

Projet 3A robotique

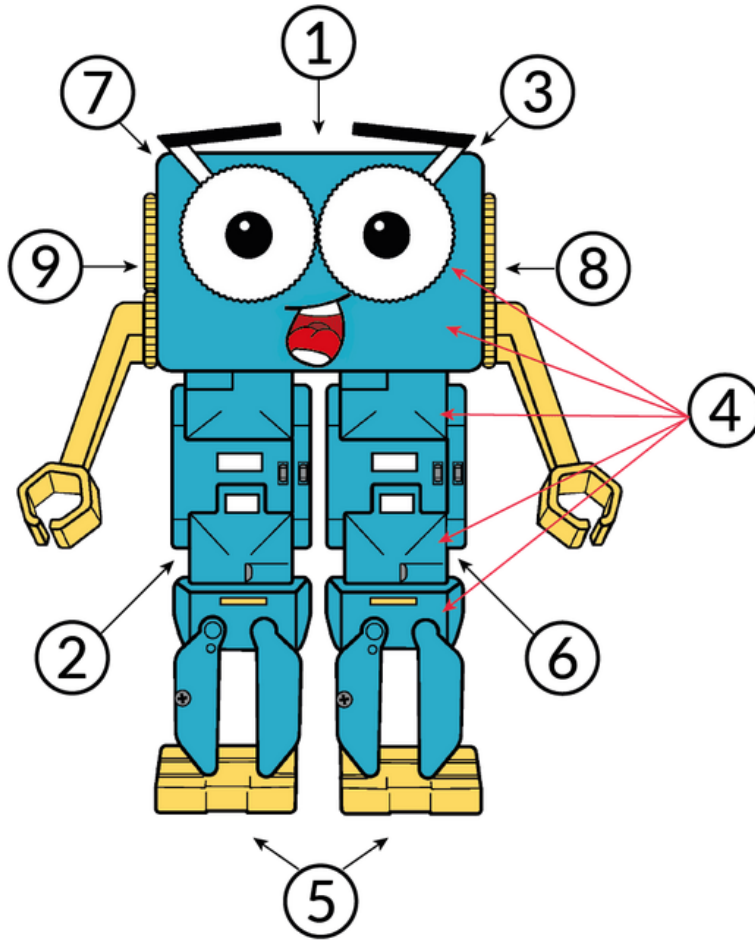
Hermine CHATOUX

2023-2024



Autres encadrants du projet : Quentin CHASSEL, Duncan LUGUERN, Meldrick REIMMER

Remerciement : Bart HEYRMAN



1 - Humanoid Form

Marty has a personality and is full of character!

2 - Unique Walking Mechanism

Walk, turn, dance, sidestep, kick a ball, wiggle

3 - Range of Expressions

Marty's eyebrows move to express emotions

4 - Motors with Position Sensors

Nine metal-g geared smart servo motors (in legs, arms & eyes)

5 - Foot Sensors

Infrared (IR) Sensor & Color Sensor for screenless coding

6 - Quality Moulded Plastic Parts

Classroom-ready, robust and built to last

7 - Acceleration & Tilt Sensor

Found in the control board in Marty's head

8 - Rechargeable Battery

With run time of 2-3 hours when fully charged

9 - Speaker

Marty speaks and plays sounds

Robot éducationnel
avec différents niveaux de difficulté

<https://robotical.io/about/all-about-marty/>

- Initiation au projet informatique avec cahier des charges succinct
- 26 h de projet par étudiant
- Groupe de 3 personnes, 4 si nécessaire
- Triple évaluation
 - Mise en place de la gestion de projet
 - Etat d'avancement
 - Démonstration finale



Principaux objectifs



- Du point de vue technique, commande à distance de deux robots par liaison wifi et résolution d'un labyrinthe
- Du point de vue gestion de projet, initiation à la programmation en groupe, partage des tâches, planification du travail...

