Introduction au Génie logiciel: outils et méthodes

Module RCPI01

Sebastien.Kramm@univ-rouen.fr

IUT R&T Rouen

2018-2019

Sommaire

UNIVERSITÉ DE ROUEN

Un joli feu d'artifice...

Question : quel a été le feu d'artifice le plus cher de tous les temps ?

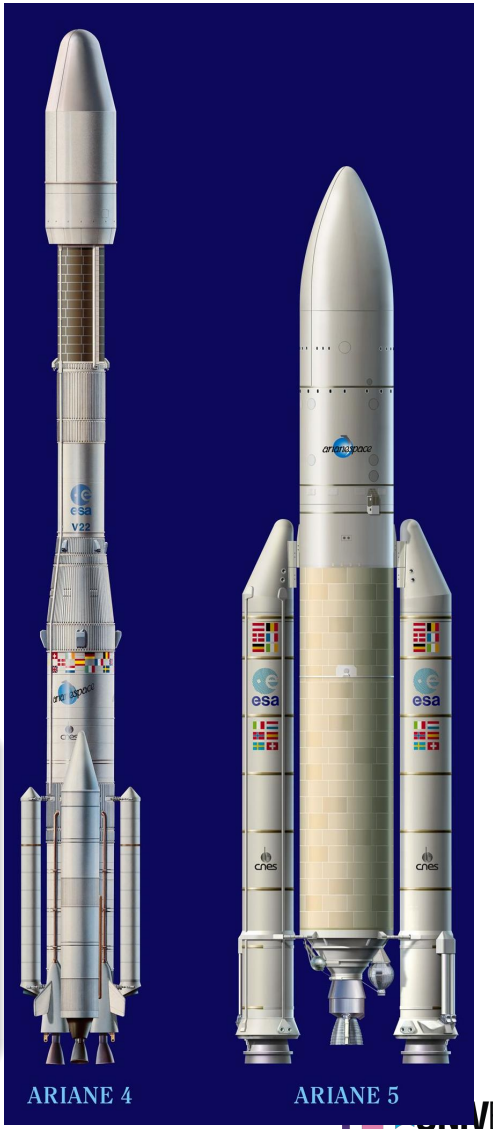
Réponse : Ariane 5 ($M 370)

Bug logiciel

- Ariane 5 utilisait le même logiciel de navigation que Ariane 4
- variable stockant l'accélération horiz. sur 8 bits
  - Sur Ariane 4 : valeur $\simeq 64$
  - Sur Ariane 5, beaucoup plus puissante : valeur $\simeq 300$
    $\Rightarrow$ dépassement de capacité et pannes en cascade...

Causes identifiées

- Tests incomplets.
- Déficiences du management dans la gestion du projet.



Développement de logiciel ?

- Consiste à étudier, concevoir, construire, transformer, mettre au point, maintenir et améliorer des logiciels.
- La conception de logiciel va suivre trois grandes phases :
  - Phase d'analyse (fonctionnelle) ou de conception
  - Phase de réalisation ou de programmation (écriture et tests des programmes)
  - Phase de livraison
- Mise en œuvre de méthodes de **gestion de projet**

UNIVERSITÉ DE ROUEN

Génie logiciel : pourquoi ?

- Lors du développement massif de l'informatique (1960-70), les logiciels sont devenus de plus en plus complexes, et par conséquent :
  - pas fiables,
  - très difficiles à réaliser dans les délais,
  - ne satisfaisaient pas le cahier des charges.
- Des problèmes apparaîssent à tous les niveaux :
  - Specifications : peuvent être incomplètes ou incohérentes (souvent, même le client ne sait pas ce qu'il veut...)
  - Développement : comment être sur qu'un module / une fonction répond au cahier des charges ?
  - Intégration : comment être sur que le produit développé va s'intégrer dans l'environnement du client ?

⇒ Il est apparu la nécessité de méthodes de gestion de projets spécifiques : **Génie Logiciel**, qui met en œuvre des **méthodes**, des **modèles**, et des **outils**.

UNIVERSITÉ DE ROUEN

Conduite de projet : ordres de grandeurs

Le *Standish Group* (cabinet de consulting) fait chaque année un sondage sur la conduite des projets informatiques.
En 2010 :

- 37 % des projets étaient conformes aux prévisions initiales,
- 42 % avaient subi des dépassements en coût et délai d'un facteur 2 à 3 avec diminution du nombre des fonctions offertes,
- 21% ont été des echecs complets.

