Génie logiciel

Nicolas Bourras

September 8, 2020

Contents

0.1 Premier cours

7 septembre 2020, 10h

0.1.1 Informations utiles

Professeurs

- HAMRI Maamar El Amine, `amine.hamri@univ-amu.fr`
- MASSAT Jean luc, `jean-luc.massat@univ-amu.fr`
- YACOUB Aznam, pas de mail donné

Volume horaire

- 9 cours de 2h
- 9 TD de 2h
- 9 TP de 2h

Modalités de contrôle des connaissances

$NoteFinale = 0.6 * ExamenTerminal + 0.4 * Projet$

Avec $Proj = \max(0.3 * Projet1 + 0.7 * Projet2, Projet2)$
L'examen terminal sera en décembre et durera 3H. Les documents ne seront pas autorisés.
Les deux projets se dérouleront sur environ 3 et 7 semaines.

Projets

Les projets sont de taille conséquente, il est donc conseillé de former des équipes de 5. Les changements d'équipe seront impossibles après le début des projets.
Il est a priori possible de se mettre en équipe avec des étudiants d'autres groupes, mais à condition que l'EDT soit bon (il faudra demander au professeur).

0.1.2 Présentation générale

NOTE : regarder le poly sur ametice sera sûrement plus représentatif du cours, l'intégralité du contenu est (sera) dessus.
Le génie logiciel étudie des méthodes et outils qui permettent de développer des logiciels de taille conséquente en gardant une qualité élevée, et une maîtrise des coûts de développement et des délais. Cette science ne se concentre pas sur le code, mais sur des solutions qui vont faciliter l'implémentation.

Plan du cours

- Intérêt d'adopter un cycle de vie
- Présentation d'une méthode de developpement logiciel
- Présenter comment coder, organiser le code, intérêt des tests, ...

Constat

Pour beaucoup de projets informatiques :

- le produit ne répond pas au cahier des charges.
- le produit est livré hors délai.
- lors de la maintenance, des erreurs se manifestent à cause d'une mauvaise hygiène de code.

L'objectif est de construire des logiciels ergonomiques, fiables, évolutifs & économiquement viables.

Indicateurs utiles sur un projet

- *Volume* : le nombre d'instructions (en KLS : Kilo Lignes Sources)
- *Effort* : le temps nécessaire pour un ingénieur (en HA : Hommes Années)
- *Délai de réalisation*
- *Durée de vie* : le temps pendant lequel le logiciel devra être maintenu

La qualité logicielle

Un logiciel peut être observé selon ses qualités, qui sont :

- soit internes, comme la modularité, la maintenabilité, …
- soit externes, comme la validité, la robustesse, la performance, …

Durant le développement, on va principalement voir les qualités internes.
Les facteurs de qualité ne sont pas nécessairement compatibles 2 à 2, il faut souvent trouver un compromis.

0.1.3 Définitions

Une méthode est un ensemble de règles qui conduisent à une solution.
Une méthodologie est une agrégation de méthodes, guides, outils, techniques de différents domaines permettant de déduire la manière de résoudre un problème.
La modélisation est une activité qui précède toute décision ou formulation.
Un modèle est une interprétation de la compréhenstion d'une situation ou d'une idée de la situation.

0.1.4 Modèle de développement : le cycle de vie

Cliquer pour voir l'article sur wikipedia
La gestion de projet nécessite la modélisation du processus de developpement lui-même. C'est ce à quoi le cycle de vie sert.
Il faut savoir que ce modèle n'est pas unique, et que d'autres modèles existent comme par exemple la méthode agile, ou la méthode merise. Ces méthodes peuvent être plus adaptées à certains types de logiciels.
Plusieurs modèles de cycle de vie ont été élaborés. Tous ont en commun au minimum les 5 phases essentielles de tout développement :

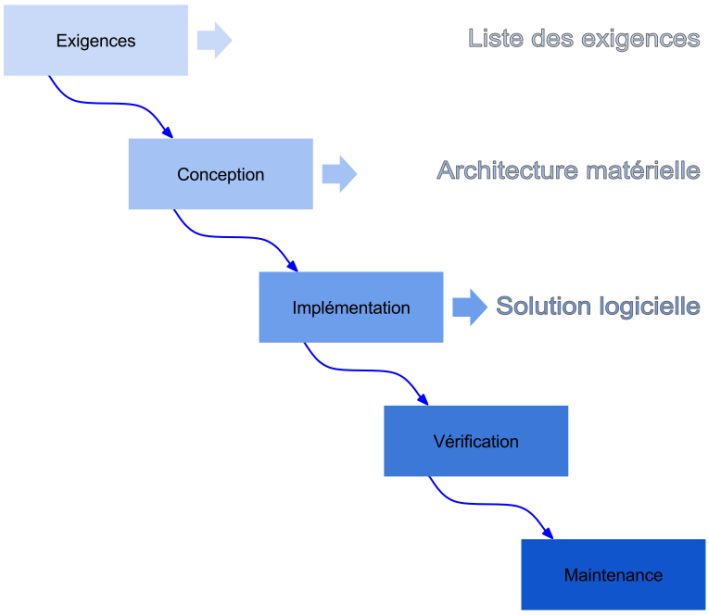
- la spécification (cahier des charges)
- la conception
- la réalisation
- les tests
- l'exploitation

