Université Moulay Ismail
Faculté des Sciences et Technique
Département d'Informatique

Gestion de Projet Informatique

- Préparé et présenté par: DR. Lakhrouit jihane
  Docteur en Architecture d'entreprise (ENSIAS)
  Ingénieur D'Etat Génie Logiciel (ENSA Marrakech)
  Licence Science Mathématique Informatique (Faculté des Sciences de Rabat)

LST, Module I617, S6

Objectifs du module

- Avoir une vision globale sur la technique de gestion de projet, pour faciliter l'intégration de l'étudiant à une équipe de travail.

Les chapitres du cours Gestion de projet Informatique

- Chapitre I : Les cycles de vie des projets Informatique (Cycle en Cascade, Cycle en V, Cycle Incrémental, etc)
- Chapitre II: L'ordonnancement et la gestion des taches (Diagramme de Gantt, Pert, etc)
- Chapitre III: Les méthodologies de gestion de projet(Les concepts d'agilité, Scrum, XP Extreme programming, etc)
- Chapitre IV: Les types de tests dans un projet Informatique(Les tests unitaires, Les tests systèmes, les tests boite blanche, boite noire ,etc)

Les Cycles de vie

PRÉPARÉ ET PRÉSENTÉ PAR: DR. LAKHROUIT JIHANE

Sommaire

Découvrir les Concepts de gestion de projet

- **Un projet** est un effort ponctuel et coordonné pour atteindre un objectif unique, incluant une dose d'incertitude quant à sa réalisation (petit Larousse).

- On appelle un projet c'est l'ensemble des actions à entreprendre afin de répondre à un besoin défini dans des délais fixés (un début et une fin). Le projet mobilise des ressources identifiées (humaines et matérielles) durant sa réalisation, celui-ci possède également un coût et fait donc l'objet d'une budgétisation de moyens.

Découvrir les Concepts de gestion de projet

- **La gestion de projet** est une action temporaire avec un début et une fin, qui mobilise des ressources identifiées (humaines, matérielles, équipements, matières premières, informationnelles et financières) durant sa réalisation, **visant à organiser de bout en bout le bon déroulement d'un projet.**

- **Une ressource** est un élément nécessaire à la réalisation d'une tâche ou d'un projet. **Une ressource peut être une personne, une équipe, un outil, de la trésorerie ou du temps**. La plupart des projets nécessite de nombreuses ressources différentes pour se dérouler.
  - Les ressources doivent être estimées et affectées avant le début du projet. Leur mauvaise planification peut entrainer un manque pendant le projet, des retards sur certaines échéances ou même la livraison finale du projet

Sommaire

Découvrir les Concepts de gestion de projet

- **Définition des parties prenantes d'un projet :**
- Il s'agit de l'ensemble des personnes et des organisations qui ont quelque chose à voir avec le projet. Soit elles sont directement impliquées dans la conduite des opérations, soit elles sont impactées par la problématique de départ, par le choix ou la mise en œuvre des solutions. Certaines parties prenantes peuvent exercer une influence à différents niveaux.

- Ces acteurs clés se situent aussi bien en **interne** - à tout niveau de la hiérarchie de l'entreprise - qu'en **externe** (un fournisseur concerné par de nouvelles méthodes d'approvisionnement d'un client, etc.)

Sommaire

Découvrir les Concepts de gestion de projet

- **Qu'est-ce qu'un Chef de projet informatique ?**

- Expert en informatique, le chef de projet informatique (CPI) peut également être appelé chef de projet intégrateur, chef de projet applicatif, Project manager ou responsable de domaine. Il a sous son égide plusieurs techniciens et ingénieurs qui ont chacun un rôle spécifique dans le traitement des demandes de clients particuliers.

- Garant de l'état d'avancement d'un projet informatique, le CPI doit ajuster les évolutions et les besoins y afférents si cela s'avère nécessaire. Il doit également tenir compte des contraintes en termes de financement et de délais. Il se doit donc de posséder de multiples capacités regroupant des compétences techniques et managériales à la fois

Découvrir les Concepts de gestion de projet

- **Qu'est-ce qu'un Chef de projet informatique ?**

Rattaché à un directeur des systèmes d'information ou à un directeur des études, le chef de projet informatique est à la tête d'un ou plusieurs services dans l'entreprise.

Accompagné par son équipe, le **CPI** a pour rôle principal de concevoir et d'intégrer un logiciel ou une solution informatique spécifique. Ses tâches sont multiples et couvrent l'ensemble de toutes les étapes du projet depuis le **brief client** (Le brief client, aussi appelé cahier des charges) jusqu'à la réception par ce dernier ainsi qu'au suivi et maintenance.

La personne doit être capable de solutionner les différentes problématiques qui risqueraient de nuire au projet. Elle doit avoir un esprit créatif pour pouvoir améliorer et sublimer sa création. Elle est amenée à trouver des idées innovantes ainsi que des stratégies optimales qui seront bénéfiques pour l'entreprise. Le chef de projet informatique est notamment spécialisé dans un langage informatique particulier.

Sommaire

Découvrir les Concepts de gestion de projet

- **Matrice d'assignation des responsabilités**
- La matrice RACI est une matrice d'attribution des responsabilités servant à décrire la participation des divers rôles, à remplir les tâches ou livrables pour un projet ou processus.
- Elle est utile pour clarifier les rôles et responsabilités dans des projets et des processus transversaux ou, d'une manière plus générale, dans un département ou service, afin d'avoir une vision claire de la répartition des tâches.
- Il s'agit donc de donner à chaque membre de l'équipe un niveau de responsabilité en fonction des tâches du projet.

Découvrir les Concepts de gestion de projet

- **Matrice d'assignation des responsabilités**




Responsible	Accountable	Consulted	Informed
Celui qui réalise la tâche	*Celui qui approuve la tâche*	*Celui qui est consulté*	*Celui qui doit être informé*
Qui ? Personne qui va exécuter la tâche : elle en est responsable.	**Qui ?** Personne qui va approuver la tâche : elle en est l'autorité.	**Qui ?** Personne qui va être consultée dans l'exécution de la tâche : elle est consultée.	**Qui ?** Personne qui sera informée lorsque la tâche est finie : elle est informée.
Sa mission : Réaliser la tâche qui lui a été attribué.	**Sa mission :** Veiller à l'exécution correcte de la tâche réalisée par le(s) responsable(s) et approuver l'activité faite.	**Sa mission :** Contribuer avec des conseils et opinions à ce que la tâche soit effectuée le plus efficacement possible.	**Sa mission :** Être tenue à jour sur les progrès réalisés, souvent à l'issue de la tâche ou du livrable.
Particularité : Il peut y avoir plusieurs responsables pour une même tâche, chacun fait une partie de la tâche.	**Particularité :** Une autorité par tâche.	**Particularité :** Il peut y avoir plusieurs personnes consultées et ce sont souvent des experts.	**Particularité :** Elle n'intervient pas activement dans la réalisation de la tâche.

Sommaire

Découvrir les Concepts de gestion de projet

- **Un projet** peut se définir comme un ensemble d'actions mises en œuvre pour atteindre un but précis, afin de répondre à un besoin spécifique. Il se caractérise par : Chaque projet doit comporter des objectifs clairement définis qui permettent la satisfaction d'un besoin spécifique et particulier. Une limite dans le temps : il a un début et une fin, marquée par l'atteinte de l'objectif .
- **Une activité** est une action qui transforme les ressources (main d'oeuvre, connaissances, l'équipement, les matières premières, le temps) en résultats attendus dans un délai de temps spécifié. Parfois, une activité est suffisante pour obtenir les résultats souhaités, mais souvent il faut passer par toute une série d'activités. Lorsque vous devez passer par la même série d'activités ou des tâches à chaque fois que vous voulez obtenir un résultat, on peut parler d'un processus. Dans le cadre logique, chaque résultat dépend d'une ou de plusieurs activités ou processus.
- **Les ressources** sont les choses qui se transforment en résultats. Lorsque nous parlons des ressources, nous pensons généralement à l'argent, le personnel, le matériel ou l'équipement. Mais il y a d'autres choses qui sont nécessaires pour un projet: le temps, les connaissances et le savoir faire, l'espace, l'infrastructure , la communication (accès à l'information) et ainsi de suite.

Découvrir les Concepts de gestion de projet

- Les résultats attendus se créent à la suite des activités du projet. Ensemble, les résultats mènent à la réalisation de l'objectif spécifique du projet. L'objectif spécifique est la situation que vous espérez atteindre lorsque le projet est terminé. Les résultats sont les biens, les services et ainsi de suite que vous souhaitez créer, au cours du projet. En tant que tel, l'achèvement des résultats est - en principe - entièrement sous votre contrôle.
- A ce niveau, la logique du projet est le plus fort: vous investissez des moyens (ressources) pour faire les activités et les activités mèneront à leur tour aux résultats concrets

Sommaire

Découvrir les Concepts de gestion de projet

Contraintes dans la gestion d'un projet

- Les contraintes de projet sont les limites générales dont vous devez tenir compte tout au long du cycle de vie du projet.
- Par exemple, une contrainte de coûts signifie que vous devez vous en tenir à un certain budget pour ce projet, tandis qu'une contrainte de temps signifie que vous devez le finaliser dans un délai précis.
- La plupart des contraintes de projet ont des répercussions les unes sur les autres, c'est pourquoi la gestion des contraintes est déterminante pour la réussite d'un projet. Si vous choisissez de prolonger le délai du projet, alors vous aurez probablement besoin de revoir également à la hausse le budget alloué. La portée de votre projet sera, elle aussi, étendue si ces deux premiers facteurs sont réajustés.

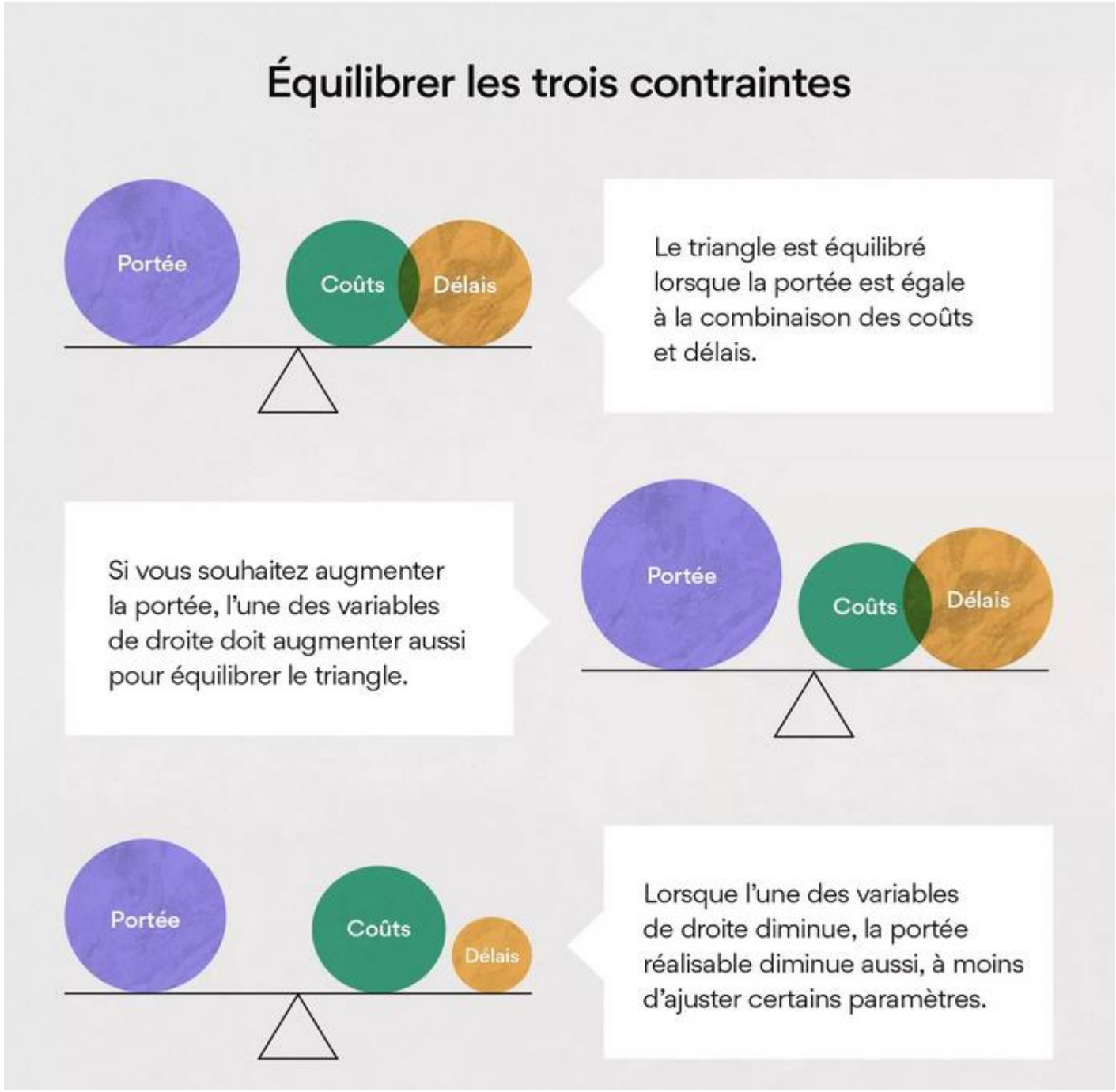
Découvrir les Concepts de gestion de projet

- **Contraintes de la portée :**
- La **portée d'un projet** correspond à son ampleur en termes de qualité, de détails et de livrables. Le délai et le budget dépendent de la portée du projet, car plus la portée s'étend, plus il faudra de temps et d'argent pour le mener à bien.
- Vous devrez avoir conscience de la **dérive des objectifs** au cours de chaque phase du projet et faire en sorte de l'éviter. Pour cela, vous pouvez élaborer des plans de projet détaillés et obtenir l'approbation des parties prenantes avant le début de la production.

Contraintes dans la gestion d'un projet

- La portée, les coûts et les délais font partie de ce que l'on appelle le triangle d'or.

- En effet, il peut s'avérer difficile de jouer sur ces trois contraintes, dépendantes les unes des autres, sans altérer la qualité du projet.
- Par exemple, si vous réduisez le budget ou augmentez la portée, vous devrez probablement compenser en accordant plus de temps à la réalisation. Vous pouvez le faire en prolongeant les délais, en ajoutant des heures de travail ou en ajustant le calendrier de votre projet.



Découvrir les Concepts de gestion de projet

- **Contraintes des délais :**
- La gestion du temps est essentielle à la réussite du projet, et vous rencontrerez diverses contraintes de temps au cours de chacune des phases de ce dernier.
- Si vous essayez d'augmenter la durée de votre projet, il y aura des conséquences comme des délais plus longs, des ajustements dans le calendrier de l'équipe ou un temps réduit pour la planification.
  - L'ensemble du calendrier du projet
  - Les heures passées à travailler sur le projet
  - Les calendriers et objectifs internes
  - Le temps alloué à la planification et à la stratégie
  - Le nombre de phases du projet

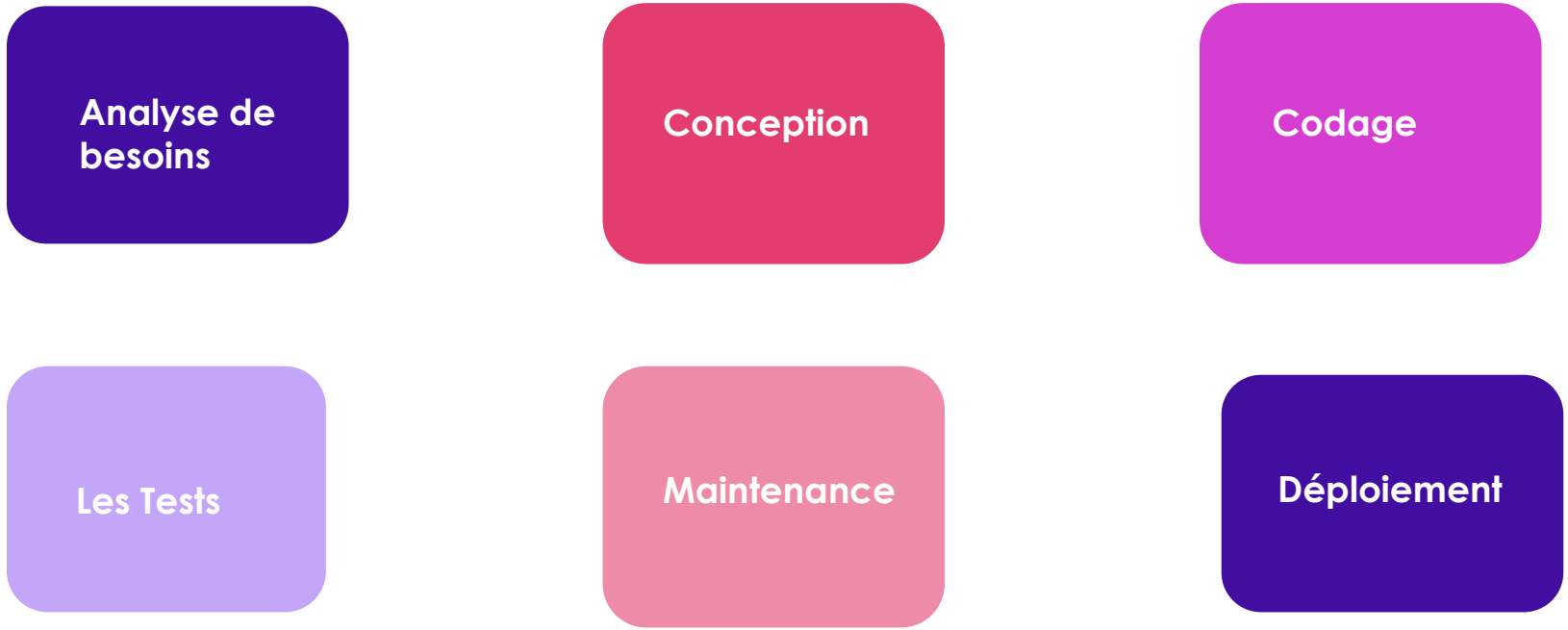
Découvrir les Concepts de gestion de projet

- Contraintes de coûts :
- Budget pour réaliser le projet
- Les contraintes de coûts comprennent le budget du projet dans son ensemble et tout élément de valeur financière nécessaire à votre projet. Voici quelques éléments qui peuvent constituer une contrainte de coûts :
  - Le coût du projet
  - Les salaires des membres de l'équipe
  - Le coût des équipements
  - Le coût des installations
  - Les coûts de réparation
  - Les coûts des matériaux
- Dans cette section on précise tous les éléments qui vous obligent à puiser dans les ressources financières de votre entreprise.