1. Un peu de technique
2. Gestion de projet
3. Gestion de code
4. Résumé



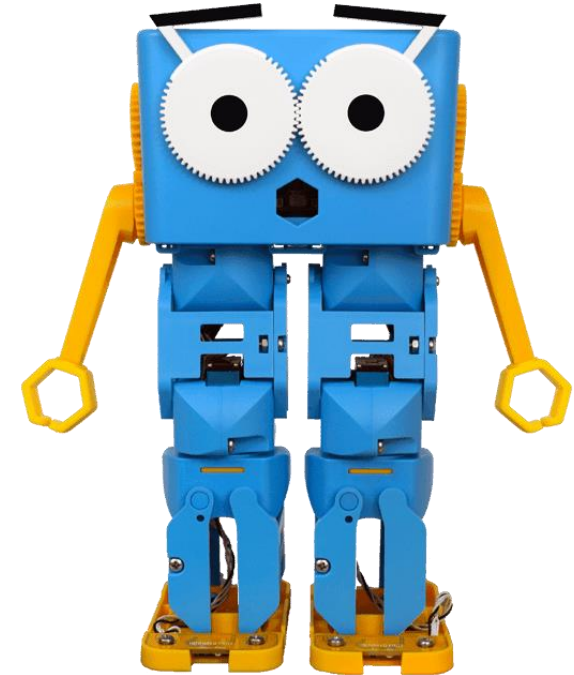
1. Familiarisation avec l'environnement de développement
2. Partie 1
 1. Compréhension des documentations techniques sur le robot
 2. Mise en œuvre du protocole de communication du robot
 3. Réalisation d'une application de contrôle du robot
3. Partie 2
 1. Communication en séquentiel de deux robots
 2. Résolution de labyrinthe via le capteur couleur
 3. Découverte de la phrase mystère grâce à la lecture de QRcode



- Langage **python**
- Librairies installées sur les machines des salles
 - Marty (https://userguides.robotical.io/martyv2/documentation/python_function_reference)
 - **Pyqt6** (<https://www.pythonguis.com/tutorials/pyqt6-creating-your-first-window/>)
 - Opencv (<https://docs.opencv.org/4.5.0/index.html>)
 - Qreader (<https://pypi.org/project/qreader/>)

En gras les langages et librairies obligatoires !

- Robot humanoïde (très mignon)
- Interface de communication via **Wi-Fi** ou Bluetooth
- Mobilité jambes, bras et yeux
- Plusieurs capteurs
 - Couleur
 - Distance
 - Obstacle
 - Niveau de batterie
- Ainsi qu'un caméra pour du traitement d'images durant le projet



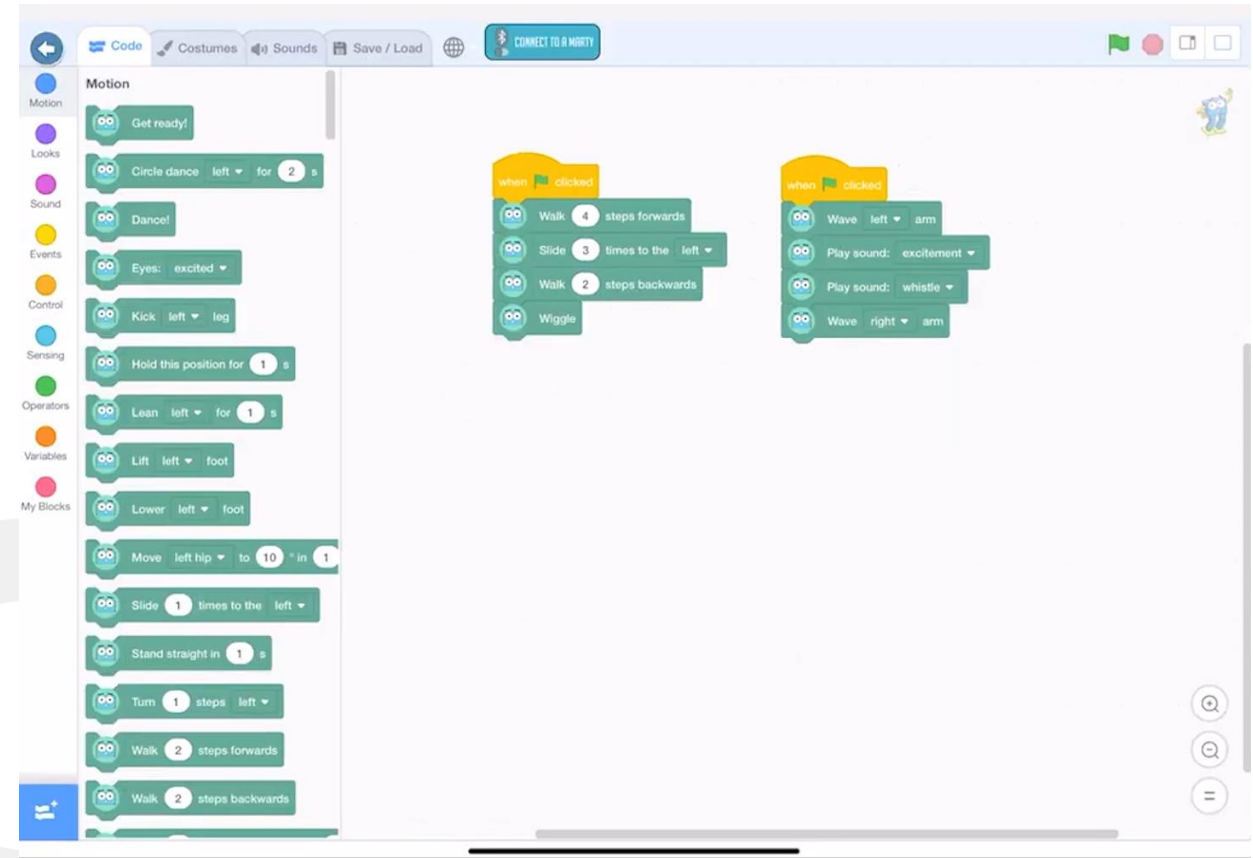
En gras le protocole obligatoire !

