CHAPITRE 4A – MÉTHODOLGIE ET GESTION DE VERSIONS

Étapes du Développement Logiciel

1 Pourquoi une méthodologie de développement ?

Lorsqu'on vous demandait, dans vos premiers cours de programmation, d'écrire un petit programme, il est fort probable que vous ayez effectué les tâches suivantes :

1. Vous avez lu l'énoncé, consolider les données et déterminer les tâches à accomplir.
2. Puis, vous avez déterminé une stratégie pour résoudre le problème par un algorithme et avec l'organisation des données avec un diagramme UML, …
3. Vous avez ensuite écrit le code, idéalement à partir de l'algorithme et des diagrammes UML.
4. Finalement, vous avez vérifié les résultats du programme en comparant avec les vecteurs de test que vous aviez préparés.

… du moins, en théorie.

En pratique, il est plus courant de voir les étudiants se lancer directement à l'étape 3, sans mettre trop d'efforts sur les autres étapes.

Pour les petits programmes, il est possible de bien s'en tirer de cette façon. Mais, lorsque la complexité augmente, et que le nombre de personnes impliquées dans un projet grossi, il est important de se donner des outils pour nous permettre de :

- **systématiser les étapes d'analyse et conception** (Cours *Méthodes de conception d'applications I : 420-G14-RO*)
- assurer l'**intégrité des fichiers sources** avec un système de gestion de versions comme GIT (voir *Chapitre 4C*)
- **organiser le travail à accomplir** pour être efficace et maintenir une progression constante en utilisant une méthode agile (voir *Chapitre 4B*).

2 Étapes de développement d'un logiciel

En plus des étapes décrites à la section précédente (analyse, conception, implémentation et validation), il faut ajouter le déploiement et la maintenance.

Voici de façon un peu plus détaillée, les différentes étapes impliquées dans le développement logiciel.

- Analyse
  - Identification des besoins
  - Estimés du temps développement
  - Études des risques et de faisabilité
  - Choix des outils
  - Recherche de solutions existantes
- Conception
  - Algorithme et diagrammes UML
- Implémentation
  - Codage
- Vérification
  - Tests unitaires (sur les classes) et fonctionnels (sur le programme entier)
- Déploiement
  - Documentation
  - Développement de programme d'installation dans son environnement d'utilisation
- Maintenance
  - Adaptation du logiciel aux demandes des utilisateurs et aux changements technologiques
  - Corrections des bugs

3 Types de développement

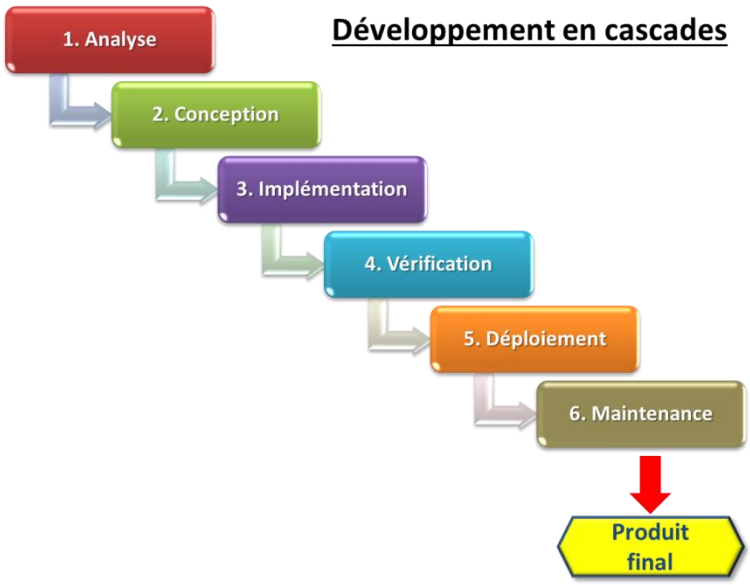
3.1 Types de développements

Il existe deux types de cheminements que l'on peut suivre pour effectuer les différentes étapes du développement logiciel : développement en cascades et développement itératif.

3.2 Méthodologie en cascade (à ne pas utiliser …)

Dans la section précédente, nous avons vu la liste des étapes à suivre pour concevoir un programme. Pour un petit programme, ces étapes étaient suivies séquentiellement de façon très intuitive. Cette stratégie était suffisante pour accomplir ce qu'il y avait à accomplir.

