GENERALITES SUR LE GENIE (L'INGENIERIE) LOGICIEL

Le génie logiciel est une branche de l'ingénierie associée au développement de logiciels utilisant des principes, méthodes et procédures scientifiques bien définis. Le résultat de l'ingénierie logicielle est un produit logiciel efficace et fiable.

Commençons par comprendre ce que signifie « le génie logiciel ». Le terme est composé de deux mots, le logiciel et l'ingénierie :

 Le logiciel est plus qu'un code de programme. En fait, un programme est un code exécutable, qui sert à des fins de calcul, par contre, le logiciel, est considéré comme une collection de code de programmation exécutable, des bibliothèques associées et de documentations. Ainsi, lorsque le logiciel, est conçu pour une exigence spécifique, est appelé un « produit logiciel ».

D'autre part, l'ingénierie (génie) consiste à développer des produits, en utilisant des principes et méthodes scientifiques bien définis.

1. DEFINITION ET CONTEXTE D'ETUDES

IEEE définit le génie logiciel comme l'application d'une approche systématique, disciplinée et quantifiable au développement, à l'exploitation et à la maintenance des logiciels; c'est-à-dire l'application de l'ingénierie aux logiciels.

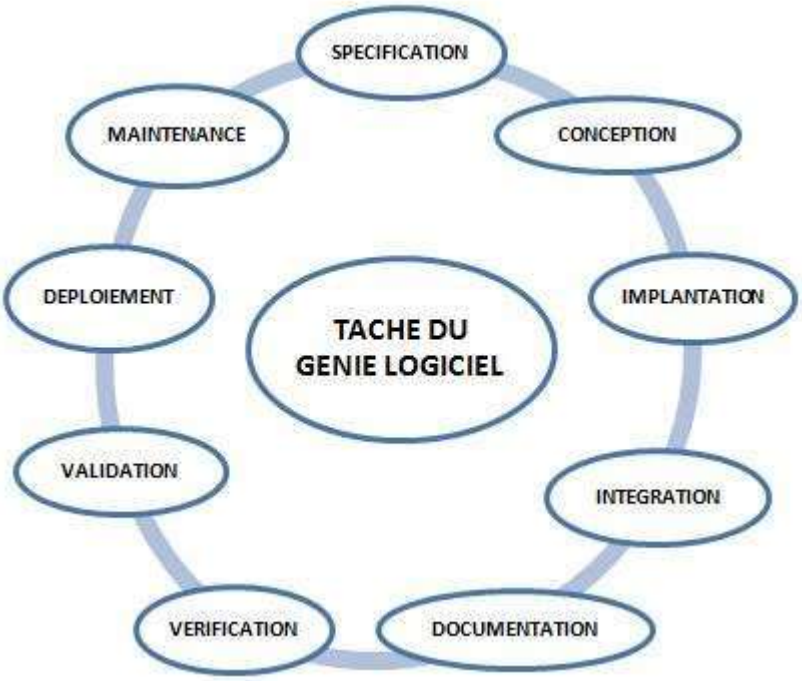
Fritz Bauer, un informaticien allemand, définit le génie logiciel comme l'établissement et l'utilisation de principes d'ingénierie afin d'obtenir des logiciels économiques, fiables et efficaces sur des machines réelles.

Dans le cadre de ce cours, le génie logiciel sera considéré comme étant un ensemble d'activités de conception et de mise en œuvre des produits et des procédures tendant à rationaliser (normaliser) la production du logiciel et son suivi. Autrement dit, le génie logiciel

est l'art de produire de bons logiciels, au meilleur rapport qualité prix. Remarquons que cette définition fait référence à deux aspects indispensables dans la conception d'un logiciel:

- Le processus ou procédure de développement des logiciels - ensemble de formalités, des marches à suivre et des démarches pour obtenir un résultat déterminé et ;
- La maintenance et le suivi des logiciels - ensemble d'opérations permettant de maintenir le fonctionnement d'un équipement informatique.