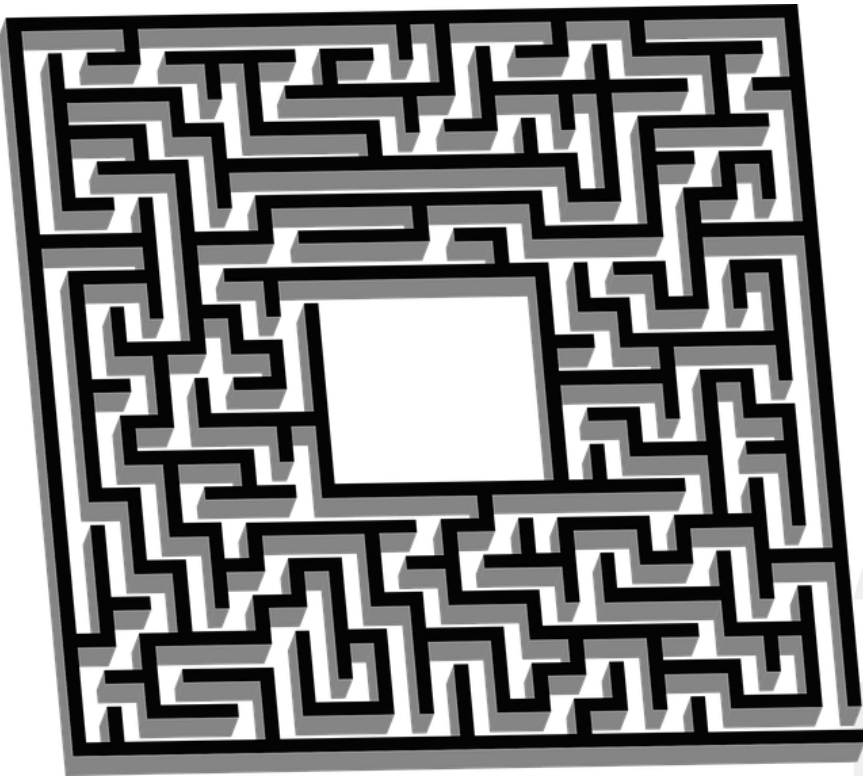
- Connexion/déconnexion au robot via le protocole de communication
- Récupération et affichage des infos des capteurs sur l'interface
- Pilotage du déplacement du robot par appui sur des boutons et par les touches du clavier
- Récupération et affichage de l'image de la webcam



Pour les groupes de 4 personnes des fonctionnalités supplémentaires sont attendues :

- Un nouvel onglet permettant de faire une liste d'instruction à donner à Marty qu'il exécutera à la suite lors du lancement de l'ensemble d'instruction
- Pilotage par manette de jeux ou autres périphériques + animation (vibration/lumière) de la manette

