Le génie pour l'industrie
Département de génie logiciel et des TI

L'assurance qualité logicielle 1
Chapitre 1 - Les connaissances fondamentales de l'AQL

Alain April et Claude Y Laporte

9 octobre 2023

1.1 - Introduction

- Les logiciels sont développés par des amateurs, des étudiants, des professionnels du logiciel et des professionnels d'autres disciplines (p.ex. le génie mécanique) dans une vaste gamme de domaines allant de la finance à l'aérospatiale.

- Il faut développer des logiciels de qualité qui correspondent aux besoins d'affaires de chaque domaine.

- L'assurance qualité logicielle est encore le parent pauvre du génie logiciel.

1.2 - Comment définir la qualité du logiciel ?

- Définition intuitive du mot ‘logiciel’
  - Un ensemble d’instructions d’un langage de programmation qui forment un programme.

- Est-ce suffisant de s’assurer de la qualité du code source pour que l’utilisateur puisse opérer un système de qualité?

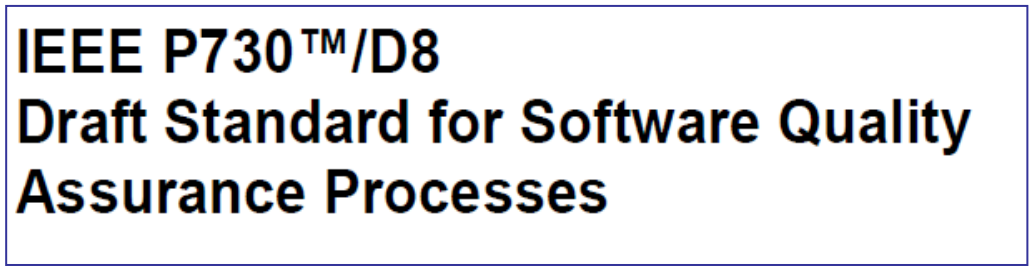
Logiciel - 1

- Un ensemble de programmes, de procédures et de documentation associée et les données qui concernent le fonctionnement d'un système informatique (ISO 24765*):
  - Programmes : Qui se traduit en code source qui a fait l'objet de spécification, de conception, d'inspection et d'essais unitaires, système et d'acceptation avec la clientèle;
  - Données : Qui ont été inventoriées, modélisées, normalisées et créées pour effectuer des scénarios d'essais lors de sa création ou d'une modification;
  - Processus : Qui sont les processus d'affaires des utilisateurs et les processus qui ont été décrits (avant l'automatisation et après), étudiés et optimisés;
  - Règles : Qui sont des règles d'affaires qui ont dû être décrites, validées, implantées et testées;
  - Documentation : Toute sorte de documentation est utile pour les utilisateurs, développeurs et mainteneurs de logiciel. La documentation permet aux différents intervenants d'une équipe de mieux communiquer, réviser et tester le logiciel. La documentation est définie et réalisée à toutes les étapes clés du cycle de vie d'un logiciel;

www.computer.org/sevocab

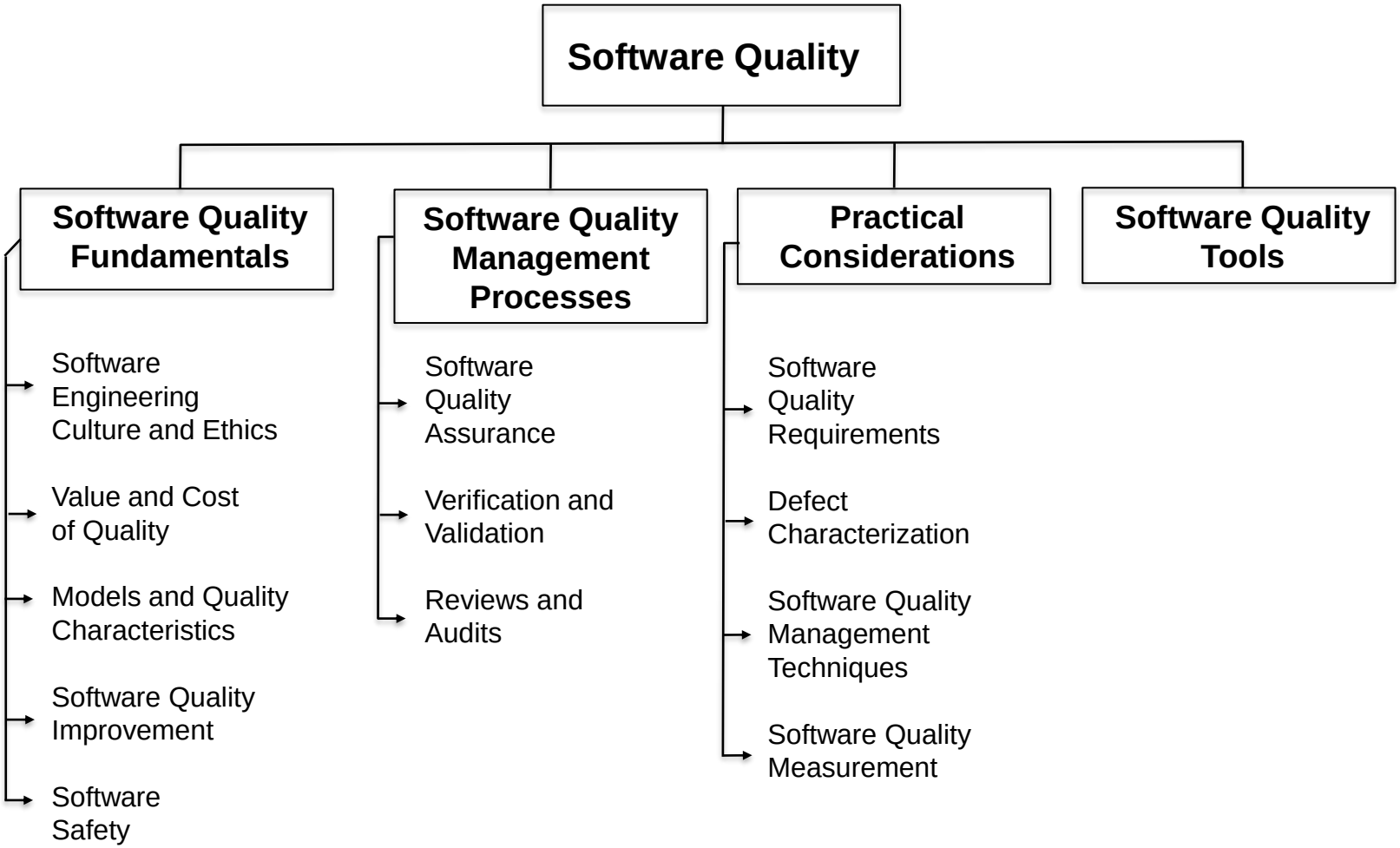
Logiciel - 2

- Le terme «logiciel» comprend le *firmware*, la documentation, les données et les instructions de contrôle d'exécution (*execution control statements (e.g., command files, job control language, etc.*) (IEEE 730-2002)



- *Software includes the software portion of firmware. The software cannot be readily modified under program control* (ISO 12207)
- **Micrologiciel *(Firmware)***
  - Combinaison d'un dispositif matériel et d'instructions d'un ordinateur ou de données informatiques qui résident, en mode lecture seulement, sur un périphérique matériel (ISO 24765).
  - Ajout à la définition du terme logiciel parce que le *firmware* pourrait échapper aux exigences auxquels sont soumis les autres éléments de la définition de l'ISO.

