Processus de Développement Introduction Génie Logiciel

Première partie

Génie logiciel

UML

Pr Hafidi Imad
imad.hafidi@gmail.com

Où se trouve un Logiciel

Le logiciel est omnipresent

- Le Logiciel est quasiment « partout »
- Beaucoup de choses de notre quotidien sont inimaginables sans le logiciel
- Bureautique, voyages aériens, scolarité, recherche scientifique, loisirs,…
- Par conséquent, notre vie dépend très fortement de la qualité des logiciels qui la gèrent

Impacts positives du logiciel

- Le logiciel a amélioré le quotidien de plusieurs manières :
- Le logiciel accélère les traitements
- Le logiciel ne « se lasse » pas
- Le logiciel résout des problèmes complexes rapidement
- Capacité de calcul, de stockage et de traitement incroyables
- Le logiciel a introduit de nouveaux loisirs Exemples :
  - Paiement électronique , Achat sur internet, Recherche d'information, Logiciels métier (ERP, tableurs, traitement de texte), Bibliothèques en ligne, …

Impact mauvaise qualité du logiciel

- La paranoïa du 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.

Définition du 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 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.

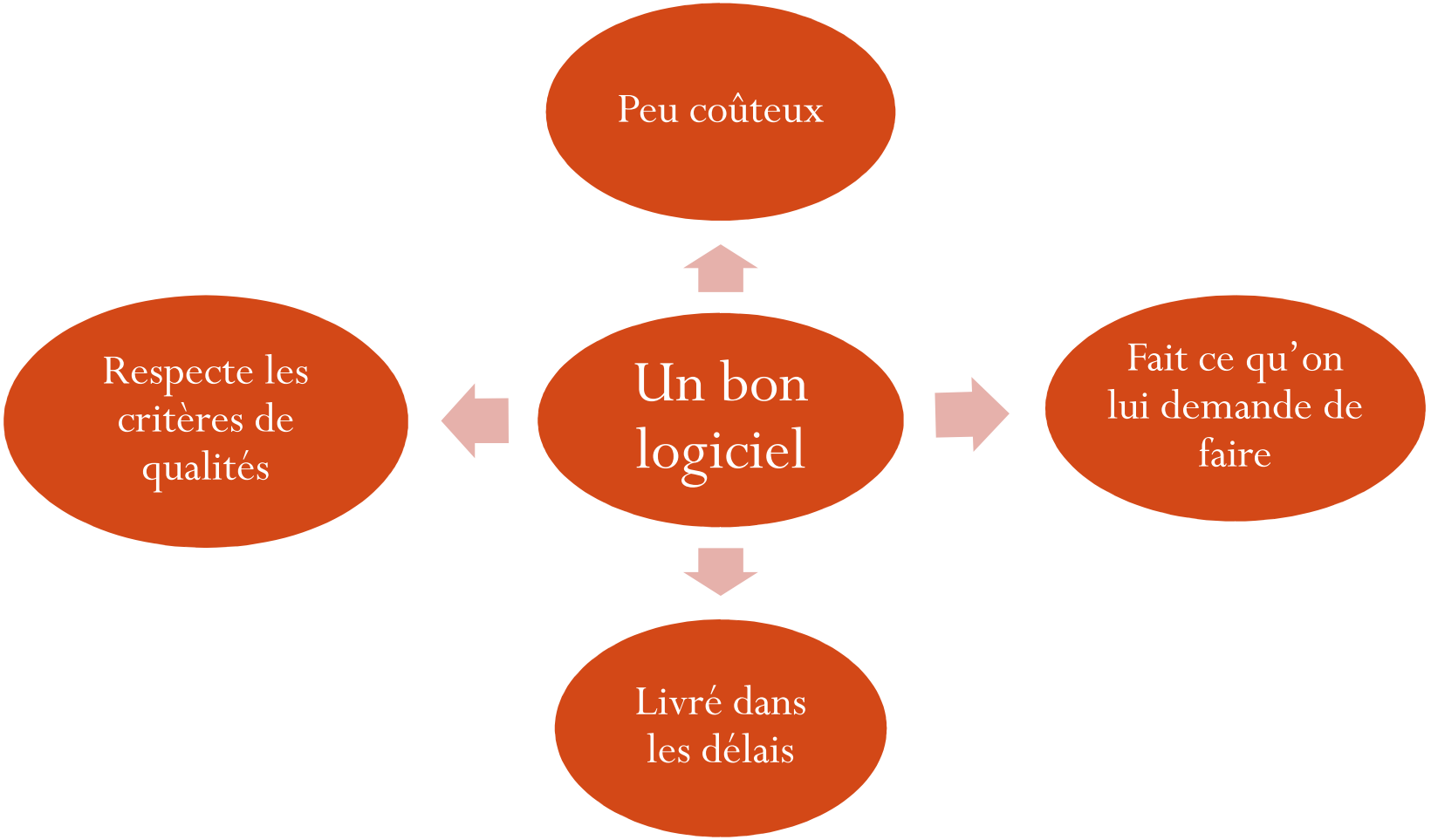
Software et Hardware

- Le « software » et le « hardware » sont indissociables :
  - Le « hardware » a besoin du « software » pour être piloté
  - Le « software » a besoin du hardware pour être exécuté
- 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 mobilités et des unités mobiles (Smartphones, tablettes, notebooks,…etc.,)