Le génie logiciel englobe les tâches suivantes :



- **La Spécification :** capture des besoins, cahier des charges, spécifications fonctionnelles et techniques
- **La Conception :** analyse, choix de la modélisation, définition de l'architecture, définition des modules et interfaces, définition des algorithmes
- **L'Implantation :** choix d'implantations, codage du logiciel dans un langage cible
- **L'Intégration :** assemblage des différentes parties du logiciel
- **La Documentation :** manuel d'utilisation, aide en ligne
- **La vérification :** tests fonctionnels, tests de la fiabilité, tests de la sûreté
- **La Validation :** recette du logiciel, conformité aux exigences du CDC
- **Le Déploiement :** livraison, installation, formation
- **La Maintenance :** corrections, et évolutions.

2. ÉVOLUTION DU LOGICIEL

Le processus de développement d'un logiciel à l'aide de principes et de méthodes du génie logiciel est appelé « évolution logicielle ». Cela inclut le développement initial du logiciel et sa maintenance et ses mises à jour, jusqu'à ce que le logiciel désiré soit développé, ce qui répond aux exigences attendues.

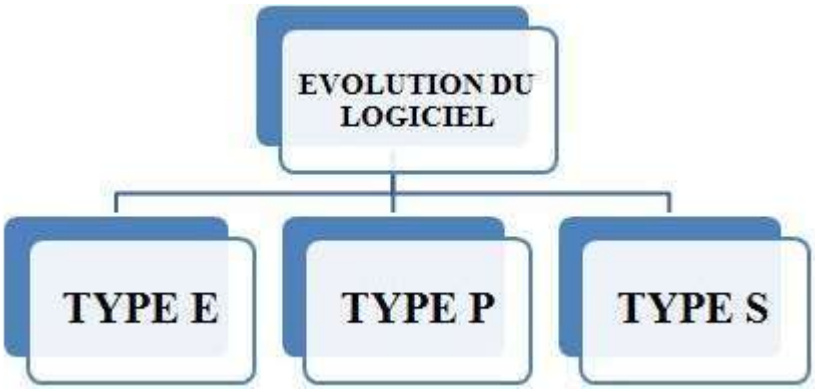
L'évolution commence par le processus de collecte des exigences. Après quoi les développeurs créent un prototype du logiciel prévu et le montrent aux utilisateurs pour obtenir leurs commentaires dès les premières étapes du développement de logiciels. Les utilisateurs suggèrent des modifications sur lesquelles plusieurs mises à jour et maintenances consécutives continuent à évoluer. Ce processus passe au logiciel d'origine, jusqu'à ce que le logiciel souhaité soit accompli.

Même lorsque l'utilisateur dispose du logiciel souhaité, la technologie évolutive et les exigences changeantes forcent le logiciel à changer en conséquence. Récréer des logiciels à partir de zéro et aller en tête-à-tête avec des exigences n'est pas réalisable.

La seule solution possible et économique consiste à mettre à jour le logiciel existant afin qu'il corresponde aux dernières exigences.

2.1. LOIS D'EVOLUTION DU LOGICIEL

Meir Lehman a donné des lois sur l'évolution des logiciels. Il a divisé le logiciel en trois catégories différentes :



- Type S (type statique) - Il s'agit d'un logiciel qui fonctionne selon des spécifications et des solutions définies. La solution et la méthode pour y parvenir, les deux sont immédiatement comprises avant le codage. Le logiciel de type s est le moins soumis à des modifications, ce qui en fait le plus simple. Par exemple, programme de calculatrice pour le calcul mathématique.

- Type P (type pratique) - Il s'agit d'un logiciel avec un ensemble de procédures.

Ceci est défini par exactement ce que les procédures peuvent faire. Dans ce logiciel, les spécifications peuvent être décrites mais la solution n'est pas évidente instantanément. Par exemple, un logiciel de jeu.

- E-type (type embarqué) - Ce logiciel fonctionne étroitement comme exigence d'un environnement réel. Ce logiciel a un haut degré d'évolution car il existe divers changements dans les lois, les taxes, etc. dans les situations réelles. Par exemple, logiciel de trafic en ligne.