Software Quality in the SWEBOK*

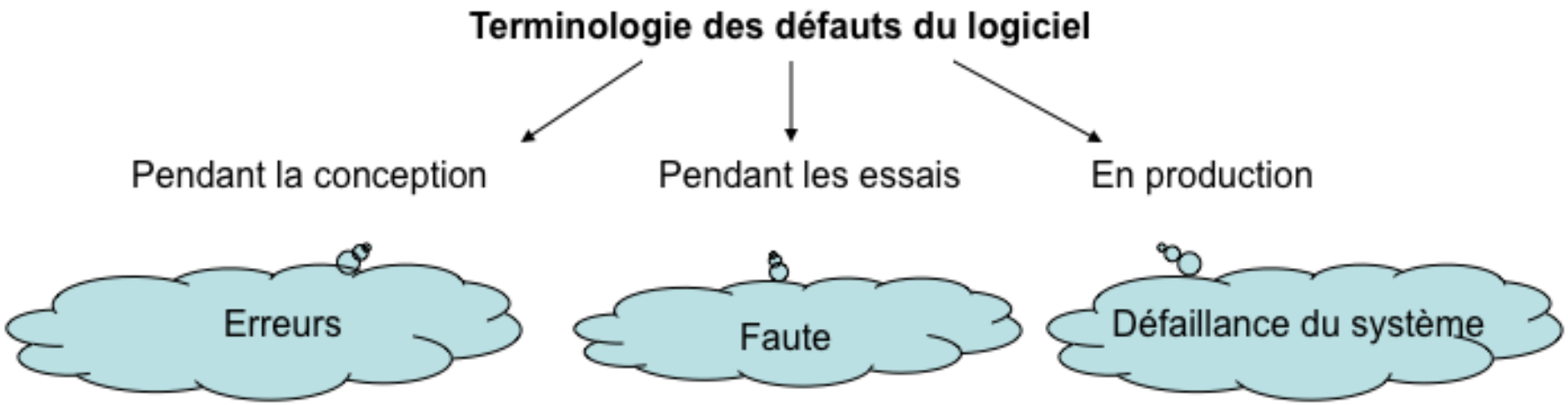


* **SWEBOK** : **S**oft**w**are **E**ngineering **B**ody **o**f **K**nowledge

1.3 - Les erreurs, fautes et défaillances du logiciel

- Termes utilisés couramment pour décrire un problème en TI:
  - le système a planté en production;
  - le concepteur a fait une erreur;
  - suite à une inspection (c.à.d. une revue), on a trouvé un défaut dans le plan de test;
  - j'ai trouvé un « bug »;
  - le système est tombé en panne;
  - le client se plaint d'un problème avec un calcul dans le rapport de paiement;
  - on rapporte la défaillance du sous-système de surveillance.

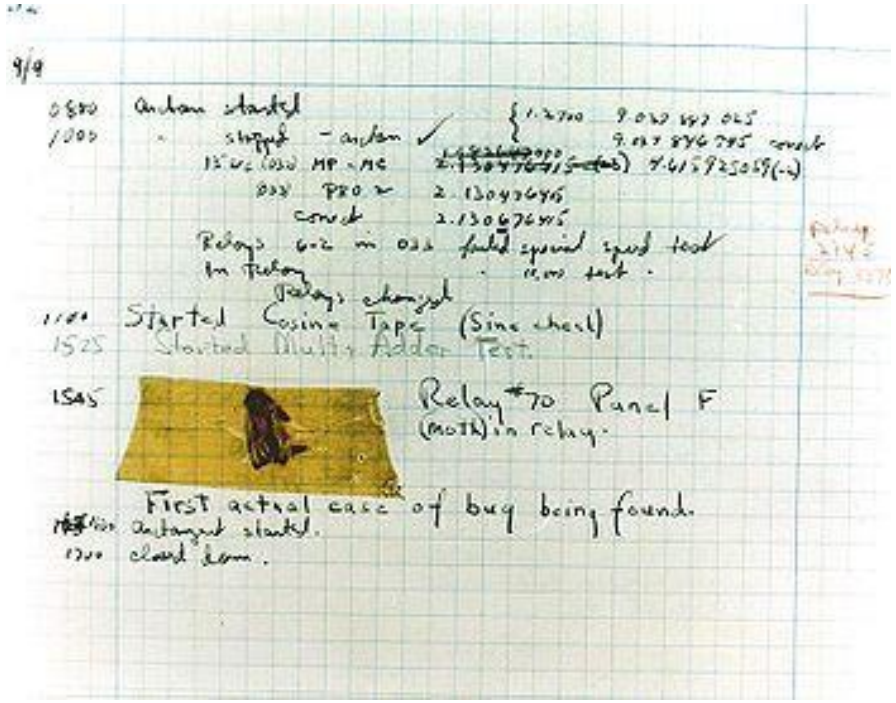
1.3 - Les grandes catégories de causes d'erreurs du logiciel



- **Erreur** *(error):* Une action humaine qui produit un résultat incorrect, comme un logiciel contenant un défaut (*fault*) (ISO 24765).
- **Défaut/Faute** (*defect*): Une faute qui, si elle n'est pas corrigée, pourra causer une défaillance (*failure*) ou produire des résultats incorrects (ISO 24765).
- **Défaillance** (*failure*): La manifestation d'une erreur dans le logiciel (ISO 24765)
  - Cessation de l'aptitude d'un produit à accomplir une fonction requise ou de son incapacité à s'en acquitter à l'intérieur des limites spécifiées précédemment (ISO 25000).

Le mot ‘Bug’

- Depuis l'époque de Thomas Edison, des ingénieurs ont utilisé le mot "bogue " de se référer à des failles dans les systèmes qu'ils ont développés. Ce mot couvre une multitude de problèmes possibles.
- Le premier cas documenté de « Bug informatique » concernait un papillon de nuit (mite) coincé dans le relais de l’ordinateur Mark II de Harvard en 1947. L’opératrice de l’ordinateur Grace Hopper colla la mite dans le journal de laboratoire sous le titre "First actual case of bug being found"



