Processus de conception de SI
2/3 – Processus unifié

Yannick Prié
Département Informatique – Faculté des Sciences et Technologies
Université Claude Bernard Lyon 1
2009-2010

Plan du cours global

- 1/3 – Méthodes et processus
- **2/3 – Processus unifié**
- 3/3 – Méthodes Agile

Plan de ce cours

- Trame du processus unifié
- Caractéristiques essentielles
- Description du processus unifié
- Illustration : deux déclinaisons

- Beaucoup de transparents, une icône pour ce qui est à retenir

Plan

- Trame du processus unifié
- Caractéristiques essentielles
- Description du processus unifié
- Illustration : deux déclinaisons

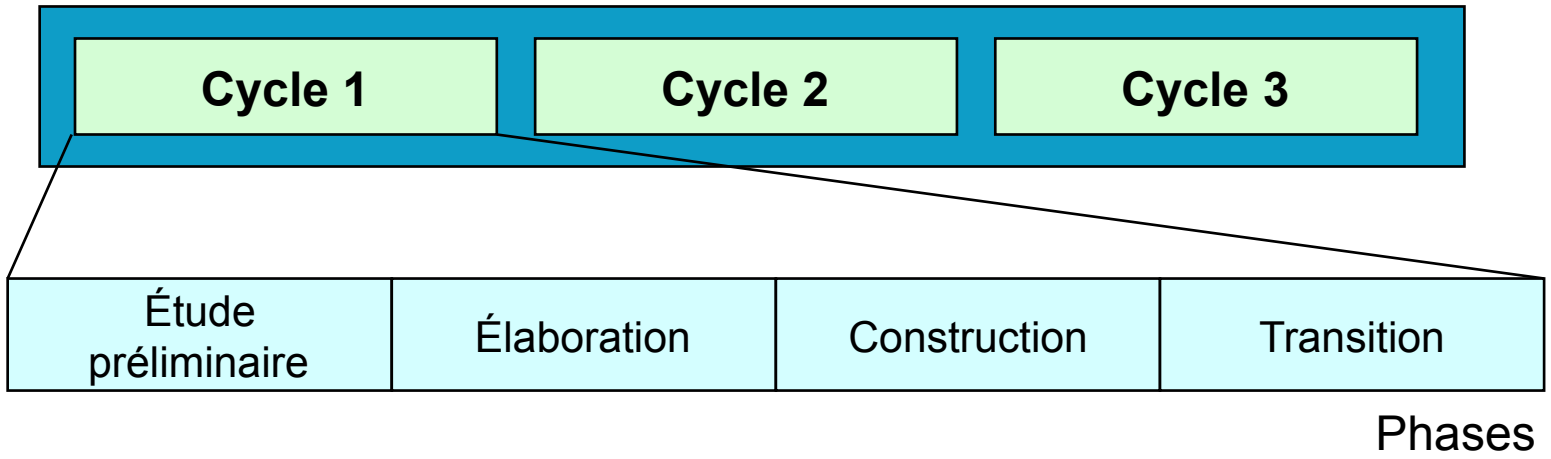
Unified Software Development Process / Unified Process (UP)

- Années 1990 : beaucoup de méthodes
  - liées à des outils, à l'adaptation à UML (comme langage de notation) de méthodes pré-existantes, aux entreprises, etc.
  - finalement, autant de méthodes que de concepteurs / projets
- USDP
  - proposée par Rumbaugh, Booch, Jacobson
    - les concepteurs originaux d'UML
  - purement objet
  - prend de la hauteur par rapport à RUP (Rational)
  - méthode / processus
  - regroupement des meilleures pratiques de développement
- Dans ce cours : beaucoup de généralités liées à USDP, qui s'appliquent à peu près à toute méthode objet

Unified Software Development Process

- Un processus capable de
  - dicter l'organisation des activités de l'équipe
  - diriger les tâches de chaque individu et de l'équipe dans son ensemble
  - spécifier les artefacts à produire
  - proposer des critères pour le contrôle de produits et des activités de l'équipe
- Bref
  - gérer un projet logiciel de bout en bout
- Regroupement de bonnes pratiques, mais
  - non figé
  - générique (hautement adaptable : individus, cultures, …)
    - choisir un UP (« cas de développement » dans RUP) qui correspond au projet du moment, appliquer
    - seul un expert peut en décider

Cycles de vie et phases



- 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
  - vue gestionnaire (manager)
  - chaque phase se termine par un point de contrôle (ou jalon) permettant aux chefs de projet de prendre des décisions

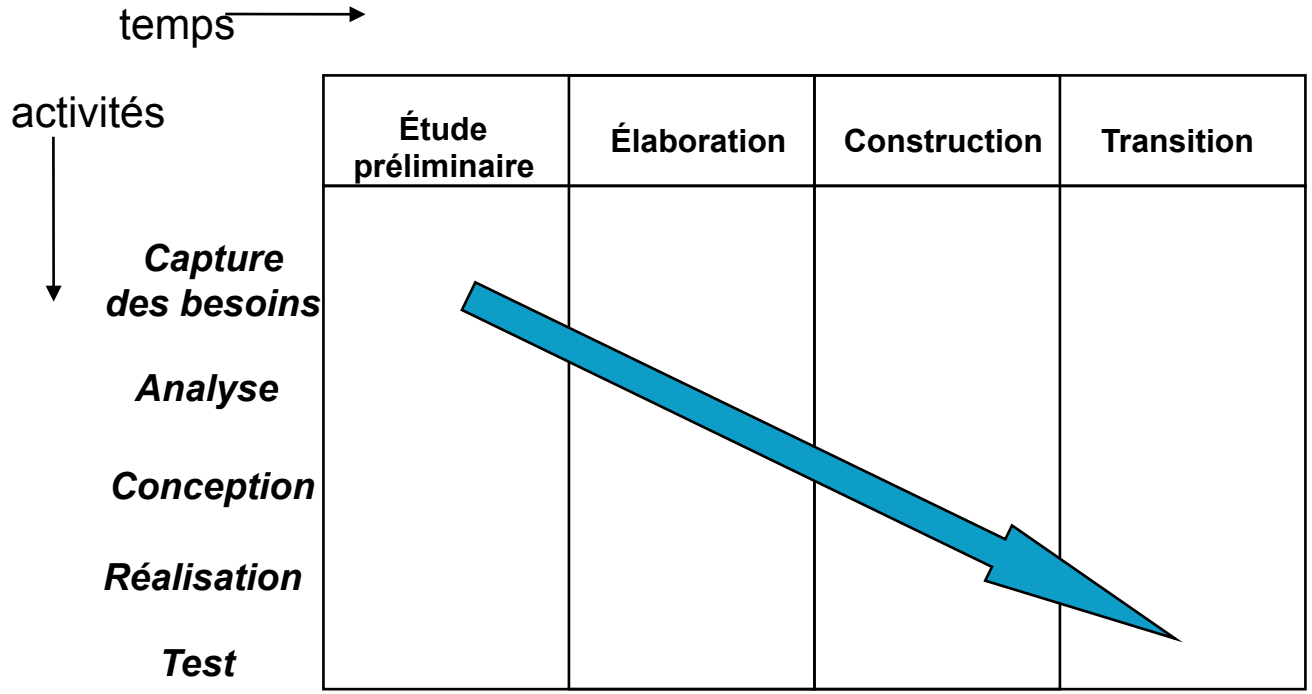
Phases 1 et 2
Etude préliminaire et Elaboration

- 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 »
- Élaboration
  - spécification de la plupart des cas d'utilisation
  - conception de l'architecture de base (squelette du système)
  - mise en œuvre de cette architecture (CU critiques, <10 % des besoins)
  - planification complète
  - besoins, architecture, planning stables ? Risques contrôlés ?
  - jalon : « architecture du cycle de vie »
- Remarque
  - phases (surtout préliminaire) effectuées à coût faible

Phases 3 et 4
Construction et Transition

- 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 »
- 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 »
- Remarque
  - construction = phase la plus coûteuse (> 50% du cycle), englobe conception/codage/tests/ intégration)