Le modèle de la cascade

Ce modèle est inspiré de l'industrie du batiment, et repose sur les principes suivants :

- on ne peut pas construire la toiture avant les fondations
- les conséquences d'une modification en amont du cycle ont un impact majeur sur les coûts en aval

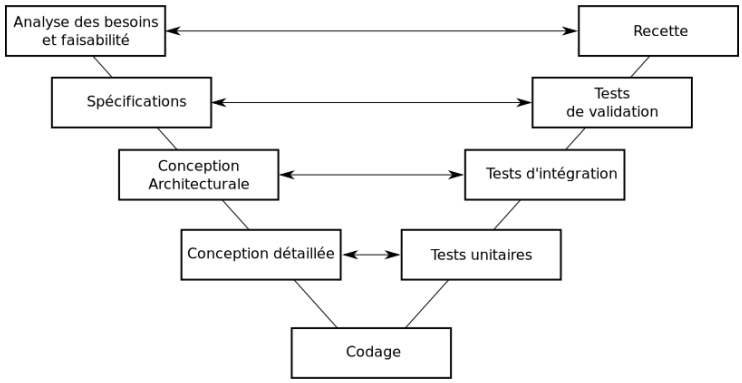
Ainsi, dans ce modèle, les tâches sont exécutées séquentiellement, et en conséquent les erreurs sont détectées tardivement (lors de la livraison du produit) et le processus pour les fixer sera donc long et coûteux.



Le modèle en V

Ce modèle, qui est encore aujourd'hui très utilisé, a été proposé pour combler les lacunes du modèle en cascade, c'est-à-dire (principalement) la tardiveté de la détection des erreurs.

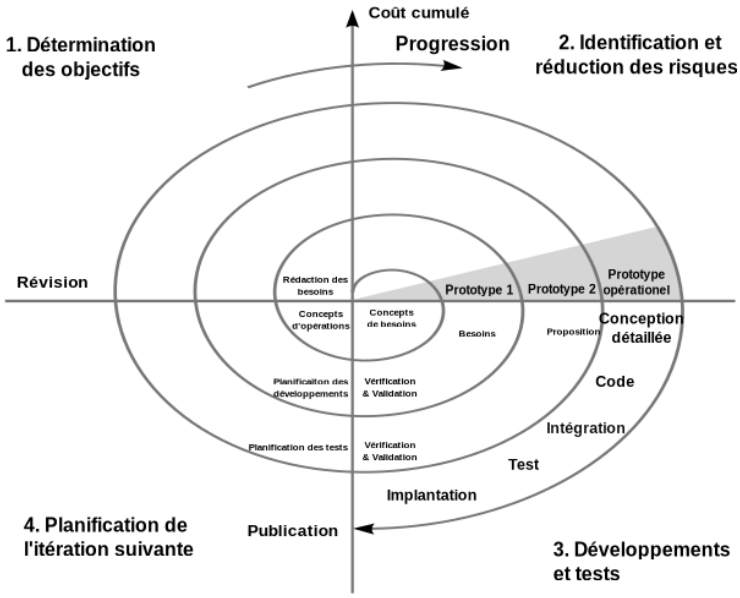
Pour cela, il définit des tests qui permettent de tester les composants indépendamment. en effet, chaque phase de développement est associée avec la phase de validation qui lui correspond.



Note : la “recette” correspond à des tests d’acceptation.
Note : C’est le modèle que l’on va adopter pour le premier projet.

Le modèle en spirale

Cette méthode reprend les différentes étapes du cycle en V. Elle est liée aux projets innovants où il est difficile de cerner les besoins du client.
Elle permet ainsi de pouvoir changer le cahier des charges plusieurs fois, ce qui serait impossible avec un modèle en spirale.



0.2 Second cours

7 septembre 2020, 14h30

0.2.1 Étude de cas

Cette partie présente une étude de cas sur un système de commande pour le chargement automatique d'un chariot.
Nous ne réalisons pour le moment que la spécification.

