LIFC

UNIVERSITE DE FRANCHE-COMTE

Cours Génie logiciel

Partie 1:

- Qualité du logiciel
- Cycle de vie

Stéphane Debricon

Plan du cours Génie Logiciel – Partie I

1 - Qualité du logiciel
- Critères de qualité
- Démarche qualité
- Plan d'Assurance Qualité

2 - Cycles de vies du logiciels
- Cycles de vie linéaires (en cascade)
- Cycles de vie itératifs (prototypage, en spirale, incrémental)
- Cycles de vie avec méthode formelle

- Ouvrage de référence : I. Sommerville, « Software Engineering », 6th edition, Addison-Wesley, 2001

Section 1

Qualité du logiciel (ou comment l'atteindre ?)

Le risque logiciel

- Diffusion massive du logiciel (logiciels embarqués, pilotage, systèmes sécuritaire, …)
- Quelques exemples d'échecs :
  - Système de réservation de ticket SNCF
  - Crash Ariane 5
  - …. et bien d'autres

Le crash d'Ariane 5 – 4 juin 1996

- Ariane 5, le lanceur européen, successeur de Ariane 4 pour le lancement de satellites en orbite terrestre
- 37 secondes après un décollage réussi, Ariane 5 se fait exploser en vol
- Des informations incorrectes avaient été envoyées à l'engin mettant le système de contrôle en défaut
- L'explosion est due, de façon directe, à un défaut du logiciel de contrôle.

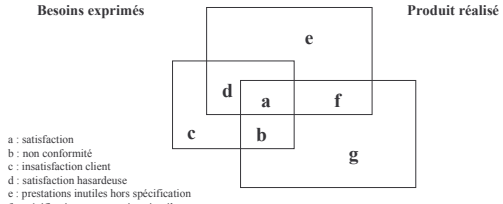
Diagnostic

- L'expertise a montré que le défaut provenait de la tentative de conversion d'un nombre flottant 64-bit en entier signé 16-bit
- Il n'y avait pas de gestion d'exceptions implantées.
- Le logiciel de sauvegarde fonctionnait exactement de la même façon.
- Cette partie du logiciel de contrôle provenait d'Ariane 4 sans avoir été modifiée. Du fait de l'accélération et de la vitesse moins importante sur Ariane 4, le problème n'était jamais apparu, ni en vol réel, ni en test.

Facteurs de non qualité du logiciel

- Mauvaises spécifications
  - Vagues, incomplètes, instables
- Mauvaises estimations
  - Fausses, oublis, précisions insuffisantes
- Mauvaise répartition des tâches
  - Organisation inadaptée, contraintes omises
- Mauvais suivi
  - Ecarts non détectés à temps
- Mauvaise réalisation technique
  - Codage, tests, documentation
- Problèmes humains
  - Mauvaise distribution des travaux
  - Conflits, rétention d'information
- Manque d'expérience du métier de Chef de projet

Le problème de la conformité aux besoins



Assurer la qualité du logiciel

- Assurer que le niveau de qualité requis est atteint
- Définir des standards de qualité et des procédures permettant d'assurer le niveau requis
- Développer une culture qualité dans les équipes de développement et de maintenance :

  "chacun est responsable"

Qu'est-ce que la qualité du logiciel?

- Qualité, signifie que le produit logiciel doit rendre les services pour lesquels il a été développé – i.e. respecter ses spécifications
- La qualité du logiciel est un problème complexe
  - Contradictions entre les besoins en qualité exprimés par l'utilisateur (ergonomie, efficacité, ...) et ceux exprimés par les développeurs (maintenabilité, réutilisabilité, etc.)
  - Les critères de qualité peuvent être difficiles à spécifier.
  - L'expression de besoins logiciels est souvent incomplète et parfois contradictoire.

Sept principes de la qualité

- Formalisation des procédures
  « Ecrire ce que l'on doit faire »
- Contrôle du respect des procédures
  « Vérifier que ce qui est fait est conforme à ce qui a été écrit »
- Traçabilité
  « Ecrire ce que l'on a fait »
- Mesure de la qualité
  « Apprécier la satisfaction du client »
- Calibrage par le retour d'expérience
  « Améliorer les procédures de façon continue »
- Unicité de responsabilité
  « Eviter la confusion des rôles »
- Séparation du contrôle et de la production
  « Eviter d'être juge et partie »

Qualité du logiciel - Normes

- Qualité du processus de production :
  - ISO 9000-3 référencée sur la série de normes d'exigences pour l'assurance qualité ISO 9000
  - ISO/CEI 12207 sur les modèles du processus du cycle de vie du logiciel
- Qualité du produit :
  - ISO/CEI 9126: Six caractéristiques génériques et 21 sous caractéristiques.