Phases et activités

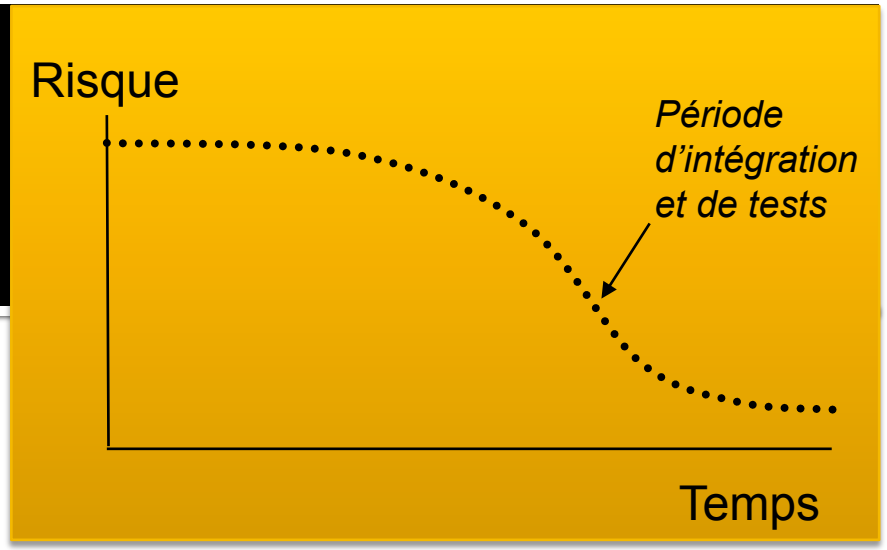


- Le cycle met en jeu des activités
  - vue du développement sous l'angle technique (développeur)
  - les activités sont réalisées au cours des phases, avec des importances variables

Plan

- Trame du processus unifié
- **Processus unifié : caractéristiques essentielles**
  - **itératif et incrémental**
  - piloté par les besoins
  - piloté par les risques
  - centré sur l'architecture
- Description du processus unifié
- Illustration : deux déclinaisons du processus unifié

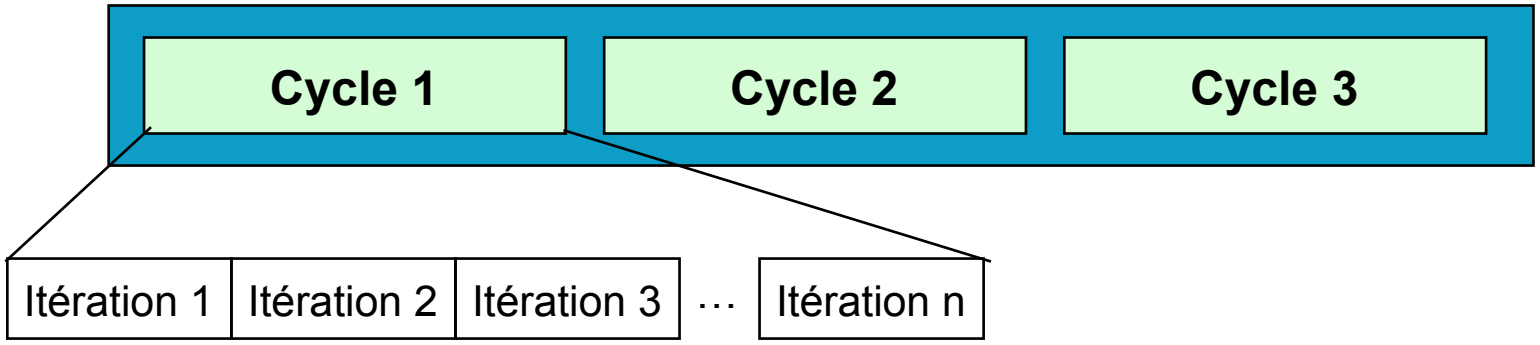
Les problèmes du cycle en cascade



- Risques élevés et non contrôlés
  - identification tardive des problèmes
  - preuve tardive de bon fonctionnement
- Grand laps de temps entre début de fabrication et sortie du produit
- Décisions stratégiques prise au moment où le système est le moins bien connu
- Non-prise en compte de l'évolution des besoins pendant le cycle
- Les études montrent :
  - 25 % des exigences d'un projet type sont modifiées (35-50 % pour les gros projet) (Larman 2005)
  - 45% de fonctionnalités spécifiées ne sont jamais utilisées (Larman 2005, citant une étude 2002 sur des milliers de projets)
  - le développement d'un nouveau produit informatique n'est pas une activité prévisible ou de production de masse
  - la stabilité des spécifications est une illusion
- Distinction entre activités trop stricte
  - modèle théoriquement parfait, mais inadapté aux humains

Anti-cascade

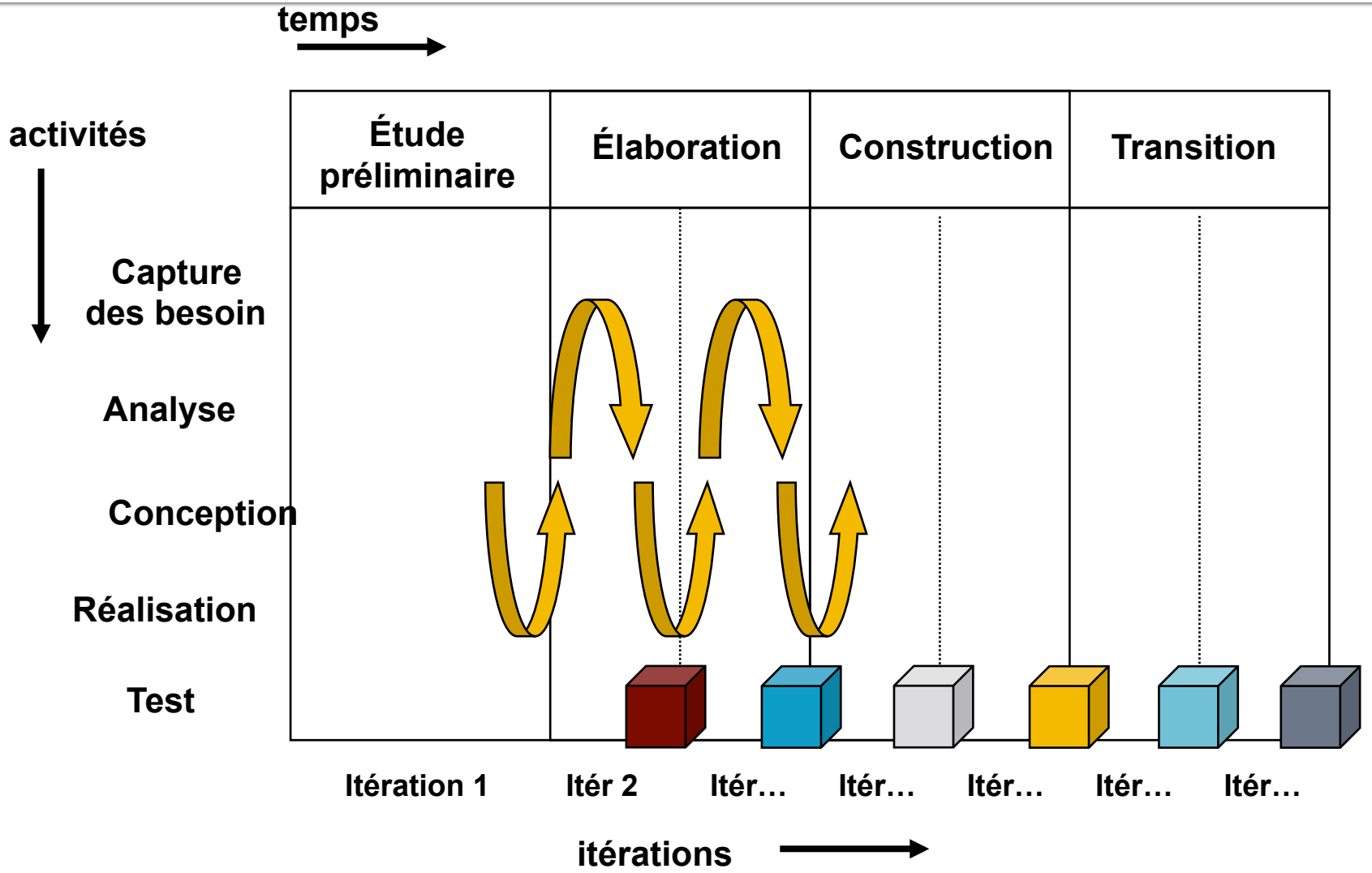
- Point commun à toutes les méthodes OO
- Nécessité de reconnaître que le changement est une constante (normale) des projets logiciels
  - feedback et adaptation : décision tout au long du processus
  - convergence vers un système satisfaisant
- Idées
  - construction du système par incréments
  - gestion 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
      - permet de savoir si on va dans une bonne direction ou non
- 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 et phases