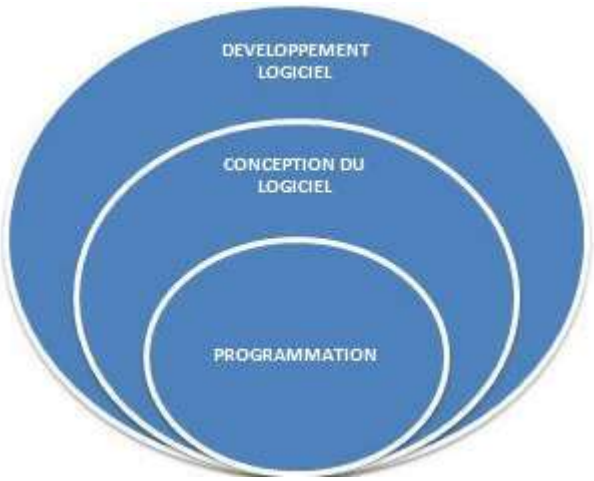
2.2. EVOLUTION DU LOGICIEL E-TYPE

Meir Lehman a donné huit lois pour l'évolution des logiciels de type E :

- **Changement continu** - Un logiciel de type E doit continuer à s'adapter aux changements du monde réel, sinon il devient progressivement moins utile.
- **Une complexité croissante** - Au fur et à mesure de l'évolution d'un système logiciel de type E, sa complexité a tendance à augmenter à moins que des travaux ne soient effectués pour le maintenir ou le réduire.
- **Conservation de la familiarité** - La familiarité avec le logiciel ou la connaissance de la façon dont il a été développé, pourquoi il a été développé de cette manière particulière, etc. doit être conservée à tout prix pour mettre en œuvre les modifications du système.
- **Poursuite de la croissance** - Pour qu'un système de type E destiné à résoudre un problème commercial, sa taille de mise en œuvre des modifications augmente en fonction des changements de style de vie de l'entreprise.
- **Réduire la qualité** - Un système logiciel de type E diminue en qualité, sauf si rigoureusement entretenu et adapté à un environnement opérationnel changeant.
- **Systèmes de rétroaction** – C'est un processus du logiciel pouvant se déclencher automatiquement après une opération défaillante, visant à provoquer une action correctrice en sens contraire. Les systèmes logiciels de type E constituent des systèmes de rétroaction multi-boucles à plusieurs niveaux et doivent être traités comme tels pour être modifiés ou améliorés avec succès.
- **Autorégulation** - Les processus d'évolution du système de type s'autorégulent, la distribution des mesures du produit et du procédé étant presque normale.
- **Stabilité organisationnelle** - Le taux d'activité global effectif moyen dans un système de type E en évolution est invariant pendant la durée de vie du produit

3. LES PARADIGMES LOGICIELS

Les paradigmes logiciels font référence aux méthodes et aux étapes qui sont prises lors de la conception du logiciel. Il existe de nombreuses méthodes proposées et sont en cours de réalisation, mais nous avons besoin de voir où se situent ces paradigmes dans le génie logiciel. Ceux-ci peuvent être combinés en différentes catégories, bien que chacun d'eux soit contenu l'un dans l'autre :



Il est à noter que le paradigme de la programmation est un sous-ensemble du paradigme de conception de logiciels, qui est en outre un sous-ensemble du paradigme de développement logiciel.

3.1. PARADIGME DE DEVELOPPEMENT LOGICIEL

Ce paradigme est connu sous le nom de paradigmes d'ingénierie logicielle, où tous les concepts d'ingénierie relatifs au développement de logiciels sont appliqués. Il comprend diverses recherches et la collecte des exigences qui aident le produit logiciel à construire. Ce paradigme fait partie du développement logiciel et inclut : Le rassemblement des besoins ; la Conception des logiciels et la planification des tâches.

3.2. PARADIGME DE CONCEPTION

Ce paradigme fait partie du développement de logiciels et comprend : la conception ; la maintenance et l'organisation des activités à exécuter.

3.3. PARADIGME DE PROGRAMMATION

Ce paradigme est étroitement lié à l'aspect programmation du développement logiciel.

Cela inclut : Le codage ; Le Test et L'intégration des besoins.

4. BESOINS DU GENIE LOGICIEL

Les besoins du génie logiciel est dû au taux de changement plus élevé des besoins des utilisateurs et de l'environnement sur lequel le logiciel fonctionne :

➢ ***Gros Logiciels (volumineux)*** - Il est plus facile de construire un mur dans une maison que dans un bâtiment, de même que la taille des logiciels devient importante.

