

Retours

- Retour sur COO
- Déroulements des TDs
- Canevas de Rapport



Retour de COO

- Faux cas d'utilisation (des interactions entre humains)
- Trop modéliser n'est pas modéliser juste. Bien délimiter le sujet, comprendre ce que l'on me demande et bien le faire.
- Les diagrammes de classe
 - les acteurs apparaissent très souvent sous forme de classe !
 - relation, classe, attributs, arités, opérations... construite au fur et à mesure.
- Les diagrammes de séquence
 - Les paramètres qui tombent du ciel,
 - des séquences illogiques qui ne mènent à rien.... –
- OCL : certains ont fait l'impasse, les autres ont réussi à condition de ne pas tomber dans le piège des collections !
- Manque de cohérence entre les diagrammes...
 - Ne pas hésiter à revenir en arrière



Déroulement des Tds

- Contrôle Continu et note(s) différenciée(s)
- Critère n°1 :
 - La méthode
 - La cohérence
 - La crédibilité
- Préparation de la séance



Canevas de rapport intermédiaire (1/3)

1. Introduction

- Résumé du sujet
- Résumé des points en interaction avec les autres équipes
- Problématiques à soulever

2. Point de vue général de l'architecture

- Un glossaire
- Une représentation générale (diagramme d'activité)

Pour la suite : TOUT DIAGRAMME DOIT ETRE EXPLIQUE



Canevas de rapport intermédiaire (2/3)

3. Point de Vue Statique

- Cas d'utilisation
 - ◆ Acteurs
 - ◆ Diagrammes de Cas d'utilisation
 - ◆ Scénarios (sous forme textuelle, un par *use case*)
 - ◆ Un morceau de Maquette IHM pour chaque use case
- Diagrammes de Classes lisible en format A4 (une découpe « logique » avec répétition de certaine classe).

4. Point de vue Dynamique

- Diagrammes de Séquence
- Chaque diagramme de séquence devra référencer un use-case.
- Morceau(x) d'IHM associé(s)
- Diagrammes de Machine d'Etat

Canevas de rapport intermédiaire (3/3)

5. Interactions avec les autres S.I.

6. Maquette de l'interface

- une maquette globale rattachée aux diagrammes présentés dans le document (synthèse des points 3-4-5)
- Lien avec les scénarios

7. Conclusion

- Analyse de votre solution : points forts et points faibles

Processus unifiés

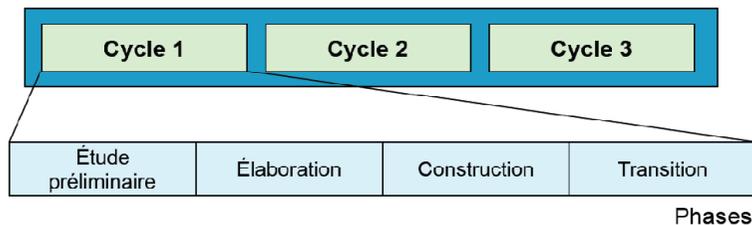
- Principes
- Description
- Déclinaison

Unified Software Development Process

- USDP
 - Résumé en UP : *Unified Process*
- Avant UML
 - Autant de notations que de méthodes
 - Focalisation sur certains aspects uniquement
- Avec UML
 - Uniformisation
- USDP
 - Processus général et méthode de conception
 - Pour gérer un projet de bout en bout
 - À décliner en fonction des notations (UML) et des processus plus particuliers utilisés

USDP : Principes

- ❑ Considérer un produit logiciel quelconque par rapport à ses versions
 - un cycle produit une version
- ❑ Gérer chaque cycle de développement comme un projet ayant quatre phases
 - chaque phase se termine par un point de contrôle (ou jalon) permettant de prendre des décisions



Phases

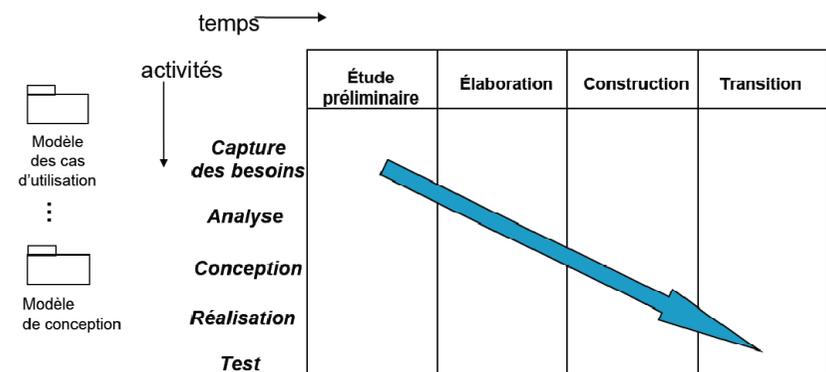
- ❑ **Phase 1 : étude préliminaire**
 - que fait le système ?
 - à quoi pourrait ressembler l'architecture ?
 - quels sont les risques ?
 - quel est le coût estimé du projet ? Comment le planifier ?
 - accepter le projet ?
 - jalon : « vision du projet »
- ❑ **Phase 2 : Élaboration**
 - spécification de la plupart des cas d'utilisation
 - conception de l'architecture de base (squelette du système)
 - mise en oeuvre de cette architecture (UC critiques, <10 % des besoins)
 - planification complète
 - besoins, architecture, planning stables ? Risques contrôlés ?
 - jalon : « architecture du cycle de vie »