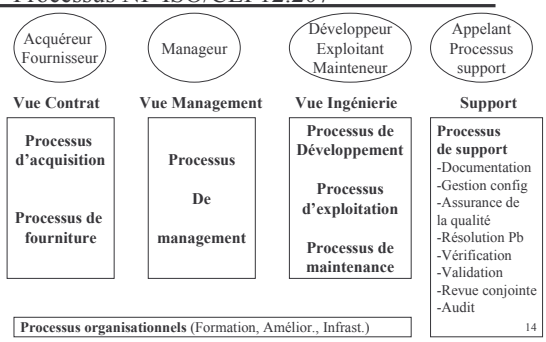
Processus d'assurance de la qualité

- Le processus d'assurance de la qualité doit fournir l'assurance que les produits du travail et les activités du projet sont conformes à leurs exigences et satisfont aux plans établis

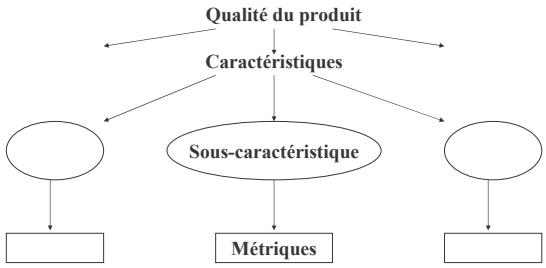
« processus qui vise à garantir que les autres processus et les logiciels associés au cycle de vie du projet sont conformes aux exigences requises et respectent les plans pré-établis »

« il convient que ce processus soit coordonné avec les processus de vérification, de validation, de revue conjointe et d'audit »

Processus NF ISO/CEI 12.207



Qualimétrie



Les critères de qualité (1)

• Validité : aptitude d'un produit logiciel à remplir exactement ses fonctions, définies par le cahier des charges et les spécifications.
  – Aptitude - exactitude - conformité réglementaire

• Fiabilité (ou robustesse) : aptitude d'un produit logiciel à fonctionner dans des conditions anormales.
  – Maturité - tolérance aux fautes - possibilité de récupération

• Intégrité : aptitude d'un logiciel à protéger son code et ses données contre des accès non autorisés.

Les critères de qualité (2)

• Réutilisabilité : aptitude d'un logiciel à être réutilisé, en tout ou en partie, dans de nouvelles applications.

• Compatibilité : facilité avec laquelle un logiciel peut être combiné avec d'autres logiciels.

• Efficacité : Utilisation optimale des ressources matérielles.
  – Comportement vis-à-vis du temps - comportement vis-à-vis des ressources

• Portabilité : facilité avec laquelle un logiciel peut être transféré sous différents environnements matériels et logiciels.

Les critères de qualité (3)

• Vérifiabilité : facilité de préparation des procédures de test.

• Extensibilité : facilité avec laquelle un logiciel se prête à une modification ou à une extension des fonctions qui lui sont demandées.
  – Facilité d'analyse - facilité de modification - stabilité

• Facilité d'emploi : facilité d'apprentissage, d'utilisation, de préparation des données, d'interprétation des erreurs et de rattrapage en cas d'erreur d'utilisation
  – Facilité de compréhension - facilité d'apprentissage - facilité d'exploitation

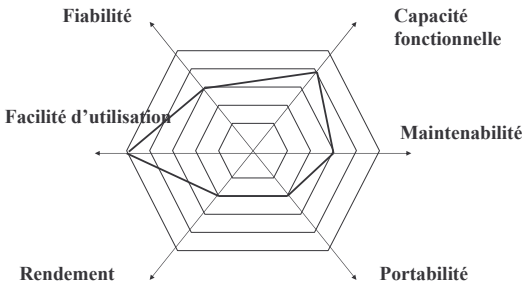
Les critères de qualité (4)

Safety	Understandability	Portability
Security	Testability	Usability
Reliability	Adaptability	Reusability
Resilience	Modularity	Efficiency
Robustness	Complexity	Learnability

Le compromis de la qualité

- Les critères de qualité peuvent être contradictoires, pour chaque projet, le choix des compromis doit s'effectuer en fonction du contexte.
  - facilité d'emploi vs intégrité
  - efficacité vs extensibilité
- La qualité s'obtient pas la mise en place de procédure et méthode dès la phase de spécification
- La démarche qualité ne consiste pas à corriger les défauts mais à les prévenir

Caractéristiques de qualité



Les activités de gestion de la Qualité

- Assurance Qualité
  - Définir les procédures organisationnelles et les standards de qualité du projet
- Gestion des procédures Qualité
  - Suivre la mise en place des procédures et standards Qualité
- Contrôle Qualité
  - S'assurer que les procédures et standards sont suivis par l'équipe de développement

☞ La gestion de la qualité doit être réalisée de façon indépendante de l'équipe de développement pour en assurer la pertinence

Manuel Qualité Logiciel

- Dispositions générales prises par une entreprise pour obtenir la qualité de ses produits et de ses services, sans tenir compte des exigences qualité d'un client ou des techniques et des outils d'un projet
  - Les règles de gestion de la qualité du logiciel,
  - Les règles et procédures en matière de conduite de projet, de production du logiciel, d'opération de tests…
  - Les plans types de documentation,
  - Ses propres règles d'évolution et l'organisation associée.
- Manuel des procédures peut être distinct

Plan d'Assurance Qualité (PAQ)

