# Notes de cours de Génie Logiciel

Yann Miguel

28 septembre 2020

## Table des matières

# 1 Introduction

## 1.1 Problèmes

L'objectif du génie logiciel est de répondre à deux problèmes principaux des projets en entreprise:

1. Comment apsser du cahier des charges au code.
2. Comment maintenir le logiciel une fois commercialisé.

En effet, il faut du temps de réflexion pour passer du cahier des charges, décrivant comment le logiciel doit fonctionner, à un logiciel qui correspond à ce cahier, et qui maintenable afin de ne pas avoir à complètement le recoder à chaque changement de système ou chaque mise à jour des machines du client.

## 1.2 Vocabulaire

- Le volume, cela correspond au nombre de lignes de code source du logiciel.
- L'effort, cela correspond au nombre d'ingénieurs mobilisé sur une unité de temps pour résoudre le problème.
- Le délai, cela correspond au délai imposé par le client ou par l'entreprise pour compléter le projet et le logiciel.
- La durée de vie, cela correspond à la durée de temps pendant laquelle le logiciel sera maintenu par les codeurs.

## 1.3 Unités

HA = Hommes par An.
KLS = KiloLineSource( millier de lignes de code source).

# 2 Cours 1

Un mauvais développement peux créer des erreures graves, mais même le plus parfait des développements ne peux être exempt d'erreurs.

Les principales sources d'erreurs sont:

- Les erreurs humaines,
- Les problèmes trop complexes,
- Les demandes du client, tel que le rajout de fonctionnalités en plein milieu du projet.

## 2.1 Qualités logicielles

### 2.1.1 Les qualités externes

Les qualités externes du logiciel sont des qualités remarquables par tout le monde, même les non programmeurs. Par exemple:

- validité, n'importe qui peux remarquer, et même des fois assez vite, si le code n'est pas valide, car il ne répondra donc pas aux besoins.
- robustesse, si le code n'est pas robuste, et qu'il casse, les utilisateurs s'en rendront compte facilement.
- performance, un programme qui lag n'est pas vraiment génial à utiliser.
- ergonomie, si l'interface n'est aps ergonomique, cela va ruiner l'expérience de l'utilisateur.

### 2.1.2 Les qualités internes

Les qualités internes sont des qualités de code remarquées principalement par des développeurs. Par exemple:

- modulabilité, si le code n'est pas miodulable, rajouter de nouvelles fonctionnalités sera une horreur.
- lisibilité, le debuggage sera presque impossible si le code est illisible.
- maintenabilité, si on ne peux aps facilement débugger le code, il va souvent crasher, et les utilisateurs ne seront pas content.

Tout ces facteurs n'étant pas forcément compatibles entre eux, il faudra compromiser, voir en prioriser certains.

Quand on design un projet, on doit satisfaire les critères CQFD, mais les facteurs coût et délai de développement peuvent être limitants.

## 2.2 Le cycle de vie

La gestion du projet est une part importante du développement logiciel, car elle permet de bien séparer les tâches, et d'organiser le projet.

### 2.2.1 Exemples de modèles de gestion

Le modèle en V a été conçu pour corriger la lacune principale du modèle en cascade, qui est le test en fin de projet.

Dans le modèle en V, chaque composante a ses tests et sa réalisation fait en parallèle, ce qui permet d'identifier plus facilement les sources d'erreurs.

Le modèle en spirale est surtout utilisé pour les projets dits innovants, c'est à dire les projets où le client ne sait pas précisément ce qu'il veut. Il consiste à présenter de multiples prototypes au client, et à ne continuer que avec ceux approuvés, ce jusqu'à réalisation du produit final.

## 2.3 Méthodes et modèles

Un bon modèle se doit d'être abstrait, raffiné et lisible.
Les modèles les plus utilisés en informatique sont les modèles conceptuels, qui se basent sur l'utilisation de symboles pour représenter des aspects qualitatifs.

# 3 Cours 2

## 3.1 Méthodes et modèles(suite)

On doit d'abord modéliser l'environnement du projet, c'est à dire les objets et leurs états.