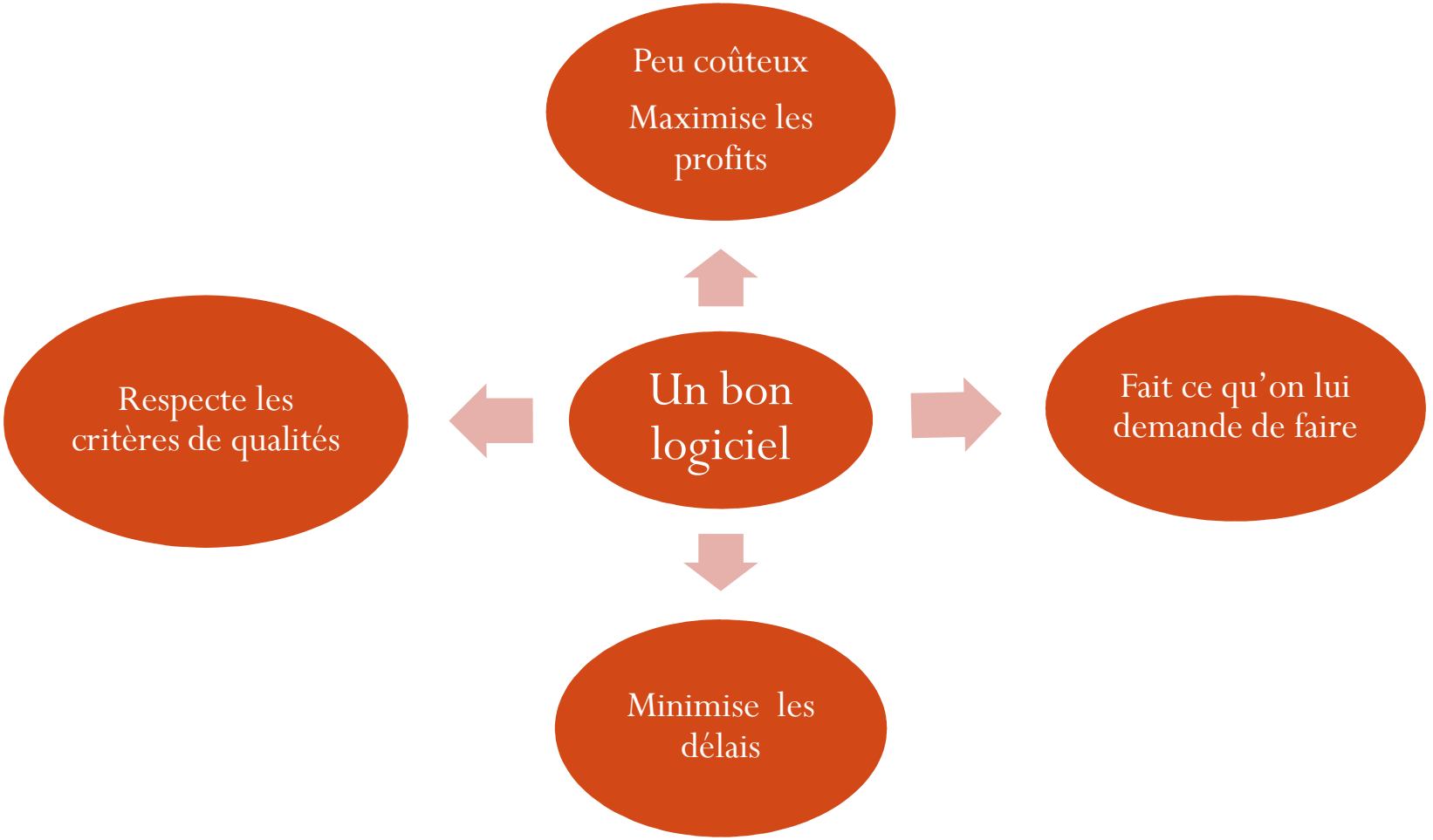
Développement d'u Logiciel

- 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
- Le client et le fournisseur