1ère phase : Modélisation de l'environnement

On commence par déterminer et caractériser les objets de l'environnement. On peut ensuite les classer en différentes catégories : données, activités et relations. C'est dans cette situation que l'on pourrait par exemple utiliser UML.
Dans notre exemple, on a 4 objets :

- le système de contrôle-commande (notre "programme")
- dans l'environnement:
  - l'opérateur
  - le chariot
  - le tapis chargeur

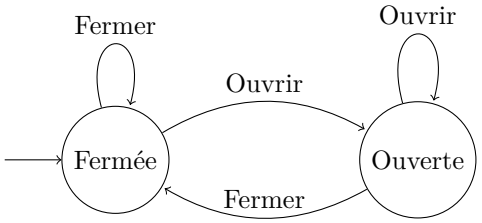
On utilise 3 variables à états discrets :

- P (position) : $P \leq P2$, $P2 < P < P1$, $P > P1$
- V (vitesse) : $+V$ (vers la droite), $-V$ (vers la gauche), 0
- T (trappe) : ouverte, fermée

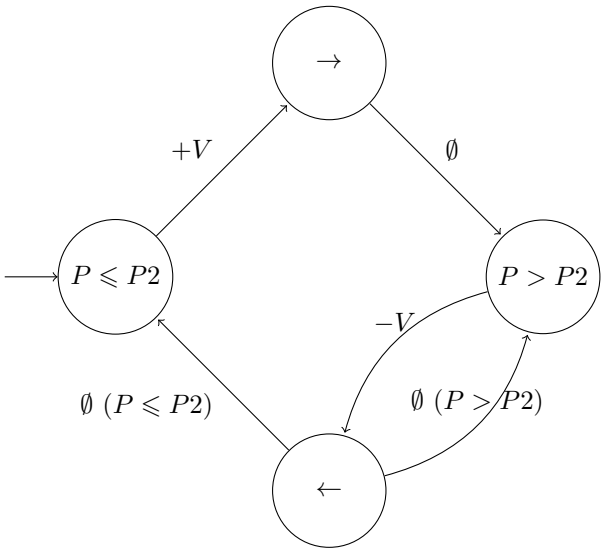
On note :

- V dépend de la commande de déplacement appliquée : Avancer, Reculer, Arrêter
- T dépend de la commande d'ouverture appliquée : Ouvrir, Fermer

On peut alors créer un automate, ici pour la trappe :



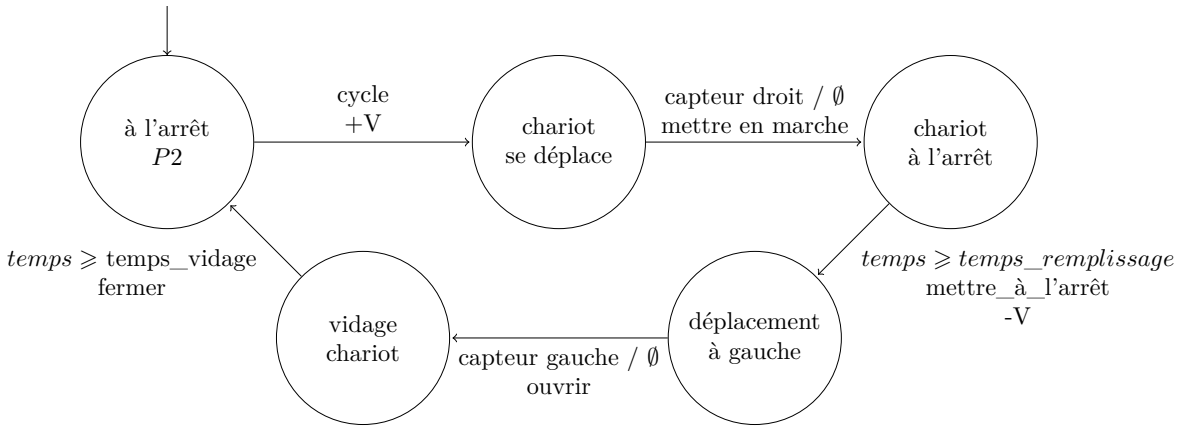
Ou alors pour la vitesse du chariot :



Avec $\leftarrow$ un déplacement à gauche, et $\rightarrow$ un déplacement à droite. Attention, on n'est pas forcément au point $P1$ à l'état $P > P2$, juste à l'arrêt à droite du point $P2$.

2e phase : détermination des entrées et des sorties

On spécifie le système qui gère les entrées et les sorties de chaque objet, c'est-à-dire les commandes que l'on envoie aux objets.



On suppose qu'à l'état initial le chariot est à l'arrêt en P2, la trappe est fermée et le tapis chargeur est à l'arrêt. Il faudra compléter ce modèle par l'arrêt d'urgence.