Itérations et risque

- Une itération
  - est un mini-projet
    - plan pré-établi et objectifs pour le prototype, critères d'évaluation,
  - comporte toutes les activités (mini-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 CU
  - 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

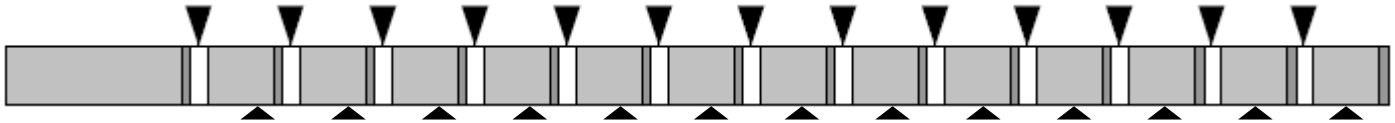
Avantages d'un processus itératif et incrémental

- Gestion de la complexité
  - pas tout en même temps, étalement des décisions importantes
- Maîtrise des risques élevés précoce
  - diminution de l'échec
  - architecture mise à l'épreuve rapidement (prototype réel)
- Intégration continue
  - progrès immédiatement visibles
  - maintien de l'intérêt des équipes (court terme, prototypes vs documents)
- Prise en compte des modifications de besoins
  - feedback, implication des utilisateurs et adaptation précoce
- Apprentissage rapide de la méthode
  - amélioration de la productivité et de la qualité du logiciel
- Adaptation de la méthode
  - possibilité d'explorer méthodiquement les leçons tirées d'une itération (élément à conserver, problèmes, éléments à essayer…)
- Mais gestion de projet plus complexe : planification adaptative

Résumé

- Démarche traditionnelle

- Méthodes itérative



Introduction (2/3)

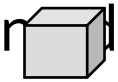
- Une solution : les méthodes agiles
  - Diffuser le processus de décision tout au long du projet
  - Enchaînement de cycles itératifs très courts

Plan

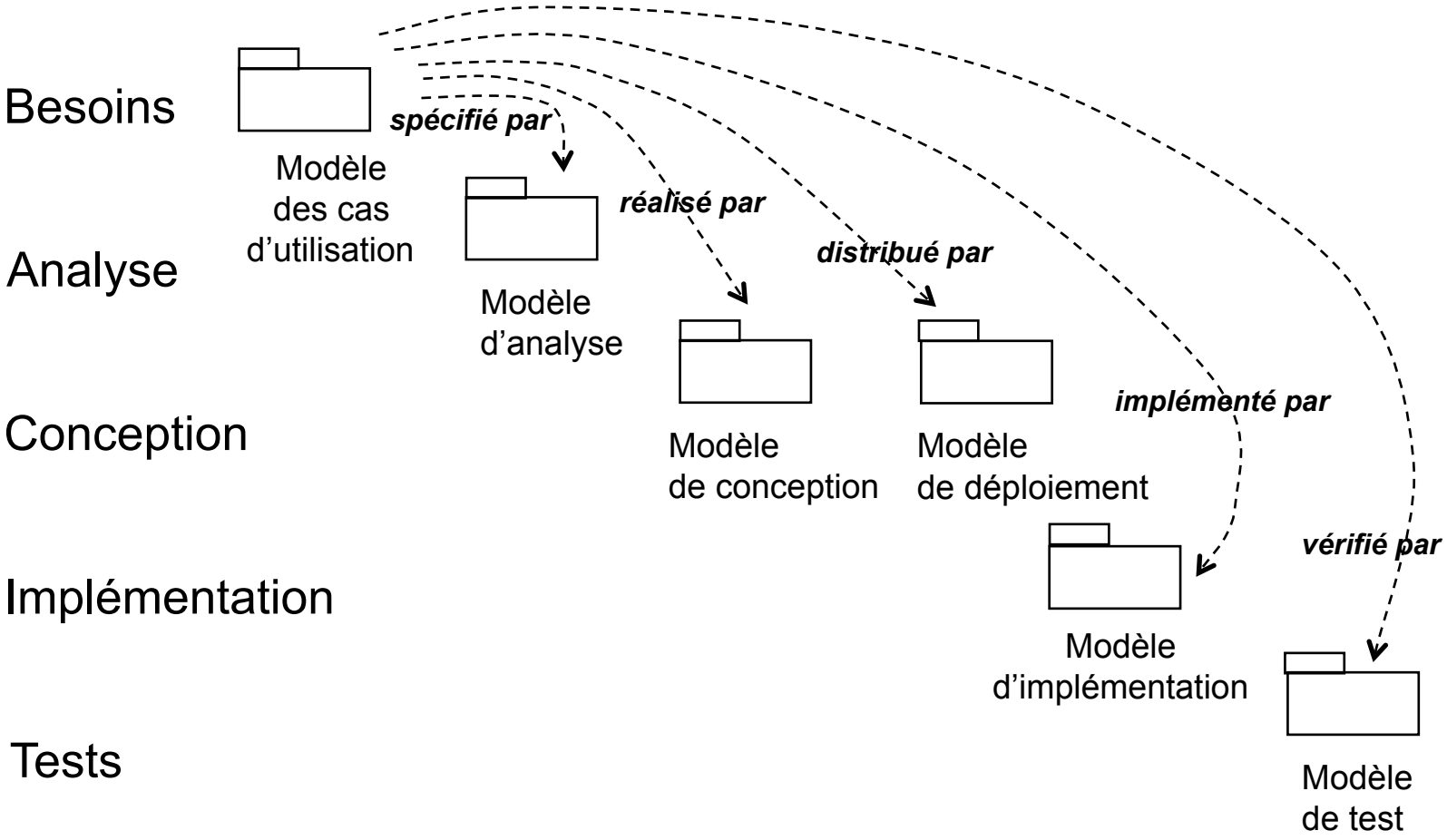
- Trame du processus unifié
- **Processus unifié : caractéristiques essentielles**
  - itératif et incrémental
  - **piloté par les besoins**
  - piloté par les risques
  - centré sur l'architecture
- Description du processus unifié
- Illustration : deux déclinaisons

Un processus piloté par les besoins

- Objectif du processus
  - construction d'un système qui réponde à des besoins
  - par construction complexe de modèles
- Cas d'utilisation = expression / spécification des besoins
  - CU portée système / objectifs utilisateurs
- CU utilisés tout au long du cycle
  - validation des besoins / utilisateurs
  - point de départ pour l'analyse (découverte des objets, de leurs relations, de leur comportement) et la conception (sous-systèmes)
  - guide pour la construction des interfaces
  - guide pour la mise au point des plans de tests



Les CU lient les modèles



Avantages des cas d'utilisation

- Centrés utilisateurs
  - support de communication en langue naturelle entre utilisateurs et concepteurs basé sur les scénarios (et non liste de fonctions)
  - dimension satisfaction d'un objectif utilisateur
    - également pour les utilisateurs informaticiens (administration)
  - besoins fonctionnels, pour acteurs humains et non humains à identifier précisément
- Assurent la traçabilité par rapports aux besoins de toute décision de conception sur l'ensemble du projet
  - tout modèle peut se référer in fine à un CU
- Fournissent une vision commune aux participants du projet
- Attention
  - créer de bons CU est un art (cf. CM rédaction de CU)