Phases (suite)

- ❑ **Phase 3 : Construction**
 - développement par incréments
 - ◆ architecture stable malgré des changements mineurs
 - le produit contient tout ce qui avait été planifié
 - ◆ il reste quelques erreurs
 - produit suffisamment correct pour être installé chez un client ?
 - jalon : « capacité opérationnelle initiale »
- ❑ **Phase 4 : Transition**
 - produit livré (version bêta)
 - correction du reliquat d'erreurs
 - essai et amélioration du produit, formation des utilisateurs, installation de l'assistance en ligne
 - tests suffisants ? Produit satisfaisant ? Manuels prêts ?
 - jalon : « livraison du produit »

Phases et activités

- ❑ Le cycle met en jeu des activités
 - vue du développement sous l'angle technique
 - Les activités sont réalisées au cours des phases, avec des importances variables
 - Les activités consistent notamment à créer des modèles



USDP : principes d'organisation

□ USDP : tout le contraire du modèle en cascade :

- Point commun à toutes les méthodes OO ?
 - ◆ le changement est... **constant**
- Feedback et adaptation : décisions nécessaires tout au long du processus
- Convergence vers un système satisfaisant

□ Principes résultants :

- construction du système par incréments
- gestion des risques
- passage d'une culture produit à une culture projet
- « souplesse » de la démarche

Itérations et incréments

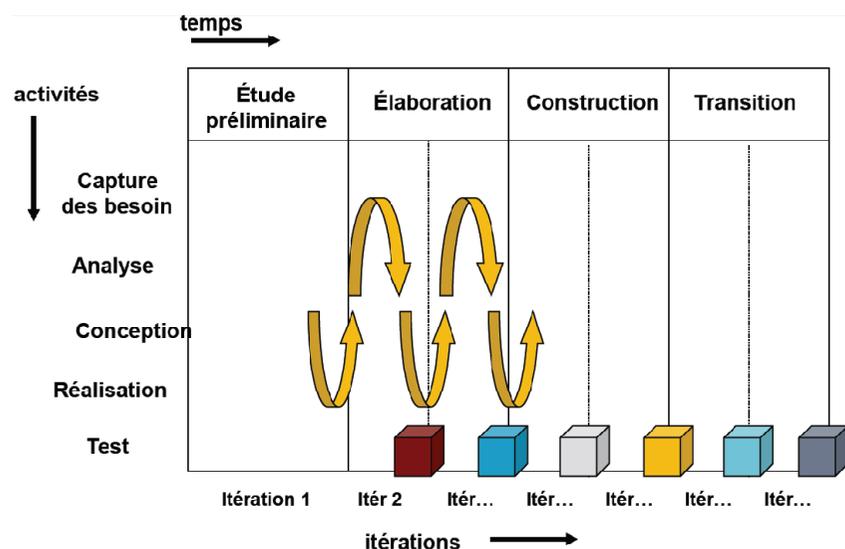
□ Des itérations

- chaque phase comprend des itérations
- une itération a pour but de maîtriser une partie des risques et apporte une preuve tangible de faisabilité
 - ◆ produit un système partiel opérationnel (exécutable, testé et intégré) avec une qualité égale à celle d'un produit fini
 - ◆ qui peut être évalué (va-t-on dans la bonne direction ?)

□ Un incrément par itération

- le logiciel et le modèle évoluent suivant des incréments
- série de prototypes qui vont en s'améliorant
 - ◆ de plus en plus de parties fournies
 - ◆ retours utilisateurs
- processus incrémental
- les versions livrées correspondent à des étapes de la chaîne des prototypes

Itérations dans les phases



Gestion du risques dans les itérations

□ Une itération

- est un mini-projet
 - ◆ plan pré-établi et objectifs pour le prototype, critères d'évaluation,
 - ◆ comporte toutes les activités (mini-processus en cascade)
- est terminée par un point de contrôle
 - ◆ ensemble de modèles agréés, décisions pour les itérations suivantes
- conduit à une version montrable implémentant un certain nombre de cas d'utilisation
- dure entre quelques semaines et 9 mois (au delà : danger)
 - ◆ butée temporelle qui oblige à prendre des décisions

□ On ordonne les itérations à partir des priorités établies pour les cas d'utilisation et de l'étude du risque

- plan des itérations
- chaque prototype réduit une part du risque et est évalué comme tel les priorités et l'ordonnancement de construction des prototypes
- peuvent changer avec le déroulement du plan

Avantages (et inconvénients) résultants

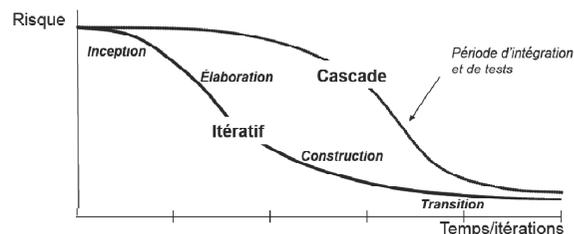
- Gestion de la complexité
- Maîtrise des risques élevés précoce
- Intégration continue
- Prise en compte des modifications de besoins
- Apprentissage rapide de la méthode
- Adaptation de la méthode
- Mais gestion de projet plus complexe : planification adaptative**

Pilotage du processus (par les cas d'utilisation)

- Objectif du processus
 - construction d'un système qui réponde à des besoins
 - par construction complexe de modèles
- Les cas d'utilisation spécifient le besoin à travers les objectifs utilisateurs
- Ils sont utilisés tout au long du cycle (ce qu'on fait naturellement en UML)
 - validation des besoins / utilisateurs
 - point de départ pour l'analyse (découverte des objets, de leurs relations, de leur comportement) et la conception
 - guide pour la construction des interfaces
 - guide pour la mise au point des plans de tests
- Les UC assurent la traçabilité !

