Approches du génie logiciel

Introduction

- Le génie logiciel est un domaine en pleine évolution qui offre une grande palette d'outils et de méthodes pour parvenir à construire du logiciel de qualité.
- Aucune de ces méthodes ne s'est imposée à ce jour : il faut donc prendre du recul sur les concepts et les conseils qu'elles préconisent et utiliser son bon sens pour les adapter à chaque situation.
- Ces méthodes se distinguent principalement par :
  - ✓ leur degré de formalisme ;
  - ✓ leur champ d'application ;
  - ✓ les contraintes de qualité qu'elles ambitionnent.

Approches formelles

- Les approches formelles utilisent des outils mathématiques et des méthodes de preuve pour construire un logiciel correct par construction dont la vérification est automatisée ou assistée.
- Exemple (approches formelles)
  - ✓Méthodes : méthode B, model-checking, logique de Hoare, . . .
  - ✓Outils et notations : Coq, Z, Atelier B, Why, Frama-C, . . .
- Ces méthodes sont utilisées pour développer des logiciels critiques.
- Elles correspondent au niveau le plus élevé de certification.
  - ✓e.g. applications de la méthode B pour développer le logiciel embarqué des lignes de métro 14 (1998) et 1 à Paris

Approches semi-formelles

- Les approches semi-formelles visent à introduire un langage normalisé pour décrire le logiciel et sa spécification.
  - ✓ Cependant, la sémantique du langage de spécification n'est pas formalisée.
  - ✓ Bien que ces approches précisent le discours du concepteur si on le compare à celui décrit à l'aide du langage naturel, elles contiennent certaines ambiguïtés et n'offrent aucune garantie sur la qualité des résultats.
- Exemple (approches semi-formelles)
  - ✓ Méthodes : Rationale Unified Process, Merise, . . .
  - ✓ Outils et notations : UML, Analyse SI, . . .
- Ces méthodes sont utilisées aujourd'hui par l'industrie du logiciel.

Approches empiriques

- Les approches empiriques mettent en avant un ensemble de “bonnes pratiques” qui ont fait leur preuve par l’expérience.
- Exemple (approches empiriques)
  - ✓Méthodes : unit testing (p.ex. la famille xUnit), peer review, relecture de. code, extreme programming, programmation défensive, . .
  - ✓Outils : plates-formes de test, gestionnaire de versions (p.ex. Git), outil de documentation automatique (p.ex. Doxygen) / literate programming, . . .

Les grands principes du génie logiciel

- Un certain nombre de grands principes (de bon sens) se retrouvent dans toutes ces méthodes.
  - ✓La rigueur;
  - ✓La décomposition des problèmes en sous-problèmes;
  - ✓La modularité;
  - ✓L'abstraction
  - ✓L'anticipation des évolutions;
  - ✓La généricité;
  - ✓La construction incrémentale

Les grands principes du génie logiciel (Rigueur)

- Les principales sources de défaillances d'un logiciel sont d'origine humaine.
- La production de logiciel est une activité créative, mais qui doit se conduire avec une certaine rigueur
- 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.
- Exemples :

→ générateurs de code, assistants de preuves, générateurs de tests, profileurs, test coverage, outil d'intégration continue, fuzzer, . . .

Les grands principes du génie logiciel (Décomposition des problèmes)

- C'est un aspect de la stratégie générale du « diviser pour régner »
- Simplifier les problèmes (temporairement) pour aborder leur complexité progressivement.
- Décorréler les problèmes pour n'en traiter qu'un seul à la fois.
- Exemple
  - ✓Comment créer dynamiquement une page internet pour visualiser et modifier le contenu d'une base donnée sans la corrompre ?
  - ✓Décomposition en trois composants :
  - ✓Modèle: son rôle est gérer le stockage des données.
  - ✓Vue: son rôle est de formater les données.
  - ✓Contrôleur: son rôle est de n'autoriser que les modifications correctes.
  - ✓Comment acheminer un email de façon sûr à travers un réseau?