Software Errors, Defects and Failures

Human Error

can lead to

Defect

can lead to

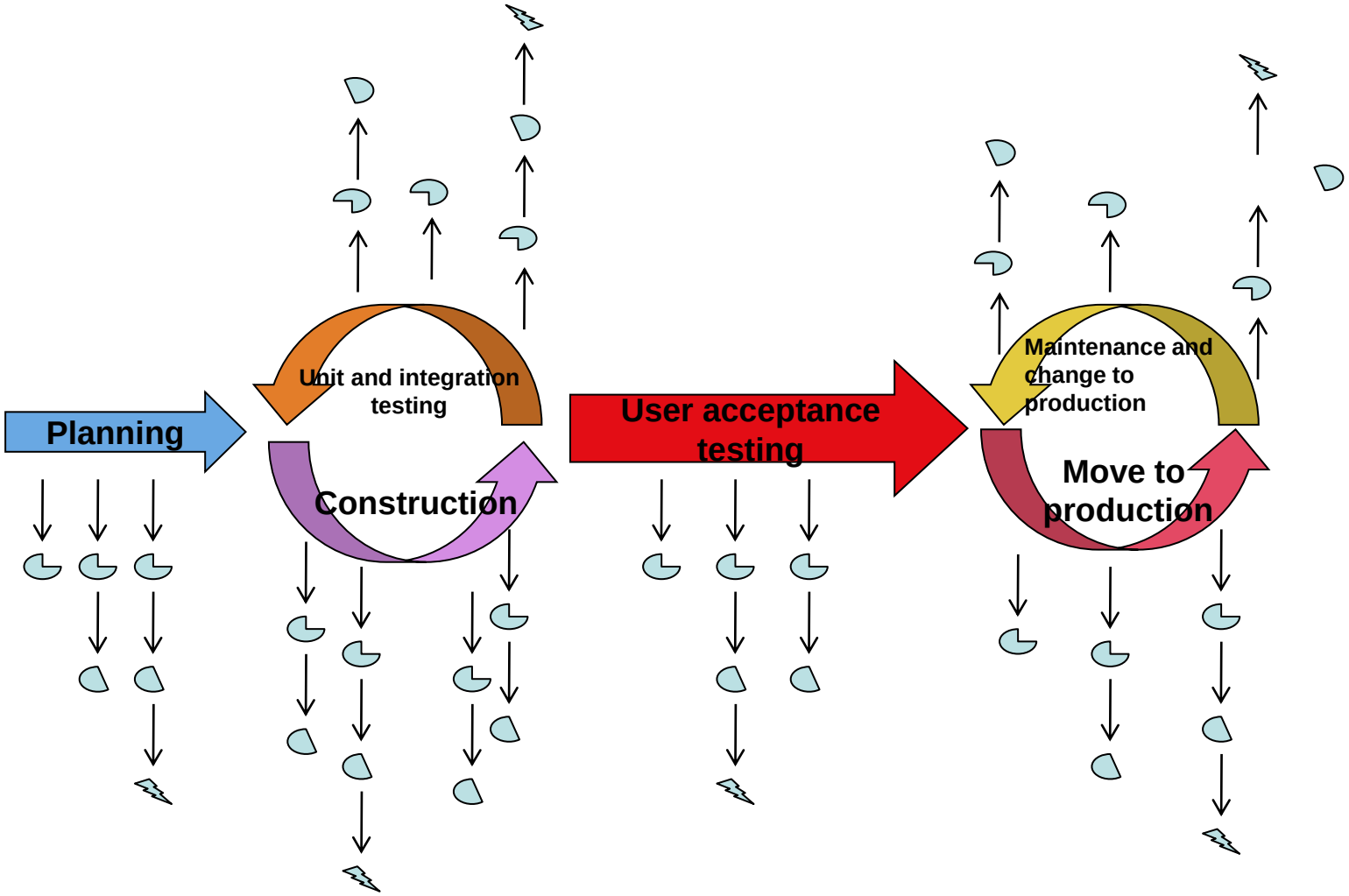
Failure

How do you prevent errors ?

How do you detect defects ?

A

Errors, defects and failures in the software life cycle



Quelques constats

- Les erreurs peuvent survenir dans toutes les étapes du développement et du cycle de vie du logiciel;*

- Les défauts doivent être identifiés et corrigés avant de devenir une défaillance;

- La cause des défaillances, des défauts et des erreurs doit être identifiée afin d'effectuer une correction.

Cycle de développement et Cycle de vie

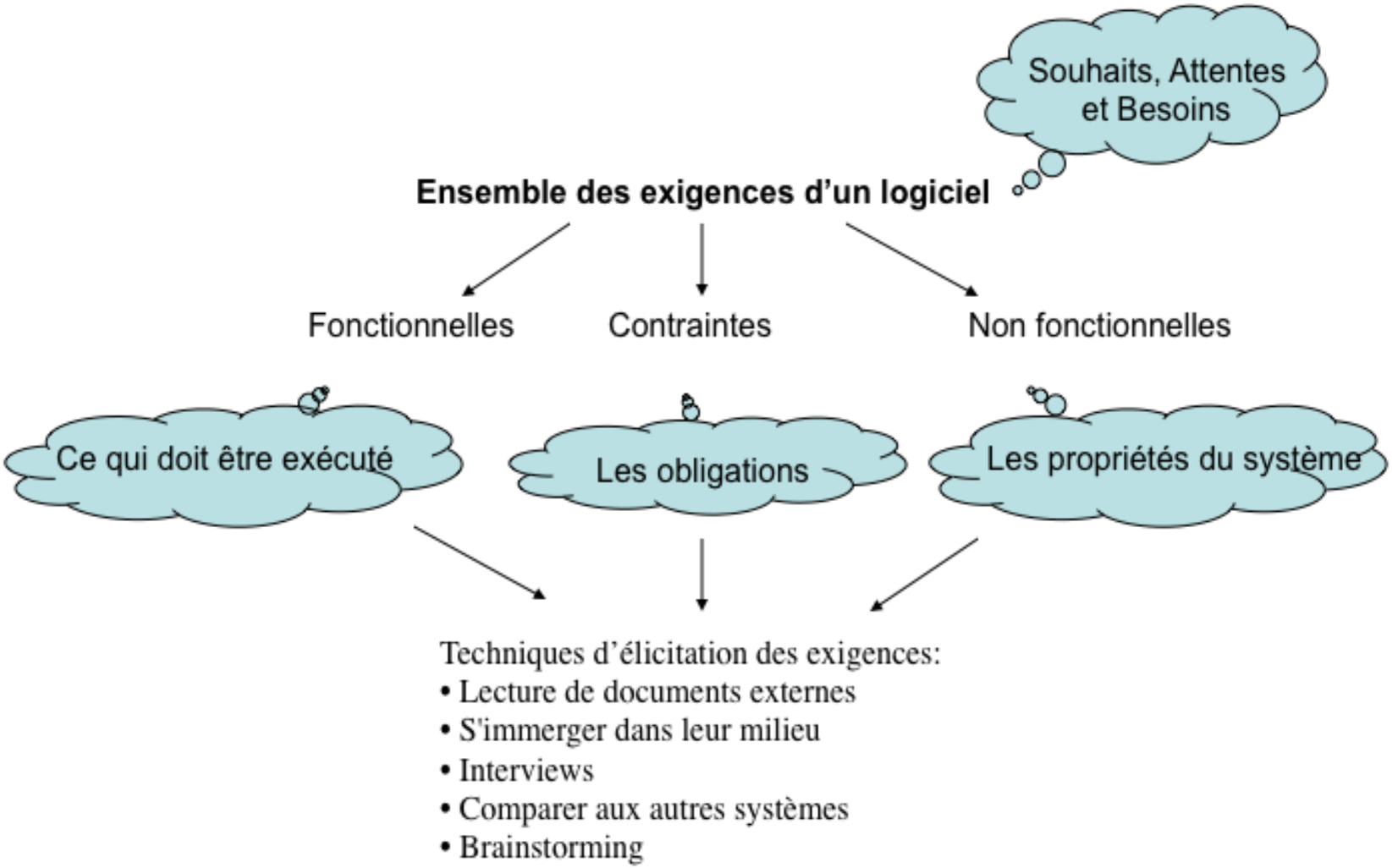
- **Cycle de développement *(Development life cycle)***
  - Processus de cycle de vie du logiciel qui comporte des activités d'analyse des besoins, de conception, de codage, d'intégration, de tests, d'installation et de soutien pour l'acceptation des produits logiciels (ISO 90003).

- **Cycle de vie *(life cycle)***
  - Évolution d'un système, produit, service, projet ou autre entité d'origine humaine de la conception à la retraite (ISO 12207).
    - *(Evolution of a system, product, service, project or other human-made entity from conception through retirement)*

Exemple de classification des causes d'erreurs

1. difficulté de définition des exigences;
2. difficulté de maintenir une communication efficace entre le client et le développeur;
3. déviations aux spécifications;
4. erreurs d'architecture et de conception;
5. erreurs de codage;
6. non-conformité avec les processus/procédures en place;
   - c.à.d. que des développeurs ne suivent pas les processus en place
7. revues et tests inadéquats;
8. erreurs de documentation.

Le contexte de l'élicitation* des exigences du logiciel



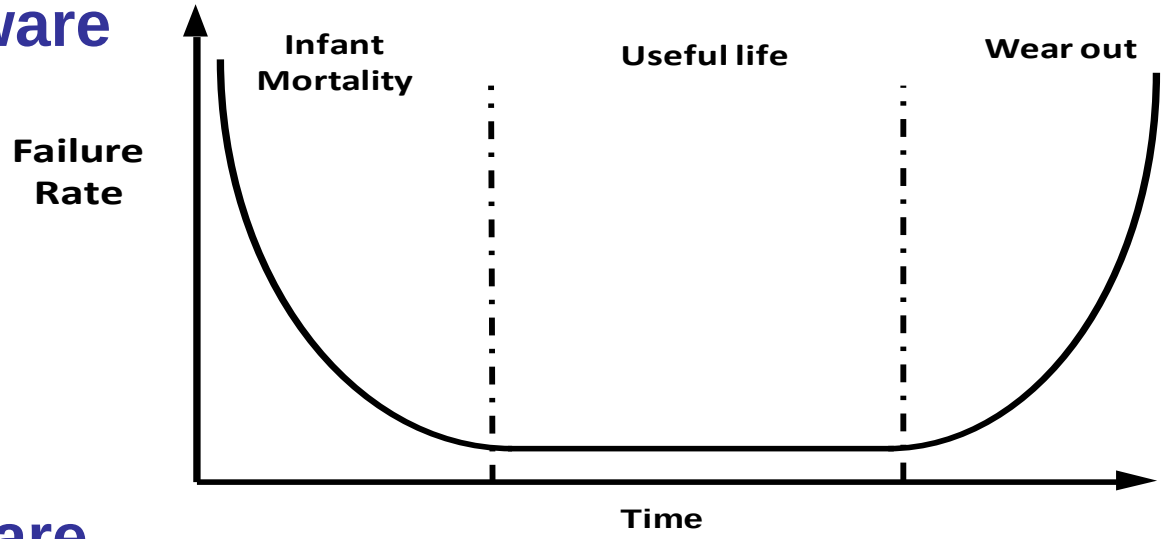
*** Élicitation ou élucidation = Obtention et explicitation**