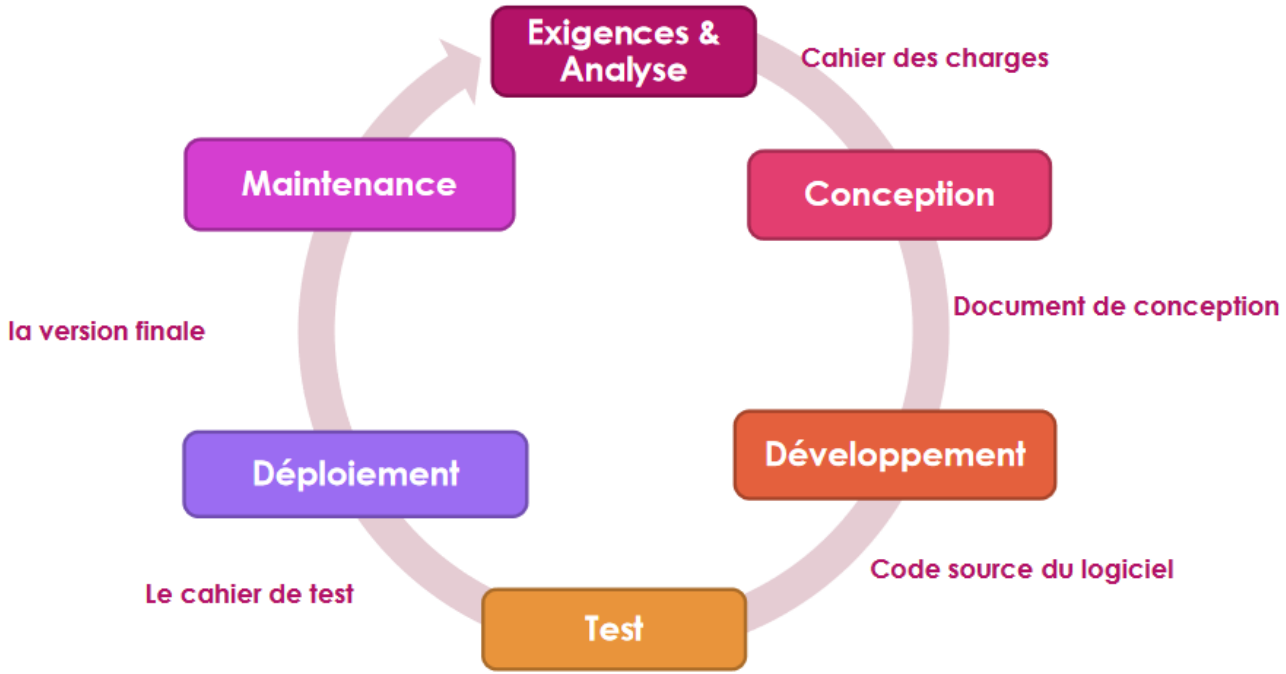
Sommaire

Les Etapes de développement

- Tous les projets de développement ont des étapes en commun :
- Chaque activité et chaque étape est conduite et réalisée par plusieurs développeurs
- Une activité a des entrées et des sorties. Les livrables font partie des sorties des activités.
- Les livrables sont des produits ou des documents produits.
- Par exemple : document, planning, code source sont des livrables



Les Etapes de développement



Les Etapes de développement

Analyse de besoins

- **Dans cette phase, les besoins des utilisateurs et les objectifs du logiciel sont identifiés et définis. Les exigences fonctionnelles et non fonctionnelles sont spécifiées, ainsi que les contraintes du projet.**
- Quels sont les besoins métier et les exigences des utilisateurs finaux ? Cette question permet de comprendre les exigences métier et les besoins spécifiques des utilisateurs finaux. Cela peut aider à établir les caractéristiques fonctionnelles du logiciel et les contraintes qui s'appliquent au projet.
- Quel est le public cible pour le logiciel ? Cette question permet de définir le public cible pour le logiciel, qui peut inclure des utilisateurs finaux, des gestionnaires, des propriétaires d'entreprise et d'autres parties prenantes.
- Quelles sont les contraintes de conception pour le logiciel ? Cette question permet de comprendre les contraintes de conception pour le logiciel, qui peuvent inclure des considérations techniques, des ressources disponibles et des contraintes de temps et de budget.

Les Etapes de développement

Conception

- **Dans cette phase, l'architecture du logiciel est conçue en détail. Les spécifications techniques sont élaborées, les composants logiciels sont identifiés et une documentation est créée.**
- Comment le système sera-t-il architecturé ? Cette question permet de définir l'architecture globale du système et de déterminer comment les différentes composantes du système s'articuleront entre elles.
- Quels sont les composants clés du système ? Cette question permet de définir les composants clés du système, y compris les modules, les bases de données, les interfaces utilisateur, etc.
- Comment la sécurité sera-t-elle gérée ? Cette question permet de déterminer les mécanismes de sécurité nécessaires pour protéger les données et les fonctions du système contre les menaces éventuelles.
- Comment le système sera-t-il testé et validé ? Cette question permet de définir les méthodologies de test et de validation qui seront utilisées pour s'assurer que le système répond aux exigences spécifiées.
- Comment le système sera-t-il déployé et maintenu ? Cette question permet de définir les exigences de déploiement et de maintenance du système, y compris les procédures de déploiement, les contrôles de qualité, les mécanismes de mise à jour, etc.

Les Etapes de développement

Implémentation

- **Dans cette phase, le code source est écrit, le logiciel est développé, les fonctionnalités sont implémentées et testées.**
- Quels sont les langages de programmation et les outils de développement nécessaires pour implémenter le système ? Cette question permet de définir les outils et les technologies nécessaires pour mettre en œuvre le système, y compris les langages de programmation, les bibliothèques de code, les environnements de développement, etc.
- Comment les erreurs de code seront-elles gérées ? Cette question permet de définir les processus et les méthodologies de gestion des erreurs, y compris les procédures de test, de débogage et de correction d'erreurs.
- Comment le code sera-t-il documenté ? Cette question permet de définir les normes de documentation pour le code, y compris les commentaires de code, les guides d'utilisation, les manuels, etc.
- Comment le code sera-t-il testé et validé ? Cette question permet de définir les méthodes de test et de validation qui seront utilisées pour s'assurer que le code répond aux exigences spécifiées.
- Comment le code sera-t-il déployé ? Cette question permet de définir les procédures de déploiement du code, y compris les environnements de test et de production, les mécanismes de mise à jour, etc.

Les Etapes de développement

Les Tests

- **Dans cette phase, le logiciel est testé pour s'assurer qu'il fonctionne correctement et qu'il répond aux exigences. Les tests comprennent des tests unitaires, des tests d'intégration et des tests système.**
- Quels sont les types de tests nécessaires pour valider le système ? Cette question permet de définir les différents types de tests à effectuer, tels que les tests unitaires, les tests d'intégration, les tests fonctionnels, les tests de performance, etc.
- Comment les scénarios de test seront-ils définis ? Cette question permet de définir les scénarios de test, qui sont des étapes détaillées permettant de tester le système en fonction de différents cas d'utilisation.
- Comment les erreurs seront-elles signalées et gérées ? Cette question permet de définir les processus de signalement et de correction des erreurs, y compris les procédures de débogage et de correction des erreurs.
- Comment les résultats de test seront-ils documentés et communiqués ? Cette question permet de définir les procédures de documentation et de communication des résultats de test, y compris les rapports de test, les enregistrements de test, etc.
- Comment le système sera-t-il validé et accepté ? Cette question permet de définir les critères de validation et d'acceptation du système, y compris les procédures de validation, les tests d'acceptation, etc.

Les Etapes de développement

Déploiement

- **Dans cette phase, le logiciel est installé et configuré sur les ordinateurs des utilisateurs finaux ou dans l'environnement de production.**
- Comment le système sera-t-il déployé ? Cette question permet de définir les procédures de déploiement du système, y compris les étapes de préparation, les configurations de l'infrastructure, les processus de migration de données, etc.
- Quels sont les environnements de déploiement ? Cette question permet de définir les environnements de déploiement, tels que les environnements de développement, de test, de pré-production et de production, et de déterminer les procédures de transfert entre ces environnements.
- Comment le système sera-t-il surveillé ? Cette question permet de définir les mécanismes de surveillance du système, tels que les outils de surveillance, les métriques de performance, les journaux d'activité, etc., et de déterminer les procédures de signalement et de correction des incidents.
- Comment les utilisateurs seront-ils formés et soutenus ? Cette question permet de définir les procédures de formation des utilisateurs finaux, y compris les guides d'utilisation, les vidéos tutoriels, les sessions de formation, etc., et de déterminer les procédures de support technique.
- Comment les mises à jour et les corrections seront-elles gérées ? Cette question permet de définir les procédures de gestion des versions et des correctifs du système en production, y compris les processus de mise à jour, de rétrogradation, de validation, etc

Les Etapes de développement

Maintenance

- Dans cette phase, **le logiciel est maintenu en bon état de fonctionnement**. Les erreurs sont corrigées, des mises à jour sont effectuées, des nouvelles fonctionnalités peuvent être ajoutées et des améliorations sont apportées pour répondre aux besoins changeants des utilisateurs.
- Comment le système est-il utilisé ? Cette question permet de comprendre comment les utilisateurs finaux utilisent le système et quelles sont les fonctionnalités les plus importantes pour eux.
- Quels sont les problèmes les plus fréquemment signalés ? Cette question permet de recueillir des informations sur les problèmes courants qui surviennent dans le système, tels que les erreurs, les pannes, les ralentissements, etc.
- Comment les problèmes sont-ils corrigés ? Cette question permet de définir les procédures de correction des problèmes, y compris les étapes de détection, de diagnostic, de correction et de test des solutions.
- Comment le système est-il amélioré ? Cette question permet de définir les procédures d'amélioration du système, y compris les processus d'analyse des performances, de conception de nouvelles fonctionnalités, de test et de déploiement des améliorations.
- Comment les mises à jour sont-elles gérées ? Cette question permet de définir les procédures de gestion des mises à jour du système en production, y compris les processus de mise à jour, de validation, de rétrogradation, etc.

Sommaire

Partie III : Les cycles de vies

- Pour maîtriser les gros projets
- Pour découper le projet et affecter correctement les tâches
- Pour anticiper et gérer les risques

Cycle de vie du développement logiciel

Définition

- Le cycle de vie du développement logiciel est une méthodologie qui décrit les différentes phases de développement d'un logiciel, depuis la conception jusqu'à la mise en production et à la maintenance ultérieure.
- Il existe plusieurs modèles de cycle de vie du développement logiciel, mais le plus commun est le modèle en cascade, qui consiste en une séquence linéaire de phases distinctes, chacune étant réalisée avant la suivante.
- Les phases typiques du cycle de vie du développement logiciel en cascade sont les suivantes :
  - Analyse des besoins
  - Conception
  - Tests
  - Intégration
  - Déploiement
  - Maintenance

Cycle de vie du développement logiciel

Types de cycles de vie

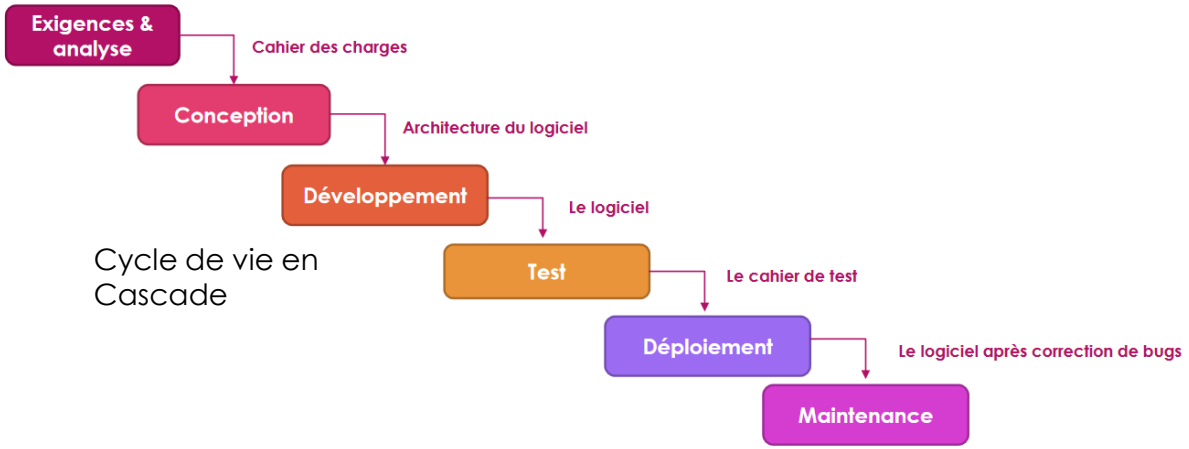
- Ils existe deux types de modèles de cycles de vie : Modèles linéaires et modèles itératifs.
- Modèles linéaires :
  - Cascade
  - Modèle en V
- Modèles non linéaires (itératifs)
  - Prototypage
  - Modèle incrémentaux
  - Modèle en spirale

Sommaire

Cycle de vie du développement logiciel

Modèle en cascade

- C'est un des premiers modèles qui a été proposé. Il date des années 70.
- Ce modèle est strict et lourd. Chaque phase doit être approuvée avant de pouvoir commencer l'autre.
- Le modèle en cascade, appelé Waterfall en anglais, tel qu'appliqué aux projets, est une approche linéaire et séquentielle des différentes phases et activités du projet nécessaires à la livraison du ou des livrables.



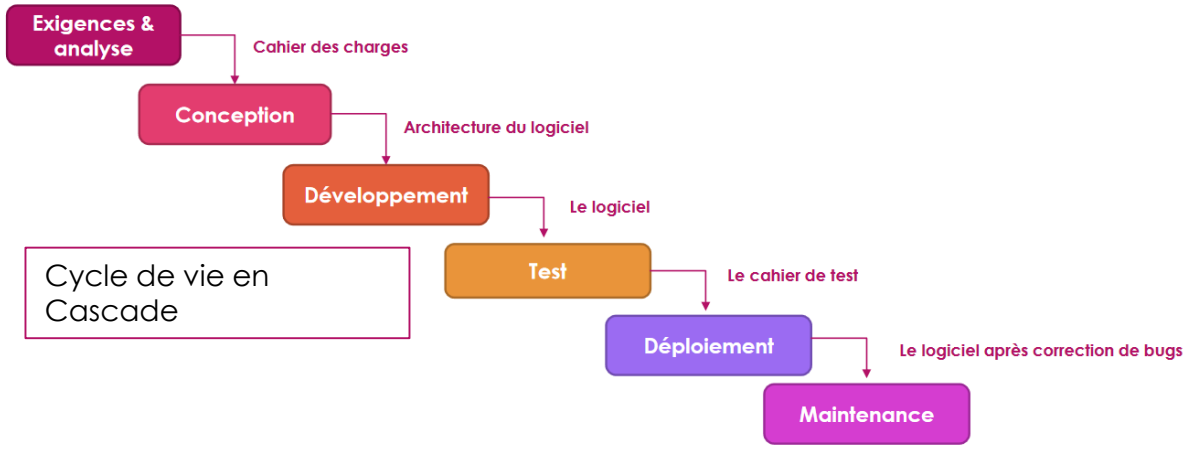
Cycle de vie en Cascade

Remarque : Les étape s'exécutent en séquence. Chaque étape dépend de l'étape précédente.

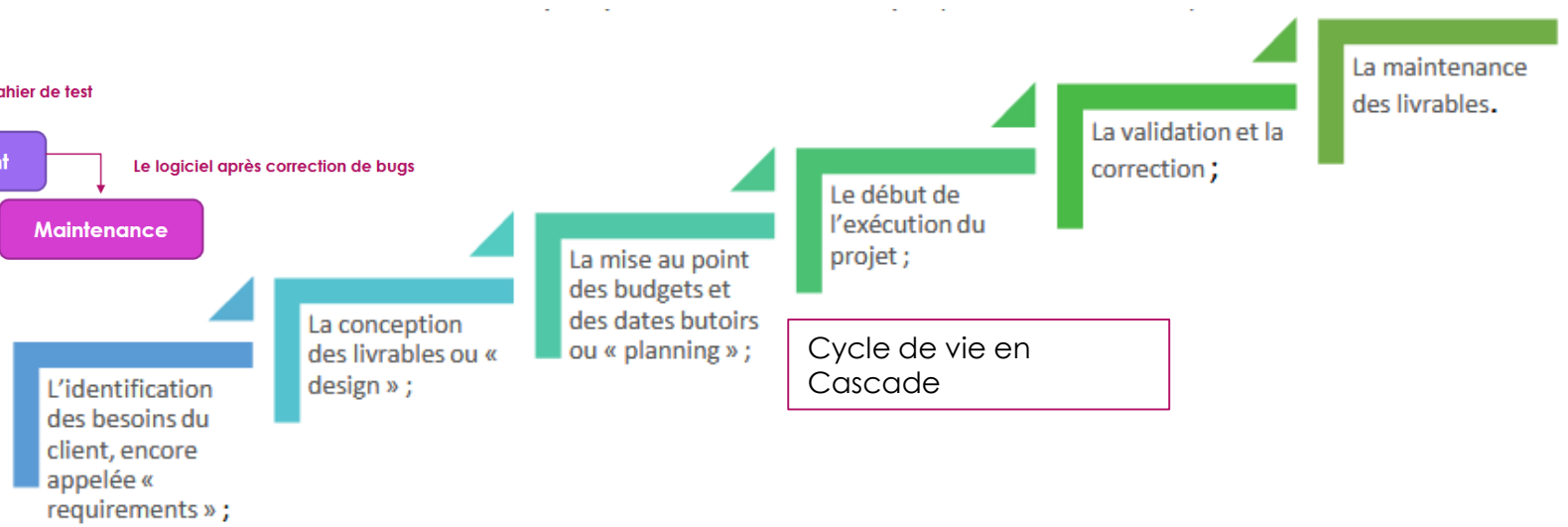
Cycle de vie du développement logiciel

Modèle en cascade

Sur le modèle d'une cascade, c'est la fin d'une phase qui mène au démarrage de la suivante. Par ailleurs, il n'y a aucune possibilité de retour en arrière.