    → Décomposition en couches utilisée sur Internet : SMP TCP ..

Les grands principes du génie logiciel (Modularité)

- C’est une instance cruciale du principe de décomposition des problèmes.
- Il s’agit de partitionner le logiciel en modules qui :
  - ont une cohérence interne (des invariants) ;
  - 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.

Les grands principes du génie logiciel (Abstraction)

- C'est encore une instance du principe de décomposition des problèmes.
- Il s'agit d'exhiber 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 ;
  - ✓ de factoriser le travail en instanciant le raisonnement général sur chaque cas particulier.
- Exemple (Support dans les langages de programmation)
  - ✓ les classes abstraites dans les langages à objets (Forme Géométrique, Carrée, rectangle, Triangle)

Les grands principes du génie logiciel (Anticipation des évolutions)

- Un logiciel a un cycle de vie plus complexe que l'habituel :

  commande → spécification → production → livraison

- La maintenance est la gestion des évolutions du logiciel.
- 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.

Les grands principes du génie logiciel (Généricité)

- proposer des solutions plus générales que le problème pour pouvoir les réutiliser et les adapter à d'autres cas
- Un logiciel réutilisable a beaucoup plus de valeur qu'un composant dédié.
- Un composant est générique lorsqu'il est adaptable.
- Exemple :

→ plutôt que d'écrire une identification spécifique à un écran particulier, écrire (ou réutiliser) un module générique d'authentification (saisie d'une identification et éventuellement d'un mot de passe).

Les grands principes du génie logiciel (Construction incrémentale)

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

- Exemple Laquelle de ses deux méthodes de programmation est la plus efficace ?
  1. Écrire l'ensemble du code source d'un programme et compiler.
  2. Écrire le code source d'une fonction ou module, le compiler, et passer à la suivante.

processus de développement logiciel

Introduction

- Un processus de développement logiciel est un ensemble (structuré) d'activités que conduisent à la production d'un logiciel

  « La qualité du processus de fabrication est garante de la qualité du produit »

- Pour obtenir un logiciel de qualité, il faut en maîtriser le processus d'élaboration
- La vie d'un logiciel est composée de différentes étapes
- La succession de ces étapes forme le cycle de vie du logiciel
- Il faut contrôler la succession de ces différentes étapes

Introduction (suite)

- En pratique:

  - ✓Il n’existe pas de processus idéal.
  - ✓La plupart des entreprises adapte les processus existants à leurs besoins.
  - ✓Ces besoins varient en fonction du domaine, des contraintes de qualité, des personnes impliquées.

Activités du développement logiciel

- Les étapes ou Activités du développement d'un logiciel
  - Analyse des besoins
  - Spécification
  - Conception
  - Programmation
  - Validation et vérification
  - Livraison
  - Maintenance
- Chaque Activité:
  - possède des entrées et des sorties
  - Les livrables font parties des sorties (Documents, Planning, code source)

Activités du développement logiciel (Analyse des besoins)

- Objectif :
  - ✓ Comprendre les besoins du client
  - ✓ Objectifs généraux, environnement du futur système, ressources disponibles, contraintes de performance...
- Entrée : Fournie par le client
  - ✓ Expert du domaine d'application, futur utilisateur
- Documents produits : Cahier des charges (+ manuel d'utilisation préliminaire)

Activités du développement logiciel (Spécification)

- Objectif :
  - ✓ Établir une description claire de ce que doit faire le logiciel (fonctionnalités détaillées, exigences de qualité, interface...)
  - ✓ Clarifier le cahier des charges (ambiguïtés, contradictions)
- Entrée : Cahier des charges + considérations de faisabilité
- Document produit : Cahier des charges fonctionnel

Activités du développement logiciel (Conception)