La qualité d'une exigence

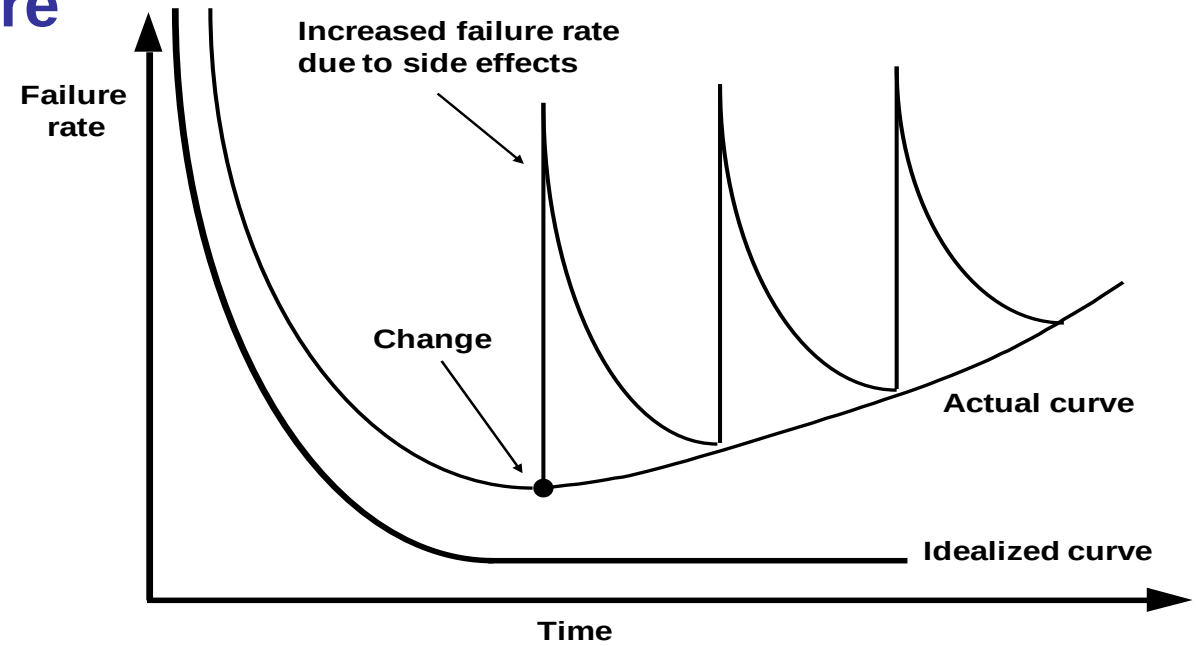
- Une exigence est dite de bonne qualité si elle rencontre les caractéristiques suivantes :
  - correcte;
  - complète;
  - claire pour chaque lecteur (p. ex. client, architecte, mainteneur, testeur);
  - concise (simple, précise);
  - consistante;
  - réalisable (réaliste, possible);
  - nécessaire (répond à un besoin du client, c.à.d. traçable);
  - indépendante de la conception;
  - indépendante de la technique d'implantation;
  - vérifiable et testable;
  - unique.

Reliability curves

For Hardware



For Software



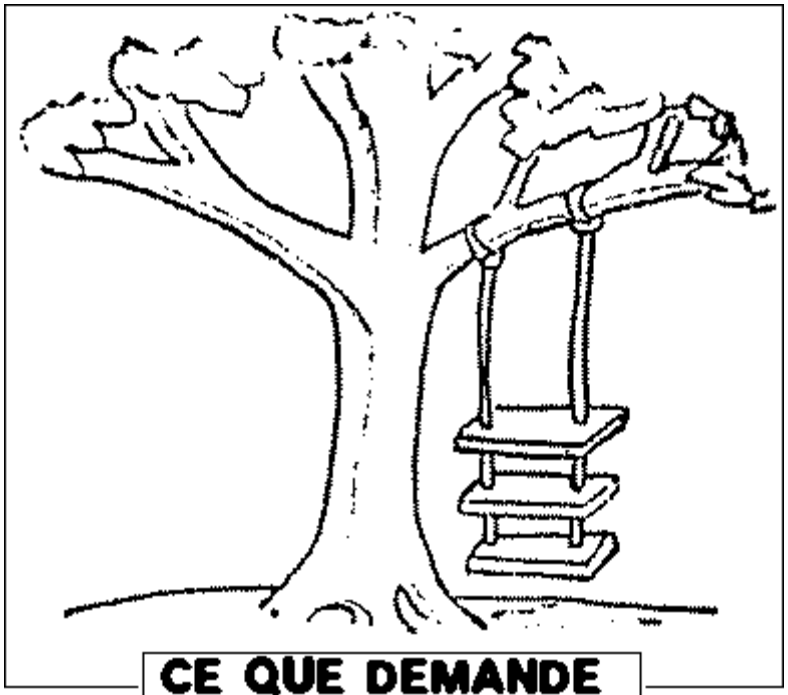
1.4 – La qualité du logiciel

- Capacité d'un produit logiciel de satisfaire les besoins exprimés et implicites (c.à.d. les attentes) quand il est utilisé dans des conditions spécifiées (ISO 24765)
- Il faut satisfaire les besoins et attentes des clients et utilisateurs
  - Les attentes ne sont pas nécessairement décrites
  - Il y a souvent plusieurs ‘utilisateurs‘ ayant chacun leurs attentes
    - L’utilisateur du logiciel
      - Dans certains projets l’utilisateur final est représenté par des ‘experts’
    - Les personnes qui vont l’installer
    - Les personnes qui vont le maintenir
    - Les personnes qui vont former les utilisateurs

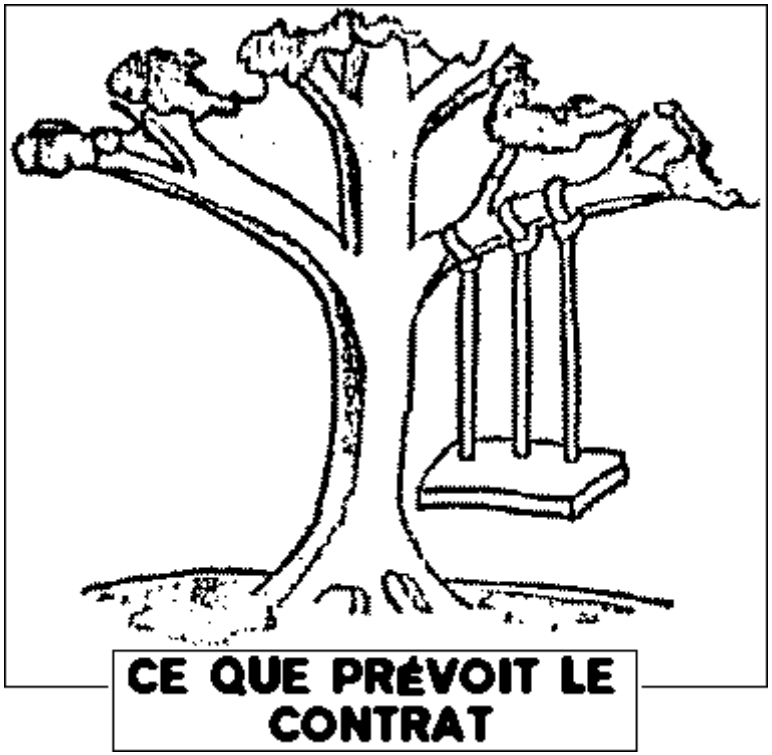
Les facteurs qui peuvent affecter la satisfaction des besoins réels du client.