En utilisant des langages comme l'UML, on peux créer une machine à états qui va décrire le comportement des objets en fonction des commandes qui leur sont envoyées.

On peux imaginer une machine à états comme un automate, dont les commandes sont les transition, et dont l'état par défault est l'état initial.

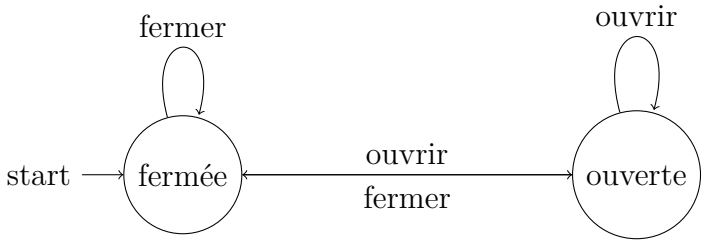


FIGURE 1 – Exemple de modélisation d'une trappe.

Si il y a plus d'une donnée, l'état devra être décrit par l'ensemble des données, ce qui est plus complexe que dans l'exemple de la figure 1.

Il faut aussi, à travers le modèle, prévoir les cas où il ne se passe rien. Per exemple, si un chariot se trouve à sa position la plus à gauche, et qu'on lui demande d'aller à gauche, il doit rester immobile.

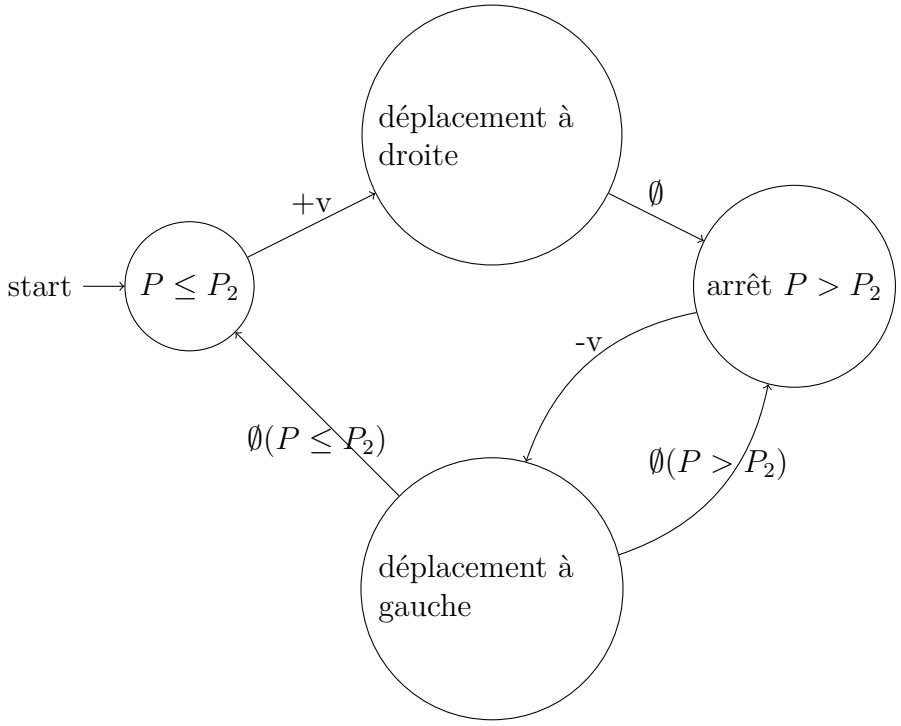


FIGURE 2 – Modèle du déplacement du chariot.

### 3.1.1 Système de contrôle

Le système de contrôle est me système qui lie tout les objets dans un environnement.

L'opérateur (humain) ordonne le démarrage, ou l'arrêt d'urgence, du système, et le système ordonne aux objets d'agir en conséquence.

Le système peux connaitre, i.e. stocker, des informations sur les objets qui reçoivent ses ordres.