- Etabli à partir du Manuel Qualité Logiciel et des caractéristiques d'un projet déterminé doté de ses propres exigences de qualité
- Décrit les procédures, les règles et les méthodes applicables au projet
- Fixe aux différents acteurs du projet leurs droits mais aussi leurs devoirs en matière de qualité
- L'établissement et le suivi du Plan Qualité Logiciel sont du ressort du Responsable Qualité du Projet

Le Plan d'Assurance Qualité (PAQ)

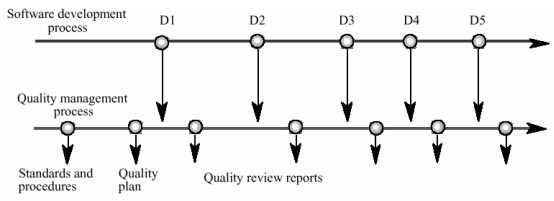
- Les objectifs qualité du projet
  - Hiérarchiser les critères
- Les moyens de la qualité
  - Quelles étapes de développements (cycle de vie)
  - Choix de méthodes et techniques
  - Structuration de l'équipe
- Le suivi de la qualité
  - Quelle mesure
- Identification des risques et prévention

☞ Le document de Plan d'Assurance Qualité possède un aspect contractuel

Exemple de PAQ

1. Introduction
   - Contexte, périmètre, procédures associées au PAQ
2. Glossaire et abréviations
3. Organisation
   - Rôles des intervenants et des structures de pilotage
4. Démarche de développement
   - Activités, documents en entrée et en résultat
5. Documentation
   - Identification, contenu, validation
6. Procédures diverses
   - Gestion de configuration, gestion de projet, gestion de points en suspens, gestion des modifications, gestion des écarts du progiciel, gestion des risques, gestion des validations, gestion de la recette
7. Reproduction, Protection, Livraison
8. Suivi de l'application du plan qualité

L'assurance qualité et le développement du logiciel



Assurance qualité : au niveau du projet

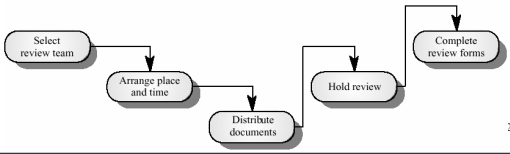
- Spécification de la qualité du produit et du processus de développement en début de projet

  ⇨ Plan d'Assurance Qualité

- Mise en œuvre des dispositions préétablies et systématiques du PAQ pour obtenir la qualité requise
- Suivi et évaluation de la qualité
  - À la fourniture des livrables, examen de la conformité,
  - Ponctuellement, conduite des revues et des audits,
  - De façon continue, appréciation de l'implication des acteurs et de l'efficacité de l'organisation
- Prévention des risques de non qualité

Contrôle Qualité : revues de projet

- Le groupe qualité examine avec précision les aspects méthodes et le logiciel produit ainsi que la documentation associée.
- Le code, la conception, les spécifications, les plans de test, les standards utilisés, etc. sont examinés.
- Le groupe qualité établi des recommandations qui doivent être mises en œuvre et vérifiée lors de l'audit suivant.



Revues de projet

- Equipe qualité : ingénieur qualité, chef de projet, responsable interface client
- L'objectif est de détecter, en amont, les défauts et inconsistances dans les documents et logiciels produits ainsi que dans les procédures
- Les résultats de la revue de projet :
  - Corrections à réaliser. Les concepteurs ou programmeurs doivent corriger les fautes identifiées
  - Modifications dans les méthodes de travail
  - RAS. Pas de corrections demandées

Revues de projet

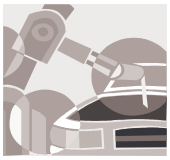
Review type	Principal purpose
Design or program inspections	To detect detailed errors in the design or code and to check whether standards have been followed. The review should be driven by a checklist of possible errors.
Progress reviews	To provide information for management about the overall progress of the project. This is both a process and a product review and is concerned with costs, plans and schedules.
Quality reviews	To carry out a technical analysis of product components or documentation to find faults or mismatches between the specification and the design, code or documentation. It may also be concerned with broader quality issues such as adherence to standards and other quality attributes.

Mesure de la qualité
Exemple - Facilité d'apprentissage

Questions	Réponse	Poids	Note
Aptitude à être mémorisé ◆ Chaque bouton correspond t-il à une fonction unique ? ◆ Existe t-il des valeurs par défaut pour les entrées ? ◆ Les procédures d'arrêt sont-elles décrites ? ◆ … ***Gestion des erreurs*** ◆ Toute erreur en entrée est-elle signalée explicitement ? ◆ Quand une commande est potentiellement destructrice, est-elle confirmée ? ◆ Existe t-il une liste des erreurs en ligne ? ◆ …			
Total			

Le Génie Logiciel

Objectifs du Génie Logiciel :
assurer la qualité du logiciel,
par la définition de cycles de vie,
par des méthodes
(spécifications, conception, test, …),
par des outils (Atelier de Génie Logiciel).