Pilotage du processus (par la gestion des risques)

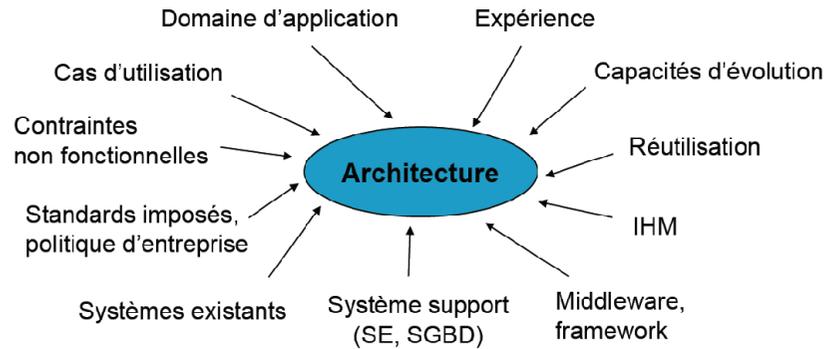
- Différentes natures de risques
 - Adéquation aux besoins, infrastructure technique, performances, personnels impliqués...
- Gestion des risques
 - identifier et classer les risques par importance
 - agir pour diminuer les risques
 - s'ils sont inévitables, les évaluer rapidement (au plus tôt)
- Construire les itérations en fonction des risques
 - provoquer des changements précoces pour stabiliser l'architecture rapidement



Processus centré sur l'architecture

- Architecture
 - Principe de génie civil difficile à adapter à l'informatique
- Définition dans notre cadre
 - Art d'assembler des composants en respectant des contraintes, ensemble des décisions significatives sur
 - ◆ l'organisation du système
 - ◆ les éléments qui structurent le système
 - ◆ la composition des sous-systèmes en systèmes
 - ◆ le style architectural guidant l'organisation (couches...)
 - Ensemble des éléments de modélisation les plus significatifs qui constituent les fondations du système à développer

Influences sur l'architecture



Différents aspects d'une architecture

□ Architecture logicielle

- organisation à grande échelle des classes logicielles en packages, sous-systèmes et couches
 - ◆ architectures client/serveurs en niveaux (tiers)
- architectures en couches
- architecture à base de composants
- patrons architecturaux (cf. cours de Master)

□ Architecture de déploiement

- décision de déploiement des différents éléments
- déploiement des fonctions sur les postes de travail des utilisateurs

Processus centré sur l'architecture

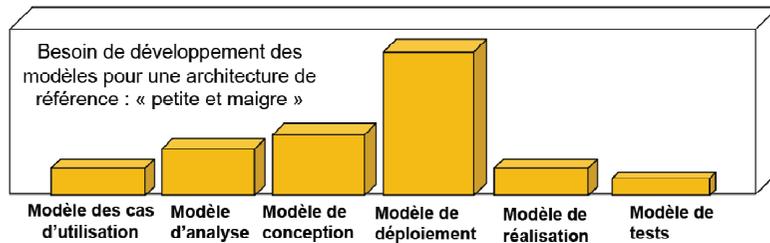
- L'architecture sert de lien pour l'ensemble des membres du projet
 - réalisation concrète de prototypes incrémentaux qui « démontrent » les décisions prises
 - vient compléter les cas d'utilisation comme « socle commun »
- Contrainte de l'architecture
 - plus le projet avance, plus l'architecture est difficile à modifier
 - les risques liés à l'architecture sont très élevés, car très coûteux
- Objectif pour le projet
 - établir dès la phase d'élaboration des fondations solides et évolutives pour le système à développer, en favorisant la réutilisation
 - l'architecture s'impose à tous, contrôle les développements ultérieurs, permet de comprendre le système et d'en gérer la complexité
- L'architecture est contrôlée et réalisée par l'architecte du projet

Construction de l'architecture : principes

- Démarrage, choix d'une architecture de haut-niveau et construction des parties générales de l'application
 - ébauche à partir
 - ◆ de solutions existantes
 - ◆ de la compréhension du domaine
 - ◆ de parties générales aux applications du domaine (quasi-indépendant des CU)
 - des choix de déploiement
- Ensuite construction de l'architecture de référence
 - confrontation à l'ébauche des cas d'utilisation les plus significatifs (un à un)
 - construction de parties de l'application réelle (sous-systèmes spécifiques)
 - stabilisation de l'architecture autour des fonctions essentielles (sous-ensemble des CU)
 - traitement des besoins non fonctionnel dans le contexte des besoins fonctionnels
 - identification des points de variation et les points d'évolution les plus probables
- Finalement réalisation incrémentale des cas d'utilisation
 - itérations successives
 - l'architecture continue à se stabiliser sans changement majeur

Elaboration de l'architecture

- ❑ Phase d'élaboration
 - aller directement vers une architecture robuste, à coût limité, appelée «architecture de référence»
 - 10% des classes suffisent
- ❑ L'architecture de référence
 - permettra d'intégrer les CU incrémentalement
 - guidera le raffinement et l'expression des CU pas encore détaillés



