Conduite de projet
Chapitre 2

D. Leclet
Module M1 Miage et Informatique
Département Informatique

Les méthodes de conduite de projets

I - Introduction

- Les seules véritables causes d'échec d'un projet, résident dans l'incapacité à communiquer entre les partenaires. Pour bon nombre de personnes, la réussite d'un projet réside dans le bon choix des méthodes de conception et de développement. Ainsi, on peut dire qu'une méthode regroupe certains "**ingrédients**".
  - Un **langage commun**, plus compliqué en multimédia, car les partenaires viennent toujours d'univers différents.
  - Un **formalisme autorisant** des représentations des données et des traitements.
  - Une **progression itérative** dans l'analyse (processus structuré en phases).
  - Une **démarche d'analyse globale** indépendante de l'environnement machine ou réseau.
- De plus, quelle que soit la méthode retenue, une série de termes communs se retrouvent employés avec des sens voisins. Il est donc important d'avoir une **définition commune**.
  - ***Tâches*** (ensemble de travaux confiés à une personne pouvant se définir par un résultat contrôlable, une date de début, une date de fin).
  - ***Phase*** (états successifs du développement du produit).
  - ***Procédure*** (activité formelle et documentée effectuée dans les différentes phases du projet de développement).
  - ***Lot*** (regroupement de tâches au sein d'une phase).
  - ***Planification*** (action d'ordonnancement des tâches à l'intérieur d'une phase en fonction des ressources disponibles).
  - ***Estimation*** (action d'évaluer une charge, un délai, un coût).
  - ***Suivi*** (actions entreprises pour surveiller le bon déroulement du processus de fabrication lors de chacune des phases).
  - ***Ingénierie de projet*** (il s'agit de regrouper dans un même ensemble, l'étude globale d'un projet multimédia, sous ses aspects techniques, économiques, financiers, sociaux, culturels, éducatifs).
- De plus, après avoir choisi la méthode, il est important d'utiliser des moyens automatisés pour supporter le projet. Un problème courant, réside dans le fait que plusieurs organisations choisissent en premier les moyens automatisés, et ensuite, dépensent beaucoup d'énergies à essayer d'appliquer ces moyens à la méthode ou d'adapter la méthode aux moyens. Un autre point

sur lequel, il est important de s'attarder : c'est la performance de la méthode. Elle peut se définir autour **différents critères**.

- ***Applicabilité*** (adéquation de la méthode de conception avec le problème posé). Cependant, les principales méthodes d'analyse informatiques (Merise, SADT, descendante, SDM/S, MCP, Method ONE, …) ne sont pas entièrement transposables au multimédia.
- ***Souplesse*** (la méthode permet-elle de dégager plusieurs solutions ?).
- ***Facilité de mise en œuvre*** (délais et moyens important sont-ils requis pour rendre la méthode opérationnelle ?).
- ***Portabilité*** (à des projets différents).
- ***Coût*** (est-il en rapport avec les risques encourus par l'absence de méthode ?).