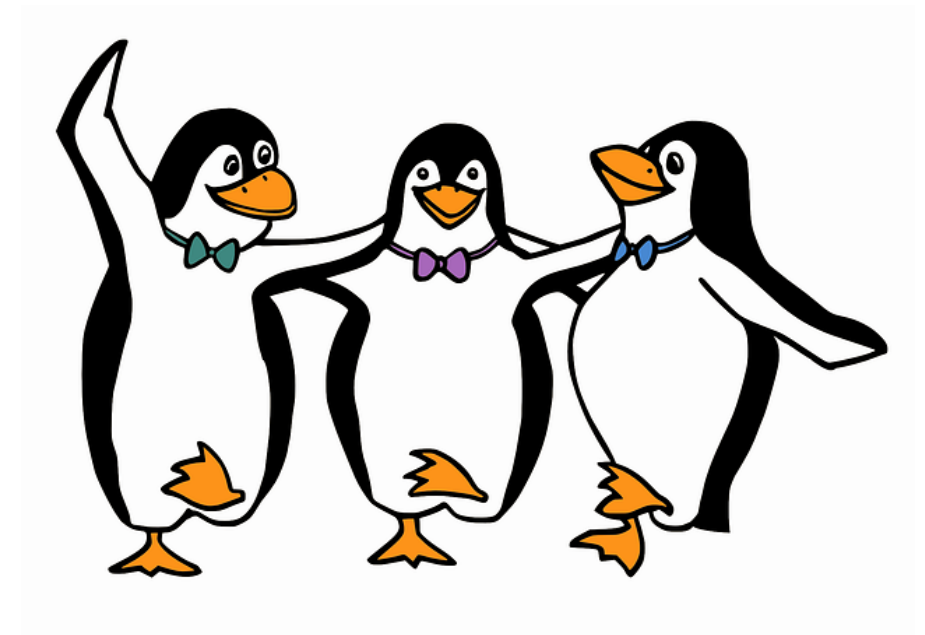




- Utilisation du capteur couleur pour déterminer les directions à suivre pour retrouver le deuxième robot au centre du labyrinthe
- Récupérer les informations contenues dans les QRcode au sol lors du déplacement grâce à la flux vidéo
- Attention, le déplacement des robots doit être séquentiel (d'un couleur à l'autre) pour découvrir le bon message

Pour les groupes de 4 personnes des fonctionnalités supplémentaires sont attendus :


- Synchroniser les robots pour présenter une chorégraphie



- Projet informatique : ensemble de tâches complexes en particulier lorsqu'il y a une équipe de plusieurs développeurs (ici 3 ou 4)
- Problèmes de définition/compréhension du travail à faire, de la répartition des tâches entre les développeurs...
- Problèmes de gestion du code source entre les développeurs : écriture du code, ajout de fonctionnalités, diffusion des modifications/corrections, retour en arrière...

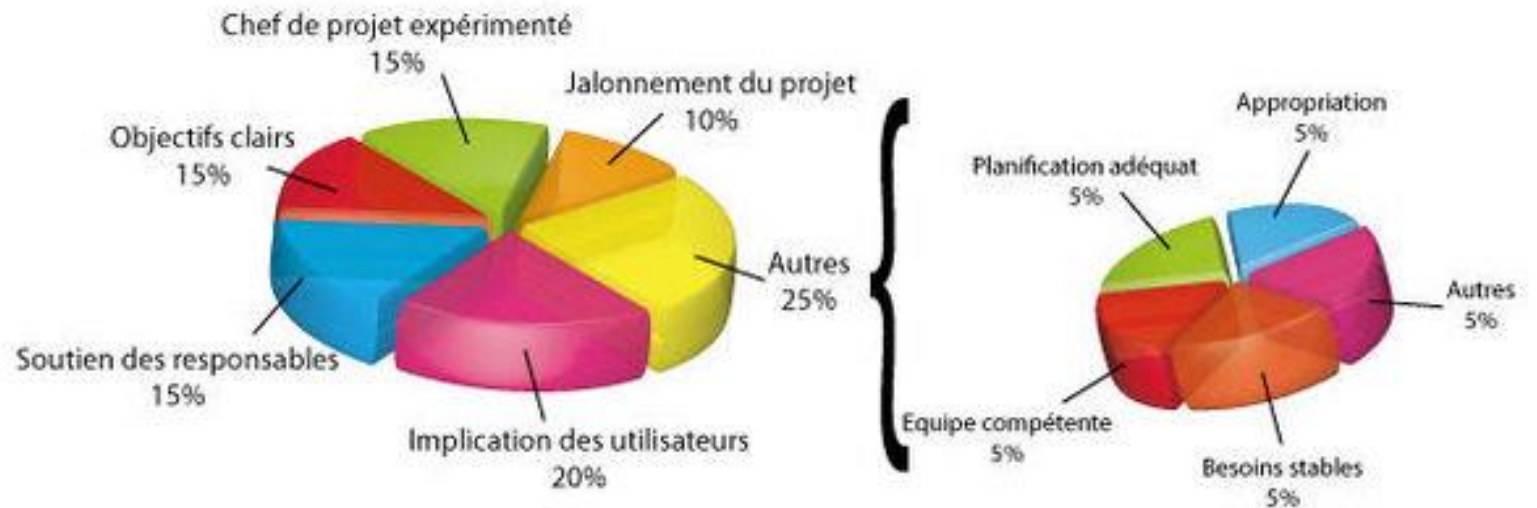
Un outil à utiliser :