Description de l'architecture

- ❑ L'architecture doit être une vision partagée
 - sur un système très complexe
 - pour guider le développement
 - tout en restant compréhensible
- ❑ **Description architecture = restriction du modèle**
 - extraits les plus significatifs des modèles de l'architecture de référence
 - vue architecturale du modèle des CU : quelques CU
 - vue du modèle d'analyse (éventuellement non maintenue)
 - vue du modèle de conception : principaux sous-systèmes et interfaces, collaborations
 - vue du modèle de déploiement : diagramme de déploiement
 - vue du modèle d'implémentation : artefacts

Description et illustration du processus unifié

USDP

- ❑ Définit un enchaînement d'activités
- ❑ Est réalisé par un ensemble de *travailleurs*
 - Avec des rôles, des métiers
- ❑ Avec pour objectifs
 - de passer des besoins à un ensemble d'*artefacts* cohérents (le système informatique résultant)
 - De favoriser le passage à une nouvelle version quand les besoins évoluent
- ❑ UP n'est qu'un cadre général de processus
 - Chaque projet est une instance de ce cadre
 - Adaptée au contexte (taille, personnels, entreprise, etc.)

Eléments du processus

☐ Artefacts (le quoi)

- Produits
- Traces
- Donc des modèles (UML), du code source, des exécutables...

☐ Travailleurs (le qui)



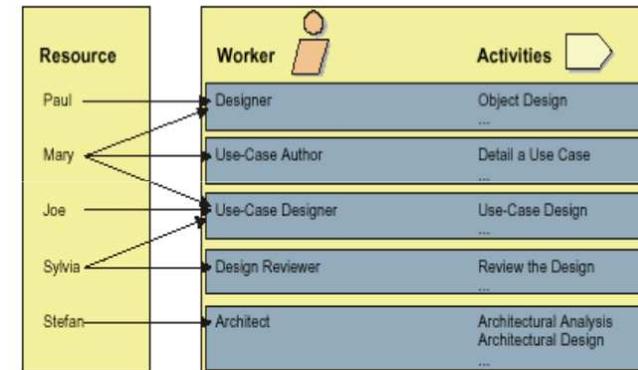
- Un rôle par rapport au projet
- Exemples : architecte logiciel, analyste métier (rédige les UC)

☐ Activités (le comment)



- 5 grandes activités
 - ◆ Plein de tâches et sous-tâches
 - ◆ Réalisées par des *travailleurs*
 - ◆ Pour manipuler de l'information dans des *artefacts*

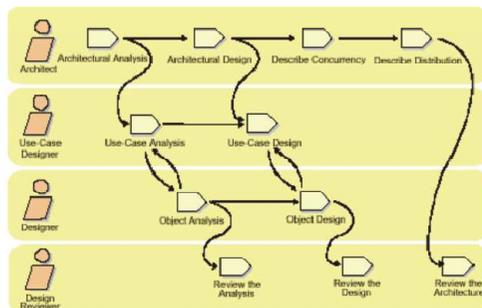
Travailleurs et activités



Organisation autour des workflows

☐ Workflow

- Enchaînement d'activités qui produisent des artefacts
- Equivalent à un diagramme d'activités UML



Les modèles utilisés

☐ Modèle des cas d'utilisation

- Système vu de l'extérieur, périmètre et insertion dans l'organisation

☐ Modèle d'analyse

- Système vu de l'intérieur
- Objets comme des abstractions des concepts manipulés par les utilisateurs, avec point de vue statique et dynamique

☐ Modèle de conception

- Modèle précédent mis en proximité des langages et plateformes de développement

☐ Modèle de déploiement

- Conception de l'architecture physique

☐ Modèle d'implémentation

- Liaison entre code et classes de conception

☐ Modèle de test

- Description des cas de tests

Activités pour passer des besoins au code

Activité : acquisition des besoins

Objectifs

- Déterminer les valeurs attendues du nouveau système informatique
- Recenser les besoins
 - ◆ les classer par priorité, évaluer leur risque
- Comprendre le contexte
 - ◆ Vocabulaire commun
 - ◆ Modèle du domaine = diagramme de classes
 - ◆ Modèle du métier (éventuellement) = UC + diag. d'activités

Modèle du domaine

- Des classes représentant des objets métiers et du monde réel
- Quelques attributs, peu d'opérations
- Des associations !
- Ce qui n'est pas encore modélisé va dans le glossaire (pour plus tard)

Modèle du métier (facultatif)

- Des liens entre des acteurs (**ce qu'on n'a pas le droit de faire d'habitude en UML**)
- Des diagrammes d'activités (des workflows)