➢ ***Évolutivité*** - Si le processus logiciel n'était pas basé sur des concepts scientifiques et techniques, il serait plus facile de recréer de nouveaux logiciels que de mettre à niveau un logiciel existant.

➢ ***Prix*** – comme L'industrie du matériel a montré ses compétences et une production importante a fait baisser le prix du matériel informatique électronique. Mais le coût du logiciel reste élevé si le processus approprié n'est pas adapté.

➢ ***Nature dynamique*** - La nature toujours croissante et adaptative du logiciel dépend énormément de l'environnement dans lequel l'utilisateur travaille. Si la nature du logiciel évolue constamment, de nouvelles améliorations doivent être apportées dans le logiciel existant. C'est là que l'ingénierie logicielle joue un rôle important.

➢***Gestion de la qualité*** - Un meilleur processus de développement logiciel fournit un produit logiciel de meilleure qualité.

5. CARACTERISTIQUES D'UN BON LOGICIEL

La caractéristique d'un logiciel est un ensemble de traits dominants ou l'expression de la correspondance entre une cause (grandeur d'entrée) et un effet (grandeur de sortie) dans la production ou le processus de développement des logiciels. Un produit logiciel doit évaluer en fonction de ce qu'il offre et de sa facilité d'utilisation. Un bon logiciel doit satisfaire les 3 catégories de critères suivants :

- Critères généraux ;
- Critères externes,
- Critères internes.

5.1. CRITERES GENERAUX

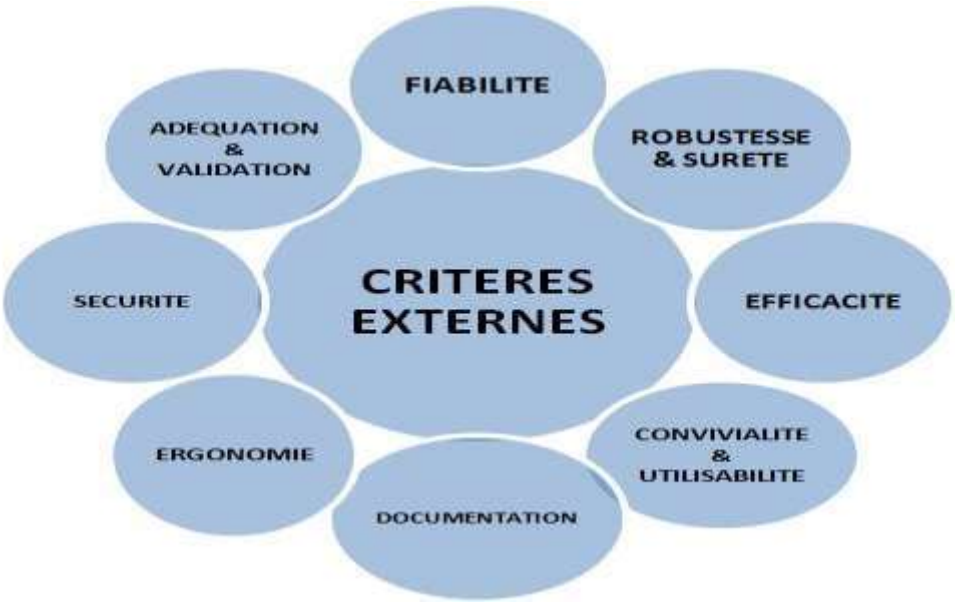
Les critères généraux sont en somme, des principes et éléments de référence qui permettent de juger ; d'estimer et de vérifier régulièrement si le processus de développement d'un logiciel possède ou non les différentes propriétés déterminées.

Cette catégorie peut se matérialiser selon 3 différents processus (aspects) :

- **Opérationnel :** Cela nous indique dans quelle mesure le logiciel fonctionne bien dans les opérations. Ces opérations peuvent être mesurées entre autre part : budgétisation, facilité d'utilisation, efficacité, exactitude, fonctionnalité, fiabilité, sécurité.
- **Transitionnel :** Cet aspect est important lorsque le logiciel est déplacé d'une plate-forme à une autre : Portabilité, Interopérabilité, Réutilisation et Adaptabilité.
- **Maintenance :** Cet aspect explique comment un logiciel a la capacité de se maintenir dans une infrastructure et environnement en constante évolution : maintenabilité, modularité, Flexibilité et Évolutivité. En résumé, le génie logiciel est une branche de l'informatique, qui utilise des concepts d'ingénierie bien définis pour produire des produits logiciels efficaces, durables, évolutifs, économiques et ponctuels.