peuvent être la même entité

Un bon logiciel d'un point de vue client



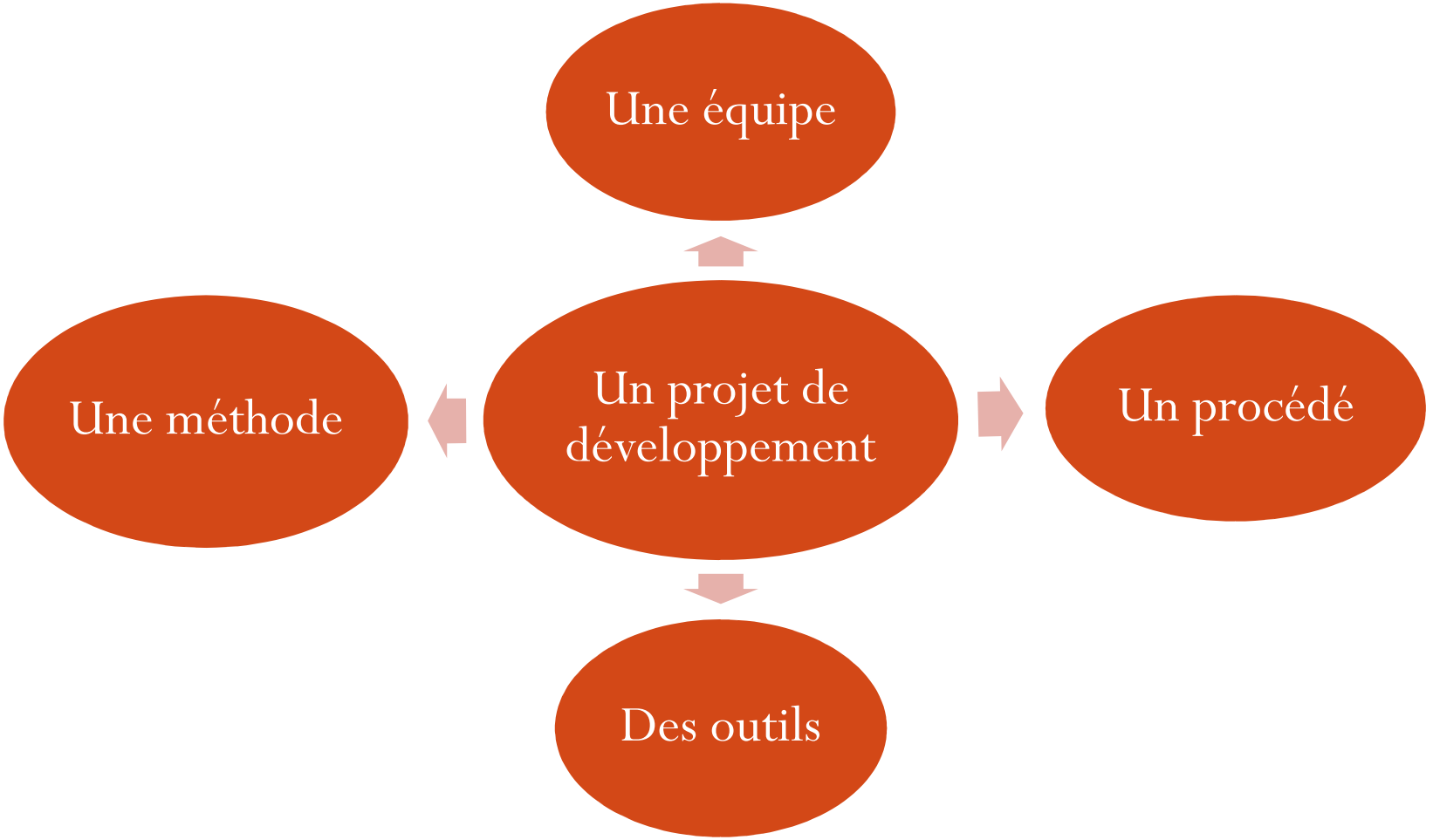
Un bon logiciel d'un point de vue Fournisseur