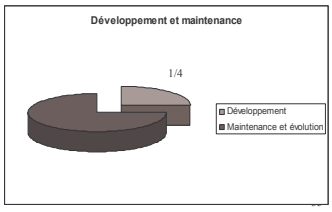
Section 2

Cycle de vie du logiciel

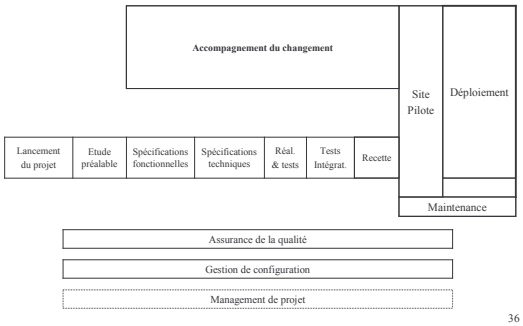
Cycle de vie du logiciel - Définition

- Cycle de vie du logiciel : suite de tâches pour spécifier, concevoir, implémenter, valider et maintenir les systèmes logiciels.

Pas seulement développer, mais aussi maintenir et faire évoluer



Vue globale des processus



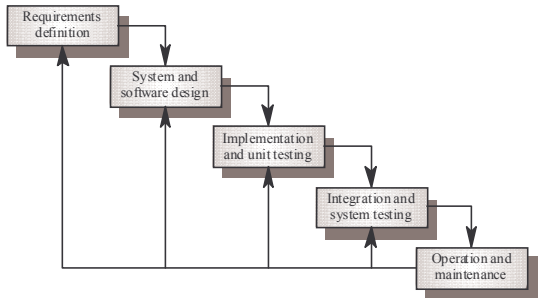
Le processus de développement (cycle de vie)

- Une suite d'étapes pour le développement et la maintenance d'un système logiciel
  - Spécification
  - Conception
  - Codage
  - Validation
  - Evolution
- Un modèle de cycle de vie constitue une représentation abstraite du cycle. Le processus peut être vu suivant un angle particulier (ex. Qualité, Gestion des risques, …)

Principaux modèles de processus de développement

1. Cycles linéaire
   - Phases distinctes et séparées de Spécification et Développement
2. Cycles itératifs – vision évolutionniste
   - Spécification et développement sont entrelacés
3. Cycles avec méthodes formelles
   - Une modélisation abstraite du système est transformée par étape formelle en une implémentation
4. Développement avec réutilisation de composants
   - Le système est conçu à partir de composants existants

1- Cycle linéaire : modèle en cascade



Les phases du modèle en cascade

- Expression de besoins et cahier des charges
- Spécification fonctionnelle détaillée
- Conception générale et détaillée
- Implémentation et test unitaire
- Intégration et test de conformité
- Mise en œuvre opérationnelle et maintenance

- Exemple : Merise (MCT, MCD, MOT, MLD, …)

Les limites du modèle en cascade

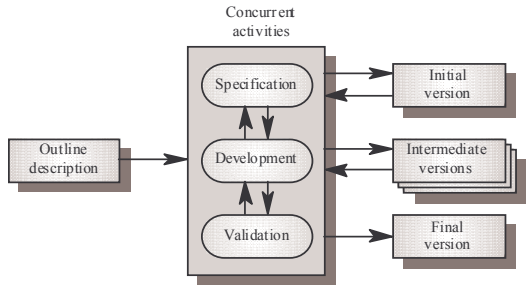
- L'inconvénient principal du modèle en cascade est « l'effet tunnel » vis-à-vis du client : après l'expression de besoins, suit un ensemble de phases (spécifications détaillées, conception et codage) qui conduit à ne pouvoir expérimenter qu'après une longue période :
  - Les besoins peuvent avoir été imparfaitement analysés,
  - Les besoins peuvent évoluer parce que le contexte change.

☞ Le modèle en cascade n'est approprié que lorsque l'expression de besoins est parfaitement claire et non soumise à évolution.

2- Cycles itératifs

- Développement exploratoire
  - L'objectif est de permettre une expérimentation rapide du logiciel en cours de développement pour consolider l'expressions de besoins et permettre la poursuite du développement
- Le prototype jetable (ou maquette) sert à valider un choix de spécification ou de conception (ex. temps de calcul d'un algorithme)
- Différents modèles itératifs :
  - Prototypage
  - Cycle en spirale (analyse du risque)
  - Développement incrémental

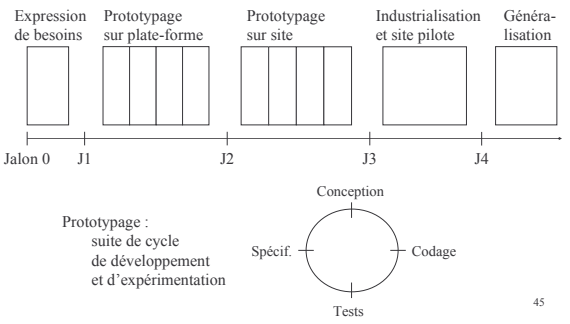
Développements itératifs



Développements itératifs

- Problèmes
  - Manque de visibilité des étapes
  - Le système résultant est souvent faiblement structuré
  - Peut nécessiter un environnement de développement adapté (e.g. langages pour le prototypage rapide)
- Applications
  - Pour les petits ou moyens systèmes interactifs
  - Pour des parties de grands systèmes (e.g. interface utilisateur)
  - Pour les systèmes à durée de vie courte