5.2. CRITERES EXTERNES

Les critères externes expriment ce qu'est un bon logiciel du point de vue des utilisateurs. Un logiciel de qualité doit être :

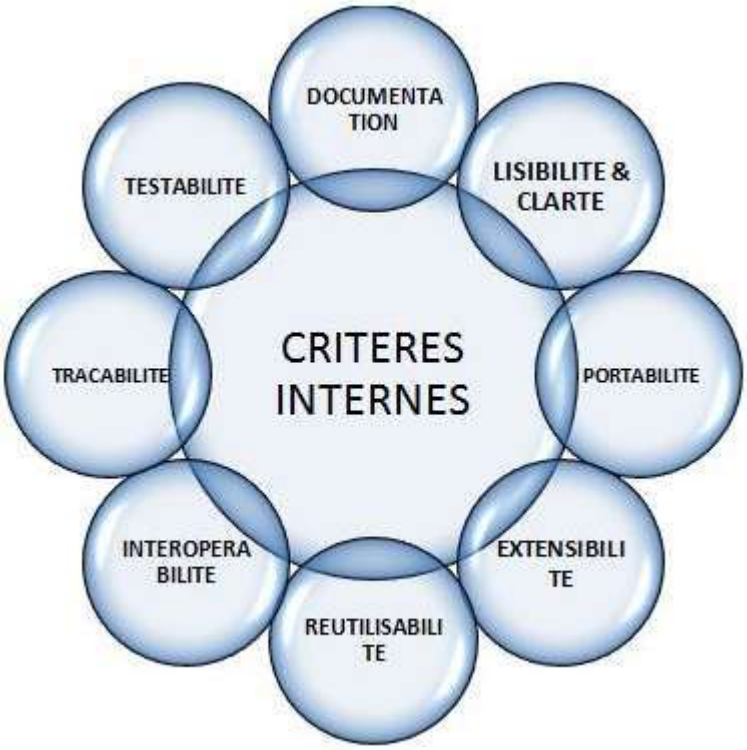


- **Fiabilité** : (correction, justesse et conformité) : le logiciel est conforme à ses spéciations, les résultats sont ceux attendus.
- **Robustesse et Sureté** :(dysfonctionnements ou ne plante pas) : le logiciel fonctionne raisonnablement en toutes circonstances, rien de catastrophique ne peut survenir, même en dehors des conditions d'utilisation prévues
- **Efficacité** : (Le logiciel fait-il bon usage de ses ressources, en terme d'espace mémoire, et temps d'exécution),
- **Convivialité et Utilisabilité** : (Est-il facile et agréable à utiliser),
- **Documentable** : (accompagné d'un manuel utilisateur, ou d'un tutoriel).
- **Ergonomique**: L'architecture du logiciel doit particulièrement être adaptée aux conditions de travail de l'utilisateur
- **Sécurité** : c'est la sûreté (assurance) et la garantie offerte par un logiciel, ou l'absence du danger lors de l'exploitation du logiciel.
- **Adéquation et validité** : c'est la conformité au maquettage du logiciel et au but qu'on se propose.

- **Intégrité** : c'est l'état d'un logiciel a conservé sans altération ses qualités et son degré originel. Autrement dit, C'est l'aptitude d'un logiciel à protéger son code et ses données contre des accès non autorisé.