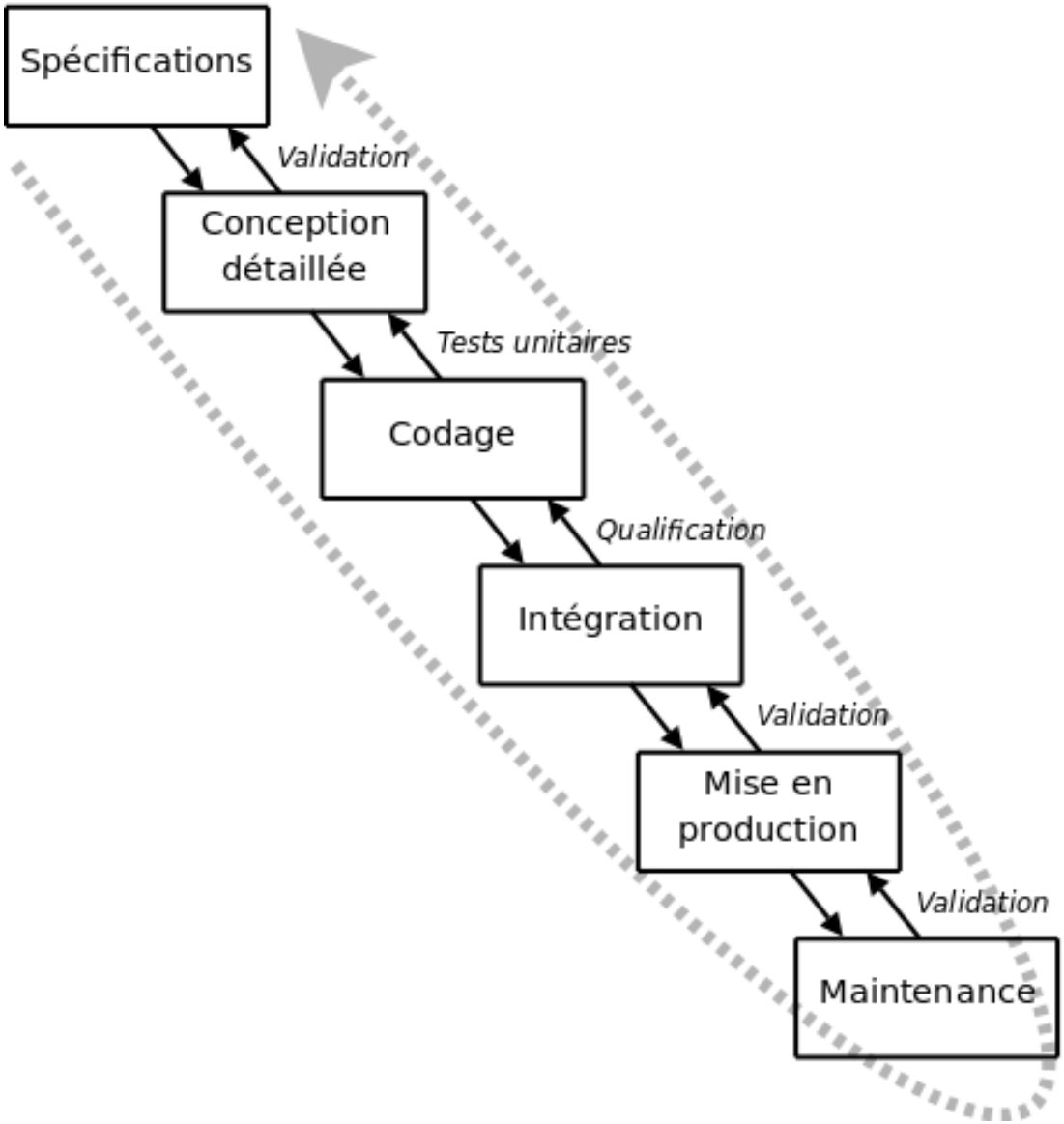
Exemple : "Louvois", logiciel de calcul de la solde des militaires français

- développé de 2001 à 2011, abandonné en 2013
- Coût : > 80 MEuro

UNIVERSITÉ DE ROUEN

Sommaire

Cycle en cascade



⇒ adapté au batiment... Pas à du developpement de logiciel !