A - Cycle de vie par Prototypage



Prototypage : suite de cycle de développement et d'expérimentation

Cycle de vie par Prototypage

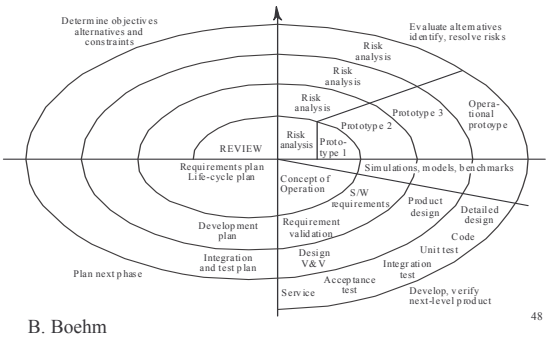
- Objectifs : permettre une expérimentation rapide du logiciel tout en conservant des techniques structurées de développement
- Séparation de la scène (le prototype) et des coulisses (les documents de modélisation et conception)
- Ce type d'approche permet un meilleur respect du critère de qualité *Validité* : par expérimentation, l'utilisateur confirme l'expression de besoins
- Application : grands systèmes logiciels avec diffusion sur plusieurs sites

B - Modèle en spirale

- Le processus est représenté par une spirale plutôt qu'une séquence d'activités avec retour arrière
- Chaque tour de la spirale représente une phase du processus de développement
- Le risque est évalué à chaque itération

☞ Ce modèle de cycle de vie est adapté aux projets importants avec un fort taux de risque

Modèle en spirale – analyse du risque



B. Boehm

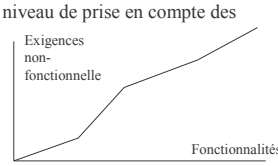
Modèle en spirale - dimensions

- Définition des objectifs
  - Les objectifs de chaque phase sont définis
- Evaluation et réduction du risque
  - Le risque est évalué et les tâches à réaliser pour le diminuer sont effectuées
- Développement et validation
  - Les tâche de développement et de validation sont réalisées
- Planning
  - Le projet est suivi et l'itération suivante de la spirale est planifiée

C - Développement incrémental

- Le développement est réalisé par suite d'incréments qui correspondent, à chaque livraison, à une partie des spécifications
- L'objectif est d'assurer la meilleure adéquation aux besoins possibles
- Le choix des incréments est un compromis entre la durée du développement et le niveau de prise en compte des spécifications.

Etapes d'un développement incrémental



Avantages du développement incrémental

- Le système peut être mis en production plus rapidement, avec la diffusion ensuite sous forme de versions
- Les premiers incréments permettent de conforter l'expression de besoins et autorisent la définition des incréments suivants
- Diminution du risque global d'échec du projet
- Les fonctions les plus fréquemment utilisées seront les plus testées, et donc les plus robustes

Extreme programming – XP (1)

- Nouvelle approche du développement incrémental basée sur la réalisation rapide, en équipe, d'incréments fonctionnels
- Caractéristiques:
  - Livraisons fréquentes,
  - Planification itérative,
  - Client sur site,
  - Rythme durable.

Extreme programming – XP (2)

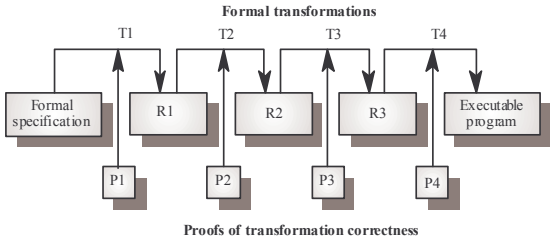
- Pratiques de développement spécifiques:
  - Conception simple (tout est utile),
  - Remaniement (du code par tous les développeurs),
  - Tests unitaires (systématiques),
  - Tests de recette (par rapport aux besoins client).
- Importante notion d'Equipe de développement:
  - Responsabilité collective du code,
  - Programmation en binômes,
  - Règles de codage,
  - Métaphore,
  - Intégration continue des modifications.

3- Développement par méthodes formelles

- S'appuie sur la transformation d'une spécification formelle, c'est à dire définie à partir de structure logique et mathématique, en un programme exécutable via une suite de représentations formelles intermédiaires
- Chaque transformation est prouvée correcte par rapport à la précédente, permettant d'assurer que l'implantation finale est correcte par rapport aux spécifications formelles de départ.

Développement formel d'un système

Transformations formelles - raffinement



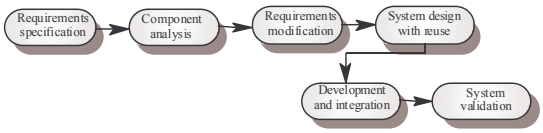
Développement formel d'un système

- Problèmes
  - Nécessite de l'équipe de développement des connaissances en formalisation et preuves du fait de l'automatisation partielle des preuves
  - Certains aspects du système sont difficiles à formaliser (ex. l'interface utilisateur)
- Applications
  - Logiciels critiques, mettant en jeu la sécurité des personnes (systèmes de transport, spatial, aéronautique, …) ou pour lesquels la présence d'une faute peut avoir des conséquences très importantes pour l'entreprise (ex. logiciel Cartes à Puces).