Plan

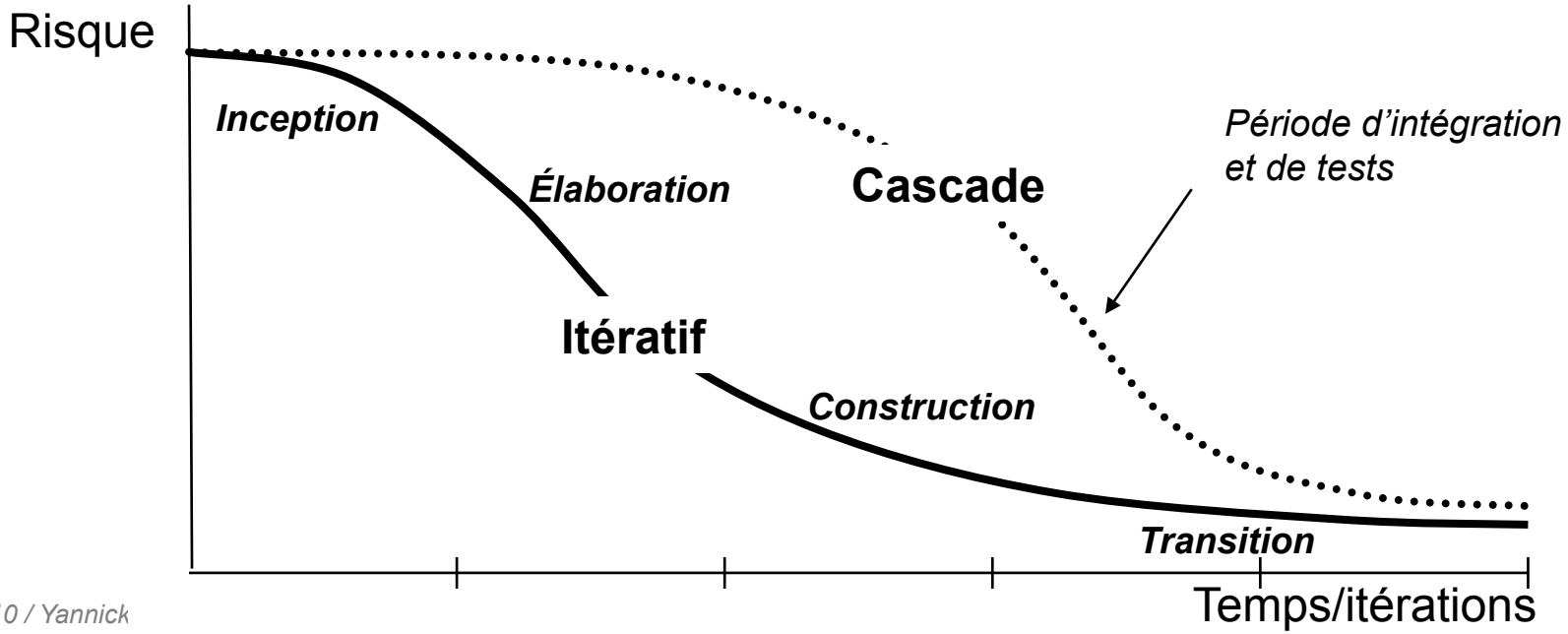
- Trame du processus unifié
- **Processus unifié : caractéristiques essentielles**
  - itératif et incrémental
  - piloté par les besoins
  - **piloté par les risques**
  - centré sur l'architecture
- Description du processus unifié
- Illustration : deux déclinaisons

Processus piloté par les risques

- Différentes natures de risques
  - besoins / technique / autres
- Exemples
  - le système construit n'est pas le bon
  - architecture inadaptée, utilisation de technologies mal maîtrisées, performances insuffisantes
  - personnel insuffisant, problèmes commerciaux ou financiers (risques non techniques, mais bien réels)
- Gestion des risques
  - identifier et classer les risques par importance
  - agir pour diminuer les risques
    - ex. changer les besoins, confiner la portée à une petite partie du projet, faire des test pour vérifier leur présence et les éliminer
  - s'ils sont inévitables, les évaluer rapidement
    - tout risque fatal pour le projet est à découvrir au plus tôt

Pilotage par les risques et itérations

- Principe
  - identifier, lister et évaluer la dangerosité des risques
  - construire les itérations en fonction des risques : s'attaquer en priorité aux risques les plus importants qui menacent le plus la réussite du projet
    - « provoquer des changements précoces »
    - ex. stabiliser l'architecture et les besoins liés le plus tôt possible



Plan

- Trame du processus unifié
- **Processus unifié : caractéristiques essentielles**
  - itératif et incrémental
  - piloté par les besoins
  - piloté par les risques
  - **centré sur l'architecture**
- Description du processus unifié
- Illustration : deux déclinaisons

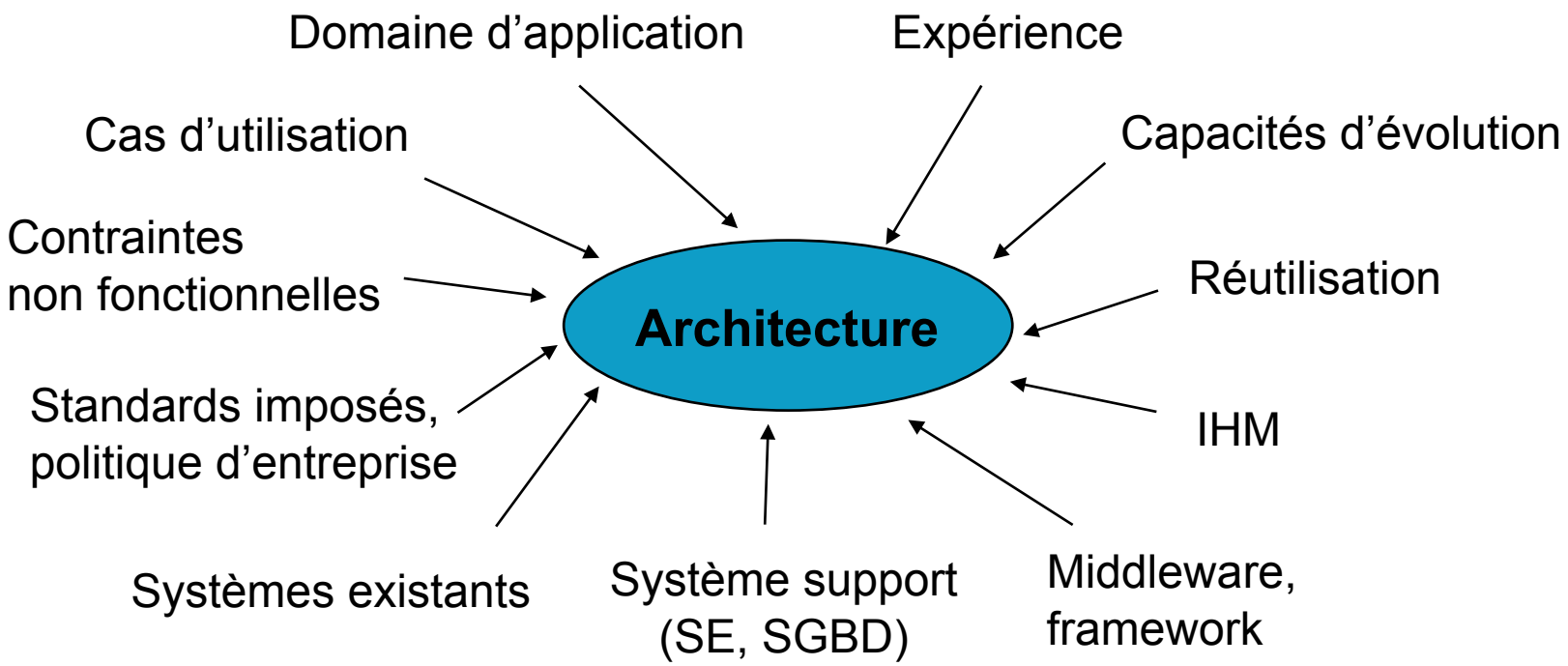
Architecture ?