Faits sur le développement

- Le développement de logiciels n'est pas une opération facile
- Le développement est un ensemble d'activités
- La programmation (le codage) n'est pas le développement mais une des activités du développement
- Il n'y a pas une seule façon de développer un logiciel donné mais plusieurs
- Il y a une différence entre développer et « développer bien »
- Les projets de développement sont souvent longs et coûteux (50 % des coûts dans la maintenance).
- Les projets de développement font souvent intervenir plusieurs personnes de compétences différentes.

Que faut il pour le développement



Equipe de développement

- L'équipe n'est pas uniquement composée de programmeurs mais d'autres acteurs : chefs de projets, testeurs,…
- Il existe une panoplie d'outils relatifs au développement : compilateurs, environnements de tests, éditeurs,…
- La communication est essentielle dans un projet de développement

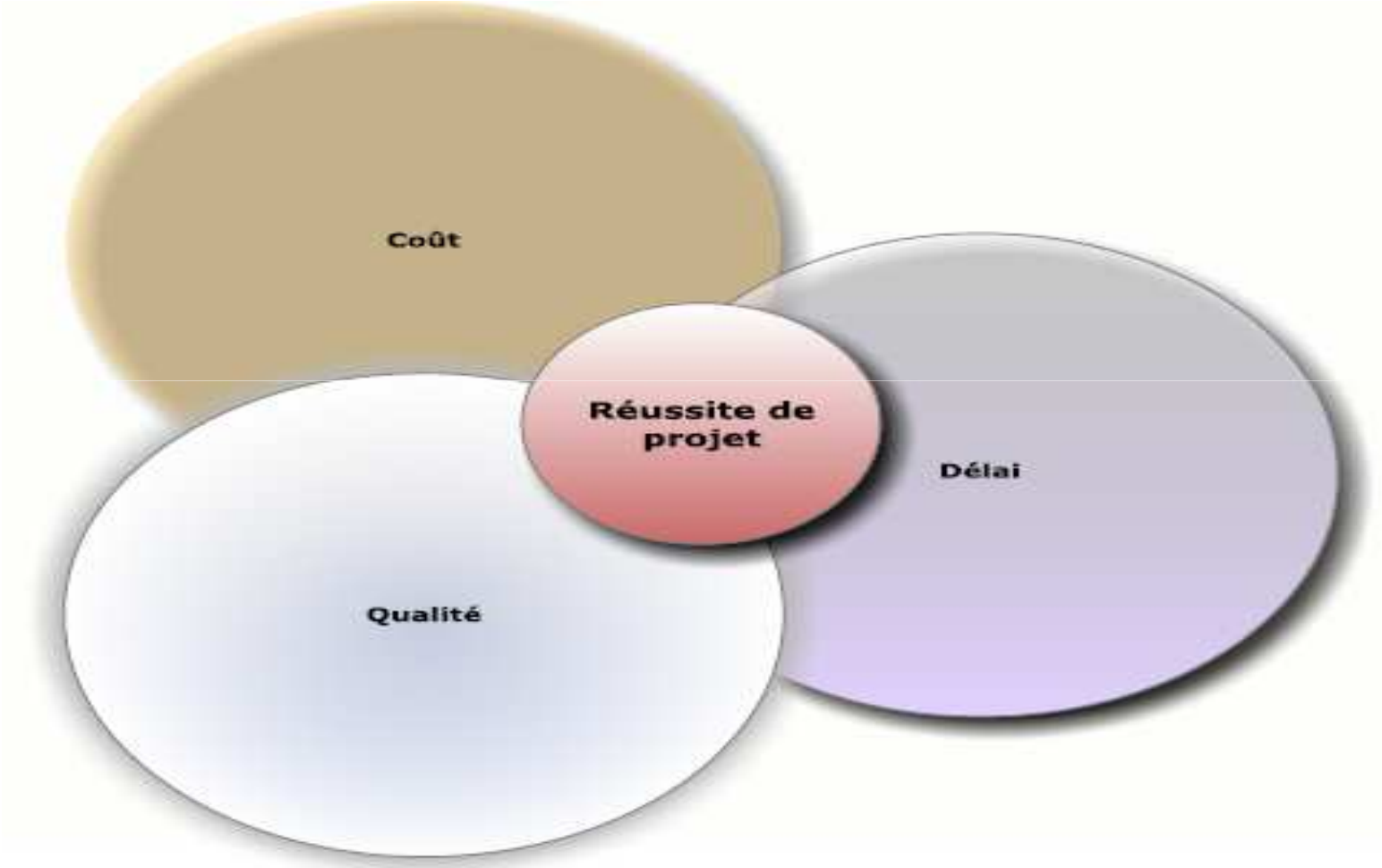
Crise Logiciel

- Étude sur 8 380 projets (Standish Group, 1995) :
  - Succès : 16 %;
  - Problématique : 53 % (budget ou délais non respectés, défaut de fonctionnalités) ;
  - Échec : 31 % (abandonné).

Le taux de succès décroît avec la taille des projets et la taille des entreprises.

- Génie Logiciel (Software Engineering) :
  - Comment faire des logiciels de qualité ?
  - Qu'attend-on d'un logiciel ? Quels sont les critères de qualité ?

Critères de réussite



Difficultés de développement

- Plusieurs difficultés caractérisent le développement de logiciels :
  - Difficile de gérer le projet et les personnes
  - Les clients arrivent difficilement à décrire leurs besoins de façon assez claire pour les fournisseurs
  - Les besoins sont en constantes évolutions ainsi que l'environnement
  - Le logiciel est non palpable (intangible)
  - Différence de langage entre les personnes techniques et non techniques
  - Difficulté de découvrir les erreurs avant la livraison du produit.
  - Le piratage de logiciels cause un énorme préjudice pour les fournisseurs

Historique Génie logiciel

- Fin des années 50, apparition du terme « Software Engineering »