4- Cycle avec réutilisation de composants

- Basé sur une utilisation systématique de composants logiciels pré-existants - COTS (Component-off-the-shelf – Composant sur étagère)
- Etapes du processus
  - Analyse des composants
  - Besoins en modification
  - Conception avec réutilisation
  - Développement et intégration
- Cette approche se répand car elle peut permettre d'importantes réductions de coûts

Cycle avec réutilisation de composants



Les Processus de support

1. Gestion de configuration
2. Gestion de la documentation
3. Revue conjointe
4. Audit
5. Vérification
6. Validation
7. Assurance de la qualité
8. Test

Unified Process

- UP a évolué depuis les années 90.
- Il s'agit d'un processus itératif mais pas agile (Extreme programming)
- Le développement est dirigé par les risques durant les premières itérations qui produisent l'architecture de base du logiciel.
- Les itérations durent en moyenne 2 à 6 semaines selon les projets.

Histoire

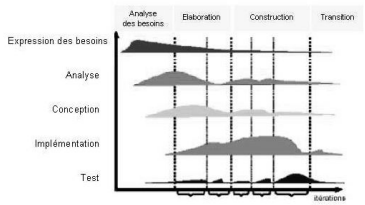
- On retrouve dans UP certain principe exposé dans le modèle en spirale de Boehm.
- Les principaux développements ont été réalisés entre 95 et 98.
- Le projet du contrôle aérien canadien a servi de test pour la mise en application des principes.
- Il existe une version commerciale, Rational Unified Process largement outillée et utilisée dans l'industrie.

Unified Process (UP)

- Il s'agit d'un processus de développement logiciel itératif.
- Les principes clés:
  - Itérations de courte durée,
  - Les risques majeurs sont traités dans les premières itérations,
  - Le client dispose toujours d'une version du développement,
  - Les changements sont pris en compte au plus tôt dans le projet,
  - Les activités sont regroupées dans une même équipe.

Unified Process

- UP est constitué d'activités et de phases



UP / Activités (1)

- **Expression des besoins**
  - inventorier les **besoins** principaux et fournir une liste de leurs fonctions
  - recenser les **besoins fonctionnels** (du point de vue de l'utilisateur) qui conduisent à l'élaboration des modèles de cas d'utilisation
  - appréhender les **besoins non fonctionnels** (technique) et livrer une liste des exigences.
- **Analyse**
  - L'objectif de l'analyse est d'accéder à une compréhension des besoins et des exigences du client. Il s'agit de livrer des spécifications pour permettre de choisir la conception de la solution.

UP / Activités (2)

- **Conception**
  - La conception permet d'acquérir une compréhension approfondie des contraintes liées au langage de programmation, à l'utilisation des composants et au système d'exploitation.
- **Implémentation**
  - L'implémentation est le résultat de la conception pour implémenter le système sous forme de composants, c'est-à-dire, de code source, de scripts, de binaires, d'exécutables et d'autres éléments du même type.
- **Test**
  - Les tests permettent de vérifier des résultats de l'implémentation en testant la construction.

UP / Phases (1)

- **Analyse des besoins**
  - L'analyse des besoins donne une vue du projet sous forme de produit fini. Cette phase porte essentiellement sur les besoins principaux (du point de vue de l'utilisateur), l'architecture générale du système, les risques majeurs, les délais et les coûts
- **Elaboration**
  - L'élaboration permet de préciser la plupart des cas d'utilisation, de concevoir l'architecture du système et surtout de déterminer l'architecture de référence.

UP / Phases (2)

- **Construction**
  - La construction est le moment où le produit s'élabore. L'architecture de référence se métamorphose en produit complet. Le produit contient tous les cas d'utilisation.
- **Transition**
  - Le produit est en version bêta. Un groupe d'utilisateurs essaye le produit et détecte les anomalies et défauts.

Quelques règles

- Toujours identifier et résoudre les risques majeurs.
- Produire des versions utilisables et exécutables dès les premières itérations.
- Privilégier les architectures orientées composants et la réutilisation de composants déjà existants.

Bonnes pratiques

- Itérations de courte durée permettant d'éviter l'effet tunnel.
- Maintenir une architecture cohérente et réutiliser les composants.
- Vérifier en permanence la qualité, par le test des éléments à chaque itération.
- Modéliser de façon visuelle pour explorer les problèmes.
- Organiser les besoins, en les regroupant et en les hiérarchisant.
- Gérer les versions à chaque itération.