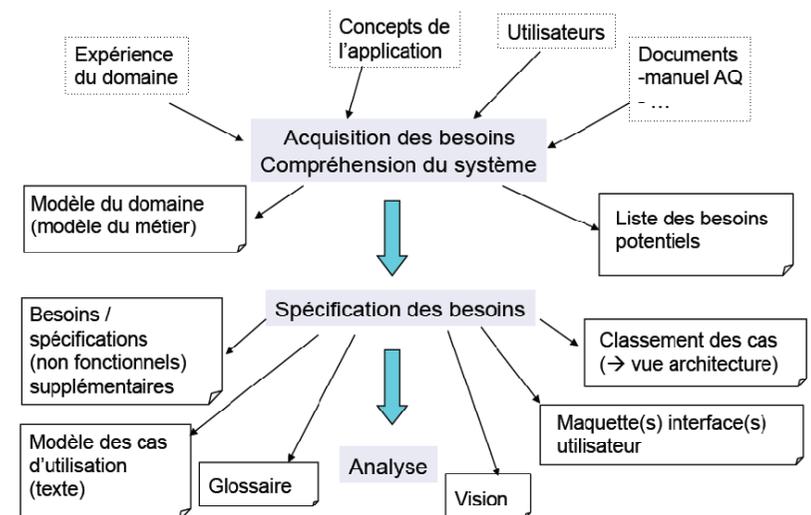
Activité : expression des besoins

Objectifs

- Définir les besoins fonctionnels
 - ◆ Scénarios
 - ◆ Acteurs / objectifs
 - ◆ Cas d'utilisation (donc construction du modèle de cas d'utilisation)
- Classer les cas d'utilisation par priorité
 - ◆ Fonction des risques, des nécessités de l'architecture ou du client
- Détailler les cas d'utilisation
 - ◆ UC: Réserver une salle
 - ◆ Portée : système de planning
 - ◆ Acteur principal : secrétaire
 - ◆ Préconditions : une salle est libre pour la période désirée
 - ◆ Scénario nominal : ...
- Appréhender les besoins non fonctionnels
 - ◆ Contraintes sur le système
 - ◆ Environnement, plateforme, fiabilité, performance

Activité : expression des besoins

Artefacts



Activité : expression des besoins

☐ Travailleurs



- Analyste du domaine
 - ◆ Modèle du domaine
 - ◆ Modèle des cas d'utilisation
 - ◆ glossaire
- Spécificateur de cas d'utilisation
 - ◆ Cas d'utilisation **détaillé**
- Concepteur d'interface utilisateur
 - ◆ Maquette IHM
- Architecte logiciel
 - ◆ Vue architecturale du modèle des cas d'utilisation

Activité : analyse

☐ Objectifs

- Construire le modèle d'analyse et préparer la conception
 - ◆ Forme/Architecture générale du système : **recherche de stabilité**
 - ◆ Haut niveau d'abstraction

☐ Mener l'analyse architecturale

- Identifier les packages d'analyse par regroupement logique
- Identifier les classes constituant le cœur de métier
 - ◆ 3 stéréotypes : frontière (interface), contrôle, entité
 - ◆ Les responsabilités doivent être évidentes
- Identifier les besoins non fonctionnels communs pour les rattacher aux UC

☐ Analyser les cas d'utilisation

- Réaliser les scénarios
- Identifier les classes, attributs et associations nécessaires
- Décrire les interactions (typiquement par des diagrammes de séquence)

Activité : analyse

- Le modèle structural doit être l'union des associations déterminées

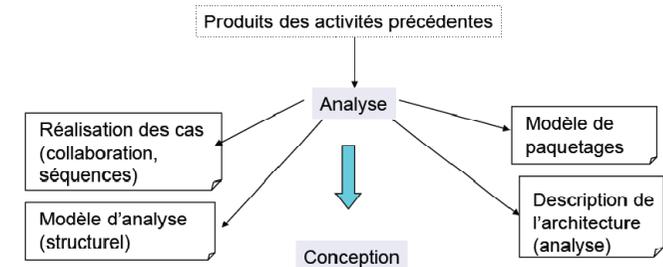
☐ Préciser les classes d'analyse

- Responsabilité ?
- Identifier attributs, associations, héritages
- Stabiliser les packages

☐ Faire ces opérations pour chaque cas d'utilisation...

Activité : analyse

☐ Artefacts



☐ Travailleurs



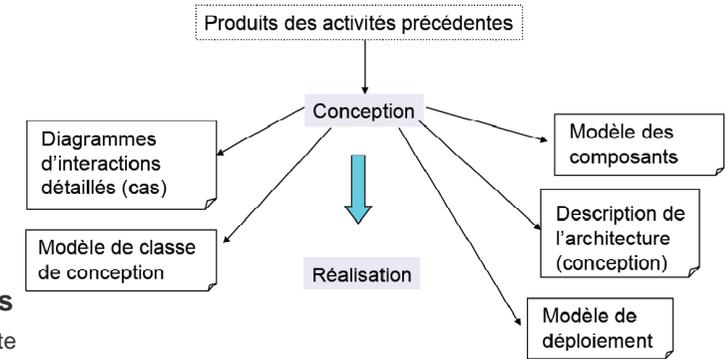
- Architecte
 - ◆ Responsable de l'intégrité du modèle d'analyse et de la description de l'architecture
- Ingénieur des UC
 - ◆ Responsable de l'analyse de chaque UC
- Ingénieur des composants
 - ◆ Responsable des classes et paquetages d'analyse

Activité : conception

- ❑ Proposer une réalisation de l'analyse et des cas d'utilisation en prenant en compte toutes les exigences
- ❑ Effectuer la conception architecturale
 - identifier les noeuds et la configuration du réseau (déploiement)
- ❑ Concevoir les cas d'utilisation
 - identifier les classes nécessaires à la réalisation des cas
- ❑ Concevoir les classes et les interfaces
 - décrire les méthodes, les états, prendre en compte les besoins spéciaux
- ❑ Concevoir les sous-systèmes
 - mettre à jour les dépendances, les interfaces, les composants réseau et/ou middleware
 - permettra de piloter le travail des développeurs