- Difficile à définir
  - Ex. bâtiment : plombier, électricien, peintre, ne voient pas la même chose.
- Définitions
  - *Software architecture is not only concerned with structure and behavior, but also with usage, functionality, performance, resilience, reuse, comprehensibility, economic and technological constraints and tradeoffs, and esthetics.* (RUP, 98)
  - *A Technical Architecture is the minimal set of rules governing the arrangement, interaction, and interdependance of the parts or elements that together may be used to form an information system.* (U.S. Army 1996)

Architecture

- Définition pour ce cours
  - 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 signifiants qui constituent les fondations du système à développer

Facteurs influençant l'architecture



- Points à considérer

  Performances, qualité, testabilité, convivialité, sûreté, disponibilité, extensibilité, exactitude, tolérance aux changements, robustesse, facilité de maintenance, fiabilité, portabilité, risque minimum, rentabilité économique…

Axes pour considérer l'architecture

- Architecture logicielle (ou architecture logique) :
  - organisation à grande échelle des classes logicielles en packages, sous-systèmes et couches
    - architectures client/serveurs en niveaux (tiers)
    - architectures en couches
      - Présentation, Application, Domaine/métier, Infrastructure métier (services métiers de bas niveau), Services techniques (ex. sécurité), Fondation (ex. accès et stockage des données)
    - architecture à base de composants
    - patterns architecturaux
- Architecture de déploiement
  - décision de déploiement des différents éléments
  - déploiement des fonctions sur les postes de travail des utilisateurs (entreprise : central/départemental/local)

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 : objectif

- Objectif : construire une architecture
  - comme forme dans laquelle le système doit s'incarner
    - les CU réalisés doivent y trouver leur place / la réalisation des CU suivant doit s'appuyer sur l'architecture.
  - qui permette de promouvoir la réutilisation
  - qui ne change pas trop
    - converger vers une bonne architecture rapidement

Construction de l'architecture : principe (1)

- Pour commencer
  - 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

Construction de l'architecture : principe (2)

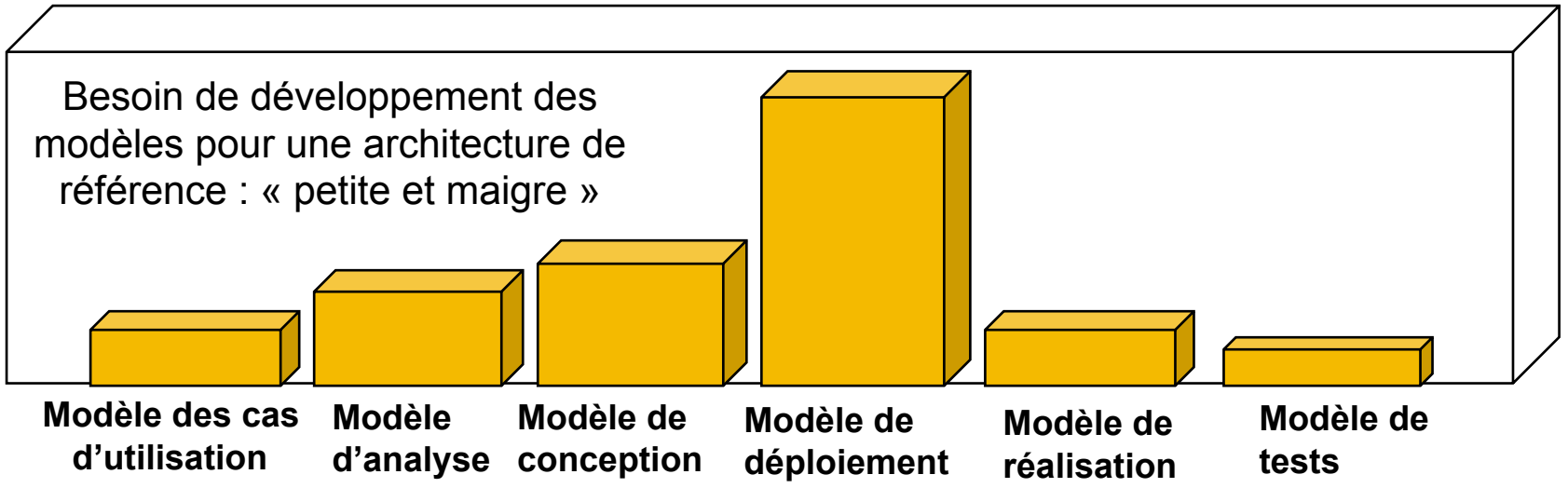
- Ensuite :
  - Construction de l'architecture de référence
    - confrontation à l'ébauche des cas d'utilisation les plus significatifs (un à un)
    - construction de parties de l'application réelle (sous-systèmes spécifiques)
    - stabilisation de l'architecture autour des fonctions essentielles (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.

Construction de l'architecture : principe (3)

- Ensuite :
  - réalisation incrémentale des cas d'utilisation
    - itérations successives
    - l'architecture continue à se stabiliser sans changement majeur
- Remarque : maturation des cas d'utilisation
  - plus faciles à exprimer et plus précis quand un début d'application est disponible

Architecture et élaboration

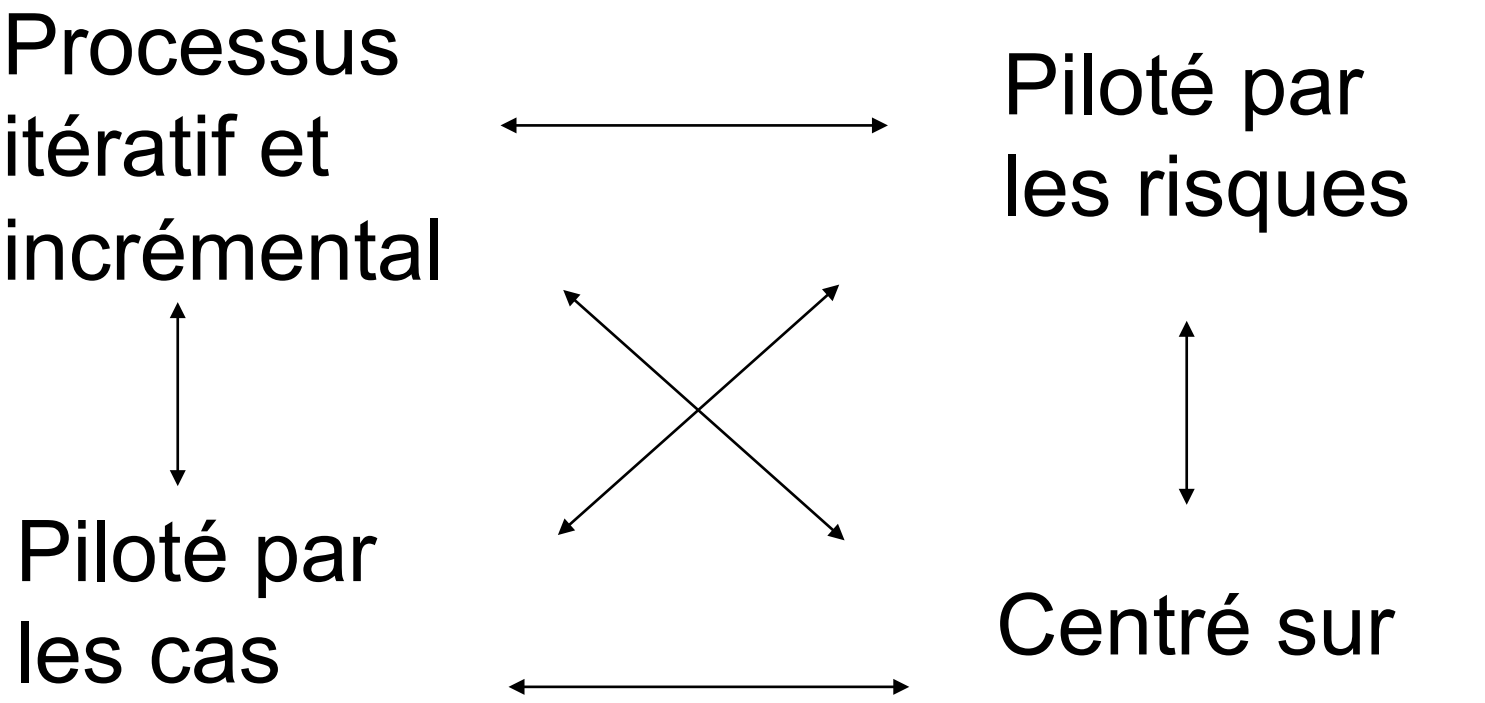
- Phase d'élaboration
  - aller directement vers une architecture robuste, à coût limité, appelée « architecture de référence »
  - 10% des classes suffisent
- L'architecture de référence
  - permettra d'intégrer les CU incrémentalement
  - guidera le raffinement et l'expression des CU pas encore détaillés