Type de besoin	Origine de l'expression	Causes notables de différentiel
Réels	"Cerveau du Commanditaire"	– Méconnaissance des besoins réels – Instabilité des besoins – Différences de point de vue entre le commanditaire (le payeur) et les utilisateurs finaux
Exprimés	Cahier des charges	– Incomplétude de la Spécification – Manque de formalisme – Insuffisance ou difficulté de communication avec le commanditaire – Insuffisance du contrôle qualité
Spécifiés	Document de Spécification du Logiciel	– Utilisation inadaptée des méthodes, techniques et outils de gestion et de production – Insuffisance des tests – Insuffisance des techniques de contrôle qualité
Réalisé	Documents et Code Produits	

Exemple

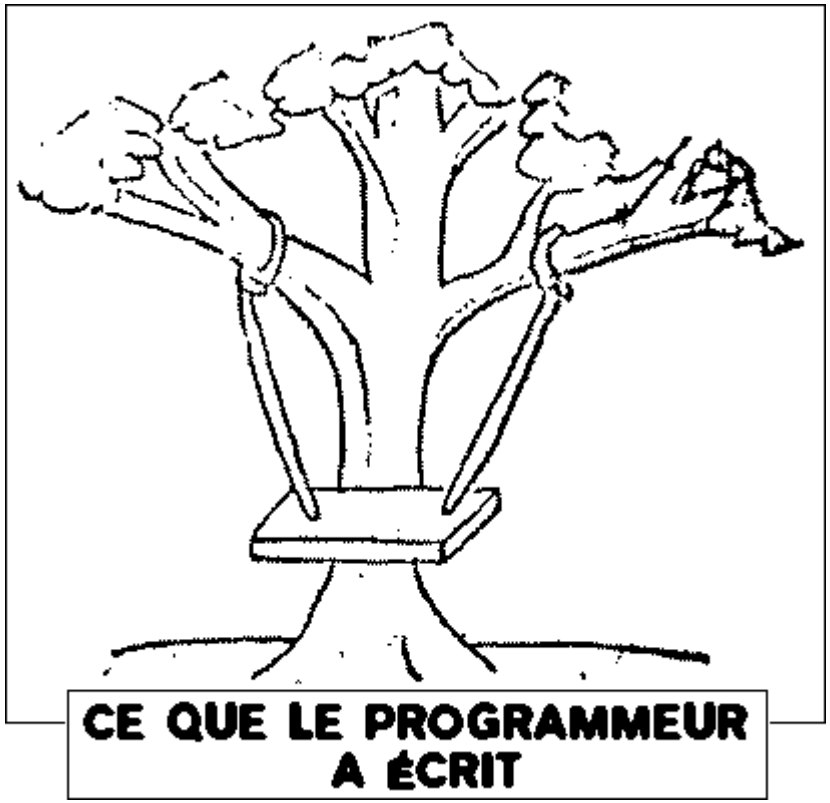


Exemple (suite)

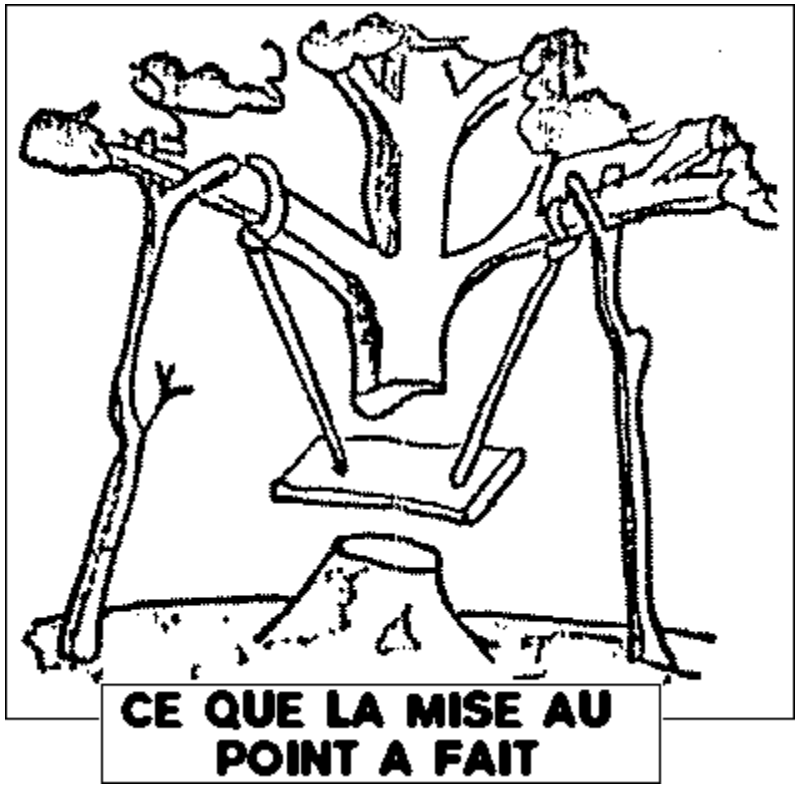


Exemple (suite)

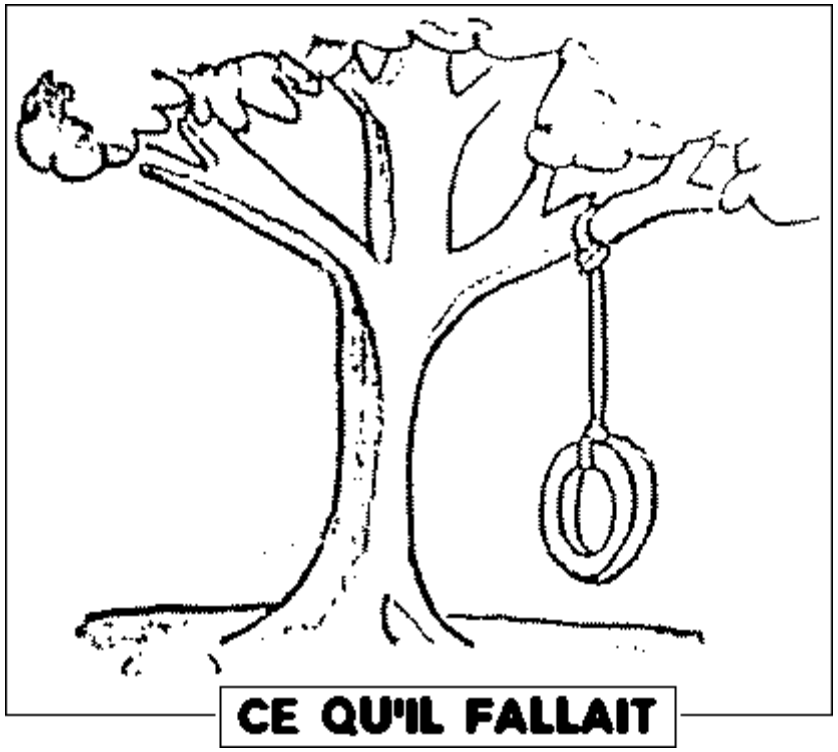
Exemple (suite)



Exemple (suite)



Exemple (suite)



1.5 – L'assurance qualité (ISO 24765)

1) Un ensemble d'activités planifiées et systématiques de toutes les actions nécessaires pour fournir une **assurance** suffisante qu'un élément produit est conforme aux exigences techniques établies;
2) Un ensemble d'activités destinées à évaluer le processus par lequel les produits sont développés ou fabriqués;
3) Les activités planifiées et systématiques mises en œuvre dans le système qualité, et démontré au besoin pour fournir une assurance suffisante qu'une entité satisfera aux exigences de qualité.