II - Méthodes de conception

- Qu'est ce que l'analyse, la conception, la modélisation d'un projet multimédia ? Dans quel contexte de réalisation doit-il être mené ? Pour quoi faire ? Voilà, certaines questions qui doivent être posées avant d'être en mesure de réaliser un projet quel qu'il en soit.
- Il est alors important de savoir, de définir les activités ou les tâches à mener à bien, d'indiquer les interactions entre celles-ci, de formuler les résultats d'une tâche, etc….
- Ainsi, avant de réaliser un projet multimédia, il est capital de commencer l'étude de ce projet par la réalisation du cahier des charges comme nous l'avons décrit dans le chapitre 1. Une fois celui-ci réalisé, il faudra précisément définir les composantes du projet, ses données, ses fonctions, ses relations, ses interfaces, etc…. L'analyse, la modélisation, la conception est alors un moyen de faire cette étude et ce, grâce aux **méthodes d'analyse**.
- On peut classer les méthodes selon différents critères, notamment, celui du cycle de vie qu'elles supportent, celui de la technologie qu'elles offrent ou celui du type d'application qu'elles permettent de concevoir.
- Ainsi, nous avons choisi de classer les méthodes selon la démarche de conception préconisée, qui est en fait primordiale dans la conception d'un projet multimédia. On retrouve ainsi, des **méthodes de conception descendante** (les approches fonctionnelles) :
  - Les méthodes de conception fonctionnelles se sont imposées assez naturellement les premières, car elles sont basées sur une séparation entre les données et le code, comme il l'existe dans l'architecture des ordinateurs.
  - Cette démarche, compréhensible dans les années 70, est inconcevable de nos jours, compte tenu de l'architecture même des ordinateurs. En effet, il n'y a aucune raison valable d'inscrire des réponses matérielles dans des solutions logicielles.
  - Parmi les méthodes de fonctionnelles, on peut distinguer les **METHODES HIERARCHIQUES** (ou appelées encore approches cartésiennes) et qui correspondent à la première génération développée durant les années 70 et
    - Ces méthodes consistent à décomposer de façon hiérarchique un problème initial en un ensemble de sous problèmes de complexité moindre, jusqu'à atteindre un niveau de décomposition suffisamment fin qui puissent être codé sous la forme de fonctions simples à réaliser.

- C'est, ce que l'on appelle une approche fonctionnelle descendante ou encore approche cartésienne classique, qui consiste donc à décomposer un problème en sous-problèmes pour en maîtriser sa complexité : " le diviser pour mieux régner ".
- Cette méthode de conception a des points forts : c'est une "discipline" bien organisée, réfléchie, logique qui favorise le développement ordonné des systèmes. Elle permet également, une certaine simplicité, car elle reflète le bon sens et une démarche assez naturelle pour aborder un problème. De plus, elle possède une certaine capacité à produire des solutions à plusieurs niveaux d'abstraction.
- Cependant, elle présente aussi des points faibles : la méthode prend en compte difficilement l'évolution des systèmes et ne facilite pas la réutilisabilité. L'utilisation des fonctions néglige quelques fois la structure de données. Elle impose également des règles de décomposition qui ne sont pas toujours explicites et qui produisent surtout des hiérarchies de décomposition différentes selon les analystes.
- Parmi les méthodes hiérarchiques, nous pouvons citer des méthodes comme **Jackson, Yourdon, ou encore, SADT.**

➢ Et les **METHODES SYSTEMIQUES** qui correspondent à la deuxième génération développée durant les années 80 :

- Dans ce type d'approche, le système d'information est considéré comme un objet complexe actif dont il faut décrire la structure (ou l'architecture) et les objectifs fonctionnels (les fonctions).
- La modélisation qui est alors faite du système est examinée selon deux points de vue complémentaires : la modélisation des données et la modélisation des traitements.
- Le modèle ainsi construit garantit d'une part la cohérence des données, et fournir, d'autres part, une spécification la plus complète possible des traitements à réaliser sur ces données.
- Cette méthode possède des **points faibles** :
  - Un manque de cohérence entre modèles de données et modèles de traitements (les deux types de modèles n'ont aucun concept commun et ne font pas explicitement référence l'un à l'autre).
  - La modélisation des traitements mélange la connaissance et le contrôle (les règles de gestion et les contraintes d'intégrité du système d'information sont intégrées dans la logique algorithmique des fonctions, ce qui ne facilité pas leur consultation et leur évolution).
  - Un cycle de développement incomplet pour les traitements (ce qui fait dire de ces méthodes que ce sont plus des méthodes de conceptions que de développement).
  - Comme exemples de méthodes dans cette catégorie, on peut citer notamment la plus connue et utilisée en France : **MERISE**.

- et des **méthodes de conception ascendante** (les approches objets).

- Vers le début des années 90, les **méthodes objets** ont vu le jour. Les informaticiens utilisent alors des méthodes de conception structurée dans un cas et dans l'autre, la conception objet.
- Ainsi, la progression vers l'analyse est opérée, toujours en exploitant le même paradigme, soit fonctionnel, soit objet. Chaque approche peut donc proposer une démarche complète, sur l'ensemble du cycle de vie du logiciel.