Activité : conception

❑ Artefacts



❑ Travailleurs



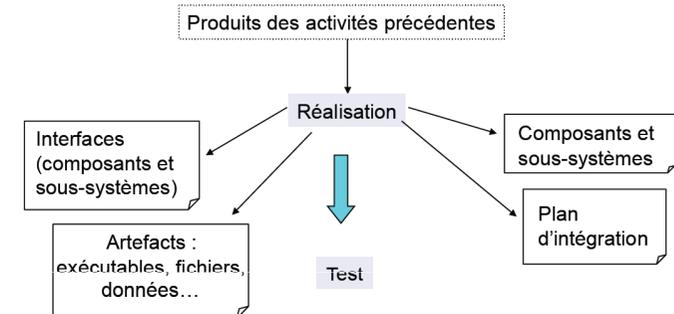
- Architecte
 - ◆ intégrité des modèles de conception et de déploiement
 - ◆ description de l'architecture
- Ingénieur UC
 - ◆ réalisation/conception des UC
- Ingénieur de composants
 - ◆ classes de conception, composants, interfaces

Activité : réalisation

- ❑ Faire la mise en œuvre architecturale
 - identifier les artefacts logiciels et les associer à des noeuds
- ❑ Intégrer le système
 - planifier l'intégration, intégrer les incréments réalisés
- ❑ Réaliser les composants et les sous-systèmes
- ❑ Réaliser les classes
- ❑ Mener les tests unitaires
 - ◆ tests de spécification en boîte noire (de structure en boîte blanche)

Activité : réalisation

❑ Artefacts



❑ Travailleurs



- Architecte
 - ◆ modèles d'implémentation et de déploiement
 - ◆ description de l'architecture
- Ingénieur de composants
 - ◆ artefacts logiciels, sous-systèmes et composants d'implémentation, interfaces
- Intégrateur système
 - ◆ plan de construction de l'intégration

Activité : test

❑ Rédiger le plan de test

- Décrire les stratégies de test, estimer les besoins pour l'effort de test, planifier l'effort dans le temps
- Tenir compte des risques (tester dès que possible)

❑ Concevoir les tests

❑ Automatiser les tests

❑ Réaliser les tests d'intégration

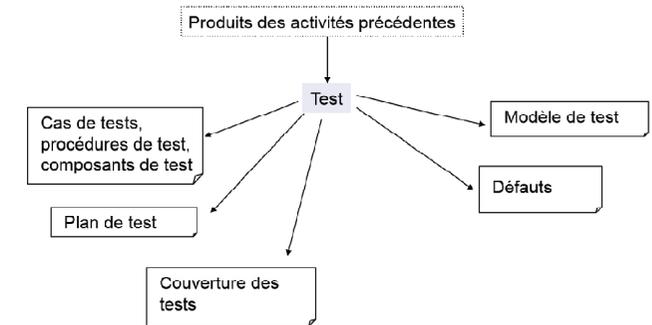
❑ Réaliser les tests du système dans son intégralité

❑ Évaluer les tests

- Sont-ils efficaces ? Pertinents ?

Activité : test

❑ Artefacts

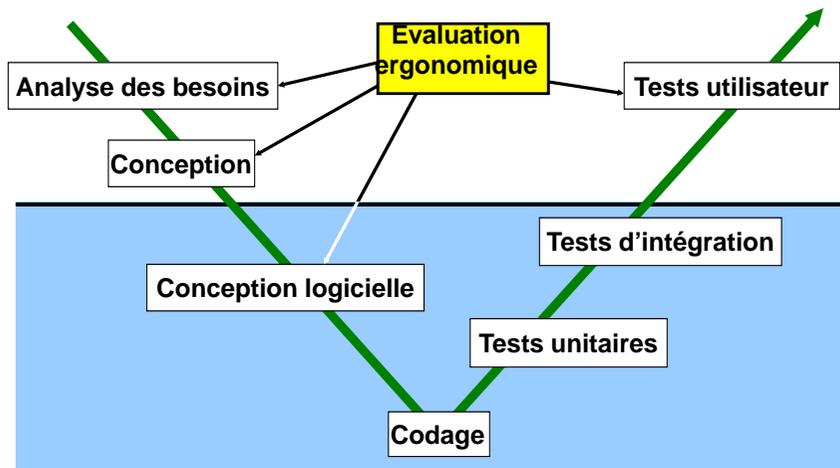


❑ Travailleurs



- Concepteur de tests
 - ◆ modèle de tests, cas de test, procédures de test, évaluation des tests, plan de tests
- Ingénieur de composants
 - ◆ test unitaires
- Testeur d'intégration
 - ◆ tests d'intégration
- Testeur système
 - ◆ vérification du système dans son ensemble

Comparaison avec une Démarche centrée Utilisateur



Note : le cycle de vie d'une interface est ici représenté en V de manière analytique...

centrée Utilisateur

❑ Analyse des besoins (IHM)

- Modèle Utilisateur
- Modèles des Tâches
 - ◆ Concept du domaine
 - ◆ Procédures

❑ Conception (IHM) : Interaction

❑ Conception Logicielle basée sur l'IHM

❑ Codage, tests... pas précisés

❑ Tests Utilisateurs (IHM)

- Bien définis
- Protocoles, méthode, etc.
- Peuvent intervenir dès la conception IHM