- **Notes**:
  - La définition du mot 'assurer'* signifie 'donner la certitude'.
  - There are both internal and external purposes for quality assurance:
    - Within an organization, quality assurance provides confidence to management;
    - In contractual situations, quality assurance provides confidence to the customer or others.

* Le terme 'assurance-qualité', comme le terme 'assurance-vie', est un peu trompeur car la mise en place de pratiques de génie logiciel ne peut pas 'assurer' la qualité d'un projet.

Unless requirements for quality fully reflect the needs of the user, quality assurance may not provide adequate confidence.

Contrôle de la qualité (ISO 24765)

- Un ensemble d'activités visant à évaluer la qualité des produits développés ou fabriqués.
- Le processus de vérification de son propre travail ou de celui d'un collègue.
  - Ce terme n'a pas de signification standardisée en génie logiciel en ce moment. (ISO 24765)
  - L'ébauche finale de la norme IEEE-730 ne comporte pas cette définition.

L'assurance qualité

- L'assurance qualité est mise en place pour réduire les risques de développer un logiciel de faible qualité.
- Cette perspective de l'assurance qualité (AQ) implique, pour le développement du logiciel, les éléments suivants :
  - la nécessité de planifier les aspects qualité d'un produit ou d'un service
  - le besoin de déployer des activités systématiques qui nous indiquent que, tout au long du cycle de vie du logiciel, certaines corrections sont requises ;
  - que le système qualité est un dispositif complet qui doit permettre, dans le cadre de la gestion de la qualité, la mise en oeuvre de la politique qualité et l'amélioration continue;
  - l'exécution de techniques d'assurance qualité qui ont pour objectif de démontrer le niveau de qualité atteint de manière à donner confiance aux utilisateurs et clients ;
  - la démonstration de la satisfaction des exigences qualité qui ont été définies pour le projet, la modification ou le service TI.

1.6 - Les modèles d'affaires

- Un modèle d'affaires énonce comment un organisme gagne *(et **perd***) de l'argent en précisant où (et comment) il se positionne dans son ou ses marchés.
- Le modèle d'affaires décrit les aspects principaux d'une activité, incluant:
  - But, offres, stratégies, infrastructure, organisations, pratiques de diffusion ou distribution et, processus et règles de fonctionnement.

(Adapté de Wikipédia)

Les modèles d'affaires et le choix des pratiques de génie logiciel

- Kathy Iberle ingénieure (sénior) de tests chez Hewlett-Packard
  - décrit son expérience dans deux domaines d'affaires d'une même compagnie:
    - les produits de cardiologie
    - les imprimantes.

Traduit de (Iberle, 2002, 2003)

Les modèles d'affaires
Les sujets traités

1. Introduction
   - Les appareils médicaux et les imprimantes
2. Les modèles d'affaires
3. Les composants du modèle d'affaires
4. Description détaillée d'un modèle d'affaires
   - Projet sous contrat
5. Survol des autres modèle d'affaires
6. La sélection de pratiques appropriées

Traduit de (Iberle, 2002, 2003)

Contexte
La division des produits médicaux

- Un domaine très réputé pour la qualité.
- Utilise les pratiques classiques d'ingénierie logiciel:
  - Spécifications détaillées écrites,
  - Utilisation intensive des inspections et des revues tout au long du cycle de vie,
  - Tests exhaustifs des exigences,
  - Critères de sortie créés au début d'un projet,
  - Un produit ne pouvait pas être expédié tant que les critères de sorties n'étaient pas ***tous*** satisfaits.
- Dans ce domaine il est possible de rater la date de fin du projet par des semaines voir des mois.
  - Ces délais sont acceptés afin de corriger les derniers problèmes.
- La qualité l'emportait ***toujours*** sur le calendrier.

Traduit de (Iberle, 2002, 2003)

Contexte
La division des imprimantes

- Produit les imprimantes à jet d'encre pour les petites entreprises et le consommateur.
- Les pratiques de ce secteur d'affaires sont différentes:
  - Les spécifications étaient beaucoup plus courtes,
  - Les critères de fin de projet étaient beaucoup moins formels,
  - L'atteinte de la date de livraison était **très** importante.
  - Les tests:
    - Testeurs qui testaient sans avoir de procédures de tests.
    - On n'essayait pas de tester toutes les combinaisons possibles d'entrées.
    - Pas consacrés aux tests reliés aux spécifications.
    - Beaucoup moins de documents de test.
- **Choc culturel de l'ingénieure**
  - "Ces gens ne se soucient pas de qualité!"
  - La définition de la qualité était différente d'un domaine d'affaires à l'autre.

Traduit de (Iberle, 2002, 2003)

1.6.2 - L'anxiété, la peur et la terreur

- **Le domaine médical**
  - Rater une date de livraison n'était pas la pire chose qui pouvait arriver.
  - Ce qui glaçait le sang de l'équipe:
    - Tuer un patient ou un technicien d'un choc électrique,
    - Causer un mauvais diagnostic,
    - L'impossibilité d'utiliser un appareil dans une situation urgente.
  - S'il y a une possibilité d'une défaillance, alors:
    - La date de livraison est repoussée automatiquement sans aucune discussion.
    - Les efforts longs et coûteux pour trouver et éliminer, de façon concluante, la cause d'un défaut étaient systématiquement approuvés.
  - **Craintes:**
    - Responsabilité légale ou être blâmé par l'agence de réglementation américaine du Food and Drug Administration (FDA).

Traduit de (Iberle, 2002, 2003)

L'anxiété, la peur et la terreur

- **Les produits de consommation**
  - Le potentiel de blessure est très faible même dans les pires conditions imaginables.
  - La véritable terreur était de ne pas rencontrer les échéanciers ou de dépasser les coûts.

  - **Craintes:**
    - Avoir des milliers d'utilisateurs incapables d'installer leur nouvelle imprimante, qui appelleraient les lignes de support à la clientèle le lendemain de Noël.
    - Une incompatibilité avec les logiciels les plus populaires ou le matériel.

Traduit de (Iberle, 2002, 2003)

L'anxiété, la peur et la terreur

- La définition de «qualité» est différente dans ces deux divisions d'affaires.
- Les clients valorisaient des choses différentes:
  - Secteur médical valorisaient l'exactitude et la fiabilité avant tout,
  - Secteur des imprimantes valorisaient la convivialité et la compatibilité beaucoup plus qu'une grande fiabilité.
  - Tout le monde veut la fiabilité:
    - Les personnes valorisent la fiabilité en fonction de la douleur qui leur est infligée par des problèmes.
      - Redémarrer leur ordinateur de temps en temps,
      - Angoisse d'un patient confronté à un problème de fonctionnement d'un défibrillateur cardiaque.
    - La définition de la «fiabilité» est très différente dans ces deux domaines d'affaires.
- **Conclusion**
  - Ce qui semblait être stupide ou bâclé, dans le domaine des imprimantes, était une façon d'aborder des problèmes différents qui ne se produisaient pas avec la même importance dans les produits médicaux.

Traduit de (Iberle, 2002, 2003)

1.6.3 - Les craintes affectent le choix des pratiques logicielles

- On choisit des pratiques qui permettraient de réduire les craintes.
  - La peur d'un diagnostic erroné conduit à effectuer de nombreuses révisions détaillées et à de multiples types de tests.
  - La peur de confondre les utilisateurs d'imprimantes conduit à effectuer des tests d'utilisabilité
- Les personnes qui sont dans les mêmes domaines d'affaires ont généralement des craintes similaires et utilisent des pratiques similaires.
- Certaines craintes peuvent aussi se retrouver dans d'autres domaines d'affaires:
  - p.ex. dans l'aérospatiale et les entreprises médicales.
- Il est également possible, pour la même entreprise, d'avoir des peurs et des valeurs différentes dans différents domaines d'affaires
  - p.ex. un système avionique et le logiciel de courriels.
- 'Groupes de pratique' *(community of practioners)*
  - Des spécialistes (p.ex. du logiciel) qui partagent des définitions communes de la qualité et ont tendance à recourir à des pratiques similaires.