III- Principaux modèles de développement

- Comme tous logiciels informatiques, une application multimédia est conçue selon un *procédé de production* ou un *processus de développement*. Ce processus possède **certaines caractéristiques**.
  - Il fait une large place à l'analyse des besoins, à la conception et à la validation.
  - Il s'opère par *raffinements successifs* (la partie technique du développement de l'application consiste en l'établissement d'une suite de descriptions de plus en plus proches d'un programme exécutable et de sa documentation).
  - Certaines étapes peuvent déclencher la révision des étapes précédentes (un manque de précision des spécifications peut être détecté lors de la conception).
- Ainsi, l'ensemble des phases qui permettent de créer une application multimédia s'appelle le *cycle de vie*. De plus, pour mieux maîtriser le processus de développement, il est important de respecter les **modèles de cycle de vie**, permettant de prendre en compte, en plus des aspects techniques, l'organisation et les aspects humains.
- De plus, il est important de préciser qu'une phase comme celle de la conception, peut faire intervenir plusieurs *activité*s (notamment celle de la spécification globale, celle du maquettage et celle de la validation). Inversement une activité comme la documentation peut se dérouler pendant plusieurs phases.
- Les relations entre les activités et les phases dépendent principalement du modèle que l'on choisit. La continuité du cycle de vie du logiciel implique qu'il faut respecter l'enchaînement des différentes phases.
- Les principaux modèles de développements utilisés lors de la conduite de projets informatiques sont :

1. Le modèle en cascade

- Le modèle en cascade est le plus vieux de tous les cycles de vie et il est le souvent utilisé parmi les **paradigmes** du génie logiciel. Loin d'être infaillible, il sert néanmoins de références à d'autres cycles plus efficaces.
- Avec ce modèle, le projet progresse étape par étape, de manière ordonnée, de la conception initiale du logiciel au test système. Chaque phase fait l'objet, d'un examen avant de passer à la suivante.
- Ainsi, si la revue effectuée entre la phase de l'analyse des spécifications et celle de la conception fonctionnelle ne se relève pas satisfaisante, le passage ne se réalisera pas avant que le problème soit résolu. Les phases sont par ailleurs discontinues (elles ne se chevauchent pas).
- Le cycle de vie en cascade est documentaire : il produit des documents pour chaque phase. Dans le modèle en cascade simple,

- Ce modèle possède certains **avantages**,
  - Il est performant pour les cycles d'un produit de définition concise et dont les méthodologies techniques sont avérées. Ce modèle permet alors de détecter, à moindres frais, les erreurs dès le début du projet.
  - La cascade simple permet de minimiser le temps système car la planification peut être menée de front. Les résultats sont tangibles, sous forme de logiciel, qu'à la fin du cycle de vie mais, pour un initié, la documentation générée fournit une somme appréciable sur ses différentes phases.
  - Ce modèle fonctionne très correctement avec des projets complexes mais pour cela il est essentiel de procéder par ordre. Il est également efficace lorsque les spécifications de qualité passent avant le coût et les exigences de planning.
- Il a également quelques **inconvénients** et la principale faiblesse du modèle ne provient pas de ses activités mais plutôt de leur traitement discontinu et séquentiel.
  - Le problème majeur de ce cycle de vie est son manque de souplesse. Il est en effet indispensable de préciser intégralement dans le cahier des charges dès le début du projet, ce qui n'est pas compatible avec les exigences commerciales actuelles.
  - Certains critiques aussi ce modèle car il n'est pas possible de faire marche arrière pour corriger d'éventuelles erreurs.
  - Le modèle en cascade présente aussi d'autres faiblesses. Certains outils, certaines méthodes et activités encombrent ses différentes phases. Ils sont difficiles à adapter aux phases discontinues de ce modèle.
  - En résumé, les faiblesses du cycle de vie en cascade simple le rendent peu adapté à l'élaboration d'un projet rapide, primordial en multimédia. Même si les avantages semblent quelques fois supérieures aux inconvénients, des modèles modifiés de cycle de vie en cascade semblent plus appropriés.