Remarque
Processus orienté composants

- On cherche à développer et à réutiliser des composants
  - souplesse, préparation de l'avenir
- Au niveau modélisation
  - regroupement en packages d'analyse, de conception réutilisables
  - utilisation de design patterns
    - architecturaux
    - objets (création, comportement, structure)
- Au niveau production
  - utilisation de frameworks
  - achats de composants

Tous les critères caractérisant les processus UP sont liés



Plan

- Trame du processus unifié
- Processus unifié : caractéristiques essentielles
- **Description du processus unifié**
  - **Activités, travailleurs, artefacts et modèles**
  - Différentes activités pour passer des besoins au code
  - Différentes phases pour piloter les activités
  - Focus sur quelques points importants
- Illustration : deux déclinaisons

Rappel : principes de conception objet

- Passer des besoins aux classes implémentées en réalisant des scénarios comme des collaborations entre objets
  - déduire les responsabilités des collaborations
- S'appuyer sur un modèle du domaine pour créer des objets métiers susceptibles d'évoluer avec les besoins
  - assumer l'évolutivité des besoins
- Favoriser systématiquement la réutilisation
  - en conception : patterns
  - en construction : composants, frameworks

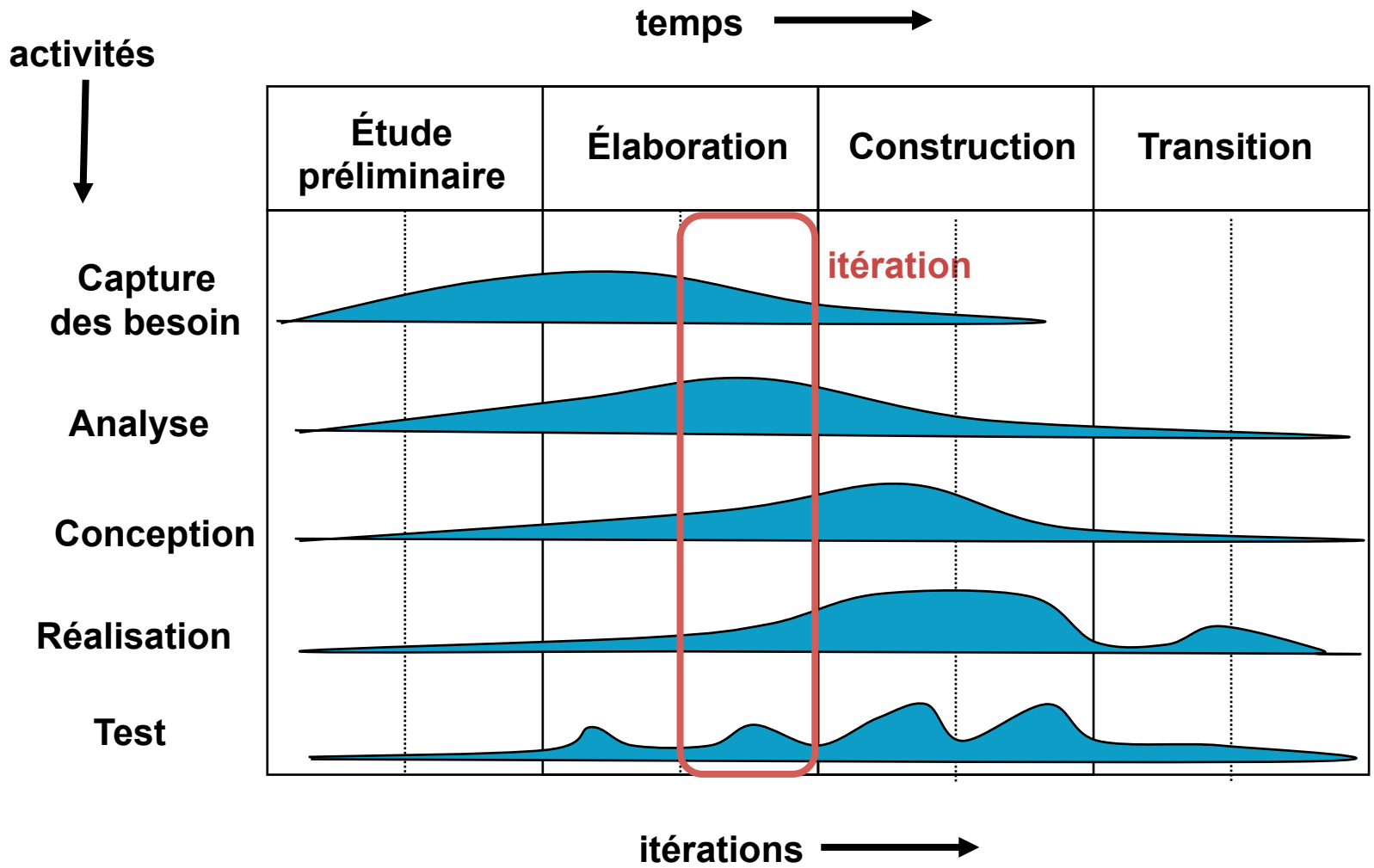
Principe général

- Décrire le modèle du domaine
  - fournira l'ossature du diagramme de classes
- Décrire les cas d'utilisation
  - besoins fonctionnels
- Réaliser les cas d'utilisation un par un
  - avec des interactions entre objets
  - pour compléter le diagramme de classe
- Coder les classes
  - compléter le prototype
- Tester / déployer

Processus unifié : activités et phases

- Activités
  - que faut-il décrire ?
  - sous quelle forme (modèles, documents textuels…) ?
  - comment obtenir les produits ?
  - description technique de la méthode
- Phases
  - planifier les itérations / phases suivantes
  - pour chaque phase :
    - quels sont les buts à atteindre ?
    - quels sont les livrables ?
    - quels aspects décrire, avec quel niveau d'abstraction ?
  - description du déroulement du projet

Les activités selon les phases

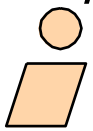


USDP

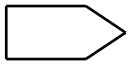
- UP définit un enchaînement d'activités
  - réalisées par un ensemble de travailleurs (rôles, métiers)
  - ayant pour objectif de passer des besoins à un ensemble cohérent d'artefacts constituant un système informatique
  - et de favoriser le passage à un autre système quand les besoins évolueront (nouvelle version)
- UP n'est qu'un cadre général de processus
  - un projet particulier est une instance de ce cadre adaptée au contexte du projet (taille, personnels, entreprise, compréhension du processus, etc.)