Traduit de (Iberle, 2002, 2003)

1.6.4 – Description de modèles d'affaires

- **Le développement à contrat**
  - L'entreprise réalise des profits en vendant des services de développement de logiciels sur mesure pour des clients.
- **Le développement à l'interne**
  - L'entreprise développe des logiciels pour améliorer son efficacité organisationnelle (p. ex. la direction des ressources informationnelles de la police)
- **Les logiciels commerciaux**
  - L'entreprise réalise des profits en développant et en vendant des logiciels à d'autres organisations (par exemple le logiciel ERP de SAP).
- **Les logiciels de masse**
  - L'entreprise fait des profits en développant et en vendant des logiciels aux consommateurs (par exemple Microsoft)
- **Les logiciels embarqués de masse** (*Firmware Embedded*)
  - L'entreprise fait des profits en vendant des logiciels qui se trouvent dans du matériel et des systèmes embarqués (p.ex. caméras numériques, graveurs de DVD)

Traduit de (Iberle, 2002, 2003)

1.6.5 - Les facteurs situationnels

- Un ensemble d'attributs ou de facteurs propres à un modèle d'affaires:
  1. La criticité du logiciel,
  2. L'incertitude des besoins et exigences des utilisateurs,
  3. La gamme d'environnements informatiques,
  4. Le coût de correction des erreurs,
  5. La réglementation
     - p.ex. les normes du domaine nucléaire, du domaine médical
  6. La taille du projet,
  7. La communication,
  8. La culture de l'organisation.

Traduit de (Iberle, 2002, 2003)

Les facteurs situationnels - 1

1. **La criticité**
   - Le potentiel de blesser l'utilisateur ou les intérêts de l'acheteur varie selon le type de produit. Certains logiciels peuvent tuer en cas de panne, d'autres logiciels peuvent faire perdre de grosses sommes d'argent de beaucoup de gens, d'autres logiciels ne font que faire perdre du temps à l'utilisateur.
2. **L'incertitude des besoins et exigences (et *attentes versus besoins*) des utilisateurs**
   - Les exigences pour un logiciel qui met en œuvre un processus connu d'entreprise sont mieux connues que les exigences relatives à un produit de consommation qui est si nouveau que les utilisateurs finaux souvent ne savent pas ce qu'ils veulent.
3. **La gamme d'environnements**
   - Un logiciel écrit pour être utilisé dans une société spécifique doit être compatible uniquement avec son environnement informatique, alors que les logiciels vendus dans le marché de masse doivent fonctionner dans un large éventail d'environnements.

Traduit de (Iberle, 2002, 2003)

Les facteurs situationnels - 2

4. Le coût de correction des erreurs

- La distribution des correctifs de certains logiciels (par ex. *logiciel embarqué*) est beaucoup plus coûteuse que de colmater un seul site web.

5. La réglementation

- Les organismes de réglementation et les clauses contractuelles peuvent exiger l'utilisation de pratiques logicielles qui autrement ne seraient pas être adoptées.
- Certaines situations exigent des audits de processus pour vérifier qu'un processus a été suivi pour fabriquer le logiciel.

6. La taille du projet

- Les projets qui s'échelonnent sur plusieurs années avec des centaines de développeurs sont courants dans certaines organisations alors que dans d'autres entreprises les projets plus courts développés par une seule (petite) équipe sont plus typiques.

Traduit de (Iberle, 2002, 2003)

Les facteurs situationnels - 3

7. La communication

- Il existe un certain nombre de facteurs, outre la dimension du projet qui peuvent augmenter la quantité de communications de personne à personne ou de rendre les communications plus difficiles.
  - **La communication Concurrente Développeur - Développeur:**
    - La communication avec d'autres personnes sur le même projet est affectée par la façon dont le travail est distribué.
      - » Les ingénieurs séniors ont conçu le logiciel et le personnel subalterne effectuera le codage et les tests unitaires
      - » La même personne qui effectue la conception, le codage et les tests unitaires d'un composant donné
  - **La communication Développeur - Mainteneur:**
    - La maintenance et les améliorations nécessitent une communication avec les développeurs. Ceci est facilité lorsque les développeurs sont dans les parages, la communication se fait donc entre eux.
  - **La communication entre Gestionnaires - Développeurs:**
    - Les états d'avancement d'un projet doivent être envoyés 'vers le haut'.
    - La quantité d'information et la forme de la communication que les gestionnaires estiment qu'ils ont besoin varient considérablement.

Traduit de (Iberle, 2002, 2003)

Les facteurs situationnels - 4

8. La culture de l'organisation

- L'organisation a une culture qui définit comment les gens fonctionnent.
- Quatre types de cultures organisationnelles:
  - **Culture de contrôle**
    - Les cultures de contrôle, comme IBM et GE, sont motivées par le besoin de puissance et de sécurité.
  - **Culture de compétence**
    - Une culture de la compétence est déterminée par le besoin de s'accomplir. Microsoft est un exemple.
  - **Culture de collaboration**
    - Une culture de collaboration telle qu'incarnée par Hewlett-Packard, est motivée par un besoin d'appartenance.
  - **Culture d'épanouissement**
    - Une culture d'épanouissement motive par la réalisation de soi,
    - Elle peut être illustrée par des organismes en démarrage.

Traduit de (Iberle, 2002, 2003)

Un modèle d'affaires: Le développement à contrat

- Le client précise exactement ce qu'il veut et promet une somme d'argent déterminée au fournisseur.
- Les profits du fournisseur dépendent de sa capacité:
  - à rester dans les limites budgétaires
  - à offrir un produit qui fonctionne comme prévu à l'intérieur du calendrier déterminé dans le contrat.
- Les applications de grandes tailles (p.ex. logiciels militaires) sont souvent écrits sur contrat.
- Le logiciel produit dans cette culture d'affaires est souvent un logiciel critique
  - p.ex. *Mission critical, Business critical*
- Le coût de la distribution de correctifs après livraison est gérable
  - Les corrections sont distribuées à un environnement connu et accessible et dans un nombre raisonnable d'emplacements.

Traduit de (Iberle, 2002, 2003)

Le développement à contrat
Les facteurs situationnels - 1

1. **Criticité**
   - Les défaillances logicielles dans des systèmes financiers peuvent endommager sérieusement les intérêts d'affaires du client.
   - Les défaillances logicielles dans les systèmes militaires peuvent mettre la vie en danger.
2. **Incertitude des exigences et des besoins des utilisateurs**
   - Puisque les acheteurs et les utilisateurs sont un groupe identifiable, ils peuvent être contactés pour savoir ce qu'ils veulent.
   - En général, ils ont une idée assez détaillée de ce qu'ils veulent.
3. **La gamme d'environnements**
   - L'organisation qui achète a généralement identifié un petit ensemble d'environnements cibles
4. **Le coût de correction des erreurs**
   - Façons peu dispendieuses pour distribuer des correctifs
     - Une grande partie du logiciel sera sur les serveurs dans un bâtiment identifiable et la localisation du logiciel du client est généralement connue.

Traduit de (Iberle, 2002, 2003)

Le développement à contrat
Les facteurs situationnels - 2

5. La réglementation

- Les logiciels pour la défense (p.ex. pour un avion de combat) doivent être rédigés en conformité avec une liste énorme de réglementations (processus de développement)
- Audits de processus
  - Pour prouver ce qui a été fait.

6. Taille du projet

- Plusieurs dizaines de personnes pendant plus de deux ans pour un projet de taille moyenne, tandis que des centaines de personnes sur plusieurs années sont requis pour les gros projets.

7. Communication

- La pratique qui consiste à répartir l'architecture et de codage entre les professionnels séniors et juniors se manifeste parfois dans cette culture.

8. Culture organisationnelle

- Les organisations qui écrivent des logiciels sous contrat ont souvent une culture de contrôle.