- De 1965 à 1985, la crise du logiciel
- Conférence de l'OTAN a Garmish, Allemagne (1968)
  - L'informatique ne répond pas aux attentes qu'elle suscite
  - L'informatique coûte très cher et désorganise les entreprises ou organisations

- A partir de 1985, conscience de la difficulté du domaine. Accord sur le fait qu'aucune méthodologie ni aucun outils n'est « universel» pour les problèmes de développement
- Les années 90, émergence d'internet et les outils RAD
- Année 2000 – Apparitions de méthodologies légères appelées « méthodes agiles »

Génie logiciel

Le terme génie logiciel (en anglais software engineering) désigne l'ensemble des méthodes, des techniques et outils concourant à la production d'un logiciel, au-delà de la seule activité de programmation.

- L'objectif du génie logiciel est de permettre le développement de logiciels :
  - Satisfaisant le client et le fournisseur
  - De qualité supérieure
  - Dans des délais raisonnables
  - Avec des coûts acceptables

Domaines 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

Critères qualité logiciel

Critères de qualités logiciel

- Utilité
- Utilisabilité
- Fiabilité
- Interopérabilité
  - Interactions avec d'autres logiciels ;
- Performance
- Portabilité
- Réutilisabilité
- Facilité de maintenance
  - Un logiciel ne s'use pas pourtant, la maintenance absorbe une très grosse partie des efforts de développement.

Utilité

- Adéquation entre
  - Le besoin effectif de l'utilisateur
  - Les fonctions offertes par le logiciel
- Solutions :
  - Emphase sur l'analyse des besoins
  - Améliorer la communication (langage commun, démarche participative)
  - Travailler avec rigueur

Utilisabilité

- Effectivité, efficacité et satisfaction avec laquelle des utilisateurs spécifiés accomplissent des objectifs spécifiés dans un environnement particulier
- Facilite d'apprentissage : comprendre ce que l'on peut faire avec le logiciel, et savoir comment le faire Facilite d'utilisation : importance de l'effort nécessaire pour utiliser le logiciel.
- Solutions :
  - Analyse du mode opératoire des utilisateurs
  - Adapter l'ergonomie des logiciels aux utilisateurs