Cycle de vie en Cascade

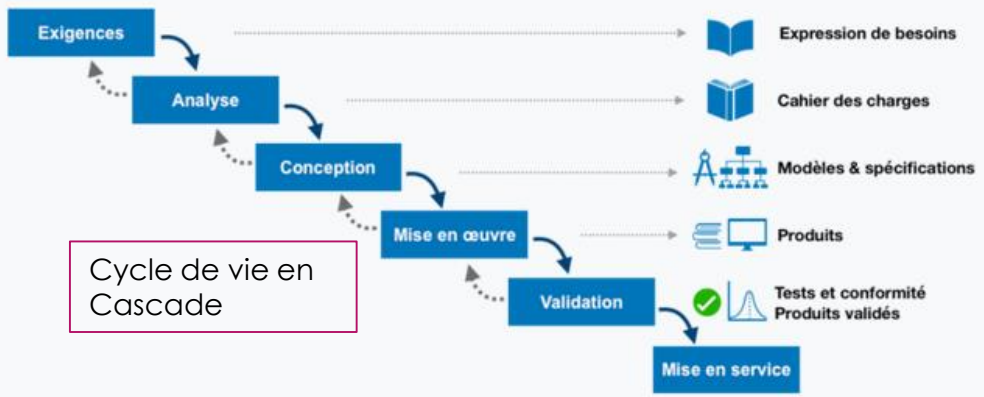


Cycle de vie en Cascade

Cycle de vie du développement logiciel

Modèle en cascade

- Avantages :
- Chaque phase est finie, elle est donc gérable tant au niveau du temps que des ressources.
- Le code est créé assez tardivement, il y a donc une bonne compréhension du système.



Cycle de vie en Cascade

- Inconvénients :
- Ce modèle ne représente pas la réalité.
- Sensibilité aux nouveaux besoins : refaire tout le procédé
- Une phase ne peut démarrer que si l'étape précédente est finie
- Le produit n'est visible qu'à la fin
- Les risques se décalent vers la fin
- Très faible implication du client
- Il y a des risques de découvrir des erreurs qu'a la fin ce qui nécessite de refaire toutes les étapes.
- L'ajout de nouvelle exigence implique qu'il faille refaire toutes les étapes. Plus un problème ou un changement doit être fait tard, plus il coûtera cher.
- L'utilisateur final est uniquement intégré dans le processus de production après la programmation.
- Les erreurs sont parfois détectées uniquement à la fin du processus de développement

Cycle de vie du développement logiciel

Modèle en cascade

Quand l'utiliser ?

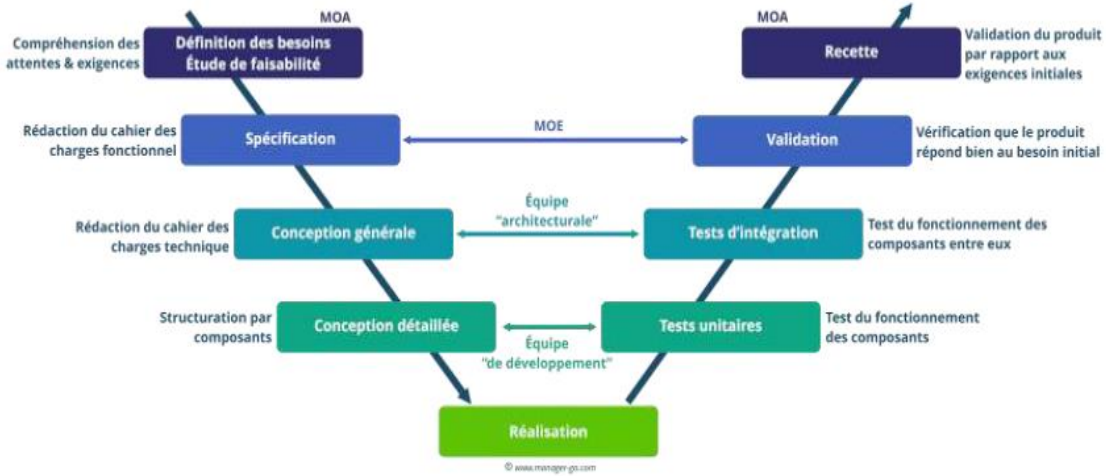
- Quand les besoins sont connus et stable.
- Quand la technologie à utiliser est maitrisée.
- Lors de la création d'une nouvelle version d'un produit existant.
- Lors du portage d'un produit sur une autre plateforme.

Sommaire

Cycle de vie du développement logiciel

Modèle en V

- Le cycle en V est un modèle de gestion de projet qui implique toutes les étapes du cycle de vie d'un projet : conception, réalisation, validation.
- Le cycle en V en gestion de projet découle du modèle en cascade théorisé dans les années 1970, qui permet de représenter des processus de développement de manière linéaire et en phases successives.
- Ce mode de gestion de projet a été développé dans les années 1980 et appliqué au champ des projets industriels, puis étendu aux projets informatiques.
- Il a été remis en cause à partir du début des années 2000, sous l'effet de l'accélération des changements technologiques, favorisant davantage les méthodes dites **« agiles ».**
- **La lettre V fait référence à la vision schématique de ce cycle**, qui prend la forme d'un V : une phase descendante suivie d'une phase ascendante. Le cycle en V associe à chaque phase de réalisation une phase de validation.

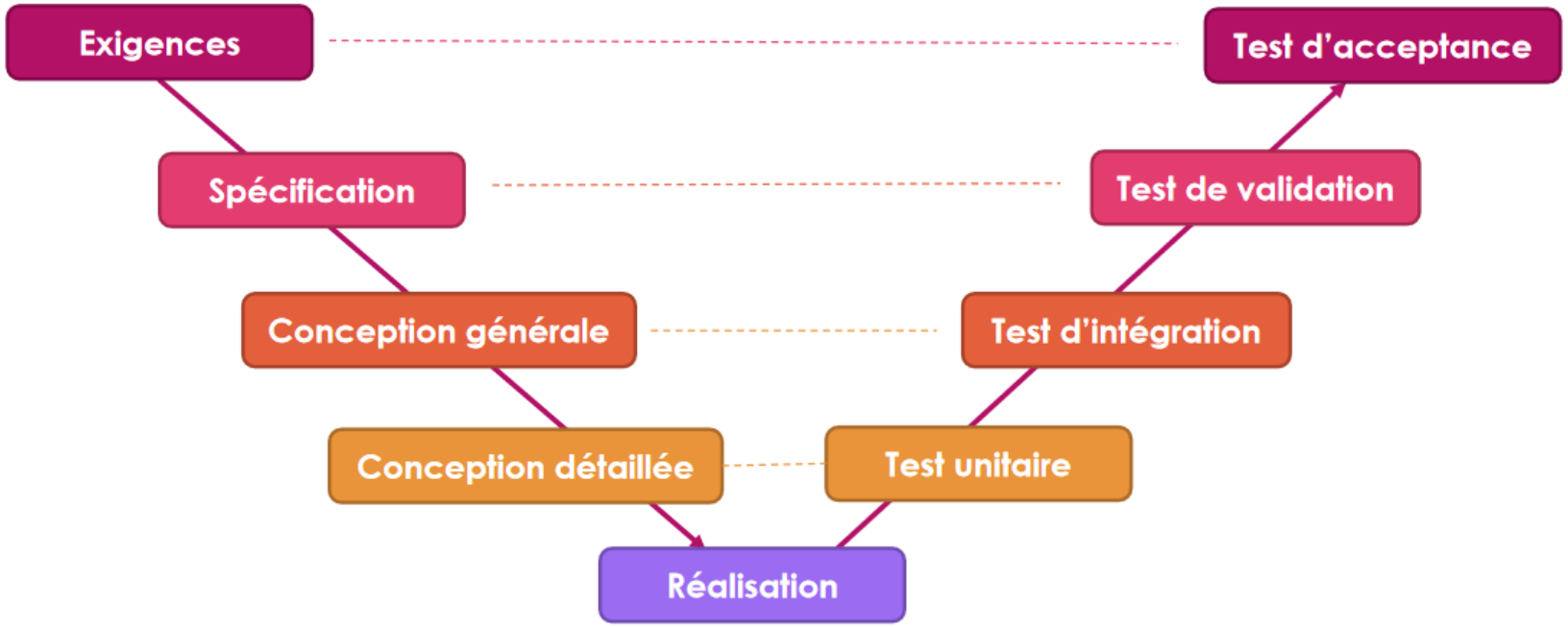


Cycle de vie en V

Cycle de vie du développement logiciel

Modèle en V

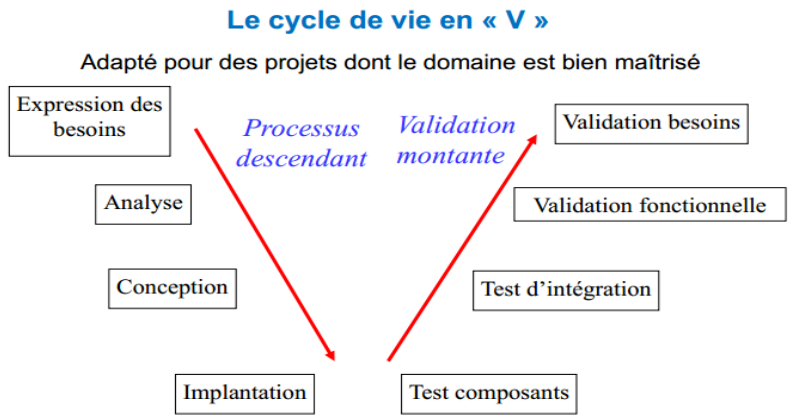
Ce modèle est convenable pour les projets complexes. C'est un modèle dérivé du modèle en cascade.



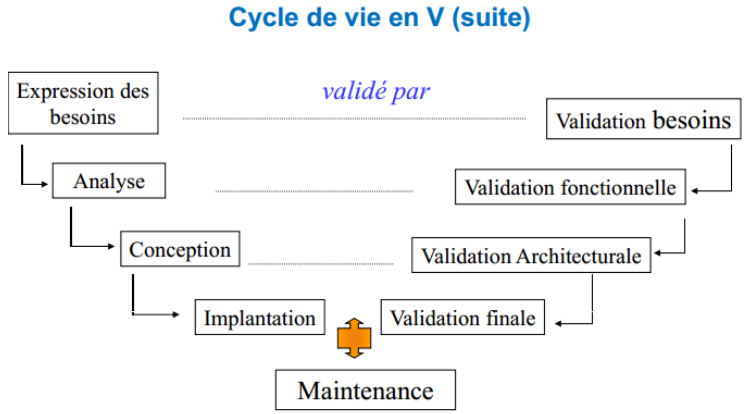
Cycle de vie du développement logiciel

Modèle en V

- Le principe de ce modèle, est que chaque étape de décomposition du système possède une phase de test.
- Chaque phase du projet à une phase de test qui lui est associé. Beaucoup de tests sont ainsi créés, ce qui implique une réflexion. On sait progressivement si on s'approche de ce que le client désire.
- Les scénarios de Tests sont crées au même moment que la phase associée.



Protocoles de validation définis par l'analyse descendante



Validation des étapes intermédiaires sous forme de documents

Cycle de vie du développement logiciel

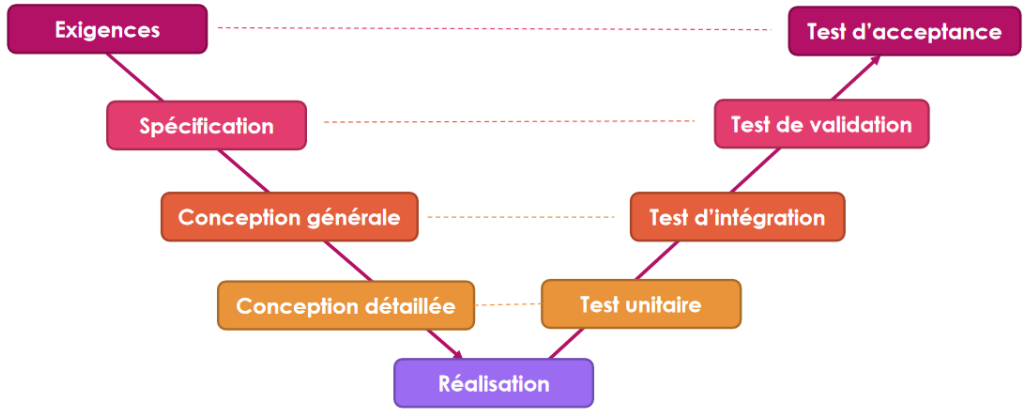
Modèle en V

- Le principal avantage du cycle en V est qu'il évite de revenir en arrière incessamment pour redéfinir les spécifications initiales.
- **Chaque phase de conception demande la rédaction d'une documentation précise et exhaustive, où chaque point doit être validé par le produit final. Dès lors qu'une étape est validée, on ne revient pas en arrière et on passe à l'étape suivante sur une base solide ; c'est la principale force du cycle en V.**
- De par son aspect à la fois rigoureux et intuitif, le cycle en V demeure un processus facile à mettre en œuvre. Le travail préalable de définition des spécifications en début de projet fait que, une fois lancé, l'ensemble des étapes est connu des collaborateurs, qui peuvent se repérer facilement dans la temporalité du projet et connaître la finalité de leurs tâches. De la même manière, **les documentations nécessaires à chaque étape sont réplicables d'un projet sur l'autre dans leur structure (cahiers des charges, cahiers de test…).**
- En général, le cycle en V est plus adapté aux structures multi sites, car il ne demande pas de réunions quotidiennes, mais seulement des réunions de pilotage actant le passage d'une phase à l'autre. Son aspect linéaire autorise donc une organisation géographique éclatée, où le côtoiement des collaborateurs n'est pas clé dans le processus. Inconvénients

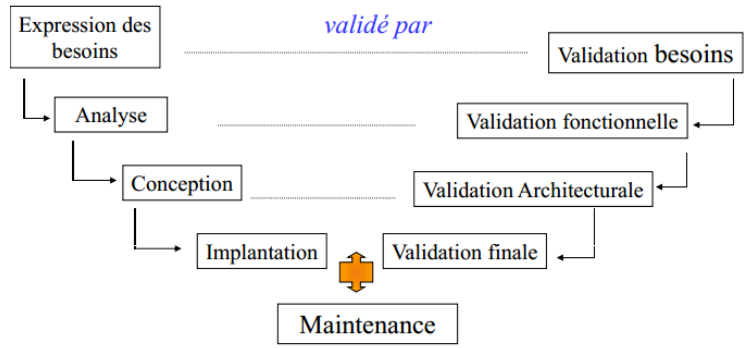
Cycle de vie du développement logiciel

Modèle en V

- Le cycle de vie en V est un modèle de développement logiciel qui représente graphiquement les phases d'un projet et montre les relations entre ces phases de manière symétrique, formant la lettre "V". Chaque étape du développement est associée à une étape de test correspondante.
- **Spécifications :** En haut de la "V", la phase de spécification démarre le processus. Les exigences du système sont définies de manière détaillée, y compris les besoins des utilisateurs et les spécifications du système.
- **Conception Architecturale ou générale :** La conception architecturale se situe en bas de la première branche de la "V". Elle implique la création d'une architecture système détaillée basée sur les spécifications.
- **Conception Détaillée :** En dessous de la conception architecturale, la phase de conception détaillée se concentre sur la conception des composants individuels du système.
- **Implémentation ou Réalisation:** La phase d'implémentation représente le bas de la "V". Elle concerne le codage et le développement réel du logiciel en se basant sur les conceptions détaillées.