Projets réalisés

- Administration canadienne de la sûreté du transport aérien
  - Système de contrôle aérien canadien
  - 10 ans, Ada et C++
  - Echec du développement en cascade: 11 ans et 2,6 billion $
- Ogre Nextgen
  - Système décisionnel concernant les cours du gaz.
  - 2 ans, Technologie Java
- QUICKcheck
  - Système de réservation de marchandise dans une chaine de magasin.
  - 1 an, 6 personnes, Technologie Java

Comment choisir une méthode

- Evaluer:
  - la durée du projet,
  - le nombre de participants au projet,
  - le facteur risque,
  - l'implication client,
  - le niveau de sécurité requis,
  - le déploiement envisagé,
  - la difficulté technique.

Section 3

Conduite de réunion

Avant la réunion

- Opportunité de la réunion
  - Avant tout chose, la raison d'être de la tenue d'une réunion doit être réfléchie :
    - » Quel est l'objectif de la réunion ?
    - » Une réunion téléphonique peut-elle être suffisante ?
    - » Une web conférence suffirait-elle ?

Avant la réunion

- Périmètre de la réunion
  - Nombre et qualité des participants.
  - Durée : Idéalement, la durée de la réunion ne devrait pas dépasser 2 heures.
  - Ordre du jour : Il s'agit du découpage horaire du temps de travail en sujets bien formulés. Il est nécessaire de minuter correctement les différents sujets de l'ordre du jour.

Avant la réunion

- Date et réservation de la salle
  - Trouver une salle de réunion libre à une date où les participants sont disponibles.
  - 15 jours à l'avance, afin de permettre l'envoi des convocations aux participants dans des délais décents.

Avant la réunion

- Contraintes à prendre en compte :
  - Capacité de la salle ;
  - Dimensions et forme de la salle (selon le type de présentation ou d'animation) ;
  - Besoin d'un accès à internet ;
  - Présence d'ordinateurs et de moyens audiovisuels (vidéo projecteur).

Avant la réunion

- Prévenir les participants
  - Diffuser l'ordre du jour, en précisant le lieu ainsi que l'heure de début et de fin de la réunion.
  - Un document préparatoire, envoyé préalablement, faisant éventuellement apparaître quelques questions clés, permettra aux participants de mieux préparer leur intervention.

Pendant la réunion

- Un « tour de table » permet à chacun de se présenter brièvement pour situer la fonction de chaque intervenant.
- Il est souhaitable de « désigner un volontaire » pour la rédaction du compte-rendu. S'il s'agit d'une série de réunion, chacun devra être rapporteur à son tour.

Pendant la réunion

- S'il s'agit d'une série de réunions, les décisions de la réunion précédentes peuvent être passées en revue.
- Récapituler brièvement l'ordre du jour de la journée, le temps imparti sur chaque sujet et les intervenants.
- Les points importants de l'ordre du jour devront préférentiellement être abordés en début de réunion.

Pendant la réunion

- Mettre à l'aise :
  - Café, jus de fruit, bouteilles d'eau.
  - Une salle correctement dimensionnée et une température adaptée.
- Un « relevé de décisions », reprends les décisions essentielles prises au cours des échanges. Un responsable doit être désigné pour la mise en œuvre de chacune des actions, avec une date prévisionnelle.
- Profiter de la présence des participants pour convenir d'une date commune pour la tenue de la réunion suivante.

Après la réunion

- Rédiger le compte-rendu « à chaud ». Il doit faire apparaître les points suivants :
  - Objet de la réunion,
  - Date de la réunion,
  - Participants (et excusés),
  - Ordre du jour,
  - Résumé de chaque point de l'ordre du jour,
  - Relevé de décisions

Après la réunion

- Le compte-rendu de la réunion a plusieurs objectifs :
  - Acter des décisions.
  - Formaliser le travail réalisé pour permettre par exemple aux excusés de pouvoir en prendre connaissance.
  - Capitaliser l'information, pour mémoire.
- Il doit être diffusé à l'ensemble des participants, pour validation.
- Après un délai de l'ordre d'une semaine, le compte-rendu final devra être envoyé.

Section 4

eXtreme Programming

Plan

- Introduction
- Valeurs et Pratiques de l'XP
- Le cycle standard

Pourquoi la méthode eXtreme Programming (XP) ?

- Réactivité par rapport aux besoins client
- Complexité des projets
- Mauvaise maîtrise des coûts et temps de développement

Qu'est-ce que l' eXtreme Programming ?

- Méthode de conduite de projet
- Méthode agile

- Autres méthodes agiles :
  - SCRUM
  - Adaptive Software Development
  - Feature Driven Development
  - Rational Unified Process
  - …

Valeurs et Pratiques

13 pratiques pour résoudre ces problèmes

1. Règles de codage
2. Refactoring
3. Conception simple
4. Tests unitaires
5. Intégration continue
6. Responsabilité collective du code
7. Travail en binômes
8. Langage commun / Métaphore
9. Rythme durable
10. Client sur site
11. Tests de recette
12. Planification itérative
13. Livraisons fréquentes

Des valeurs à respecter

- Communication :
  au sein de l'équipe, avec le client
- Feedback :
  itérations rapides, permet la réactivité
- Simplicité :
  du code, livrables ne contenant que les exigences du client
- Courage :
  il faut parfois faire des choix difficiles, cela est facilité par les 3 premières valeurs