Activités, travailleurs, artefacts

- Artefacts (quoi)
  - documentent le système et le projet (traces et produits)
  - ex. : modèle architectural, code source, exécutable, modèle des CU, etc.
- Travailleurs ou discipline (qui)

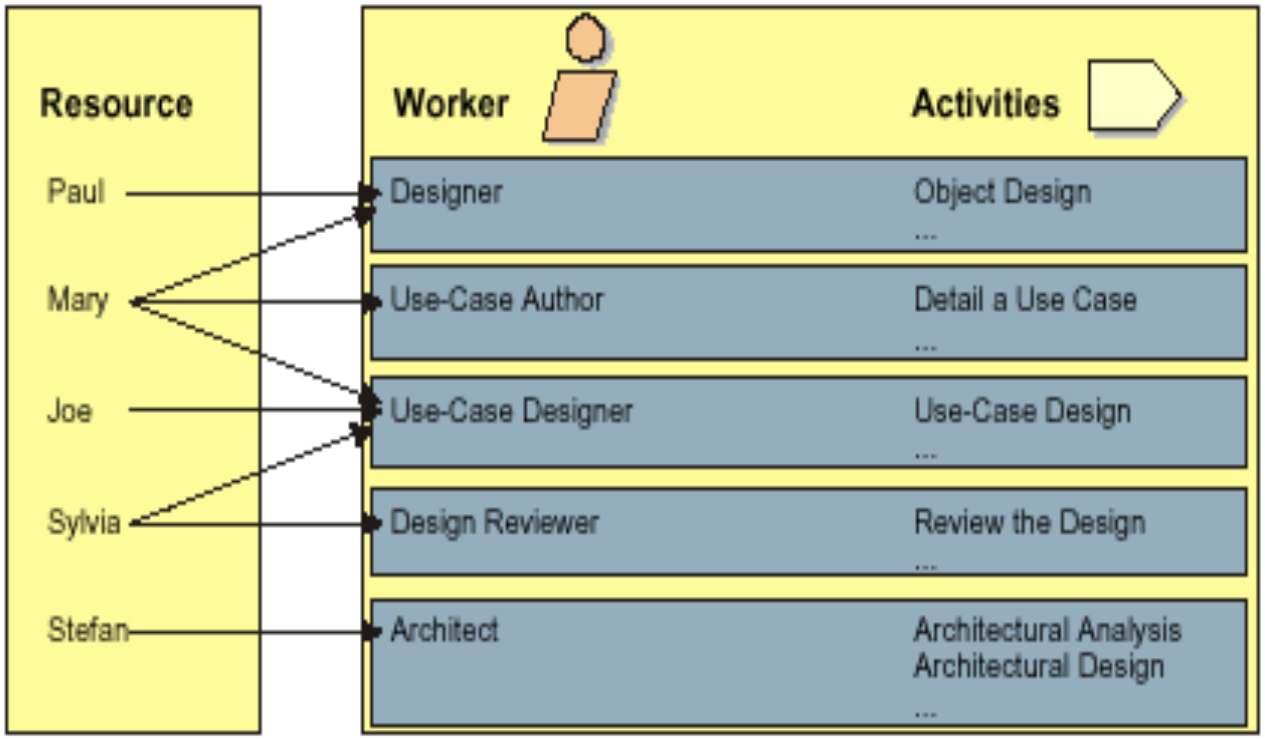


  - rôle par rapport au projet
  - ex. : architecte, analyste de CU
- Activités (comment)



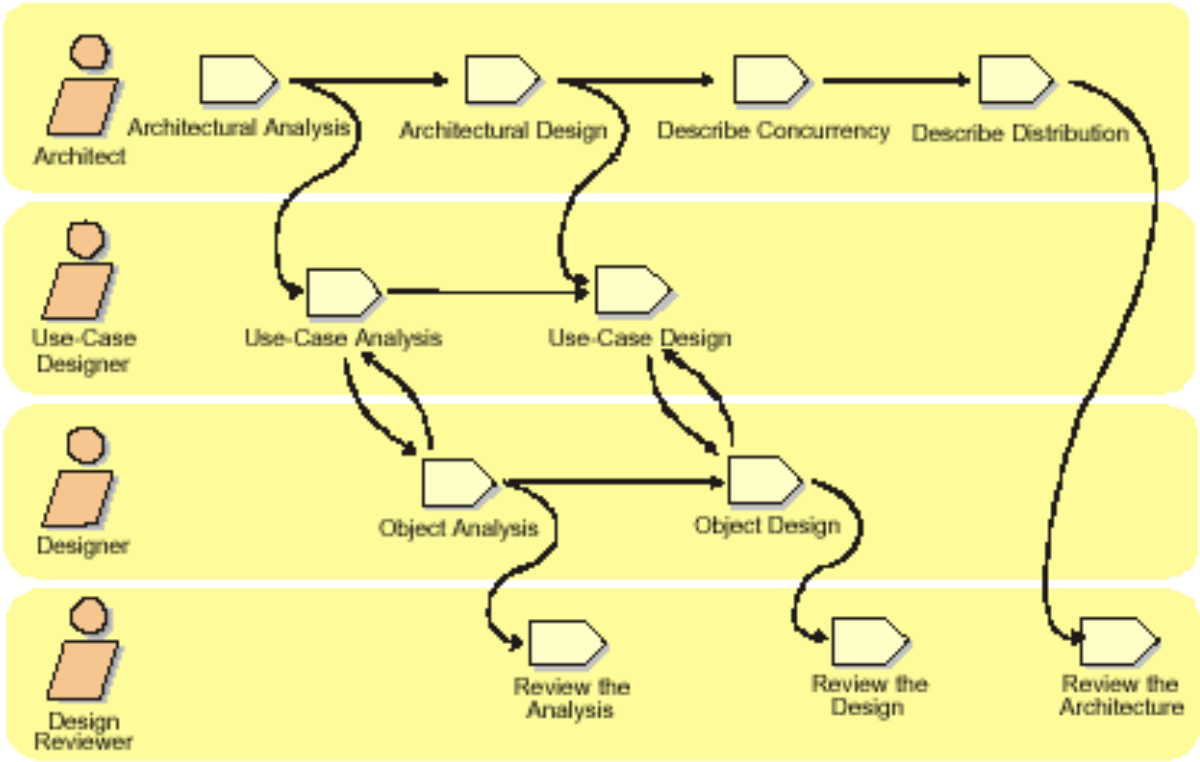
  - 5 grandes activités, multiples sous-activités
    - tâches réalisée par un travailleur, impliquant une manipulation d'information
  - exemples
    - concevoir une classe, corriger un document, détailler un CU, etc.