Traduit de (Iberle, 2002, 2003)

Le développement à contrat
Les craintes

- Les craintes des développeurs de ces systèmes sont habituellement:
  - Résultats incorrects,
  - Dépassement du budget,
  - Pénalités pour livraison tardive,
  - Ne pas livrer ce que le client a demandé,
  - Des litiges.

Logiciel critique

Un logiciel dont l'échec pourrait avoir un impact sur la sécurité, ou pourrait entraîner des pertes financières, environnementales ou sociales.

Traduit et adapté du glossaire IEEE 610.12 (IEEE 1990)

Traduit de (Iberle, 2002, 2003)

Le développement à contrat
Les hypothèses

- Les facteurs situationnels permettent de déduire un ensemble d'hypothèses pour ce modèle d'affaires:
  - La livraison dans les délais et le budget est très importante,
  - Un logiciel fiable et correct est très important,
  - Les exigences peuvent et doivent être connus dans le détail au début du projet,
  - Les projets seront grands et les canaux de communication sont nombreux,
  - Nous devons être capables de prouver que nous faisions ce que nous avons promis,
  - Nous avons besoin de plans et de produire des rapports d'étape régulièrement qui sont acheminés à la direction du projet et au client (ou à son représentant (p.ex. département des achats)).

Traduit de (Iberle, 2002, 2003)

Les pratiques prédominantes du modèle d'affaires 'Développement à contrat'

- **Beaucoup de documentation**
  - Moyen de communication lorsque la taille du projet est importante
  - On a recourt à des fournisseurs externes.
  - Souvent plus efficace que les discussions de couloir quand les voies de communication sont complexes,
  - Certains documents sont souvent nécessaires pour prouver que nous avons fait ce qui est établi dans le contrat.
  - Exigences connues en détail au début du projet:
    - Documentation et de nombreuses révisions des exigences avant d'aller en appel d'offre.

Traduit de (Iberle, 2002, 2003)

Les pratiques prédominantes du modèle d'affaires 'Développement à contrat'

- **Les répertoires des pratiques exemplaires ***
  - Les répertoires de pratiques exemplaires tels que le modèle CMMI® pour le développement (CMMI-Dev) du SEI
    - Utilisés pour développer des clauses contractuelles.
    - Utilisés pour évaluer un fournisseur.
  - Les répertoires sont utilisés pour 'encadrer':
    - La planification (p.ex. estimation) et la gestion de projet pour rencontrer le calendrier et le budget prévu au contrat
    - La rédaction des plans et des rapports d'avancement

* Appelé parfois malheureusement '*Best Practices*'

Traduit de (Iberle, 2002, 2003)

Les pratiques prédominantes du modèle d'affaires 'Développement à contrat'

- **Utilisation du cycle de développement en cascade**
  - Pour donner aux grands projets TI suffisamment de structure pour être en mesure de planifier et d'orienter la livraison à temps.
- **Des audits de projet**
  - Des audits sont souvent spécifiés dans les contrats de ce modèle d'affaires.
  - Utilisé pour prouver, au client/juge ou lors de poursuites, que ces éléments ont été satisfaits:
    - les clauses contractuelles:
    - le respect du calendrier,
    - la qualité
    - les fonctionnalités

Traduit de (Iberle, 2002, 2003)

Modèle d'affaires: Le développement à l'interne

- En utilisant ses propres employés, les aspects économiques sont différents de ceux qui font développer des logiciels sous contrat.
- La valeur du travail dépend de l'amélioration de l'efficience ou l'efficacité des opérations de l'organisation.
- L'accent mis sur le calendrier est souvent moindre puisque les projets sont souvent suspendus ou repris en fonction de budgets.
- Les systèmes peuvent être critiques pour l'organisation ou de nature expérimentale.
- Les corrections sont distribuées à un nombre limité d'emplacements.
- Leurs peurs:
  - De produire des résultats incorrects
  - De limiter la capacité des autres employés à faire leur travail
  - Que leur projet soit cancellé

Traduit de (Iberle, 2002, 2003)

Modèle d'affaires: Les logiciels commerciaux

- Un logiciel vendu à d'autres organisations plutôt que pour un consommateur individuel.
- **Le profit**
  - Consiste à vendre de nombreuses copies du même logiciel plus cher qu'il ne coûte à développer et à en faire des copies.
- Le logiciel est souvent critique pour l'organisation ou au moins très important pour le fonctionnement de l'organisation du client.
- La distribution des corrections peut être très coûteuse.
  - Logiciel est dans les mains de nombreux clients dans de nombreux endroits,
- Ces clients ont aussi l'habitude de poursuivre en justice ou de causer d'autres ennuis si le logiciel est très déficient
- Les vendeurs de systèmes commerciaux craignent généralement:
  - Les poursuites
  - Les rappels
  - D'entacher leur réputation

Traduit de (Iberle, 2002, 2003)

Modèle d'affaires: Les logiciels de masse

- Logiciels sont vendus à des consommateurs individuels souvent à un volume très élevé.
- Profits
  - Vendre les produits supérieurs au coût, souvent dans le marché de niche ou à certains moments de l'année comme la période de Noël.
- Défaillances du logiciel sur le client sont généralement moins graves
  - Les clients sont moins susceptibles d'exiger des réparations pour les dommages encourus.
  - Peuvent affecter considérablement le bien-être financier de l'utilisateur comme pour un logiciel de préparation d'impôt.
- Les craintes typiques dans cette culture:
  - De rater la fenêtre de marché
  - Un taux élevé d'appels de soutien
  - De mauvaises critiques dans la presse

Traduit de (Iberle, 2002, 2003)

Modèle d'affaires: Les logiciels embarqués de masse

- Le coût de la distribution de correctifs est souvent extrêmement élevé
  - Les circuits électroniques qui doivent être changés sur place.
  - Les corrections ne peuvent pas être simplement être envoyées au client *.
- Le logiciel contrôle un dispositif (p.ex. freinage)
  - L'impact des pannes dans les logiciels embarqués de masse est potentiellement grave
  - Les défaillances du logiciel peuvent avoir des conséquences fatales
- Les craintes typiques de cette culture:
  - Un comportement incorrect du logiciel dans certaines situations
  - Les rappels (coûteux)
  - Les poursuites (coûteuses en $ en 'réputation')

* Graveur de DVD japonais

Traduit de (Iberle, 2002, 2003)

Les modèles d'affaires et le choix des pratiques de génie logiciel

- La connaissance des modèles d'affaires et de la culture des organisations aide à:
  - Évaluer l'efficacité de nouvelles pratiques pour une organisation ou un projet spécifique;
  - Apprendre les pratiques logicielles d'autres domaines ou d'autres cultures;
  - Comprendre le contexte qui aide à travailler avec les membres d'autres cultures;
  - S'intégrer plus facilement dans un nouvel emploi d'une autre culture.

Traduit de (Iberle, 2002, 2003)

Matériel supplémentaire

Les négligences

- Negligence imposes a set of expectations on behavior through the concept of duty.
- Since the duty in negligence focuses on behavior, it yields constraints on development processes and not the products themselves.
- Processes that fail to meet negligence constraints may be a basis for liability.
  - Processes that satisfy these constraints will not be the basis for liability, even if the product caused harm to an innocent party.
- Evaluation of processes for satisfaction of negligence constraints is a matter of evidence.
  - The evidence usually comes in the form of engineers' testimony and process documentation. It must be evaluated by Court approved experts in the relevant field who look for strengths and weaknesses in engineering tradeoffs. The Court evaluates the tradeoff process according to a social risk-benefit analysis.
  - Documentary evidence is persistent and not easily dismissed. Total reliance on human testimony about intricate process details (often years after the fact) is a very risky strategy
- Well informed software organization will prepare to produce evidence regarding the following:
  1. the state of the art in software process for the given domain; and,
  2. a precise and accurate picture of the process in question.
- The Court's ultimate judgment is negligence liability if 2 is significantly outside the bounds established in 1. On the other hand, if the organization can demonstrate that their process is well within the bounds of 1, they are judged non-negligent.

Green, M., Determining Negligence in Computer Error and Injury
http://expertpages.com/news/computer_error.htm