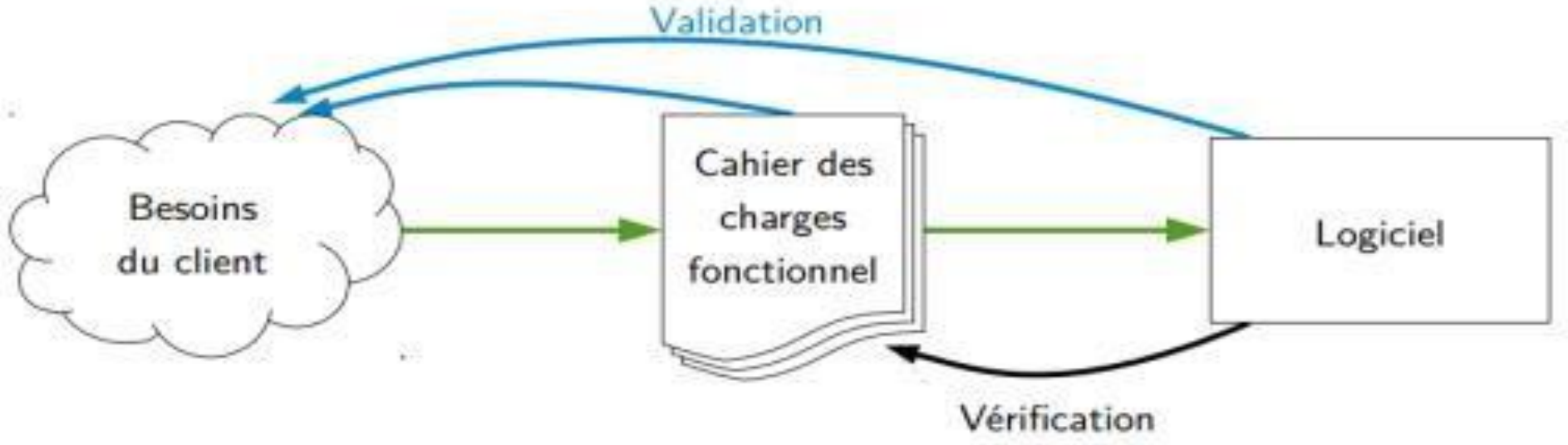
- Objectif :
  - ✓ Élaborer une solution concrète réalisant la spécification
  - ✓ Description architecturale en composants (avec interface et
  - ✓ fonctionnalités)
  - ✓ Réalisation des fonctionnalités par les composants (algorithmes, organisation des données)
  - ✓ Réalisation des exigences non-fonctionnelles (performance, sécurité...)
- Entrée : Cahier des charges fonctionnel
- Document produit : Dossier de conception

Activités du développement logiciel (Programmation)

- Objectif :
  - ✓ Implémentation de la solution conçue
  - ✓ Choix de l'environnement de développement, du/des langage(s) de programmation, de normes de développement...
- Entrée : Dossier de conception
- Documents produits : Code documenté + manuel d'utilisation

Activités du développement logiciel : Validation et vérification

- Objectifs :
  - ✓Validation : assurer que les besoins du client sont satisfaits (au niveau de la spécification, du produit fini...)
  - ✓Vérification : assurer que le logiciel satisfait sa spécification



Activités du développement logiciel : Validation et vérification

- Essayer le logiciel sur des données d'exemple pour s'assurer qu'il fonctionne correctement
  - ✓ Tests unitaires : faire tester les parties du logiciel par leurs développeurs
  - ✓ Tests d'intégration : tester pendant l'intégration
  - ✓ Tests de validation : pour acceptation par l'acheteur
  - ✓ Tests système : tester dans un environnement proche de l'environnement de production
  - ✓ Tests Alpha : faire tester par le client sur le site de développement
  - ✓ Tests Bêta : faire tester par le client sur le site de production
  - ✓ Tests de régression : enregistrer les résultats des tests et les comparer à ceux des anciennes versions afin de vérifier le taux de compatibilité