Cette façon de travailler s'appelle un **cheminement en cascades ou séquentiel**, où chacune des étapes énumérées ci-contre doit être complétée avant de passer à la suivante

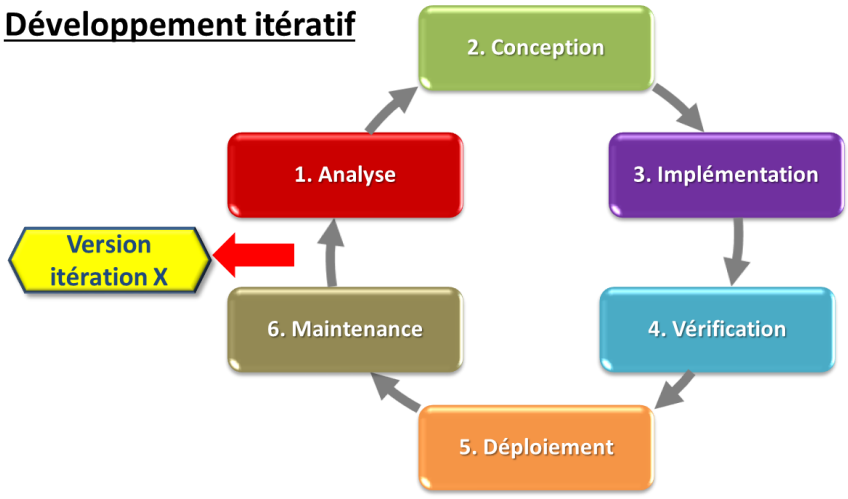


Pour les plus gros projets, beaucoup d'efforts sont mis dans les phases initiales dans le but d'avoir des spécifications solides et une architecture de logiciel stable avant de commencer l'implémentation du code.

Malheureusement, comme vous allez le constater dans votre vie de programmeurs professionnels, les besoins changent tellement rapidement que lorsque vous serez rendu à l'étape de la vérification, votre programme ne correspondra que très peu aux nouvelles exigences. Il est fort probable que vous aurez mis beaucoup de temps à établir les exigences et à concevoir du code pour des fonctionnalités qui ne correspondent plus à exactement à ce que le client avait vraiment besoin.

3.3 Méthodologie itérative (à utiliser)

Pour mieux s'adapter à la réalité des besoins changeants des clients, il existe une approche plus appropriée, soit le cheminement itératif.



Avec le **cheminement itératif**, on sépare la durée du projet en une série de courtes itérations de durée fixe dans lesquelles toutes les phases du développement sont abordées en parallèle. À la fin de chaque itération, une nouvelle version du logiciel fonctionnelle et testée doit être complètement terminée. On obtient donc, une série de versions de logiciels, qui contiennent de plus en plus de fonctionnalités et qui s'approchent de plus en plus de la version finale. On parle alors de stratégie de **développement itératif et incrémental** ou **itératif et évolutif**.

On commence donc, par concevoir, implémenter et tester les fonctionnalités les plus importantes. Puis, lorsque les exigences pour des fonctionnalités secondaires changeront ou apparaitront (comme c'est garanti qu'elles vont le faire…), elles seront adressées dans une des prochaines itérations. L'essentiel du logiciel sera présent, et on n'aura pas perdu de temps à concevoir des aspects secondaires de l'utilitaire mal spécifiées ou simplement inutiles.

3.4 Méthodes éprouvées de conception logicielles

Comme nous allons le voir plus tard dans le cours (Chapitre4B), il existe plusieurs méthodologies éprouvées pour concevoir des logiciels. Toutes ces méthodes utilisent le développement itératif.

Parmi les méthodes les plus connus, il y a la méthode du **processus unifié** qui utilise la notation UML pour la conception de logiciels. Il existe également des approches complémentaires qui permettent de réaliser un logiciel le plus rapidement possible, et ce, avec la meilleure qualité possible. Parmi ces approches on retrouve :

- **la programmation extrême** (Travail en pair ou en binôme avec révision respective du code de l'autre)
- **le développement agile** (Philosophie de travail qui mise avant tout sur la simplicité et l'efficacité)

- **le développement Scrum** (Réalisation de rencontres quotidienne avec objectifs de courte durée à réaliser pour la prochaine réunion)
- …