- **Git**
- [Tutoriel gestion de projet](#)
- [Tutoriel gestion de code](#)

- Analyse fonctionnelle et définition des objectifs
 - Conception détaillée (découpage, planification, répartition)
 - Développement
 - Test
 - Recette (validation) et mise en production
 - Maintenance
- 
- Cahier des charges
 - Diagramme de Gantt
 - Cahier de recettage
 - Cahiers de test
 - Livrable
 - Mise à jour

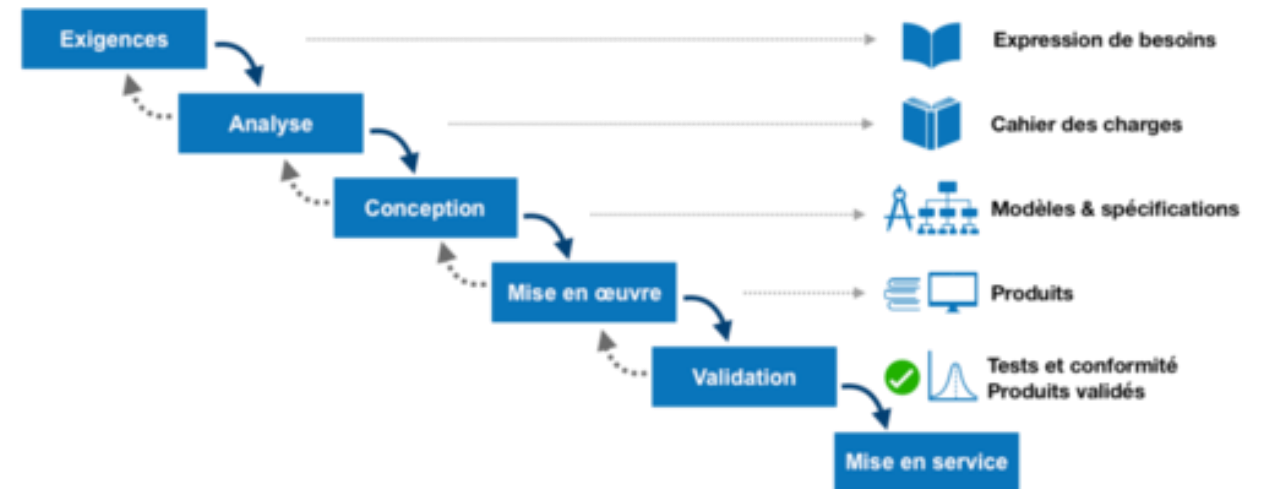
- **Cascade**
 - Traditionnel
- **Cycle en V**
 - « évolution » du cascade
- **Spirale ou itératif**
 - Partir d'une base et itérer
- **AGILE**
 - Croisement