Cycle de vie en V (suite)

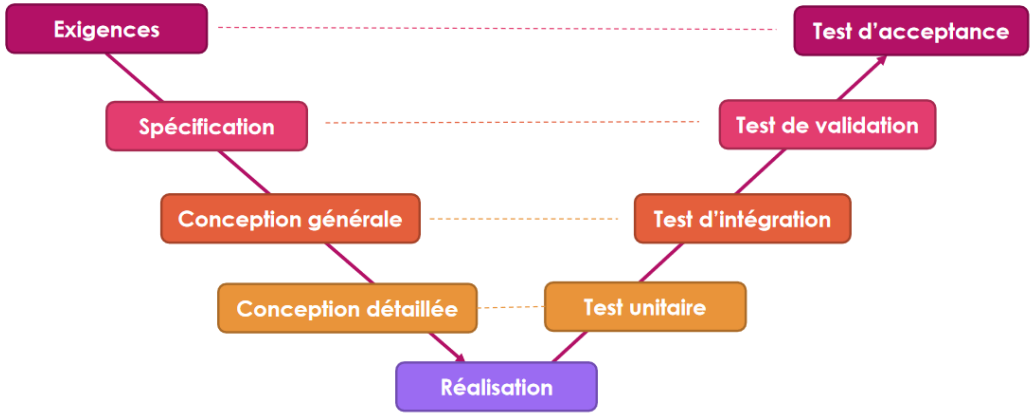


Validation des étapes intermédiaires sous forme de documents

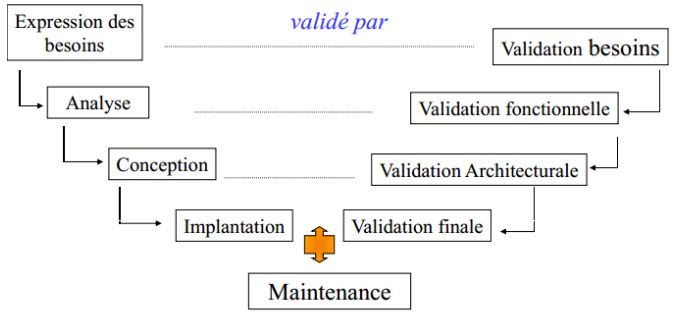
Cycle de vie du développement logiciel

Modèle en V

- **Tests Unitaires :** Les tests unitaires sont effectués pour chaque composant individuel du logiciel afin de s'assurer qu'ils fonctionnent correctement.
- **Tests d'Intégration :** Les composants sont intégrés pour former des sous-systèmes, et les tests d'intégration sont réalisés pour garantir que les différentes parties interagissent correctement.
- **Tests de Validation :** La validation est effectuée pour s'assurer que le système correspond aux attentes des utilisateurs finaux et qu'il fonctionne dans son environnement prévu.
- **Tests d'Acceptation :** En haut de la deuxième branche de la "V", les tests d'acceptation sont réalisés pour s'assurer que le système répond aux exigences métier et peut être accepté par l'utilisateur final.
- **Livraison et Maintenance :** La livraison du produit final se trouve à l'extrémité droite de la "V". Après la livraison, la maintenance peut être nécessaire pour traiter les problèmes post-livraison et effectuer des mises à jour.
- **L'idée centrale du cycle de vie en V est que chaque phase de développement a une phase de test correspondante. Cela permet de détecter et de corriger les erreurs à un stade précoce du processus de développement, réduisant ainsi le risque de défaillance du système..**



Cycle de vie en V (suite)



Validation des étapes intermédiaires sous forme de documents

Cycle de vie du développement logiciel

Modèle en V

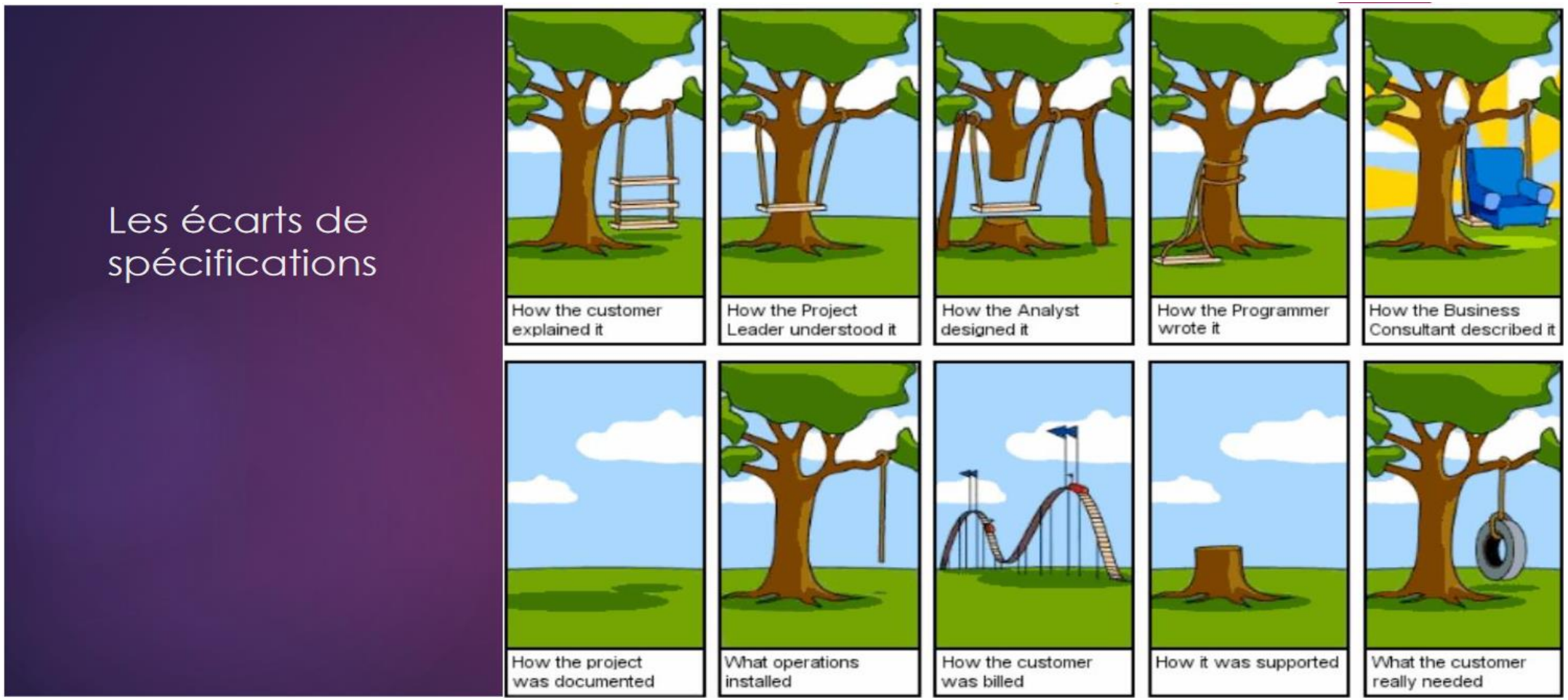
- **Avantages** :
- Met l'accent sur les tests et la validation et par conséquent, ça accroît la qualité du logicie l
- Chaque livrable doit être testable
- Facile à planifier dans une gestion de projets
- Facile à utiliser

- **Inconvénients** :
- Ne gère pas explicitement les changements des spécifications :Après une phase de définition précise du produit auquel doit l'équipe doit aboutir, le projet est lancé dans un « tunnel » constitué des phases évoquées plus haut. Mais que faire si les spécifications initiales sont dépassées ? Si le besoin du client vient à changer, ou a été mal exprimé ? **Le cycle en V supporte donc mal les changements, ce qui est à la fois sa force et sa principale faiblesse.**
- Les tests sont souvent difficiles à rédiger (prévoir les erreurs et les défaillances)
- Le cout du logiciel devient élevé.
- Ce cycle de vie offre moins de réactivité par rapport au contexte technologique et économique, aux demandes du client, aux événements inopinés. L'effet tunnel est aussi induit par le travail conséquent de production de la documentation en début de projet, qui n'est plus rectifiable par la suite. **Enfin, l'image du tunnel illustre le temps (parfois très) long qui sépare l'expression du besoin de la recette du produit final.**

Questions de réflexions :

- Avez-vous déjà développé un logiciel où le besoin était bien claire? Justifiez vos réponses.
- Avez-vous déjà annulé un processus de développement à cause des contraintes technologiques? Lequel? Quelle solutions vous avez envisagé? (expression de besoin, modélisation, …)
- Trouvez vous que le cycle en V est convenable pour un projet basé sur une nouvelle idée innovante? Pourquoi?
- Trouver vous que le cycle en V est convenable pour la création d'une version Web d'une application existante? Pourquoi?

L'importance d'une communication permanente



Sommaire

Cycle de vie du développement logiciel

Prototypage

- Le projet se fait sur plusieurs itérations
- Les développeurs construisent un prototype selon les attentes du client
- Le prototype est évalué par le client
- Le client donne son feedback
- Les développeurs adaptent le prototype selon les besoins du client
- Quand le prototype satisfait le client, le code est normalisé selon les standards et les bonnes pratiques

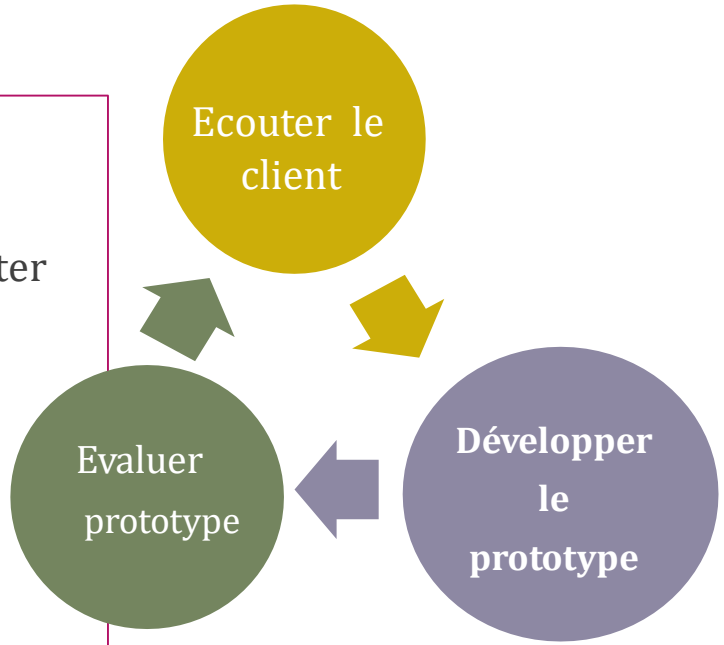
Cycle de vie du développement logiciel

Les Modèles Non linéaires(Agile)

Prototypage

- Avantages :
- Implication active du client.
- Le développeur apprend directement du client
- S'adapte rapidement aux changements des besoins
- Progrès constant et visible
- Une grande interaction avec le produit

- Inconvénients :
- Degré très faible de maintenabilité
- Le processus peut ne jamais s'arrêter
- Très difficile d'établir un planning
- Le prototypage implique un code faiblement structuré



Cycle de vie Prototypage

Sommaire

Cycle de vie du développement logiciel

Modèle Incrémental

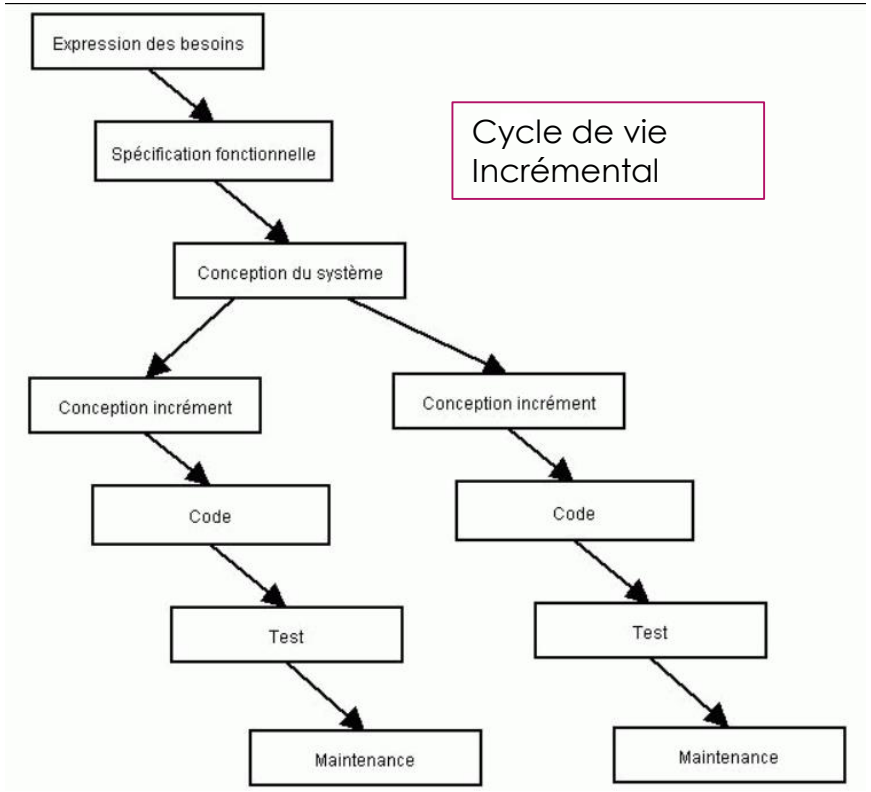
- Dans ce modèle, un seul composant est développé à la fois.
- On débute par définir les exigences et on les décompose en sous-système.
- À chaque version du logiciel, de nouvelles fonctionnalités venant combler les exigences sont ajoutées.
- On continue de la sorte jusqu'à ce que toutes les fonctionnalités demandées soient comblées par le système. Chaque incrément peut utiliser un autre modèle (v, en cascade...)

Cycle de vie du développement logiciel

Les Modèles Non linéaires(Itératifs)

Modèle Incrémental

- L'image n'affiche que deux incrémentations, mais il peut y en avoir plusieurs.
- Le développement sont ainsi moins complexe, l'intégration est moins brutal et le client a plus rapidement un système même s'il n'est pas complet.
- Le cœur du système peut être à revoir lorsqu'on passe à une autre incrémentation du système. Il peut ainsi s'avérer complexe d'ajouter certaines fonctionnalités.
- Ce modèle convient au projet de grande envergure. L'architecture doit être bien pensée dès le départ afin de réduire les risques par la suite.

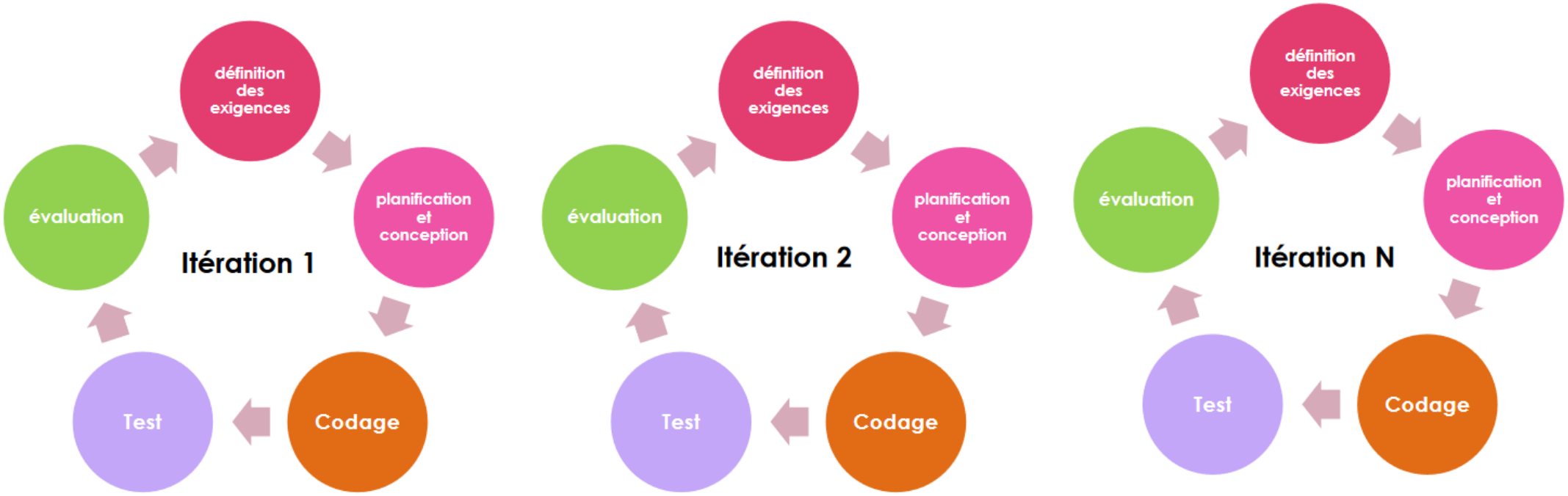


Cycle de vie Incrémental

Cycle de vie du développement logiciel

Les Modèles Non linéaires(Itératifs)

Modèle Incrémental



Cycle de vie du développement logiciel

Modèle Incrémental

Phase d'Initiation :

Dans cette phase, les besoins initiaux sont identifiés et le projet est défini.

Les objectifs généraux sont établis, et une vision initiale du produit est définie.

Premier Incrément :

Une première partie du produit est développée, généralement avec des fonctionnalités de base.

Cette version initiale est souvent appelée "Version 1.0" ou "Premier Incrément".

Évaluation et Rétroaction :

La première itération est évaluée par les parties prenantes, les utilisateurs ou l'équipe de développement.

Les retours d'expérience et les commentaires sont recueillis pour guider les améliorations futures.

Itérations Successives :

Le développement se poursuit par cycles itératifs, chaque incrément ajoutant de nouvelles fonctionnalités ou améliorant celles déjà existantes.

Chaque itération est évaluée et ajustée en fonction des retours.

Livraison de Versions Successives :

À chaque itération, une nouvelle version du produit est livrée.

Les utilisateurs peuvent commencer à bénéficier des fonctionnalités ajoutées à chaque incrément.

Répétition des Itérations :

Le processus d'itération se répète jusqu'à ce que le produit atteigne sa maturité, avec toutes les fonctionnalités nécessaires.