2. Le modèle "programmer-corriger"

- Le modèle "programmer-corriger" est rarement utile, mais il est cependant d'usage fréquent. C'est généralement le modèle que l'on utilise par défaut.
- Le chef de projet dispose éventuellement d'un cahier des charges où est définit une idée générale du résultat souhaité. Il est alors possible d'utiliser toutes les combinaisons possibles d'analyse, de programmation de correction et de méthodologies de tests jusqu'à obtenir un produit prêt à être lancé.
- Ce modèle présente deux **avantages principaux**.

- Le temps système n'entre pas en ligne de compte. Il est donc inutile de se soucier de la planification, de la documentation, de l'assurance qualité, de l'application des normes ou de toute activité autre que le codage. La progression est immédiatement perceptible et il n'est pas nécessaire d'être expert pour l'utiliser (toute personne ayant déjà décrit un programme informatique connaît forcément ce modèle).
- Pour les projets de faible envergure et d'espérance de vie courte, ce modèle peut être utile. Cela concerne de petits programmes, les démos de courte durée ou les prototypages à usage unique.

- Il possède également certains **inconvénients**.
  - Il peut s'avérer dangereux pour les projets de plus grande envergure. En effet, si le temps système est important et que le développeur ne se consacre qu'à la programmation sans aucun moyen de contrôle de qualité ou d'identification de risques éventuels et, qu'une fois le travail presque achevé il se rend compte que l'analyse est lourdement entachée, il ne lui restera plus qu'à tout reprendre depuis le début. D'autres modèles permettraient alors de détecter et de corriger plus tôt de telles erreurs.
  - Ce modèle de cycle de vie ne présente aucun intérêt pour l'élaboration rapide d'un projet, exception faite des petits projets cités précédemment.

3. Le modèle en V

- Avec ce modèle, toutes les décompositions doivent être décrites avec la recomposition, et que toute description d'un composant est accompagnée de tests qui permettront de s'assurer qu'il correspond à sa description.
- Ceci rend explicite la préparation des dernières phases (validation-vérification) par les premières (construction du logiciel), et permet ainsi d'éviter un écueil bien connu de la spécification du logiciel (énoncer une propriété qu'il est impossible de vérifier objectivement après la réalisation).
- Avec ce modèle, on peut alors différencier deux sortes de dépendances :
- L'enchaînement et l'itération se déroulent essentiellement de gauche à droite.
- La préparation des phases ultérieures. Par exemple à l'issue de la conception architecturale, le protocole et les jeux de test de l'intégration doivent êtres complètement décrits.
- *Cela a pour conséquences : l'obligation de concevoir les jeux de test et leurs résultats et la réflexion et retour sur la description en cours.*
- Il est important aussi de noter que les activités de chaque phase peuvent être réparties en 5 catégories : *l'assurance qualité*, la *production*, le *contrôle technique*, la *gestion* et le *contrôle de la qualité*.

4. Le modèle en spirale

- Proposé par B. Boehm en 1988, le modèle en spirale est beaucoup plus sophistiqué que celui le modèle "programmer-corriger". Ce modèle met l'accent sur l'activité *d'analyse des risques*. Il découpe le projet en plusieurs petits sous ensembles. Chacun gère un ou plusieurs risques principaux jusqu'à l'examen exhaustif de ces derniers.
- Les risques peuvent être liés à des spécifications ou une architecture mal comprises, à des problèmes de performance du logiciel, à la technologie sous-jacente. Une fois tous les risques principaux passés en revue, le modèle en spirale s'achève selon le schéma de vie en cascade.