Cycles de développement

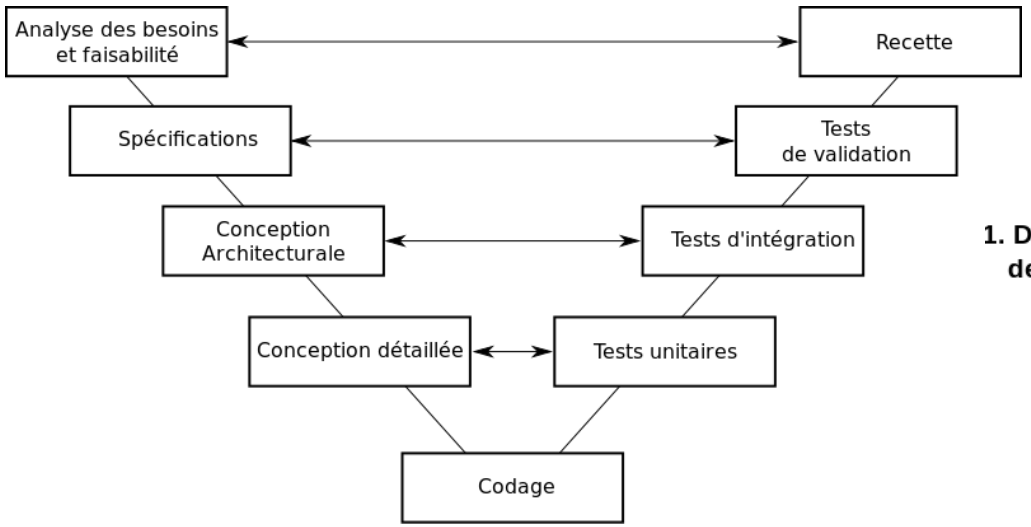
Il existe plusieurs modèles conceptuels de gestion de projet pour du logiciel :

- Cycle en V
- Cycle en spirale
- Cycle semi-itératif
- Cycle itératif & Méthodes dites "Agiles" (Scrum, Extreme programming, ...)

Cycles en V et en spirale

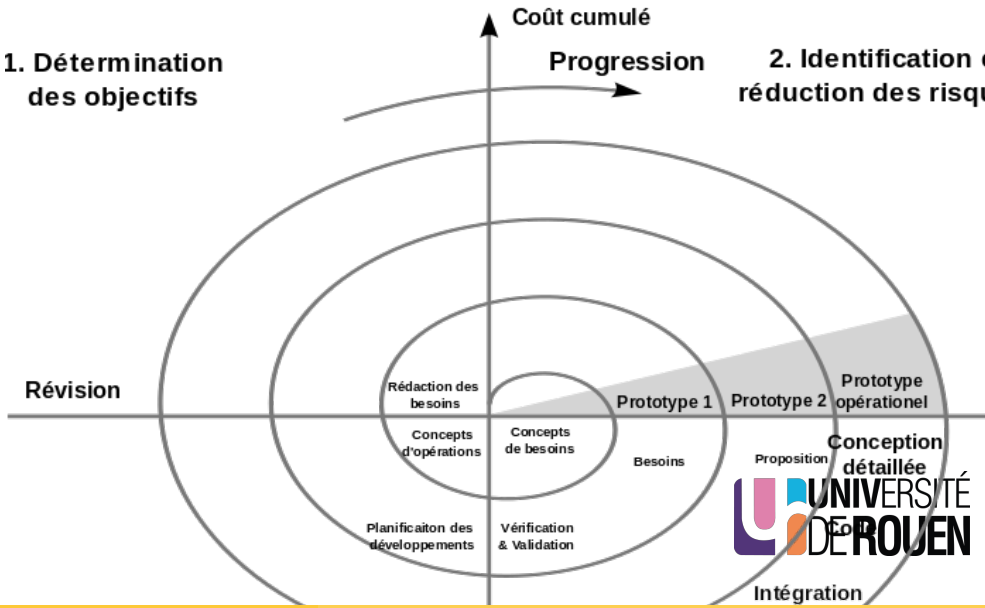
Cycle en V

- Plutôt pour des gros projets très étalés dans le temps



Cycle en spirale

- Implémentation de versions successives, le cycle recommence en proposant un produit de plus en plus complet et robuste.



Sommaire

Sommaire

Phase d'analyse

Via une méthode d'analyse normalisée, on procède à :

- Etudes des données
- Etude des traitements à effectuer

Il en ressort une description des bases de données, des programmes à développer et des différentes interactions.

1. Spécification
2. Conception
3. Définition de l'architecture

Phase d'analyse

Spécification

Ensemble explicite d'exigences à satisfaire par un matériau, produit ou service.