Finalisation et Maintenance :

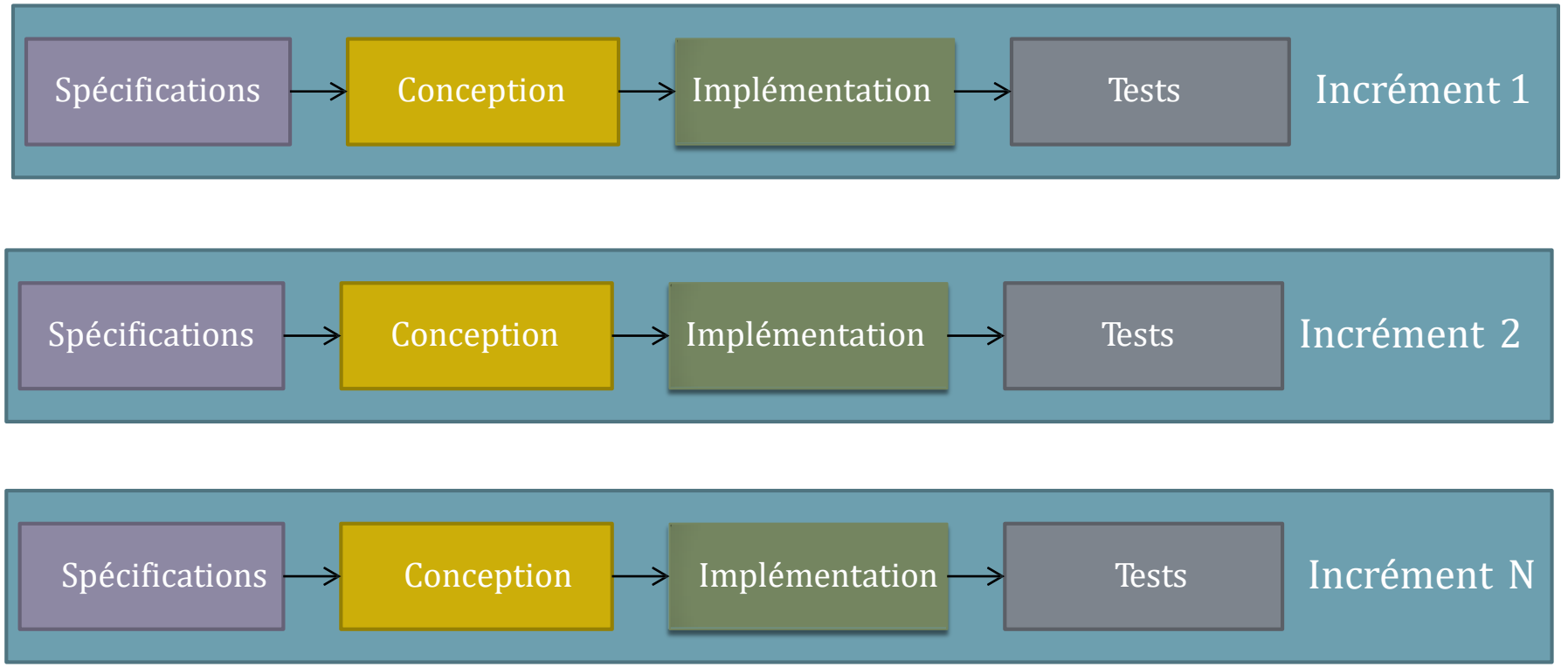
Une fois que toutes les fonctionnalités ont été développées et que le produit répond aux exigences, il est finalisé.

La maintenance continue peut être nécessaire pour les correctifs de bogues, les mises à jour de sécurité, etc.

Cycle de vie du développement logiciel

Les Modèles Non linéaires(Itératifs)

Modèle Incrémental



Cycle de vie du développement logiciel

Les Modèles Non linéaires(Itératifs)

Modèle Incrémental

- **Les Avantages :**

- Chaque incrément donne un produit fonctionnel.
- Le client intervient à la fin de chaque incrément.
- Utiliser l'approche « diviser pour régner ».
- Meilleur Intégration du client dans le processus. Le client entre en relation avec le produit très tôt.
- Développement de fonctionnalités à risque en premier

- **Les Inconvénients :**

- Exige une bonne planification et une bonne conception.
- Exige une vision sur le produit fini pour pouvoir le divi ser en incréments.
- Le coût total du système peut être cher.

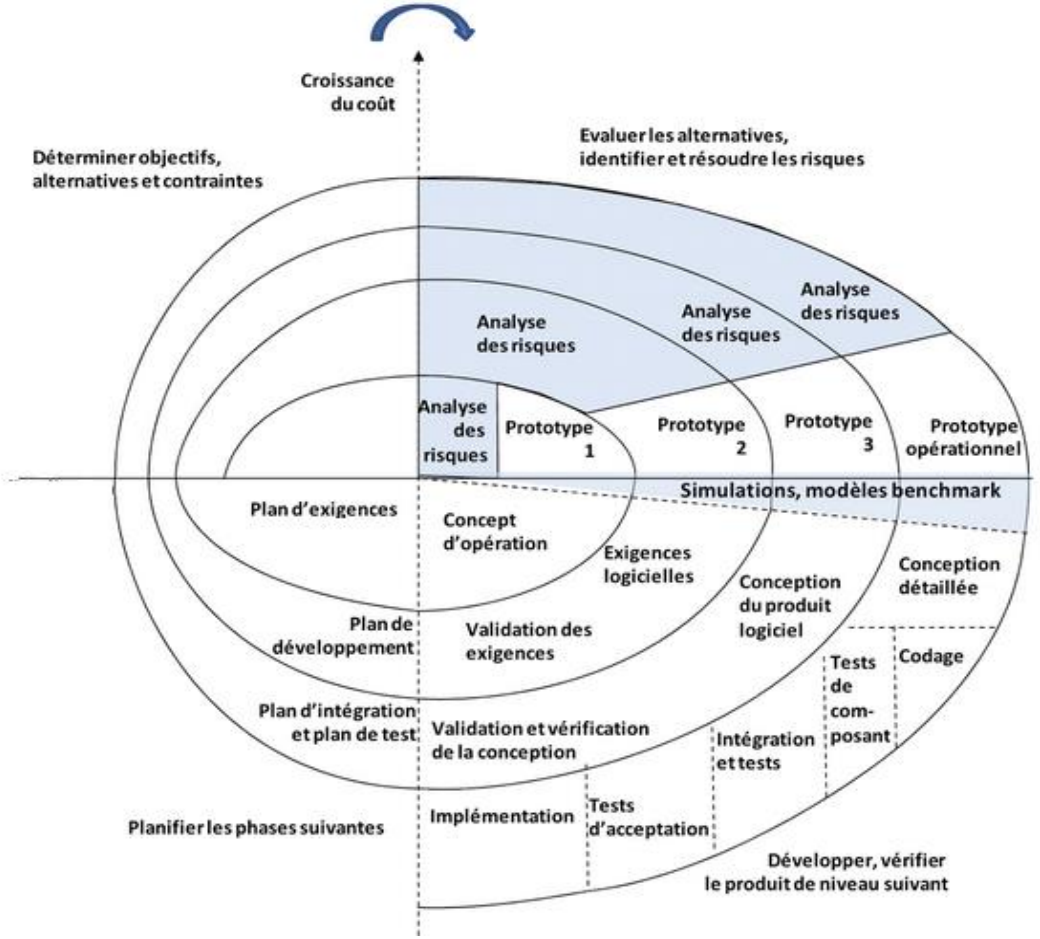
Sommaire

Cycle de vie du développement logiciel

Les Modèles Non linéaires(Itératifs)

Modèle Spirale

- Des incréments sous forme de cycles.
- À la fin de chaque cycle on détermine les objectifs du cycle suivant.
- Chaque cycle est composé des même activités que du modèle en cascade.
- Inclut l'analyse de risque et le prototypage.



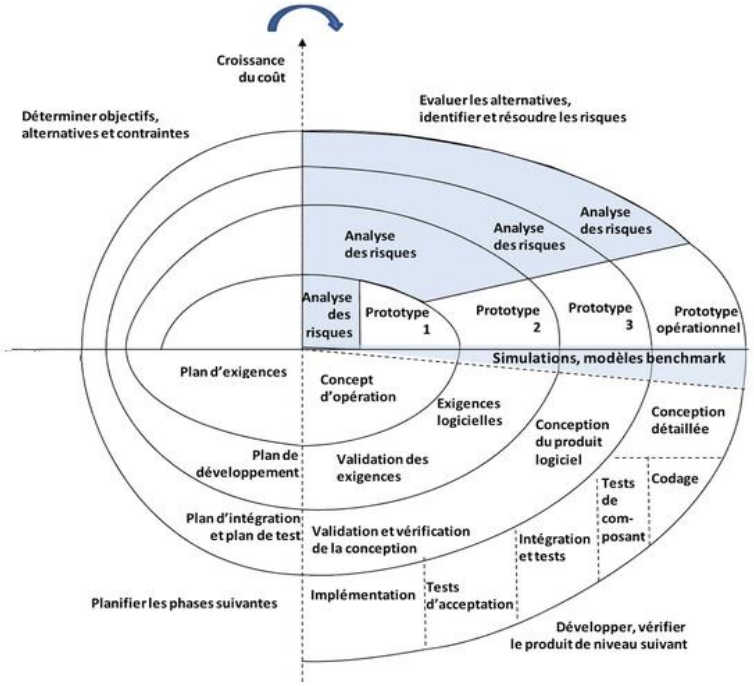
Cycle de vie du développement logiciel

Les Modèles Non linéaires(Itératifs)

Modèle Spirale

- **Il a été proposé par Barry Boehm dans les années 1980. Le modèle en spirale met l'accent sur la gestion des risques tout au long du cycle de vie du projet et est itératif par nature. Il est particulièrement adapté aux projets complexes et à haut risque.**

- **Définition des Objectifs :** La première phase commence par l'identification des objectifs spécifiques du projet, ainsi que des contraintes et des besoins des parties prenantes. Cette étape établit la base pour les phases suivantes du modèle.
- **Évaluation des Risques** : Une évaluation des risques est effectuée pour identifier et évaluer les risques potentiels associés au projet. Les risques peuvent inclure des aspects tels que des technologies non éprouvées, des contraintes budgétaires, ou des incertitudes dans les exigences.
- **Développement et Validation d'un Prototype** : Une fois les risques identifiés, une version prototype ou un concept initial est développé pour valider les idées et tester les concepts. Cela permet d'obtenir des retours d'expérience et de vérifier la faisabilité technique.
- **Planification** : La planification du projet est revue et mise à jour en fonction des retours obtenus. Les ressources nécessaires, les échéanciers et les coûts sont réévalués pour tenir compte des nouvelles informations.

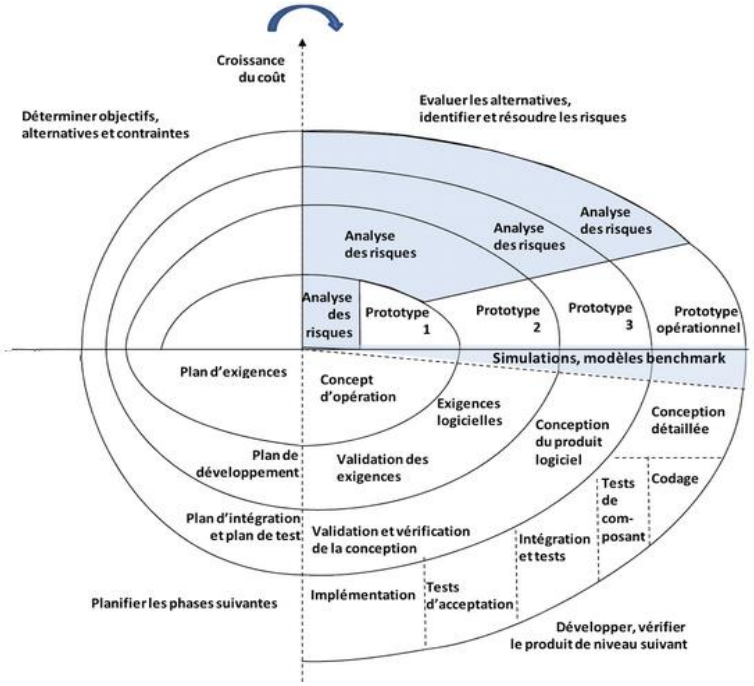


Cycle de vie du développement logiciel

Les Modèles Non linéaires(Itératifs)

Modèle Spirale

- **Implémentation :** Cette étape implique le développement réel du produit basé sur les résultats des étapes précédentes. Les itérations peuvent être utilisées pour ajouter progressivement des fonctionnalités et améliorer le produit.
- **Évaluation :** L'évaluation est réalisée à chaque itération pour vérifier si les objectifs du projet sont atteints et pour identifier tout risque ou problème potentiel. Cette étape peut également inclure des évaluations de performance et de sécurité.
- **Planification de la Prochaine Étape :** En fonction des résultats de l'évaluation, une planification est effectuée pour la prochaine itération ou phase du projet. Cela inclut la définition des objectifs, la réévaluation des risques et la planification des activités futures.
- **Révision avec les Parties Prenantes :** Les résultats et les plans pour la prochaine étape sont examinés avec les parties prenantes du projet. Cela garantit une communication continue et une prise de décision éclairée.
- **Le modèle en spirale est cyclique, et ces étapes se répètent à plusieurs reprises jusqu'à ce que le produit atteigne sa maturité et réponde aux exigences définies. Chaque itération de la spirale représente un cycle complet de planification, d'implémentation, d'évaluation et d'ajustement. L'approche en spirale met l'accent sur la gestion proactive des risques, ce qui en fait un modèle adapté aux projets complexes où l'incertitude et la variabilité sont élevées.**



Cycle de vie du développement logiciel

Modèle Spirale

- Identification et Evaluation des risques :
- Etudier les alternatives de développement
- Identification des risques : technologie non maîtrisées, équipe peu expérimentée, planning trop serré, ...etc.
- Evaluation des risques : voir si les risques peuvent impacter le projet et à quel niveau

		Impact: quelle est la sévérité du risque ?		
Probabilité que le risque aura lieu	Très probable	Risque **Acceptable Moyen 2**	Risque **Inacceptable Elevé 3**	Risque **Inacceptable Extrême 5**
	Probable	Risque **Acceptable Bas 1**	Risque **Acceptable Moyen 2**	Risque **Inacceptable Elevé 3**
	Improbable	Risque **Acceptable Bas 1**	Risque **Acceptable Bas 1**	Risque **Acceptable Moyen 2**
		Limité	Moyen	Important

Cycle de vie du développement logiciel

Modèle Spirale

- **Les Avantages :**

- Identification rapide des risques.
- Impacts minimaux des risques sur le projet
- Fonctions critiques développées en premier
- Feedback rapide du client
- Une évaluation continue du procédé

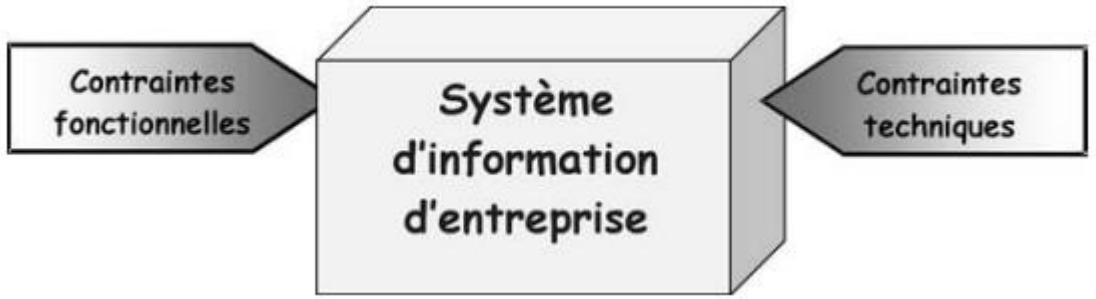
- **Les Inconvénients :**

- L'évaluation du risque peut prendre beaucoup de temps.
- Le modèle est très complexe
- La spirale peut s'éterniser
- Les développeurs doivent être réaffectés pendant les phases de non-développement
- Les objectifs ne sont pas souvent faciles à formuler

Comment appliquer un processus unifié ?

Définition 2TUP

- 2TUP signifie «2 Track Unified Process ». C'est un processus UP qui répond aux caractéristiques que nous venons de citer.
- «2 Track » signifie littéralement que le processus suit deux chemins. Il s'agit des chemins « fonctionnels » et «d'architecture technique », qui correspondent aux deux axes de changement imposés au système informatique



Définition 2TUP

- L'axiome fondateur du 2TUP consiste à constater que toute évolution imposée au système d'information peut se décomposer et se traiter parallèlement, suivant un axe fonctionnel et un axe technique. Pour illustrer cet axiome, prenons les trois exemples suivants :

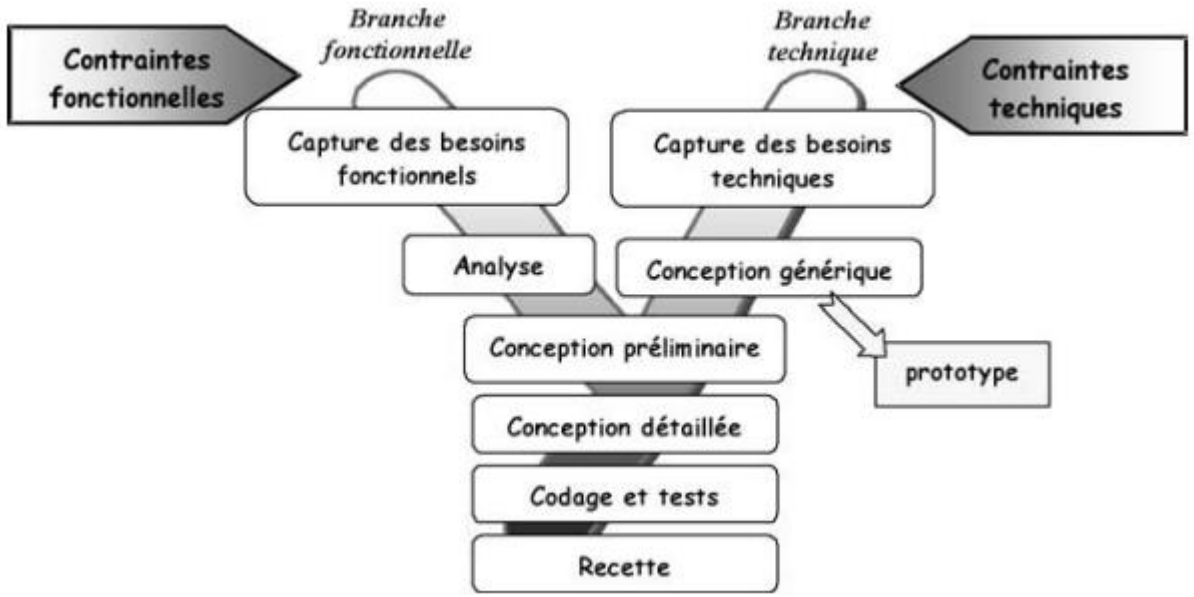
- 1. une agence de tourisme passe des accords avec une compagnie aérienne de sorte que le calcul des commissions change. En l'occurrence, les résultats issus de la branche fonctionnelle qui évoluent pour prendre en compte la nouvelle spécification ;

- 2. cette même entreprise décide d'ouvrir la prise de commande sur le Web. Si rien ne change fonctionnellement, en revanche, l'architecture technique du système évolue ;

- 3. cette entreprise décide finalement de partager son catalogue de prestations avec les vols de la compagnie aérienne. D'une part, la fusion des deux sources d'informations imposera une évolution de la branche fonctionnelle, d'autre part, les moyens techniques de synchronisation des deux systèmes conduiront à étoffer l'architecture technique du système. L'étude de ces évolutions pourra être menée indépendamment, suivant les deux branches du 2TUP.