Facteurs de succès des projets



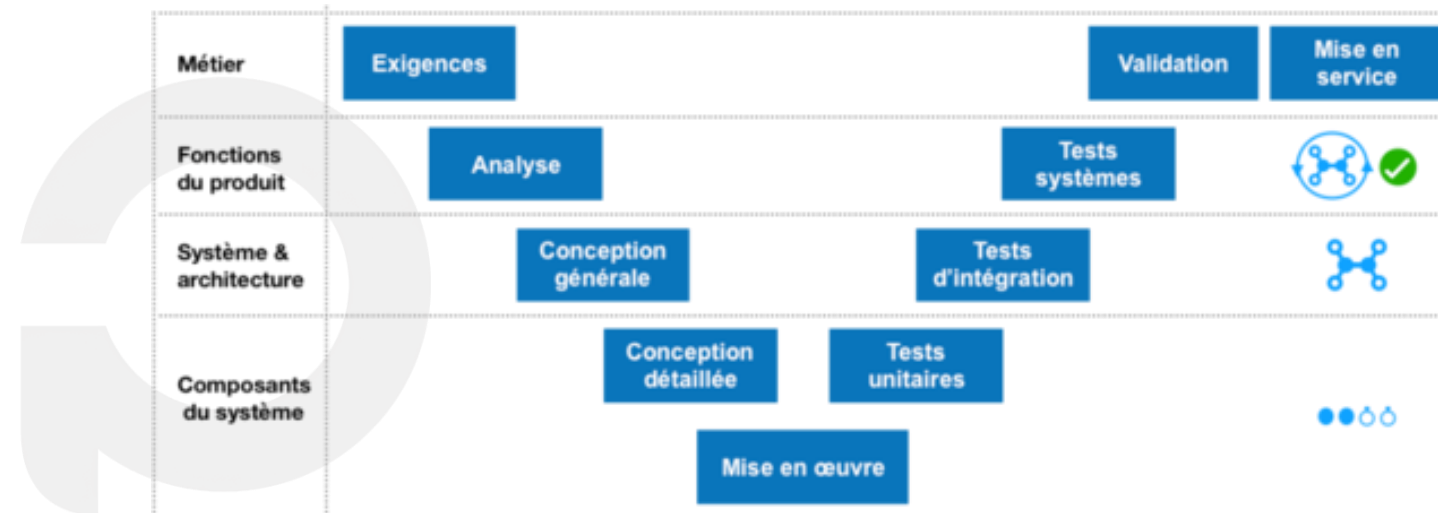
Méthode cascade

- Une étape après l'autre
- Chaque rôle est défini, **compartimenté**
- Rigide
- **Effet tunnel**