Conception

- Formalisation des étapes préliminaires du développement d'un système afin de rendre ce développement fidèle aux besoins du client.
- En Génie logiciel : UML

Architecture

- Structure générale d'un système informatique, organisation des différents éléments du système (logiciels et/ou matériels et/ou humains et/ou informations) et des relations entre les éléments.
- Lié à un ensemble de décisions stratégiques prises durant la conception du système.

Sommaire

Phase de réalisation

- Phase d'écriture des programmes informatiques dans un langage de programmation informatique
- Met en œuvre des techniques et des outils divers :
  - Algorithmique
  - Programmation
  - Gestion de versions
  - Tests unitaires (TDD : *Test Driven Developement*)
  - *Profiling*, optimisation du code et *Refactoring*

UNIVERSITÉ DE ROUEN

Ecriture du code

Algorithmique

- Un algorithme est une suite finie et non-ambiguë d'instructions permettant de donner la réponse à un problème.
- Indépendant de tout langage de programmation.

Programmation

- Rédaction du code source du logiciel : "encodage" dans un langage donné des algorithmes.

En pratique, ces deux phases sont aujourd'hui très liées :
fusion des métiers "analyste" et "programmeur".

UNIVERSITÉ DE ROUEN

Phase de réalisation : Gestion de versions

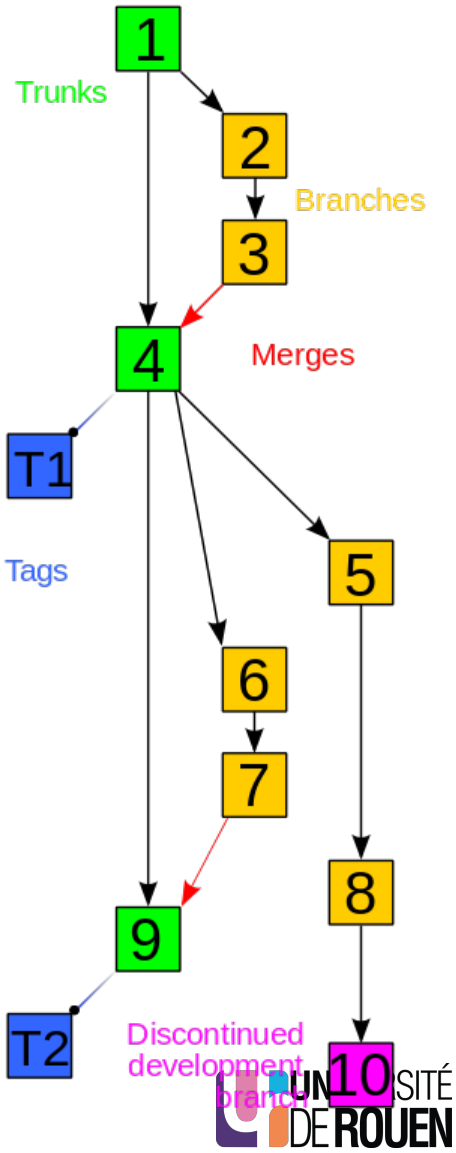
- Exemple de situation 1
  - L'équipe de développement produit des versions successives : v0.9 → v1.0 → v1.1 → ... → v1.8.2
  - Un client de la version 1.1 signale un bug et ne peux pas upgrader vers la dernière version (raisons diverses).
  - Problème : comment revenir en arrière, corriger le bug, repackager le logiciel et porter le correctif sur les versions actuelles ?
- Exemple de situation 2
  - L'équipe A fait des "*fix* " sur la base de code, qui doit rester "propre" (= build sans problèmes)
  - L'équipe B doit implémenter une nouvelle fonctionnalité
  - Problème : comment faire un sorte que le travail de B ne perturbe pas la base de code principale, tout en permettant à B de bénéficier des corrections de A ?