USDP

→ Des points communs

- Souvent négligé
- Similaire
 - Interaction en moins

→ Plutôt vague

→ Comp | basée Obj

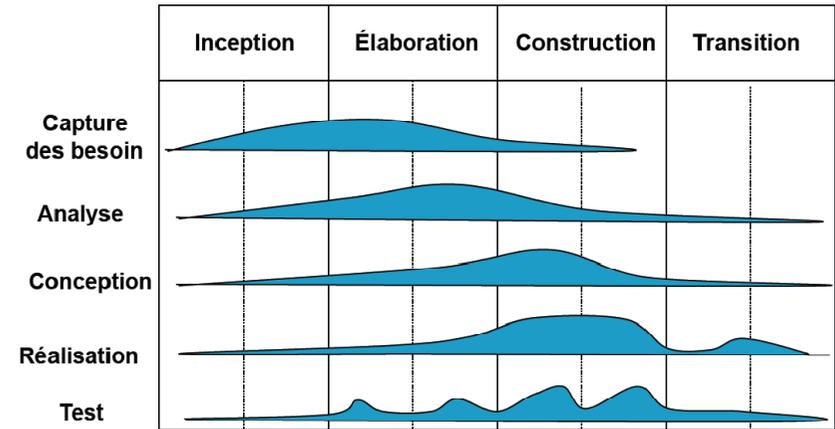
→ Bien définie

→ Pas trop abordé

Phases de pilotage des activités

Rappel sur les phases

- Chaque phase spécifie les activités à effectuer



Gestion des phases

□ Planifier les phases

- allouer le temps, fixer les points de contrôle de fin de phase, les itérations par phase et le planning général du projet

□ Dans chaque phase

- planifier les itérations et leurs objectifs de manière à réduire
 - ◆ les risques spécifiques du produit
 - ◆ les risques de ne pas découvrir l'architecture adaptée
 - ◆ les risques de ne pas satisfaire les besoins
- définir les critères d'évaluation de fin d'itération

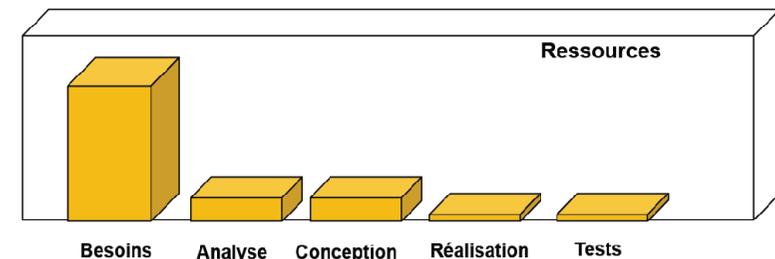
□ Dans chaque itération

- faire les ajustements indispensables (planning, modèles, processus, outils...)

Phase d'étude préliminaire (inception)

□ Objectif : lancer le projet

- établir les contours du système et spécifier sa portée
- définir les critères de succès, estimer les risques, les ressources nécessaires et définir un plan (petite planification)
- **à la fin de cette phase, on décide de continuer ou non**
- attention à ne pas définir tous les besoins, à vouloir des estimations fiables (coûts, durée), sinon on fait de la *cascade*



Activités (principales) de cette phase

☐ Capture des besoins

- comprendre le contexte du système (50 à 70% du contexte)
- établir les besoins fonctionnels et non fonctionnels (80%)
- traduire les besoins fonctionnels en cas d'utilisation (50%)
- détailler les premiers cas par ordre de priorité (10% max)

☐ Analyse

- analyse des cas d'utilisation (10% considérés, 5% raffinés)
- pour mieux comprendre le système à réaliser, guider le choix de l'architecture

☐ Conception

- première ébauche de la conception architecturale : soussystèmes, noeuds, réseau, couches logicielles
- examen des aspects importants et à plus haut risque

Livrables de cette phase

☐ Première version du modèle du domaine ou de contexte de l'entreprise

- parties prenantes, utilisateurs

☐ Liste des besoins fonctionnels et non fonctionnels

☐ Ébauche des modèles de cas, d'analyse et de conception

☐ Esquisse d'une architecture

☐ Liste ordonnée de risques et liste ordonnée de cas

☐ Grandes lignes d'un planning pour un projet complet

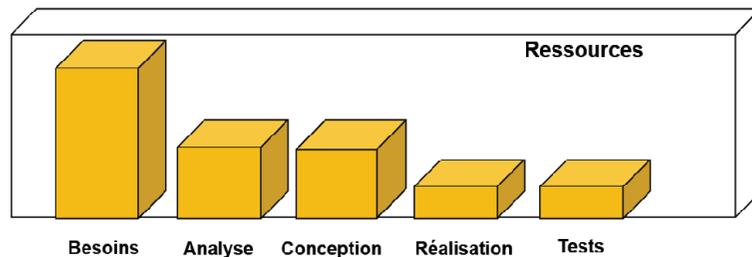
☐ Première évaluation du projet, estimation grossière des coûts

☐ Glossaire

Phase d'élaboration

☐ Objectif : analyser le domaine du problème

- capturer la plupart des besoins fonctionnels
- planifier le projet et éliminer ses plus hauts risques
- établir un squelette de l'architecture
- réaliser un squelette du système



Activités principales de l'élaboration

☐ Capture des besoins

- terminer la capture des besoins et en détailler de 40 à 80%
- faire un prototype de l'interface utilisateur (éventuellement)

☐ Analyse

- analyse architecturale complète (packages...)
- raffinement des cas d'utilisation (pour l'architecture, < 10%)

☐ Conception

- terminer la conception architecturale
- effectuer la conception correspondant aux cas sélectionnés

☐ Réalisation

- limitée au squelette de l'architecture
- faire en sorte de pouvoir valider les choix