Fiabilité

- Correction, justesse, conformité : le logiciel est conforme a ses specifications, les résultats sont ceux attendus
- Robustesse, sureté : le logiciel fonctionne raisonnablement en toutes circonstances, rien de catastrophique ne peut survenir, même en dehors des conditions d'utilisation prévues

Interopérabilité, coulabilité

- Un logiciel doit pouvoir interagir en synergie avec d'autres
- logiciels
- Solutions :
  - Bases de données (découplage données/traitements)
  - < Externaliser > certaines fonctions en utilisant des <Middleware > avec une API (Application Program Interface) bien définie
  - Standardisation des formats des fichiers (XML…) et des protocoles de communication (CORBA…)
  - Les ERP (Entreprise Resources Planning)

Performance

- Les logiciels doivent satisfaire aux contraintes de temps d'éxecution
- Solutions :
  - Logiciels plus simples
  - Veiller a la complexité des algorithmes
  - Machines plus performantes

Portabilité

- Un même logiciel doit pouvoir fonctionner sur plusieurs machines
- Solutions :
  - Rendre le logiciel indépendant de son environnement d'execution (voir interopérabilité)
  - Machines virtuelles

Réutilisabilité

- On peut espérer des gains considérables car dans la plupart des logiciels :
  - 80 % du code est du < tout venant > qu'on retrouve a peu près partout
  - 20 % du code est spécifique
- Solutions :
  - Abstraction, généricité (ex : MCD générique)
  - Construire un logiciel a partir de composants prêts a l'emploi
  - < Design Patterns >

Maintenance

- Un logiciel ne s'use pas
- Pourtant, la maintenance absorbe une très grosse partie des efforts de développement

	Répartition effort dév.	Origine des erreurs	Coût de la maintenance
Définition des besoins	6%	56%	82%
Conception	5%	27%	13%
Codage	7%	7%	1%
Intégration Tests	15%	10%	4%
Maintenance	67%		

Maintenance corrective

- Corriger les erreurs : défauts d'utilité, d'utilisabilité, de fiabilité…
  - Identifier la défaillance, le fonctionnement
  - Localiser la partie du code responsable
  - Corriger et estimer l'impact d'une modification
- Attention
  - La plupart des corrections introduisent de nouvelles erreurs
  - Les coûts de correction augmentent exponentiellement avec le délai de détection
- La maintenance corrective donne lieu a de nouvelles livraisons (release)

Maintenance adaptative

- Ajuster le logiciel pour qu'il continue a remplir son rôle compte tenu du l'évolution des
  - Environnements d'exécution
  - Fonctions a satisfaire
  - Conditions d'utilisation
- Ex : changement de SGBD, de machine, de taux de TVA, an 2000…

Maintenance évolutive

- Accroître/ameliorer les possibilités du logiciel
- Ex : les services offerts, l'interface utilisateur, les performances…
- Donne lieu a de nouvelles versions

Facilités de Maintenance

- Objectifs
  - Réduire la quantité de maintenance corrective (zéro défaut)
  - Rendre moins coûteuses les autres maintenances
- Enjeux
  - Les coûts de maintenance se jouent très tôt dans le processus d'élaboration du logiciel
  - Au fur et a mesure de la dégradation de la structure, la maintenance devient de plus en plus difficile

- Solutions :
  - Réutilisabilité, modularité
  - Vérifier, tester
  - Structures de données complexes et algorithmes simples
  - Anticiper les changements a venir
  - Progiciels

Principe du Génie logiciel

- Généralisation : regroupement d'un ensemble de fonctionnalités semblables en une fonctionnalité paramétrable (généricité, héritage)
- Structuration : façon de décomposer un logiciel (utilisation d'une méthode bottom-up ou top-down)
- Abstraction : mécanisme qui permet de présenter un contexte en exprimant les éléments pertinents et en omettant ceux qui ne le sont pas

- Modularité : décomposition d'un logiciel en composants discrets
- Documentation : gestion des documents incluant leur identification, acquisition, production, stockage et distribution
- Vérification : détermination du respect des spécifications établies sur la base des besoins identifies dans la phase précédente du cycle de vie