5.3. CRITERES INTERNES

Les critères de qualité internes expriment ce qu'est un bon logiciel du point de vue du développeur. Ces critères sont essentiellement liés à la maintenance d'un logiciel. Un bon logiciel doit être facile à maintenir, et pour cela doit être :



- **Documentable** : (a-t-il été précédé par un document de conception ou architecture).
- **Lisibilité et Clarté** : (est-il écrit proprement, et en respectant les conventions de programmation),
- **Portabilité**: Un même logiciel doit pouvoir fonctionner sur plusieurs machines ainsi le rendre indépendant de son environnement d'exécution ;
- **Extensibilité** : (est-il souple ou flexible ? ou permet-il l'ajout possible de nouvelles fonctionnalités).
- **Réutilisabilité** : (des parties peuvent être réutilisées pour développer d'autres logiciels similaires).
- **Interopérabilité et coulabilité** : Un logiciel doit pouvoir interagir en synergie avec d'autres logiciels.
- **Traçabilité** : c'est la possibilité de suivre un produit aux différents stades de sa production, de sa transformation et de sa commercialisation.
- **Testabilité et vérifiabilité** : c'est la possibilité de soumettre un logiciel à une épreuve de confirmation de la tâche à exécuter.

6. CATEGORIES DE LOGICIELS

Il existe plusieures catégories des logiciels notamment :

- LOGICIELS GENERIQUES : c'est sont des logiciels qui vendus comme les produits courants sur le marché informatique. Dans cette catégorie, on en distingue autant :
  - Logiciels amateurs : Il s'agit de logiciels développés par des « amateurs» (par exemple par des gens passionnés ou des étudiants qui apprennent à programmer). Bref, ce sont des logiciels sans impact économique significatif sur l'ensemble.
  - Logiciels « jetables » ou « consommables » : Il s'agit de logiciels comme par exemple les logiciels des traitements de texte ou les tableurs pour les entreprises. Ces logiciels ne coûtent pas très cher, et peuvent être remplacés facilement au sein d'une entreprise sans engendrer des risques majeurs. Ils sont souvent largement diffusés.
- LOGICIELS SPECIFIQUES : ce sont des logiciels développés pour une application précise et destinés à un seul client. Dans cette catégorie, on en distingue autant :
  - Logiciels essentiels au fonctionnement d'une entreprise : Ce type de logiciel est le fruit d'un investissement non négligeable et doit avoir un comportement fiable, sûr et sécurisé.
  - Logiciels critiques : Il s'agit de logiciels dont l'exécution est vitale, pour lesquels une erreur peut coûter très cher ou coûter des vies humaines.

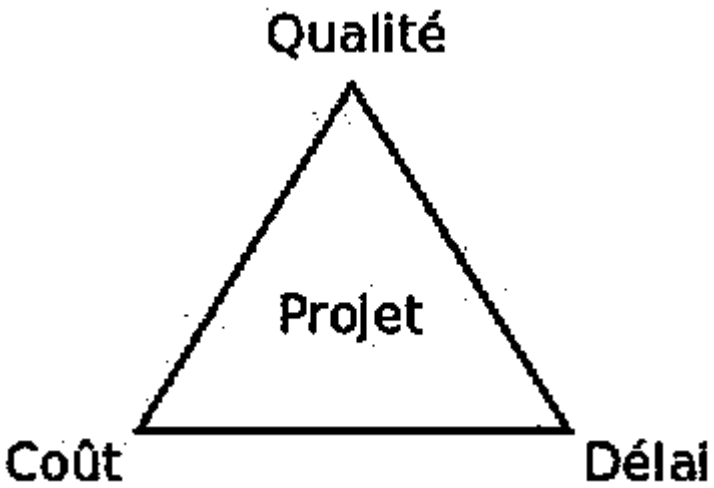
Exemple : domaines du transport, de l'aviation, de la médecine, de l'armement, etc.

L'objectif de qualité d'un logiciel est différent suivant la catégorie de logiciel. En particulier, les logiciels essentiels au fonctionnement d'une entreprise, et plus encore les logiciels critiques, doivent avoir un haut niveau de qualité.

7. ENJEUX DU GENIE LOGICIEL

Un enjeu peut être considéré comme un ensemble des risques encourus par un développeur pour en mise en œuvre d'un logiciel. Le génie logiciel vise à rationaliser et à optimiser le processus de production d'un logiciel. Les enjeux associés sont multiples :

- Adéquation aux besoins du client.
- Respect des délais de réalisation prévus.
- Maximisation des performances et de la fiabilité.
- Facilitation de la maintenance et des évolutions ultérieures.