☐ Test

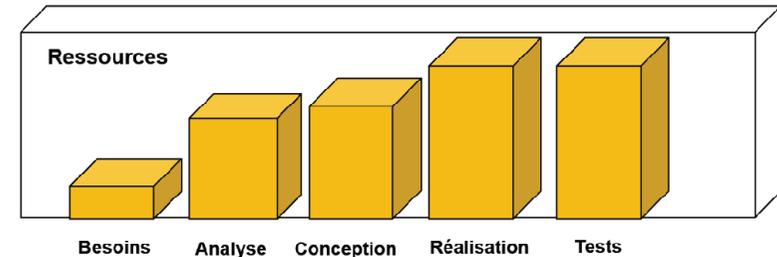
- du squelette réalisé (attention, peut être coûteux)

Livrables de l'élaboration

- Un modèle de l'entreprise ou du domaine complet
- Une version des modèles : cas, analyse et conception (<10%), déploiement, implémentation (<10%)
- Une architecture de base exécutable
- La description de l'architecture (extrait des autres modèles)
 - document d'architecture logicielle
- Une liste des risques mise à jour
- Un projet de planning pour les phases suivantes
- Un manuel utilisateur préliminaire (optionnel)
- Évaluation du coût du projet

Phase de construction

- Objectif : Réaliser une version beta

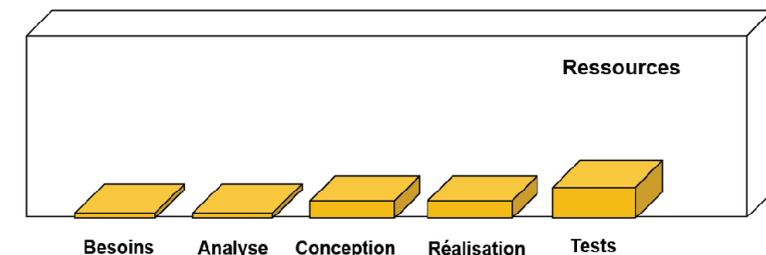


Activités et livrables de la construction

- Capture des besoins
 - spécifier l'interface utilisateur
- Analyse
 - terminer l'analyse de tous les cas d'utilisation, la construction du modèle structurel d'analyse
- Conception
 - l'architecture est fixée et il faut concevoir les sous-systèmes
 - dans l'ordre de priorité (itérations de 1 à 3 mois, max. 9 mois)
 - concevoir les cas d'utilisation puis les classes
- Réalisation
 - réaliser, passer des tests unitaires, intégrer les incréments
- Test
 - toutes les activités de test : plan, conception, évaluation...
- Livrables
 - Un plan du projet pour la phase de transition
 - L'exécutable et son packaging minimal
 - Tous les documents et les modèles du système
 - Une description à jour de l'architecture
 - Un manuel utilisateur suffisamment détaillé pour les tests

Phase de transition

- Objectif : mise en service chez l'utilisateur
 - test de la version beta, correction des erreurs
 - préparation de la formation, la commercialisation



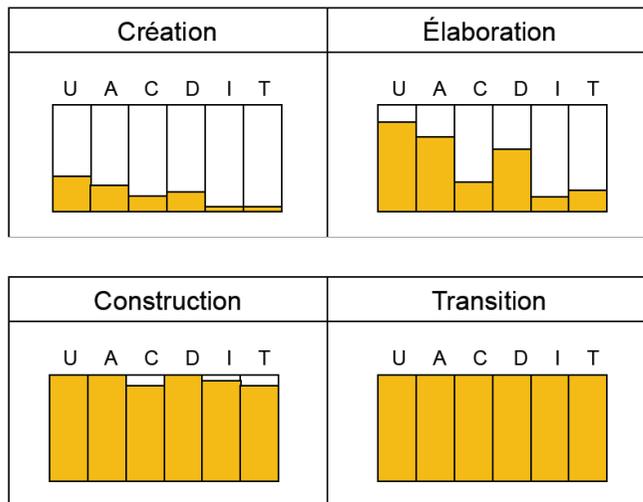
Activités principales de transition

- Préparer la version beta à tester
 - Installer la version sur le site, convertir et faire migrer les données
- Gérer le retour des sites (retour de déploiement)
 - Le système fait-il ce qui était attendu ? Erreurs découvertes ?
 - Adapter le produit corrigé aux contextes utilisateurs (installation...)
- Terminer les livrables du projet (modèles, documents...)
- Déterminer la fin du projet
- Reporter la correction des erreurs trop importantes (nouvelle version)
- Organiser une revue de fin de projet (pour apprendre)
- Planifier le prochain cycle de développement

Livrables de la transition

- L'exécutible et son programme d'installation
- Les documents légaux : contrat, licences, garanties, etc.
- Un jeu complet de documents de développement à jour
- Les manuels utilisateur, administrateur et opérateur et le matériel d'enseignement
- Les références pour le support utilisateur (site Web...)

Répartition modèles/phases



U : modèle des cas d'utilisation
A : modèle d'analyse
C : modèle de conception
D : modèle de déploiement
I : modèle d'implémentation
T : modèle de tests

Mini-conclusion

- UP
 - Est gros
 - Mais très structurant
- Il décrit un ensemble de processus applicables
- Mais il faut s'adapter aux besoins du projet
- Exemples d'application (à venir)
 - 2TUP
 - UP « agiles »
- Et on peut faire de l'agile sans faire de l'UP...