- Le modèle en spirale se déroule en **quatre phases :**
  - La détermination, à partir des résultats des cycles précédents ou de l'analyse préliminaire des besoins, des objectifs du cycle, des alternatives pour les atteindre et des contraintes.
  - L'analyse des risques, l'évaluation des alternatives et, éventuellement le maquettage.
  - Le développement et la vérification de la solution retenue, un modèle " classique " (cascade ou en V) peut être utilisé ici.
  - La revue des résultats et la vérification du cycle suivant.
- L'idée fondamentale est de débuter à petite échelle, d'explorer les risques éventuels, d'élaborer un plan pour résoudre puis établir une approche vers l'itération suivante.
- Chaque itération permet d'élargir l'échelle de travail, en partant du centre et en remontant la spirale et en vérifiant que chaque boucle est parfaitement aboutie, avant de progresser vers la suivante.
- Ce cycle de vie permet d'élaborer à **moindres frais** les toutes premières itérations.
- Ce modèle présente un **avantage** :
  - En effet, à mesure que les coûts augmentent, les risques diminuent. Plus on insiste en temps et en moyens financiers, moins on prend de risques. Le développeur est ainsi, à même de développer rapidement votre projet.
  - Le suivi de la gestion du projet est d'un niveau de performances équivalentes à celui de la cascade. En effet, un contrôle est effectué à l'issue de chaque itération. Ce modèle étant orienté risques, les risques majeurs sont décelés. Si le projet ne peut être mené à terme pour des raisons techniques, le chef de projet s'en aperçoit assez vite pour pouvoir l'abandonner avant que le coût ne devienne prohibitif.
- L'unique **inconvénient** de ce modèle provient de sa complexité :
  - Il requiert une grande attention ainsi que d'excellentes qualités de gestionnaire. Il peut s'avérer difficile de définir de manière objective les étapes permettant de passer à la boucle suivante.
  - Mais, si le développement du produit est simple et les risques peu élevés, il sera inutile de recourir à la souplesse d'utilisation et la gestion des risques du modèle en spirale.

5. Le prototypage évolutif.

- Le prototypage évolutif est un modèle permettant de développer le système tout au long du projet.
- La procédure habituelle consiste à élaborer les aspects les plus importants du système puis de faire une démonstration au client. On achève ensuite le développement du prototype en prenant en considération ses commentaires. S'il est jugé suffisamment satisfaisant, il ne reste plus alors qu'à mettre au point les dernières caractéristiques du système avant de lancer le prototype.
- Le prototype évolutif s'avère particulièrement utile lorsque les spécifications sont fréquemment modifiées, que votre client n'arrive pas à arrêter son choix ou lorsque vous n'arrivez à aucun accord sur la zone d'application. Il est également utilisé lorsque les programmeurs ne savent pas quelle architecture optimale ni quels algorithmes utiliser.
- Comme tout modèle, il possède un **avantage** essentiel :

- Le prototypage évolutif fournit des indications régulières et précises sur la progression du projet, qui constituent un atout non négligeable lorsque la priorité est donnée à la vitesse de développement.

- Et des **inconvénients** :
  - L'inconvénient majeur de ce type de prototype réside dans l'impossibilité d'évaluer au début du projet le temps nécessaire à son élaboration. Il est également impossible de connaître le nombre d'itérations nécessaires. On pallie cet inconvénient en maintenant le client régulièrement informé de l'état d'avancement du projet.
  - Cette approche peut également devenir un prétexte pour utiliser la technique "programmer-corriger". Un véritable prototype évolutif inclut l'analyse des spécifications, les conceptions fonctionnelles et détaillées et une programmation efficace. La différence notable avec une autre approche se situe au niveau des surplus de paliers.

IV- Quelques outils

- Il est très difficile de définir une liste exhaustive d'outils informatiques de suivi de projet, car généralement chaque entreprise ou société de développement possèdent leurs propres outils (interne) ou utilisent des outils hybrides.
- En matière de multimédia, il n'existe pas encore d'outils standards qui permettent de coupler le suivi de projet et les méthodes de conception.
- Cependant, on peut recenser parmi les outils informatiques de suivi de projet, les principaux outils suivants :
  - **Artemis** de Lucas management Systems,
  - **PSN 6** de Le Bihan et Cie,
  - **Super Project** de Computer Associates,
  - **Project** de Microsoft.