Activités du développement logiciel : Validation et vérification

- Méthodes de validation :
  - ✓Revue de spécification, de code
  - ✓Prototypage rapide
  - ✓Développement de tests exécutables
- Méthodes de vérification :
  - ✓Test (à partir de la spécification) :exécution d'un programme sur des données choisies dans le but de détecter des non-conformités par rapport à la spécification
  - ✓Preuve de programmes :preuve mathématique qu'un programme satisfait sa spécification en termes de pré- et post-conditions
  - ✓Model-checking: analyse d'un modèle du programme dans le but de prouver mathématiquement qu'il vérifie certaines propriétés dynamiques

Activités du développement logiciel (Livraison)

- Fournir au client une solution logicielle qui fonctionne correctement Installation :
  - ✓rendre le logiciel opérationnel sur le site du client
  - ✓Formation: enseigner aux utilisateurs comment se servir du logiciel
  - ✓Assistance : répondre aux questions des utilisateurs

Activités du développement logiciel : Maintenance

- Lois de Belady et Lehman:
  - ✓ Un logiciel est en constante évolution

    La livraison n'est pas une fin en soi, après sa livraison un logiciel peut être modifié
  - ✓ Lorsqu'un logiciel évolue, il devient:
    - moins structuré
    - Plus complexe

Activités du développement logiciel : Maintenance

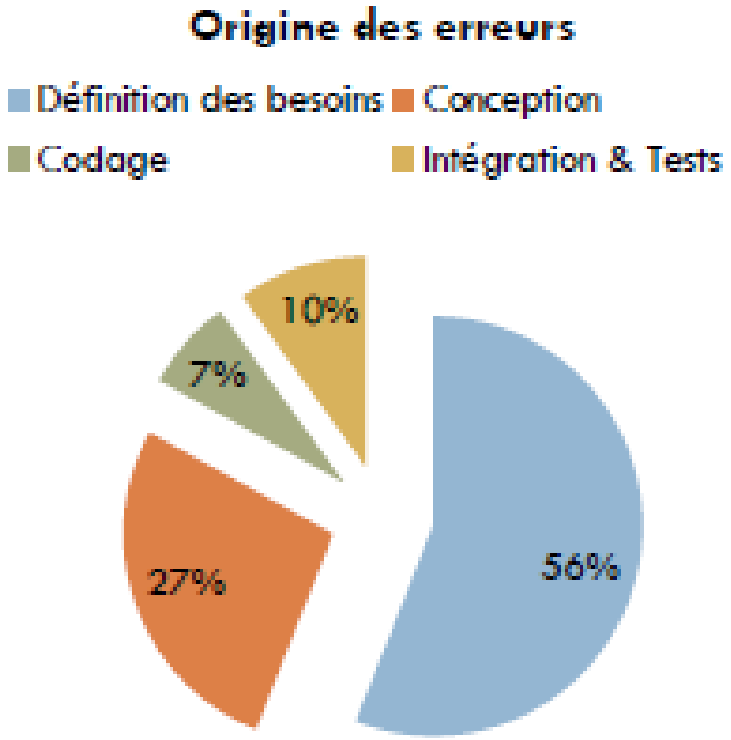
On distingue trois types de maintenance:

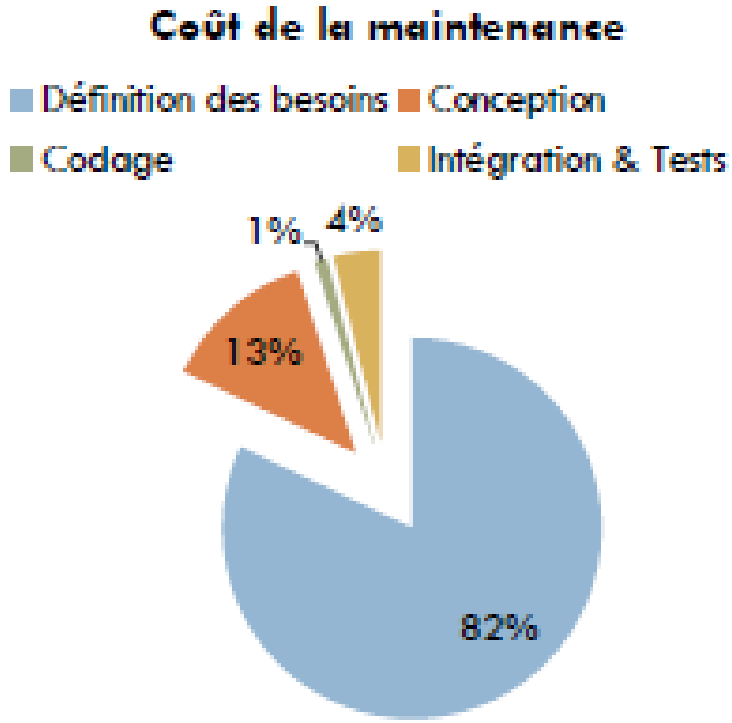
- Maintenance Corrective : identifier et corriger des erreurs trouvées après la livraison:
  - ✓Identifier la défaillance;
  - ✓le fonctionnement Localiser la partie du code responsable;
  - ✓Corriger et estimer l'impact d'une modification
- Attention!!!
  - ✓La majorité des corrections introduisent de nouvelles erreurs
  - ✓Les coûts de correction augmentent considérablement avec le délai de détection
- La maintenance corrective donne lieu à des nouvelles livraisons

Activités du développement logiciel : Maintenance

- Maintenance Adaptative : Adapter le logiciel afin qu'il continue à fonctionner correctement en tenant compte des changement enregistrés dans l'environnement:
  - ✓format des données,
  - ✓environnement d'exécution,
  - ✓changement de SGBD,
- Maintenance Perfective : comprend tous les changements faits sur un système afin de satisfaire aux besoins de l'utilisateur.
  - ✓améliorer la performance,
  - ✓ajouter des fonctionnalités,
  - ✓améliorer la maintenabilité.

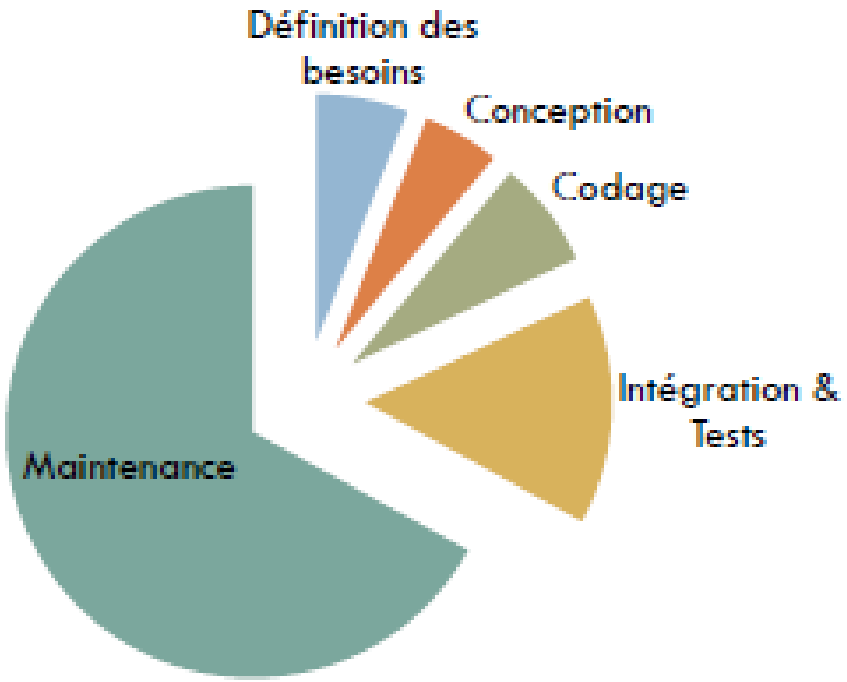
Activités du développement logiciel : Quelques statistiques





Source Martin & Leffinel

Activités du développement logiciel : Quelques statistiques



Source Martin & Leffinel

Répartition d'effort de développement