Description	Organiser le logiciel afin qu'il puisse satisfaire les exigences de la spécification. Faire les principaux choix techniques pour satisfaire les exigences de la spécification.
Entrées	Une spécification.
Sorties	Une description des décisions de conception. Des procédures de tests qui permettent de vérifier que les décisions de conception sont correctement implémentées en code source et qu'elles contribuent à satisfaire les exigences de la spécification.

d) *Implantation* : (COMMENT ?)

Activité	Coder et tester
Description	Ecrire le code source du logiciel. Tester le comportement du code source afin de vérifier qu'il réalise les responsabilités qui lui sont allouées.
Entrées	Spécification, conception.
Sorties	Code source. Tests unitaire. Documentation.

e) *Intégration :*

Activité	Intégrer
Description	Assembler le code source du logiciel (éventuellement partiellement) et dérouler les tests d'intégration.
Entrées	Conception, code source, tests d'intégration (tests).
Sorties	Un rapport de tests d'intégration.

f) Validation :

Activité	Valider
Description	Construire le logiciel complet exécutable. Dérouler les tests de validation sur le logiciel complet exécutable.
Entrées	Logiciel complet exécutable à valider. Tests de validation.
Sorties	Rapport de tests de validation.

g) Maintenance :

Activités :

- Maintenance corrective : correction des bugs.
- Maintenance adaptative : ajuster le logiciel.
- Maintenance perfective, d'extension : augmenter / améliorer les possibilités du logiciel.

Productions :

- Logiciel corrigé ;
- Mises à jour ;
- Documents corrigés.

3. Les difficultés du développement

- ✓ La flexibilité : Absence de contrainte physique,
- ✓ Absence de limite à la complexité (a priori),
- ✓ Possibilité de modifier le programme a posteriori
- ✓ Possibilité de commencer le programme à priori (avant d'avoir défini précisément la tâche à réaliser),
- ✓ Un succès local n'implique pas la réussite du projet global,
- ✓ L'absence de références standards,
- ✓ Chaque logiciel est un prototype,
- ✓ Une application est souvent une innovation.