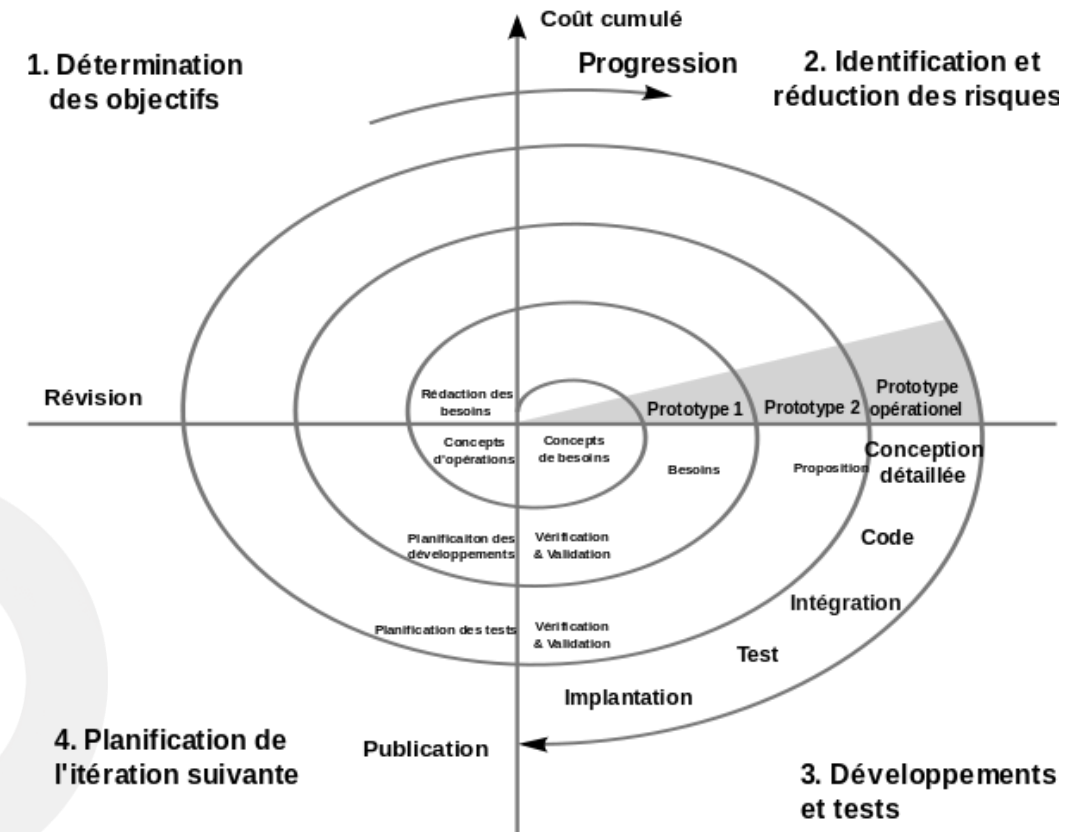


Modèle en V

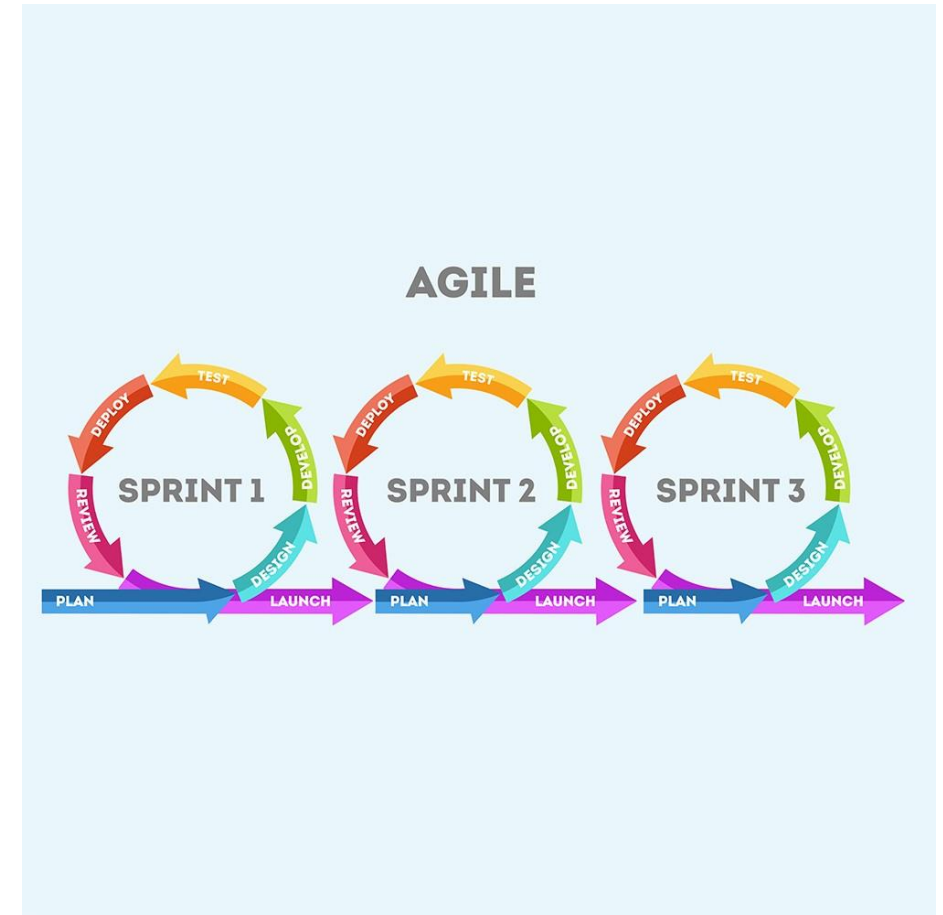
- Validation grâce à une double communication
- Moins d'effet tunnel
- Mais gestion toujours globale



- On reprend le cycle en V
 - Mais avec de courtes versions
- **Travail par itérations**
 - A chaque cycle
 - Rédaction des besoins
 - Conception
 - Vérification et validation
 - Et on planifie le prochain
- Limite les risques
 - **A chaque itération, le client voit le produit**
 - Reste quelques soucis
 - Suivi au fil des versions



- Privilégie le dialogue entre tous les acteurs (clients, développeurs...)
- Se base généralement sur un cycle proche de la spirale
 - **Cycles courts**
 - Documents de suivi rapides et efficaces
 - **Implication de chaque partie à quasiment chaque niveau**
- Devenu la référence entre 2010 et 2020
- Se dérive en différentes méthodes
 - [Scrum](#), [eXtreme Programming](#), etc.



Egoless programming

- **Peer review, Pair programming**
- Avantages
 - Plusieurs points de vue pour les tâches complexes
 - Communication renforcée
 - Approche liée à l'entreprise et non à soi
- Désavantages
 - Peut devenir plus complexe
 - Tensions possibles
 - Repose fortement sur la communication



```
src/doctree.py
39 +         out[item.name] = indent_char*times_indented + item.name + doc
40 +         else:
41 +             my_dirs_docs[item.name] = item.name + doc
42 +         for o in out:
```

 Tadaboddy a day ago Owner + 😊 ...