Solution : *Version control system* (VCS)

Outils logiciels permettant de gérer le cycle de vie d'un ensemble de sources définissant l'intégralité d'un logiciel (code source, scripts de *build*,

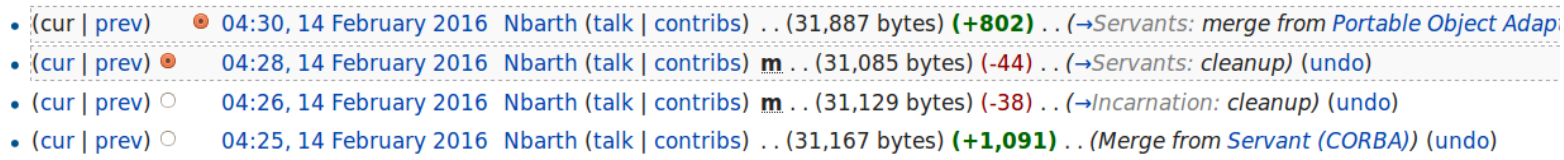
Logiciels de gestion de versions

- Permettent le travail collaboratif à distance.
- En début de journée, le développeur récupère l'état actuel des sources : opération de "mise à jour" (*update* ou *pull*).
- A chaque modification significative, le développeur procède à un *"commit"* : enregistrement dans une base de données de l'état courant des sources.
  ⇒ on peut retrouver **tous** les états intermédiaires.
- Permettent la fusion des diverses modifications :
  - Si plusieurs personnes font des modifs sur un fichier, le VCS procède à une fusion ("*merge* ") de façon transparente.
  - L'utilisateur n'est sollicité que s'il y a ambiguïté.
- Permettent la gestion de branches ("*branch* ") séparées, qui pourront à terme être fusionnées (ou pas).



Logiciel de gestion de versions

- Utilisation simplifiée et grand public de ces logiciels : CMS de type Wiki (Wikipédia ou autre)



Types de logiciels de gestion de versions

- Deux types :
  - Modèle **centralisé** : Mise en place d'un serveur de référence, les développeurs font des *checkout* et des *commit* directement sur le serveur.
  - Modèle **distribué** (*peer-to-peer*) : chaque développeur a l'intégralité de l'historique localement et peux "committer" dessus. Il peut propager (*"push"*) son dépôt aux autres et récupérer le travail des autres (*"pull"*)
- Logiciels dominants :
  - Modèle centralisé : Subversion ("svn")
  - Modèle distribué : Git (crée par les développeurs du kernel Linux)
- Ces logiciels sont souvent associés à un site web de type **forge logicielle**, offrant une vue web ainsi que des services annexes (gestion de projet, pages web, intégration continue, forum, gestionnaires de bug, etc.)

Test unitaires

- Deux objectifs :
  1. Valider que toutes les fonctions élementaires font bien ce qu'on leur demande.
  2. Vérifier à tout moment qu'un ajout de code ne "casse" pas une autre partie du programme (test de "non-regression").
- Principe : pour chaque fonction élementaire, on écrit des tests correspondants qui doit vérifier que la fonction répond à son cahier des charges.
- Exemple : soit une fonction `calcule()` qui fait des opérations arithmétiques sur des entiers, l'opération étant donnée sous forme de chaîne de caractère.
  On écrira une fonction de test avec par exemple le code suivant :

```
int sum1 = calcule("1+2");
assertEquals( 3, sum1 );
int sum2 = calcule("1-2");
assertEquals( -1, sum2 );
```

  En pratique, on se limitera aux cas limites.

TDD : *Test Driven Developement*

Le TDD est une technique de développement qui pousse ce principe à l'extrême :

1. Avant d'écrire une fonction, on écrit le test correspondant.
2. On exécute et on vérifie que le test ne passe pas (normal : la fonction n'a pas été écrite).
3. On écrit la fonction et on vérifie que le test passe.
4. On itère ensuite refactoring et tests, jusqu'à ce que les perfs soient conformes aux exigences.

Sommaire

UNIVERSITÉ DE ROUEN

Phase de livraison

- Intégration
- Validation
- Documentation logicielle
- Packaging :

- Production d'une archive type 7z ou zip contenant les binaires
- archive auto extractible, créee via des "installeurs" du type Inno Setup, NSIS
- paquet Debian (.deb)
- fichier d'installation Windows (.msi)

UNIVERSITÉ DE ROUEN

Bonne pratiques pour réussir en équipe

Dans un contexte de travail en équipe, quelques principes de base : (cf. Joël test et l'original)

1. Utiliser un système de gestion de versions
2. Faire un *build complet* en une seule étape à partir des sources
3. Faire des daily (ou nightly) builds complets, avec les tests !
4. Utiliser un gestionnaire de bugs
   (description du bug, qui l'a signalé, quelle plateforme, qui le gère, résolu ou pas, etc.)
5. Avoir un planning
6. Avoir des specifications
7. ...

Coût

Le coût pour corriger un bug est faible en début de cycle, mais augmente de façon exponentielle avec le temps.