Comme tout projet, la réalisation d'un logiciel est soumise à des exigences contradictoires et difficilement conciliables (triangle coût-délai-qualité).La qualité d'un logiciel peut s'évaluer à partir d'un ensemble de facteurs tels que :

- Le logiciel répond-il aux besoins exprimés ?
- Le logiciel demande-t-il peu d'efforts pour évoluer aux regards de nouveaux besoins ?
- Le logiciel peut-il facilement être transféré d'une plate-forme à une autre ? ...

Sans être une solution miracle, le génie logiciel a pour objectif de maximiser la surface du triangle en tenant compte des priorités du client.

I.8. DIMENSIONS DU GENIE LOGICIEL

La dimension peut être envisagée comme étant un ensemble de mesures de chacune des grandeurs nécessaires à l'évaluation du logiciel. Le génie logiciel couvre l'ensemble du cycle de vie d'un logiciel. Il étudie toutes les activités qui mènent d'un besoin à la livraison du logiciel, y compris dans ses versions successives, jusqu'à sa fin de vie. Les dimensions du génie logiciel sont donc multiples :

- Analyse des besoins du client.
- Définition de l'architecture du logiciel.
- Choix de conception.
- Règles et méthodes de production du code source.
- Gestion des versions.
- Test du logiciel.
- Documentation.
- Organisation de l'équipe et interactions avec le client.
- ...

9. LES PRINCIPES DU GENIE LOGICIEL

Un certain nombre de grands principes (de bon sens) se retrouvent dans toutes ces méthodes. En voici une liste proposée par Ghezzi7 :

1. La rigueur : Les principales sources de défaillances d'un logiciel sont d'origine humaine. À tout moment, il faut se questionner sur la validité de son action. Des outils de vérification accompagnant le développement peuvent aider à réduire les erreurs. Cette famille d'outils s'appelle (CASE "Computer Aided Software Engineering"). C'est par exemple: typeurs, générateurs de code, assistants de preuves, générateurs de tests, outil d'intégration continue, fuzzer, . . .

2. La Généralisation : regroupement d'un ensemble de fonctionnalités semblables en une fonctionnalité paramétrable (généricité8 et héritage)

3. La Structuration : c'est la manière de décomposer un logiciel (utilisation d'une méthode bottom-up ou top-down). C'est la décomposition des problèmes en sous problèmes indépendants.

Outre, Il s'agit de la Décorrélation. Les problèmes pour n'en traiter qu'un seul à la fois. Ou de la Simplification. Les problèmes (temporairement) pour aborder leur complexité progressivement. C'est par exemple : le modèle TCP.

4. La modularité : Il s'agit de partitionner le logiciel en modules qui ont une cohérence interne (des invariants) ; et possèdent une interface ne divulguant sur le contenu du module que ce qui est strictement nécessaire aux modules clients.

L'évolution de l'interface est indépendante de celle de l'implémentation du module. Les choix d'implémentation sont indépendants de l'utilisation du module.

Ce mécanisme s'appelle le camouflage de l'information (information hiding).

5. L'abstraction : Il s'agit de présenter des concepts généraux regroupant un certain nombre de cas particuliers et de raisonner sur ces concepts généraux plutôt que sur chacun des cas particuliers. Le fait de fixer la bonne granularité de détails permet de raisonner plus efficacement et de factoriser le travail en instanciant le raisonnement général sur chaque cas particulier. C'est par exemple : les classes abstraites dans les langages à objets, le polymorphisme de Caml et le generics de Java, les foncteurs de Caml et le templates de C++, . . .

6. La construction incrémentale: Un développement logiciel a plus de chances d'aboutir s'il suit un cheminement incrémental (baby-steps)

7. La Documentation : correspond à la gestion des documents incluant leur identification, acquisition, production, stockage et distribution. Il est primordial de prévoir les évolutions possibles d'un logiciel pour que la maintenance soit la plus efficace possible. Pour cela, il faut s'assurer que les modifications à effectuer soient le plus locales possibles. Ces modifications ne devraient pas être intrusives car les modifications du produit existant remettent en cause ses précédentes validations. Concevoir un système suffisamment riche pour que l'on puisse le modifier incrémentalement est l'idéal.

8. La Vérification : c'est la détermination du respect des spéciations établies sur la base des besoins identités dans la phase précédente du cycle de vie.