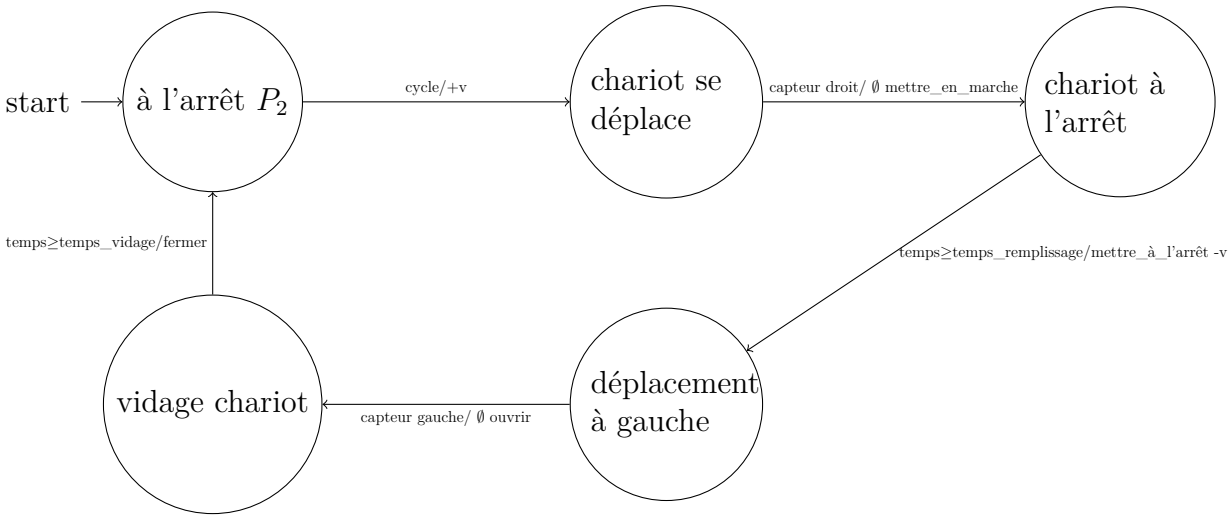


FIGURE 3 – Modèle d'un système de contrôle.

# 4 Cours 3

## 4.1 Méthodes et modèles (suite)

Certaines grosses entreprises utilisent les cycles de développements classiques, comme le cycle en V, alors que d'autres ont fabriqués leurs propres cycles.

Le but de la spécification est de comprendre les besoins du client.

Il est primordial que les modèles designés soient déterministes, et les objets non intelligents ne doivent rien avoir d'automatisé, car ils ne doivent pas se gérer tout seuls.

## 4.2 L'UML

L'UML est une boîte à outils de développeur logiciel.

Le premier diagramme a développer ets le diagramme de cas d'utilisation, il exprime une liste de fonctions décrites par du texte.

# 5 Cours 4

## 5.1 Diagramme de classe

La classe est représentée par un rectangle qui contient le nom. Il peux aussi contenir des méthodes et variables, mais cela est inutile pour la phase de spécification.

Il est possible tracer un trait entre deux classes pour les associer. Sur le trait, on peut mettre des valeurs de cette forme:

- rien, un seul
- * de 0 à $+\infty$
- 0..n, de 0 à n

Il est possible, et conseillé, de marquer sous la valeur ce que ça représente. On peux aussi marquer sur le trait l'action que fait la valeur.

L'agrégation indique que l'objet duquel par la flèche est identifiable par l'objet qui reçoit la flèche, mais l'inverse n'est pas vrai.

Dans la composition, la destruction de l'élement qui reçoit la flèche détruit les élements desquels partent la flèche. L'inverse n'est pas vrai.

# 6 Cours 5

Il y a aussi un diagramme pour des packages. Il représente le liens entre les packages.

L'UML sert donc à décrire intégralement le projet, et les interaction entre ses éléments.

# 7 Informations importantes

Il y aura un examen final sans documents qui vaudra 60% de la note finale.
Il y aura deux projets en équipe valant en tout 40% de la note finale.
Donc, $NF = 0.6 * ET + 0.4 * Projets$.
La note de projet sera calculée ainsi: $max(0.3 * projet1 + 0.7 * projet2, projet2)$, ce qui veut dire que le premier projet peux ne pas compter si il abaisse trop la note finale.
Les équipes pour le projet seront normalement composées de 5 élèves.
Les polys de cours et les infos importantes sont présentes sur la page Amétice du cours.