- À l'issue des évolutions du modèle fonctionnel et de l'architecture technique, la réalisation du système consiste à fusionner les résultats des deux branches

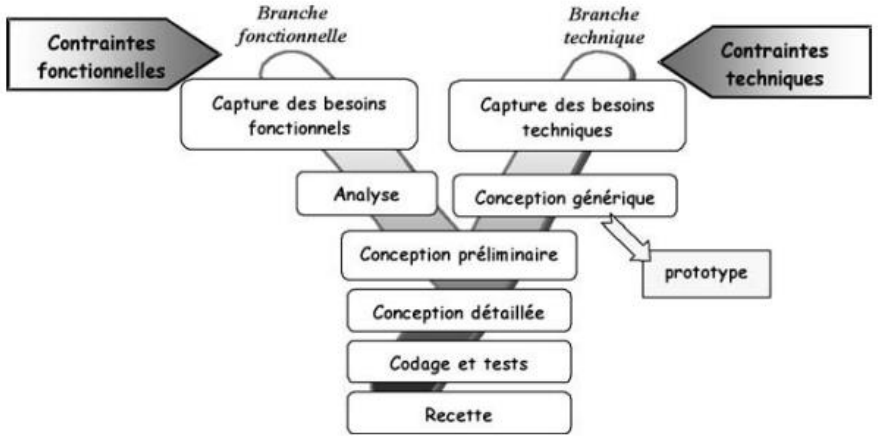
Cycle de vie du processus 2TUP

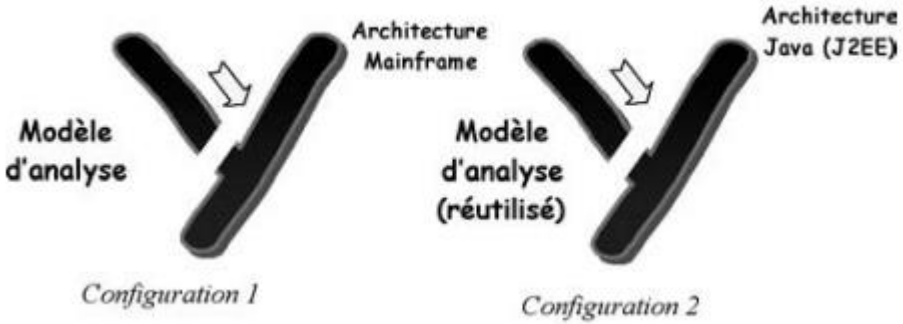


Cycle de vie du processus

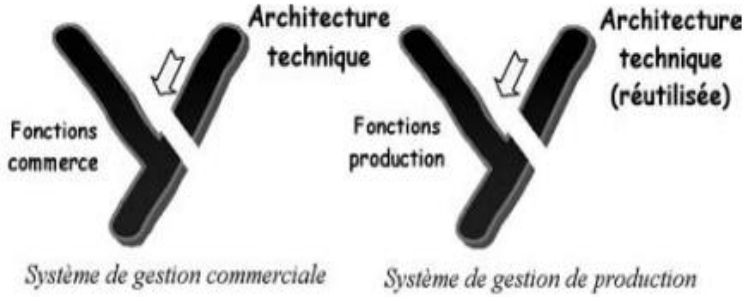
Cycle de vie du processus 2TUP : la branche fonctionnelle

- **La branche gauche (fonctionnelle) comporte : la capture des besoins fonctionnels, qui produit un modèle des besoins focalisé sur le métier des utilisateurs**.
  - Cette branche consiste à étudier précisément la spécification fonctionnelle de manière à obtenir une idée de ce que va réaliser le système en termes de métier. Les résultats de l'analyse ne dépendent d'aucune technologie particulière.
- La branche gauche capitalise la connaissance du métier de l'entreprise. Elle constitue généralement un investissement pour le moyen et le long terme.
- Les fonctions du système d'informations sont en effet indépendantes des technologies utilisées. Cette évidence n'a malheureusement pas souvent été mise en pratique, car dans bien des cas, la connaissance fonctionnelle d'un produit se perd dans les milliers de ligne de code de sa réalisation.
- L'entreprise qui maintient le modèle fonctionnel de sa branche gauche est pourtant à même de le réaliser sous différentes technologies. Il suffit de« greffer » une nouvelle architecture technique pour mettre à jour un système existant.





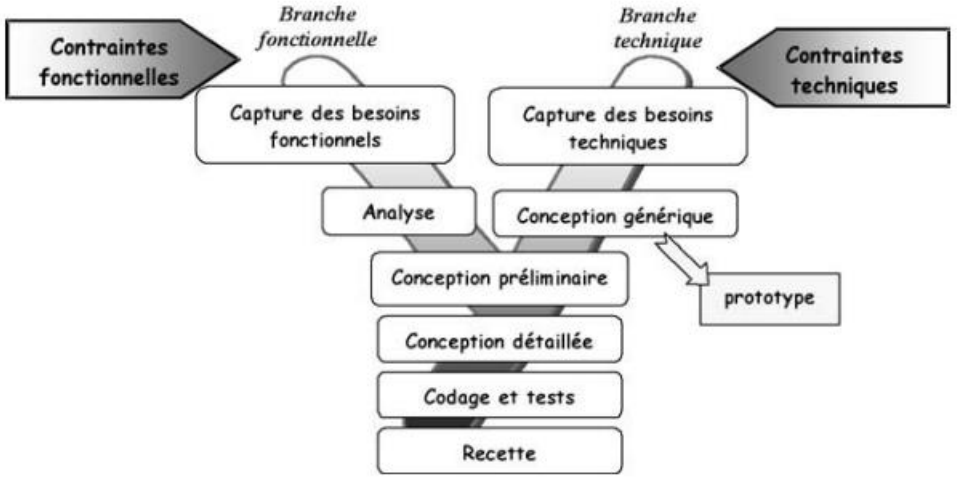
Cycle de vie du processus 2TUP : la branche technique



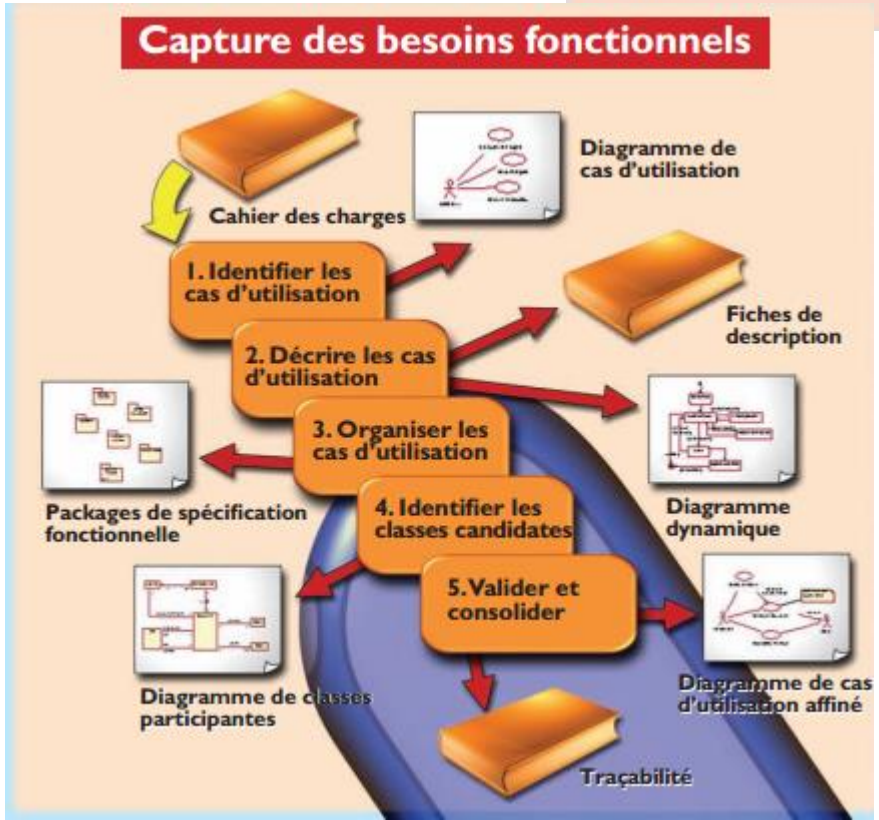
- **La branche droite (architecture technique) comporte : la capture des besoins techniques, qui recense toutes les contraintes et les choix dimensionnant la conception du système. Les outils et les matériels sélectionnés ainsi que la prise en compte de contraintes d'intégration avec l'existant conditionnent généralement des prérequis d'architecture technique ;**
  - Cette conception est la moins dépendante possible des aspects fonctionnels. Elle a pour objectif d'uniformiser et de réutiliser les mêmes mécanismes pour tout un système. L'architecture technique construit le squelette du système informatique et écarte la plupart des risques de niveau technique. L'importance de sa réussite est telle qu'il est conseillé de réaliser un prototype pour assurer sa validité.
- La branche droite capitalise quant à elle un savoir-faire technique. Elle constitue un investissement pour le court et le moyen terme. Les techniques développées pour le système peuvent l'être en effet indépendamment des fonctions à réaliser.
- L'architecture technique est d'ailleurs de moins en moins la préoccupation des services informatiques dont l'entreprise n'a pas vocation à produire du code. L'existence de produits tels que les serveurs d'application ou la standardisation des services Web reflète cette tendance à pouvoir disposer sur le marché d'architectures techniques « prêtes à intégrer ».
- Une architecture technique est en effet immédiatement réutilisable pour les différentes composantes fonctionnelles

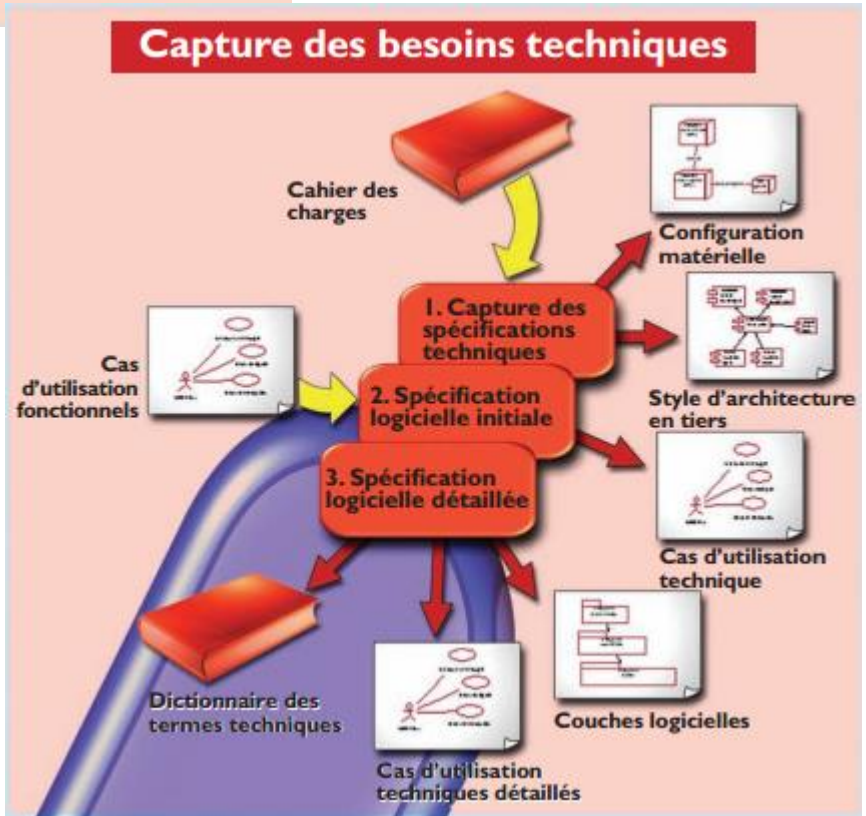
Cycle de vie du processus 2TUP

- La branche du milieu comporte : la conception préliminaire, qui représente une étape délicate, car elle intègre le modèle d'analyse dans l'architecture technique de manière à tracer la cartographie des composants du système à développer ;
  - La conception détaillée, qui étudie ensuite comment réaliser chaque composant ;
  - l'étape de codage, qui produit ces composants et teste au fur et à mesure
    les unités de code réalisées ;
  - l'étape de recette, qui consiste enfin à valider les fonctions du système
    développé.

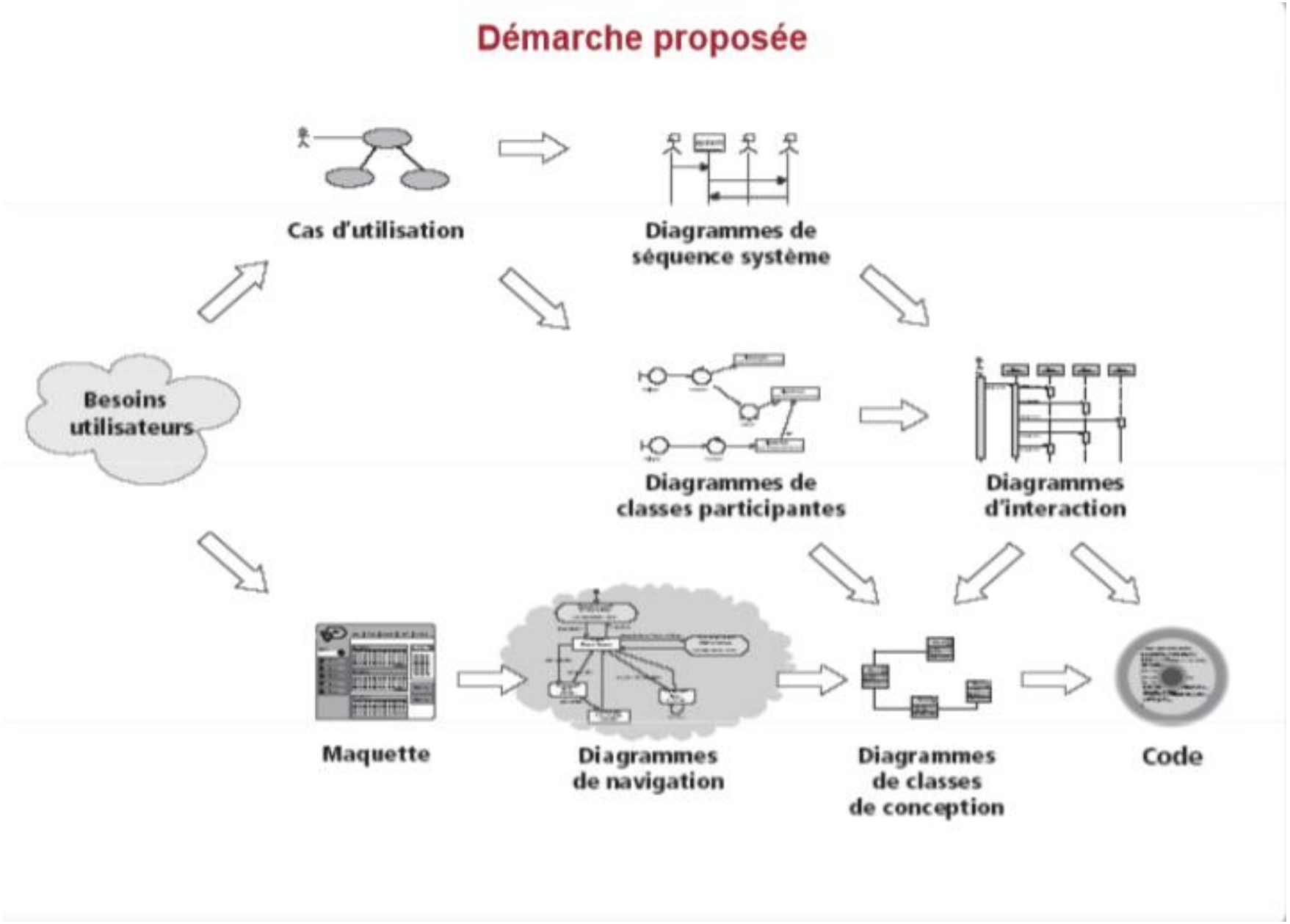


LES Étapes 2TUP



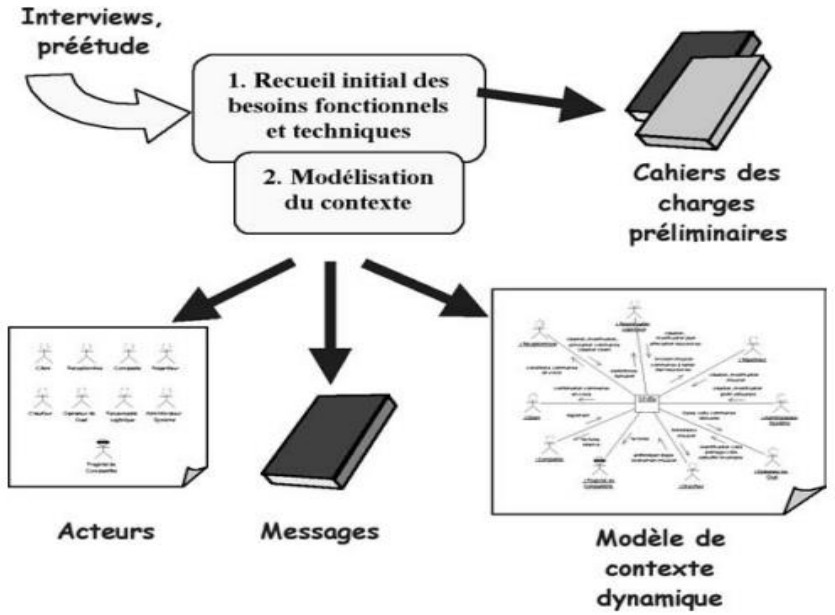


Démarche proposée



Etude préliminaire

- L'étude préliminaire (ou pré étude) est la toute première étape de notre processus de développement.
- Elle consiste à effectuer un premier repérage des besoins fonctionnels et opérationnels, en utilisant principalement le texte, ou des diagrammes très simples.
- Elle prépare les activités plus formelles de capture des besoins fonctionnels et de capture des besoins techniques.