Pratiques concernant l'équipe

- Responsabilité collective du code
  - Polyvalence des développeurs

- Travail en binôme
  - Partage des compétences
  - Prévention des erreurs
  - Motivation mutuelle

- Langage commun / Métaphore

- Rythme durable

Pratiques concernant le code

- Refactoring
  - Investissement pour le futur
  - Réduction de la dette technique
- Conception simple
  - Facilite la reprise du code
  - Facilite l'ajout de fonctionnalités

Pratiques concernant le code

- Tests unitaires
  - Test first
- Intégration continue
  - Ajout de valeurs chaque jour
- Règles de codage
  - Ex : java → Standards SUN
  - Ex : java → Formatage automatique Eclipse

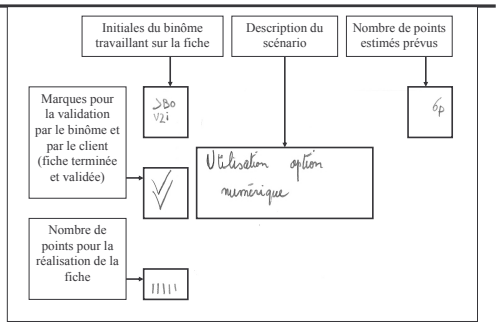
Pratiques concernant le projet

- Client sur site
- Tests d'acceptation
  - Conformité par rapport aux attentes du client
- Planification itérative et livraisons fréquentes
  - Fiches et Itérations

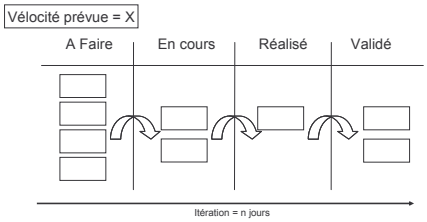
L'utilisation des fiches

- Rédigées pour la plupart au début du projet, puis tout au long du projet
- Une fonctionnalité (scénario) = Une fiche
- Sur chaque fiche on doit renseigner :
  - La description du scénario correspondant
  - Un ordre de priorité
  - La durée prévue pour la réalisation (en jours ou en nombre de points)
    - » Un point = une unité de temps ou un niveau de difficulté
  - La durée écoulée depuis le début de la réalisation
  - Le binôme qui réalise le développement

L'utilisation des fiches



L'utilisation des fiches



En fin d'itération, on connaît le nombre de points réalisés pour la durée prévue, on peut donc estimer la vélocité de la prochaine itération, et prévoir les délais de réalisation du projet

Les 13 pratiques, résumé

PROJET	CODE	EQUIPE
Livraisons fréquentes : l'équipe vise la mise en production rapide d'une version minimale du logiciel, puis elle fournit ensuite régulièrement de nouvelles livraisons en tenant compte des retours du client.	**Conception simple** : on ne développe rien qui ne soit utile tout de suite.	**Programmation en binômes** : les développeurs travaillent toujours en binômes, ces binômes étant renouvelés fréquemment.
Planification itérative : un plan de développement est préparé au début du projet, puis il est revu et remanié tout au long du développement pour tenir compte de l'expérience acquise par le client et l'équipe de développement.	**Tests unitaires** : les développeurs mettent en place une batterie de tests de non régression qui leur permettent de faire des modifications sans crainte.	**Responsabilité collective du code** : chaque développeur est susceptible de travailler sur n'importe quelle partie de l'application.
Client sur site : le client est intégré à l'équipe de développement pour répondre aux questions des développeurs et définir les tests fonctionnels.	**Remaniement (refactoring)** : le code est en permanence réorganisé pour rester aussi clair et simple que possible.	**Rythme durable** : l'équipe adopte un rythme de travail qui lui permet de fournir un travail de qualité tout au long du projet.
Tests de recette : les testeurs mettent en place des tests automatiques qui vérifient que le logiciel répond aux exigences du client. Ces tests permettent des recettes automatiques du logiciel.	**Intégration continue** : l'intégration des nouveaux développements est faite chaque jour.	**Métaphore** : les développeurs s'appuient sur une description commune du design.
	Règles de codage : les développeurs se plient à des règles de codage strictes définies par l'équipe elle-même.	

Le cycle standard XP

1. Le client écrit ses besoins sous forme de scénarii.
2. Les développeurs évaluent le coût de chaque scénario, si le coût est trop élevé pour un scénario ou s'il ne peut pas être estimé, le client le découpe en plusieurs scénarii et les soumet à nouveau à l'évaluation des développeurs.
3. Le client choisit les scénarii à intégrer à la prochaine livraison (en accord avec les développeurs).
4. Chaque binôme de développeurs prend la responsabilité d'une tâche.

Le cycle standard XP

5. Le binôme écrit les tests unitaires correspondant au scénario à implémenter.
6. Le binôme procède à l'implémentation proprement dite, puis
7. Le binôme réorganise le code (refactoring).
8. Le binôme intègre ses développements à la version d'intégration, en s'assurant que tous les tests de non régression passent.

Le cycle standard