Suggested change Beta ⓘ [Give us feedback](#)

```
42 -         for o in out:
42 +         for key,val in out.items():
```

or

Suggested change Beta ⓘ [Give us feedback](#)

```
42 -         for o in out:
42 +         for value in out.values():
43 +             print(value)
```

Resolve conversation

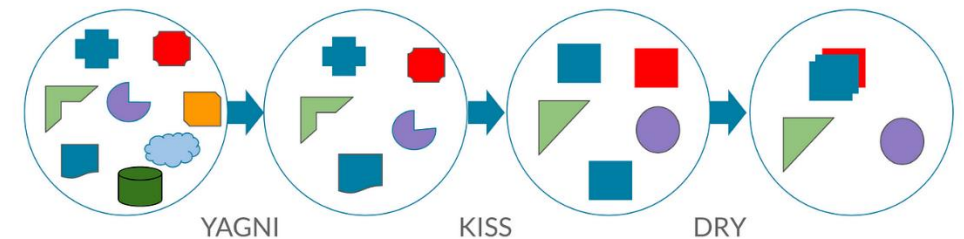
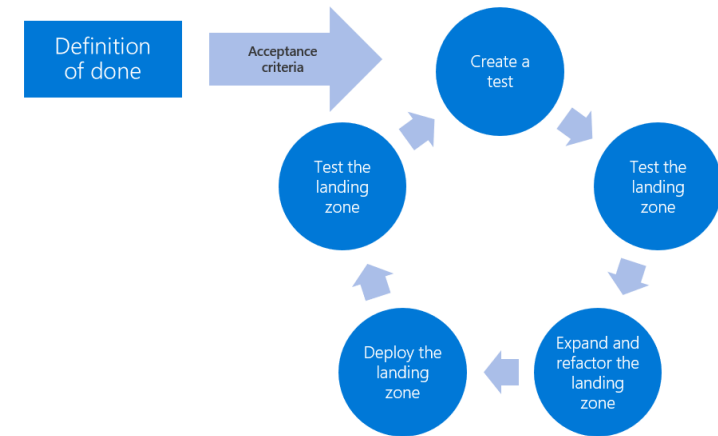
Gestion des tâches dans le temps

- Kanban
- Poker planning



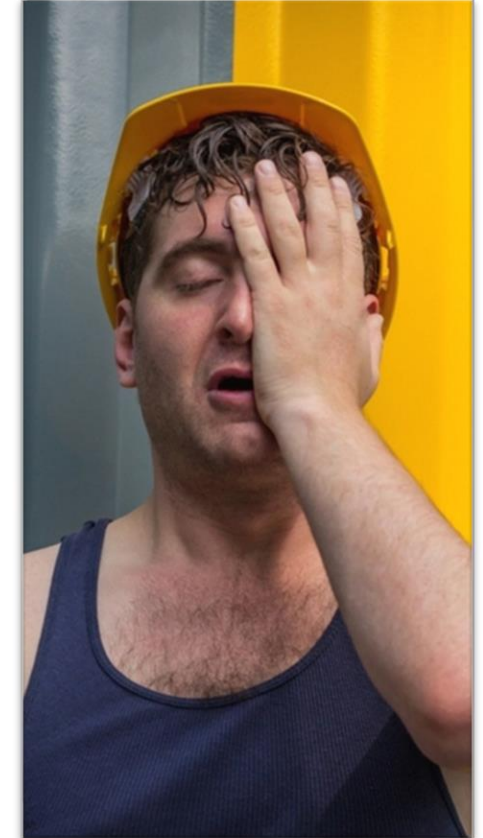
Optimisation de code

- Test-driven development ([TDD](#))
- Don't Repeat Yourself (DRY)
- Keep It Simple, Stupid (KISS)
- You aren't gonna need it (YAGNI)



Top 5 des pires phrases prononcées par des élèves ingénieurs en prog.

- “Monsieur, mon code était sur le PC hier et aujourd’hui, il n’y a plus rien !”
- “C’est quoi le dossier “backup1234” ?”
- “J’ai écrit plein (trop) de code. Ca pourra servir un jour !”
- “Tout ce qui ne compile pas, je l’ai mis en commentaire ! ”
- “Je vous jure Monsieur ! Mon programme marchait hier ! J’ai rien changé (ou presque rien) et ça plante aujourd’hui !”



- Indispensable pour le travail en équipe
- Applicable à tout type de code source : app, script, web...
- Historique de toutes les opérations
- Retours en arrière et corrections toujours possibles
- Travaux en parallèle sur plusieurs branches

Cf cours de Dominique Ginhac



- Initiation au projet informatique avec cahier des charges succinct
 - Gestion de version de code avec **Git**
 - Gestion de projet avec **Git**
 - Création d'un code fonctionnel à plusieurs robots avec **Python**
- Plus de travail attendu pour les groupes de 4
- Triple évaluation
 - T0+2 h (mise en place de la gestion de projet)
 - T0+13 h (état d'avancement)
 - T0+26 h (démon)
 - Grilles d'évaluation à venir

**Prenez le temps de
vous former avant de
vous jeter dans le
code !**