Étape 1 : Modélisation Métier

Résultats :

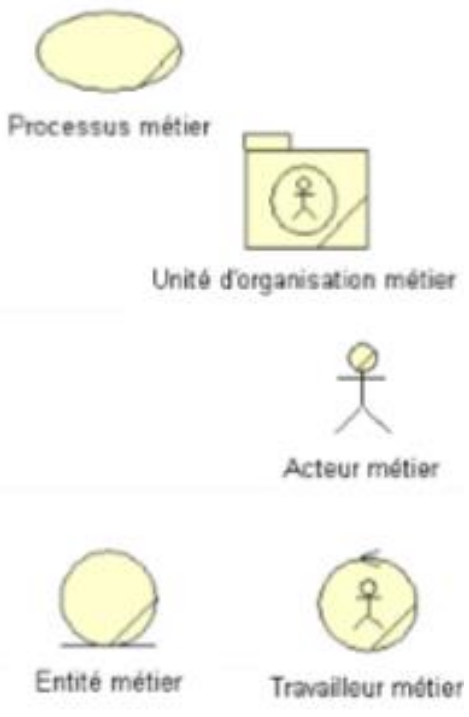
- Diagramme d'activité
- Cahier des charges fonctionnelles

Démarche :

❑ Description du processus métier à l'aide d'un diagramme de cas d'utilisation stéréotypé

STEREOTYPES DE JACOBSON
Représentent en modélisation métier les concepts UML d'acteur, cas d'utilisation, classe, package, etc., (fournit par Rational/Rose) :

- *Cas d'utilisation Stéréotypé, appelé processus métier*
- *Acteur stéréotypé, représentant une entité externe à l'entreprise*
- *Classe stéréotypée représentant un humain agissant à l'intérieur de l'entreprise*
- *Classe stéréotypée, représentant une entité passive manipulée par un travailleur métier*
- *Package stéréotypé, structurant le modèle métier*



Étape 2 : Spécification des exigences par les cas d'utilisation

Résultats :

- Diagramme de cas d'utilisation

Démarche :

- Identifier les acteurs
- Identifier les cas d'utilisation
- Structurer les cas d'utilisation en packages
- Ajouter les relations entre cas d'utilisation
- Établir les priorités entre cas d'utilisation en tenant compte des deux facteurs :
    - la priorité fonctionnelle (service Marketing)
    - le risque technique (chef de projet)
- Planification du projet en itérations

Concepts liés : voir cours modélisation fonctionnelle

Structuration des cas d'utilisation

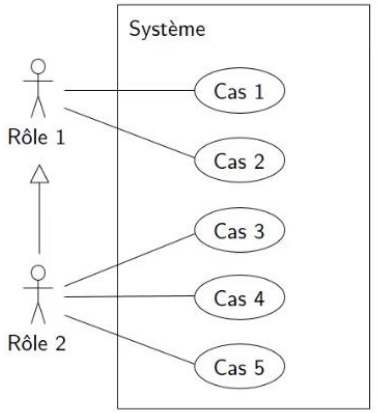
Structuration des cas d'utilisation en packages.

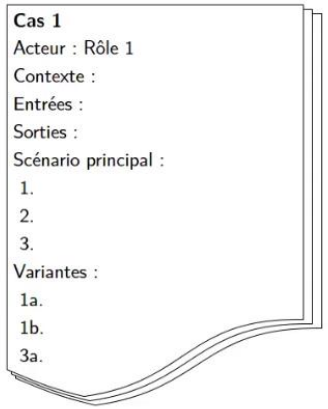
- Pour rendre plus lisible un diagramme de cas d'utilisation, on fait un regroupement en un ensemble de package.
- Plusieurs stratégies sont possibles : par acteur, par domaine fonctionnel,….

Exp. : cas du GAB : Regroupement par acteur principale

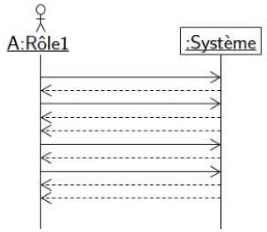
Étape 3 : Spécification détaillée des exigences

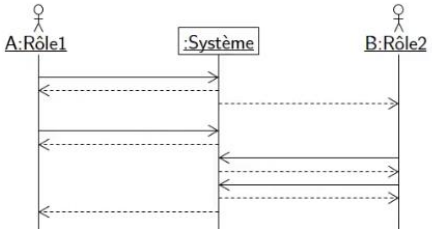
Diagrammes de cas d'utilisation + Description textuelle





+





Résultats pour chaque itération :

- Maquette
- description textuelle
- diagramme de séquence système SN
- DS des S alternatif, erreur + D activité

Démarche :

- décrire textuellement chaque cas d'utilisation
- élaborer les diagrammes de séquence système pour chaque cas d'utilisation

Description textuelle du cas d'utilisation

Cas d'utilisation détaillé

Description textuelle d'un cas d'utilisation

- Nom du cas d'utilisation
- Brève description
- Acteurs
- Contexte
- Données en entrée et pré-conditions
- Données en sortie et post-conditions
- Scénario principal pour ce cas d'utilisation : étapes à suivre pour réaliser ce cas
- Variantes, cas d'erreur : déviations des étapes du scénario principal, scénarios alternatifs, scénarios d'erreur

Description textuelle du cas d'utilisation

Nom du cas d'utilisation
Acteur principal
Acteurs secondaires
Objectifs
Préconditions
Postconditions
Scénario nominal
1.
2.
3.
....
Alternatives
1a
1.
2.
....
1b
1.
2.
3.
....
2a
1.
2.
....

Nom : Commander

Acteur : Client

Données d'entrée : Produits sélectionnés par le client

Le cas d'utilisation commence lorsque le client clique sur le bouton « Commander »

Scénario principal :

1. Le système demande au client de saisir son identifiant et son mot de passe
2. Le client saisit son identifiant et son mot de passe et valide
3. Le système demande au client de choisir son adresse de livraison parmi sa liste d'adresses ou d'en saisir une nouvelle
4. Le client choisit une adresse de livraison et valide
5. Le système demande au client de choisir un mode d'expédition parmi une liste prédéfinie (à préciser)
6. Le client choisit un mode d'expédition et valide
7. Le système affiche un récapitulatif de la commande, indique le montant total de la livraison et demande au client de choisir un mode de paiement parmi une liste prédéfinie (à préciser)
8. Le client choisit un mode de paiement et valide
9. Le système demande au client de saisir ses informations de paiement
10. Le client saisit ses informations de paiement et valide
11. Le système informe le client que la transaction s'est effectuée correctement et un e-mail récapitulatif de la commande est envoyé au client

Description textuelle du cas d'utilisation

Scénario d'erreur : Client inconnu

3a. Le client n'est pas connu du système. Le système affiche un message d'erreur. Retour à l'étape 1.

Scénario alternatif : Nouvelle adresse de livraison

4a. Le client saisit une nouvelle adresse de livraison et valide.

Le scénario reprend à l'étape 5

Scénario alternatif : Modifications des choix de livraison

8a. Le client demande à modifier son adresse de livraison.

Retour à l'étape 3.

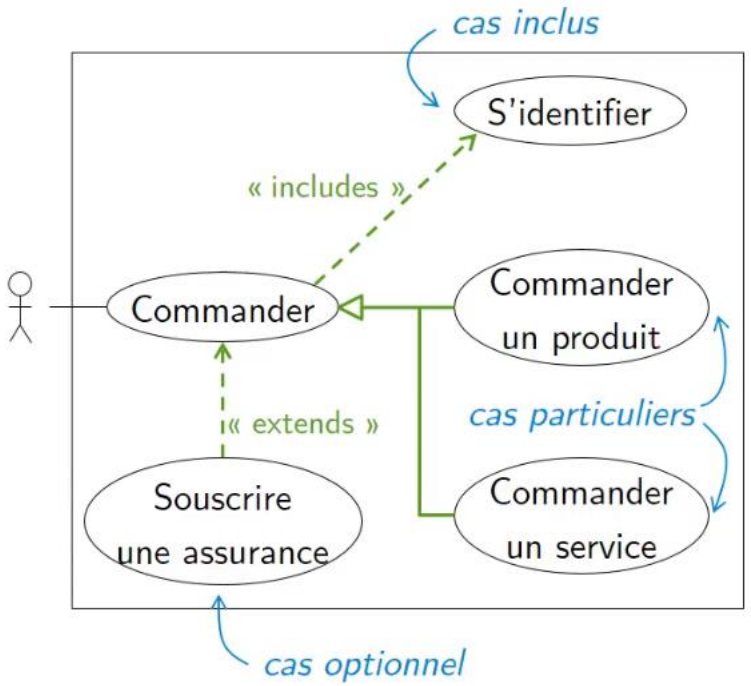
8b. Le client demande à modifier le mode de livraison.

Retour à l'étape 5.

Scénario d'erreur : Transaction impossible

11a. Le système informe le client que ses informations de paiement sont incorrectes. Retour à l'étape 9.

Les relations entre les diagrammes



Commander

Acteur : Client

Le cas commence lorsque le client clique sur « Commander »

Scénario principal

1. Déclenchement du cas S'identifier
2. Fin du cas S'identifier
3. ...

4.1 L'objet de la commande est un produit.

Déclenchement du cas Commander un produit.

4.2 L'objet de la commande est un service.

Déclenchement du cas Commander un service.

Scénario alternatif

3a. Le client choisit de prendre une assurance.

Étape 4 : Réalisation des cas d'utilisation : classe d'analyse

Résultats :

• modèle de domaine (diagramme de classe métier)

• diagramme d'état (pour les classes ayant un comportement dynamique complexe)

• diagramme de classe participante (diagramme de classe d'analyse) pour chaque itération

Démarche :

• identification des concepts de domaine

• ajout des associations et des attributs

• Élaboration du (des) diagramme(s) d'état pour les classes présentant un comportement dynamique complexe

• identification des classes d'analyse participantes pour chaque itération

Étape 5 : Modélisation de la navigation

Résultats *pour chaque itération* : Diagramme d'état de la navigation (utile pour les applications web)

Démarche :

- Identification des états (type de dialogues)
- Structuration de la navigation : par acteur, cas d'utilisation (indépendants).

Concepts liés : *Notation de base* : voir modélisation dynamique

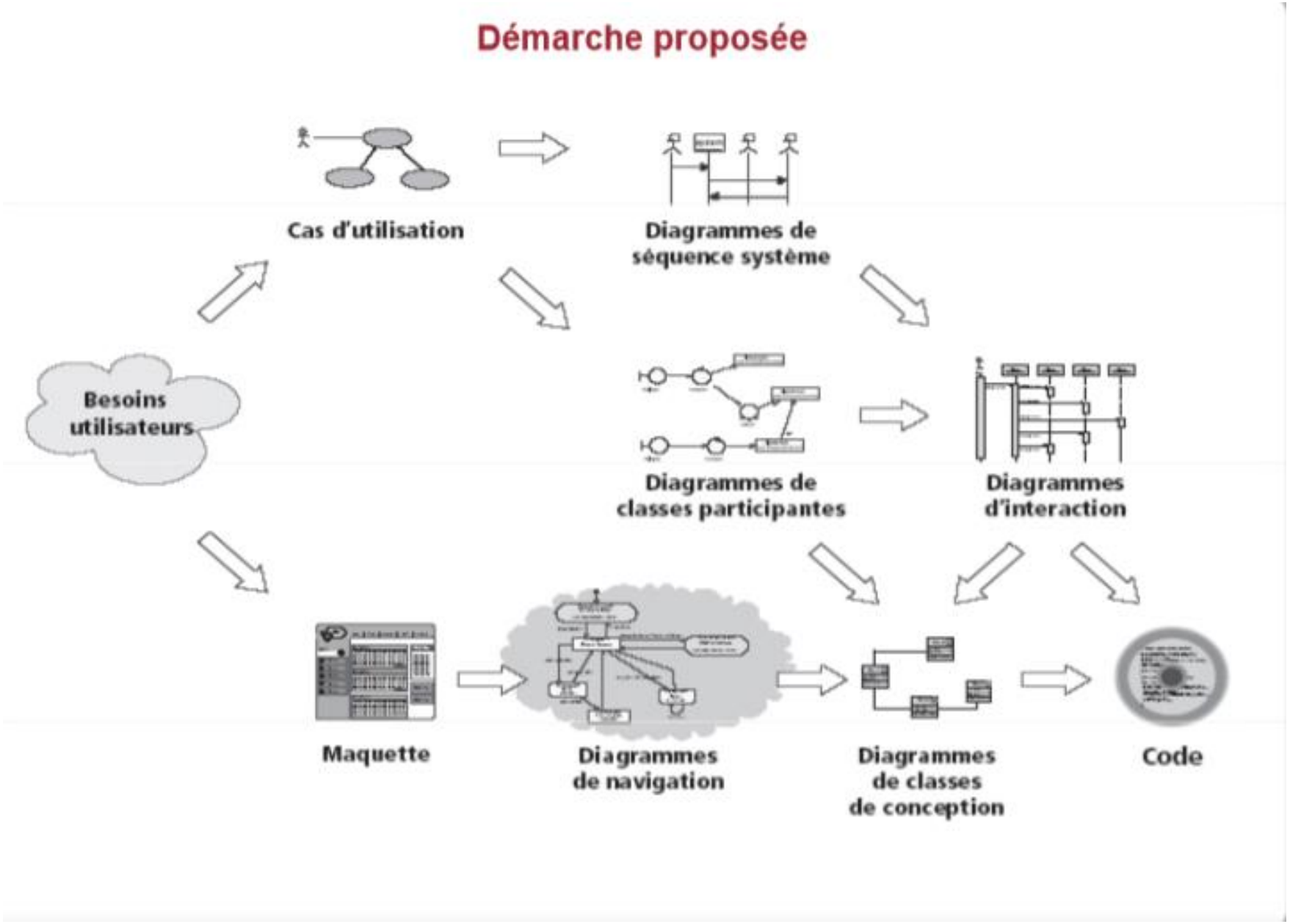
Modélisation de la navigation :

- des états pour représenter les classes dialogues
- des transitions entre états déclenchées par des événements et pouvant porter des conditions, pour représenter les actions IHM.

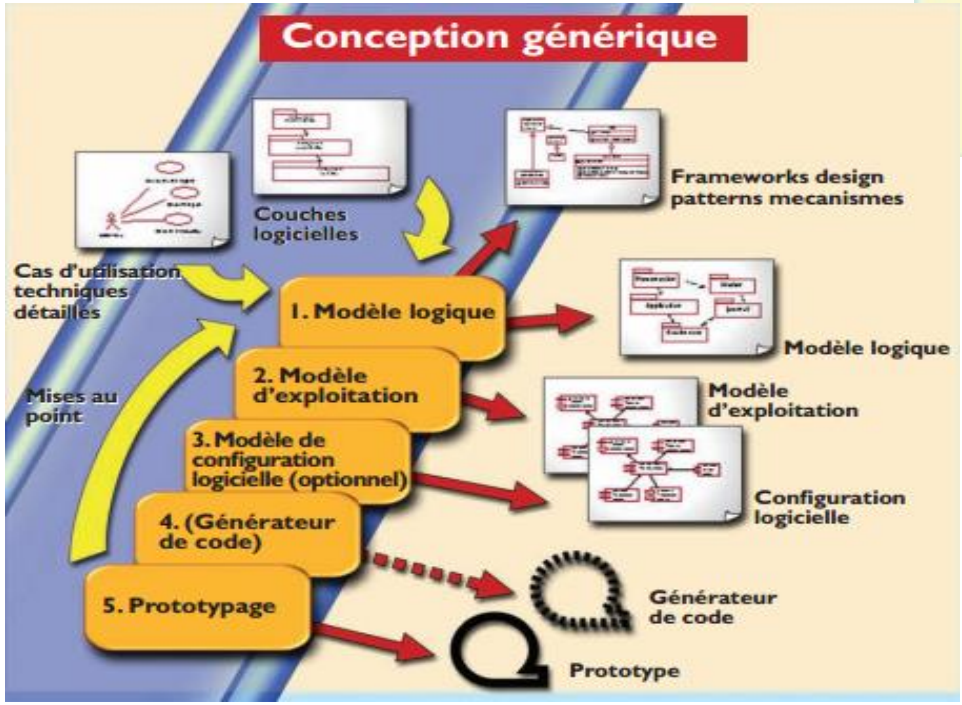
Conventions spécifiques :

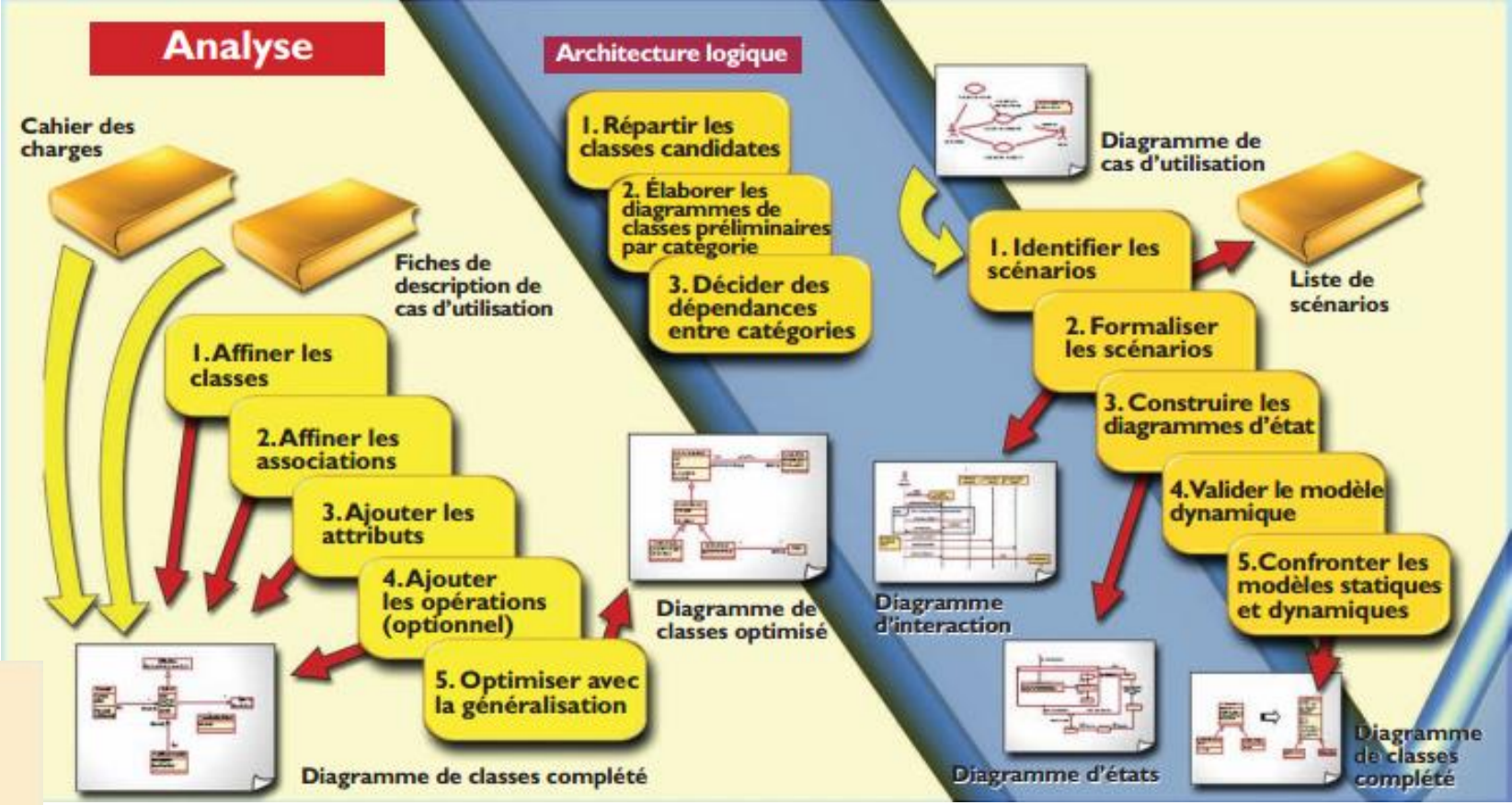
- une page complète du site (« page »)
- un frame particulier à l'intérieur d'une page (« frame »)
- une erreur ou un comportement inattendu du système (« exception ») (gris intermédiaire)
- une liaison avec un autre diagramme de navigation pour des raisons de structuration et de lisibilité (« connector »).

Démarche proposée

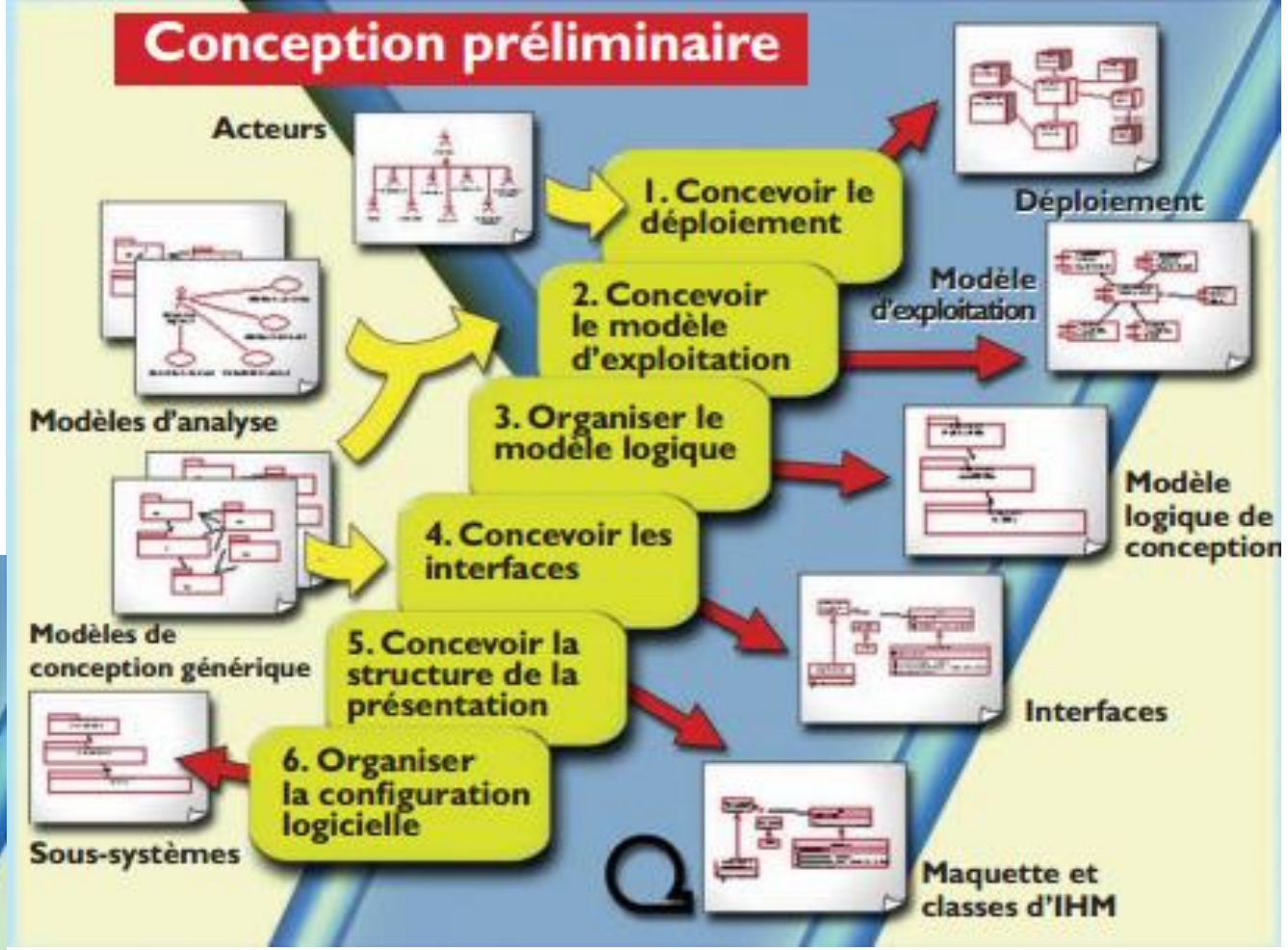


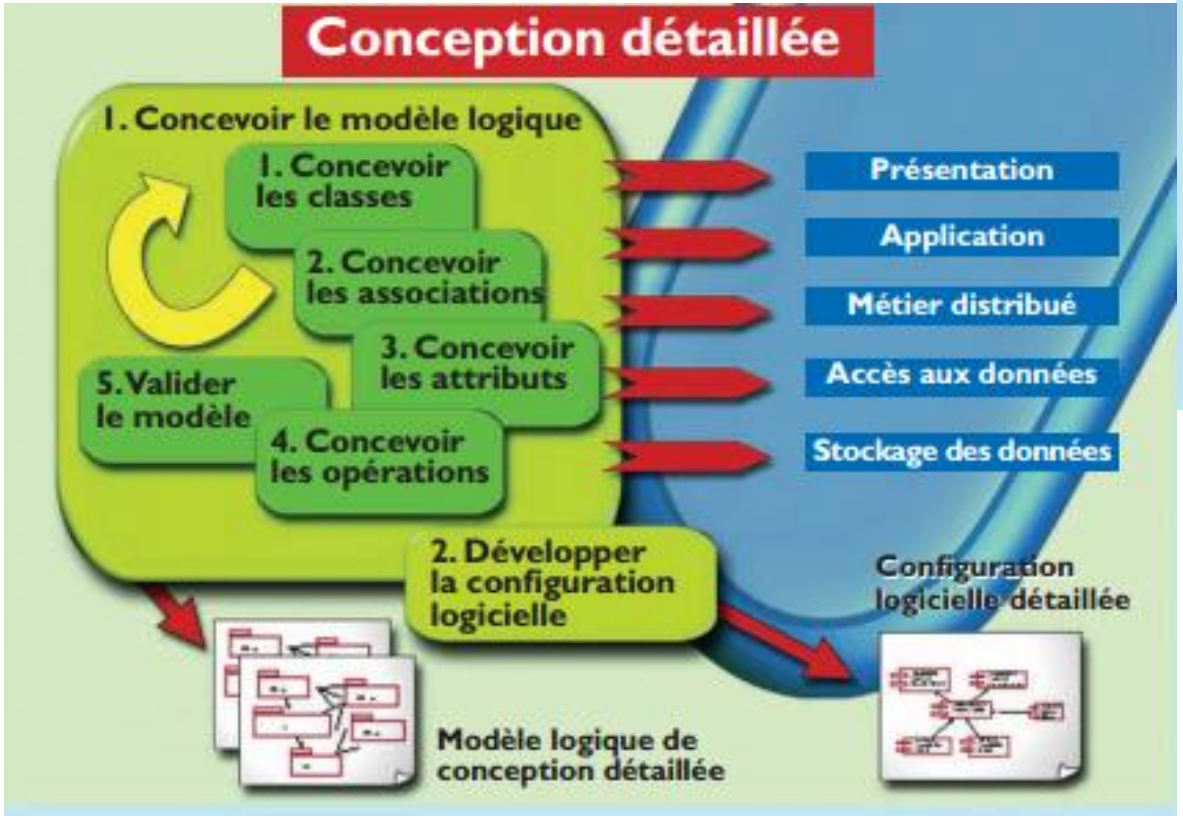
LES Étapes 2TUP



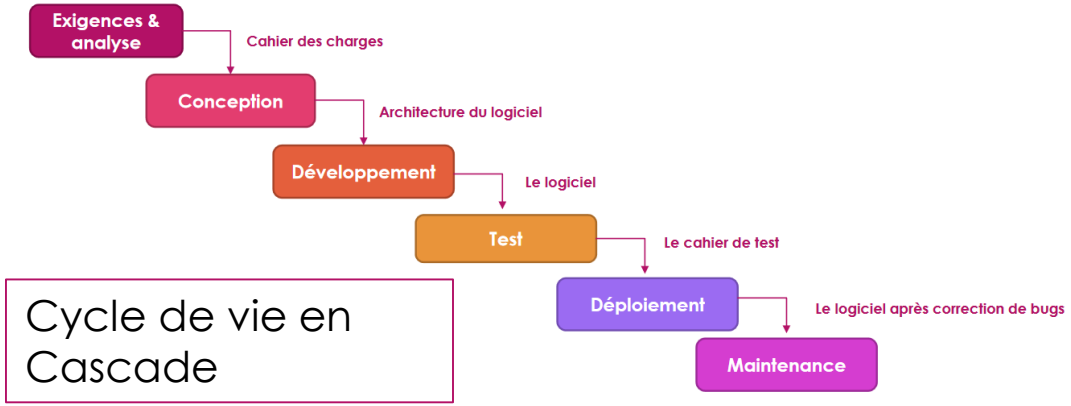


LES Étapes
2TUP

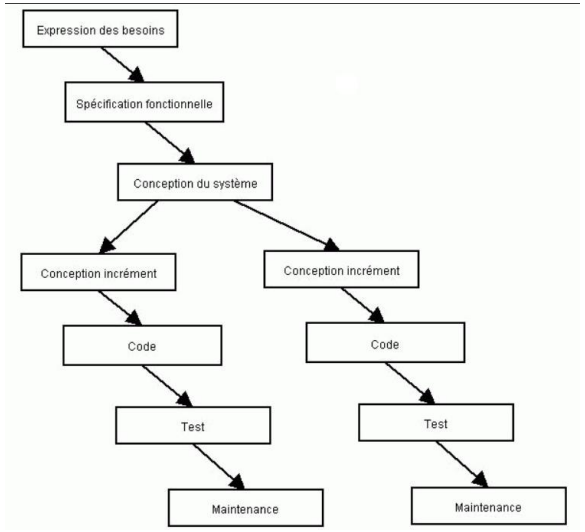




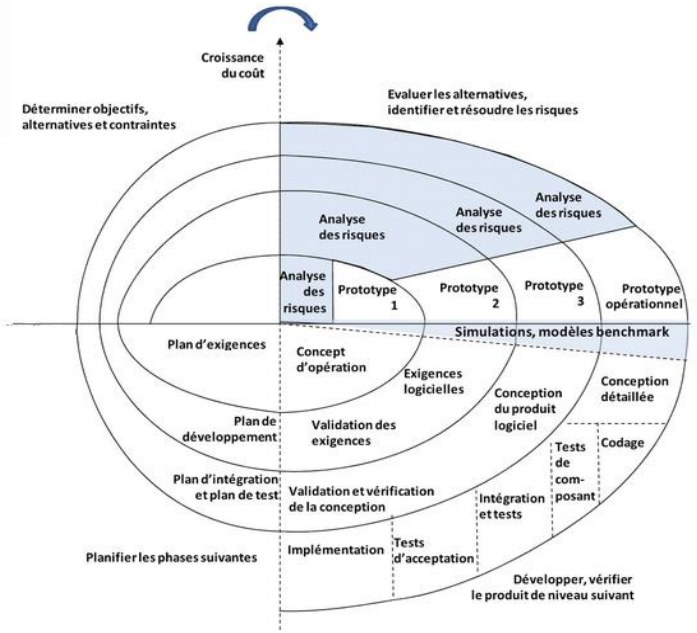
Comparaison & Discussion



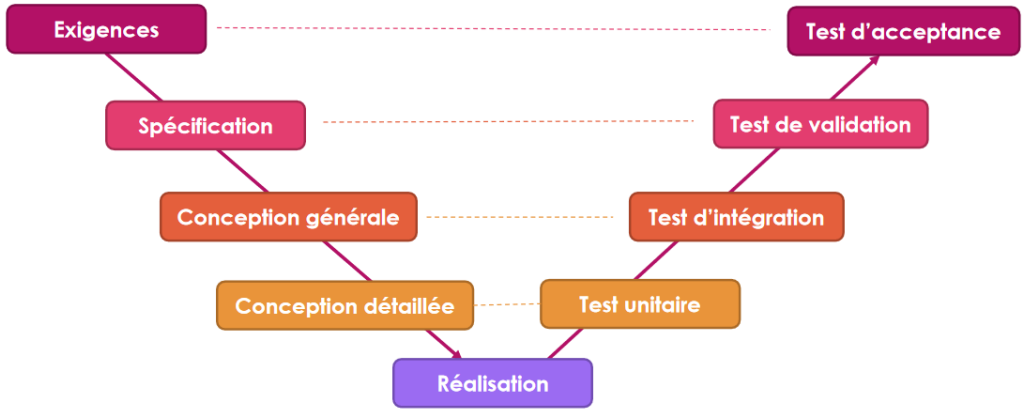
Cycle de vie en Cascade



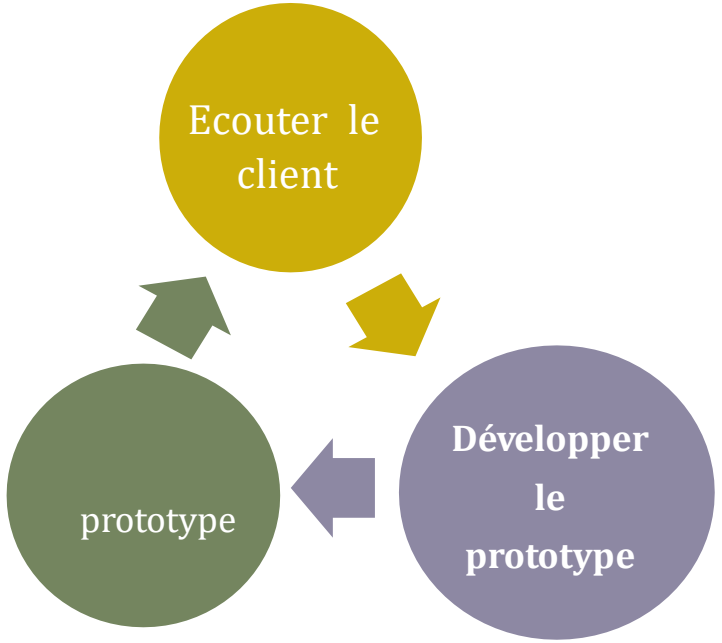
Cycle de vie Incrémental



Cycle de vie Spiral



Cycle de vie en V



Cycle de vie Prototypage