Travailleurs et activités

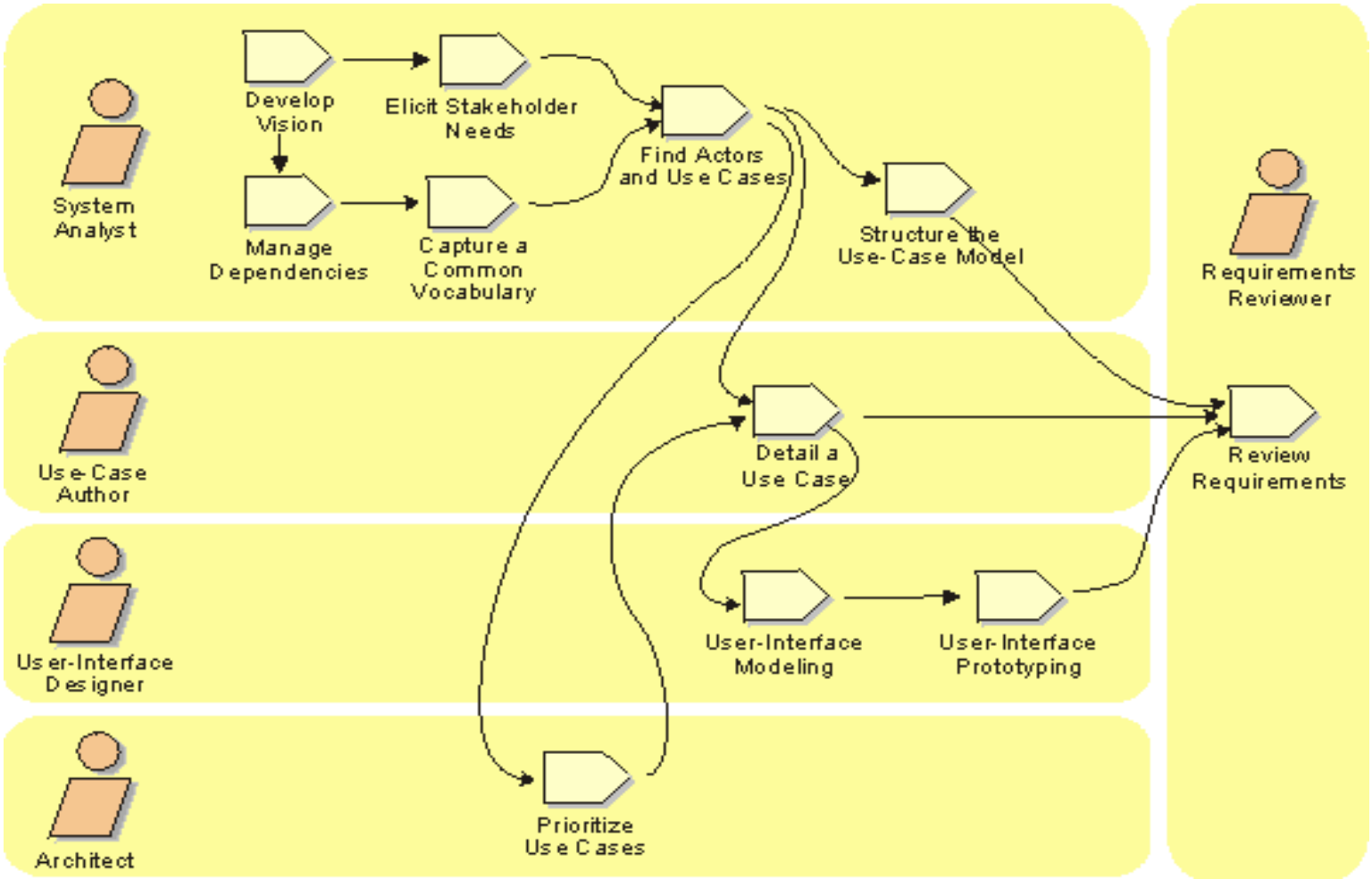


Workflows

- Enchaînement d'activités qui produisent des artefacts
- Diagramme d'activité



Exemple RUP : requirement workflow

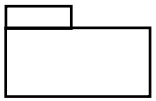


Les modèles du processus

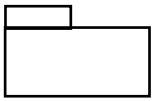
- Objectif de la modélisation
  - créer un modèle global composé de modèles liés aux différentes activités
    - et pas une montagne de documents
- Les modèles sont liés
  - par les cas d'utilisation
  - par les enchaînements d'activités qui ont permis de les mettre en place
- Rappel
  - la description de l'architecture utilise une sous-partie des modèles

Modèles d'un projet USDP

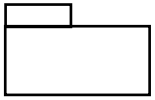
- Les activités consistent (entre autres) à créer des modèles



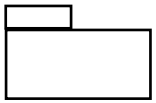
Modèle des cas d'utilisation



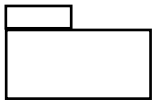
Modèle d'analyse



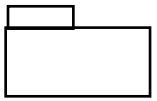
Modèle de conception



Modèle de test



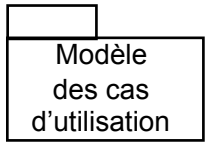
Modèle de déploiement

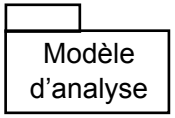


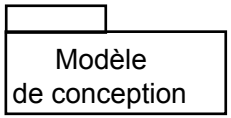
Modèle d'implémentation

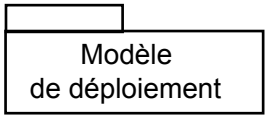
Vue globale des modèles

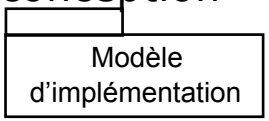
- Capture des besoins
  - le modèle des CU représente le système vu de l'extérieur, son insertion dans l'organisation, ses frontières fonctionnelles.
- Analyse
  - le modèle d'analyse représente le système vu de l'intérieur. Les objets sont des abstractions des concepts manipulées par les utilisateurs. Point de vue statique et dynamique sur les comportements.
- Conception
  - le modèle de conception correspond aux concepts utilisés par les outils, les langages et les plateformes de développement.
  - le modèle de déploiement spécifie les nœuds physiques et la distribution des composants. Permet d'étudier, documenter, communiquer et d'anticiper une conception
- Implémentation
  - le modèle d'implémentation lie le code et les classes de conception
- Tests
  - le modèle de tests décrits les cas de tests

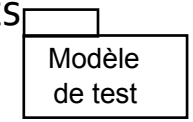












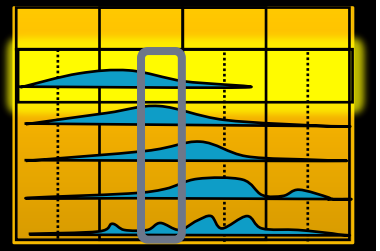
Avertissement sur la suite

- Ce n'est pas forcément du UP générique complet
  - insistance sur certains points plutôt que d'autres, utilisation de plusieurs sources à peu près cohérentes
- S'appuie largement sur le cours de JL Sourrouille (INSA-Lyon)
  - trame description / exemple

Plan

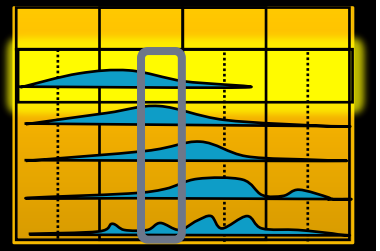
- Trame du processus unifié
- Processus unifié : caractéristiques essentielles
- **Description du processus unifié**
  - Activités, travailleurs, artefacts et modèles
  - **Différentes activités pour passer des besoins au code**
  - Différentes phases pour piloter les activités
  - Focus sur quelques points importants
- Illustration : deux déclinaisons

Activité *acquisition des besoins :* objectifs



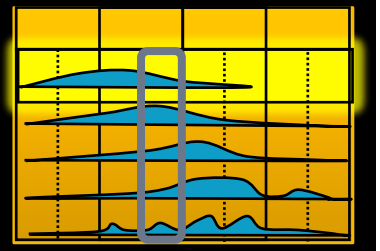
- Déterminer les valeurs attendues du nouveau système et de la nouvelle organisation
  - arriver à un accord client / développeurs
- Recenser les besoins potentiels
  - caractéristiques potentielles, priorité, risques…
- Comprendre le contexte du système
  - association d'analystes et d'experts métier pour construire un vocabulaire commun
  - modèle du domaine (ou modèle de l'entreprise)
    - diagramme de classes
  - modèle du métier (éventuellement)
    - modélisation des processus (CU métier, diagrammes de séquences, activité)
  - glossaire

Activité *acquisition des besoins :*
Liste initiale des besoins



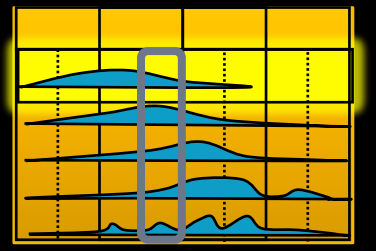
- Une petite chaîne d'hôtels a décidé de mettre sur le web un système de réservation de chambres ouvert à tout client, et d'autre part elle veut automatiser la gestion de ses hôtels. Un hôtel est géré par un directeur assisté d'employés.
- Pour réserver à distance, après avoir choisi hôtels et dates, le client fournit un numéro de carte bleue. Lorsque le retrait a été accepté, la réservation devient effective et une confirmation est envoyée par mail. Les clients sous contrat (agences de voyage…) bénéficient d'une réservation immédiate.
- Le directeur de l'hôtel enregistre les réservations par téléphone. Si un acompte est reçu avant 72h, la réservation devient effective, sinon elle est transformée en option (toute personne ayant payé a la priorité). Si la réservation intervient moins de 72h avant la date d'occupation souhaitée, le client doit se présenter avant 18h.
- Le directeur fait les notes des clients, perçoit l'argent et met à jour le planning d'occupation effectif des chambres.
- Une chambre est nettoyée soit avec l'accord du client lorsqu'il reste plusieurs jours, soit après le départ du client s'il s'en va, et dans ce cas avant occupation par un nouveau client. Les employés s'informent des chambres à nettoyer et indiquent les chambres nettoyées au fur et à mesure. Pour cela les chambres vides à nettoyer doivent être affichées, et les employés doivent pouvoir indiquer les chambres nettoyées de façon très simple. Un historique des chambres nettoyées par chaque employé est conservé un mois.

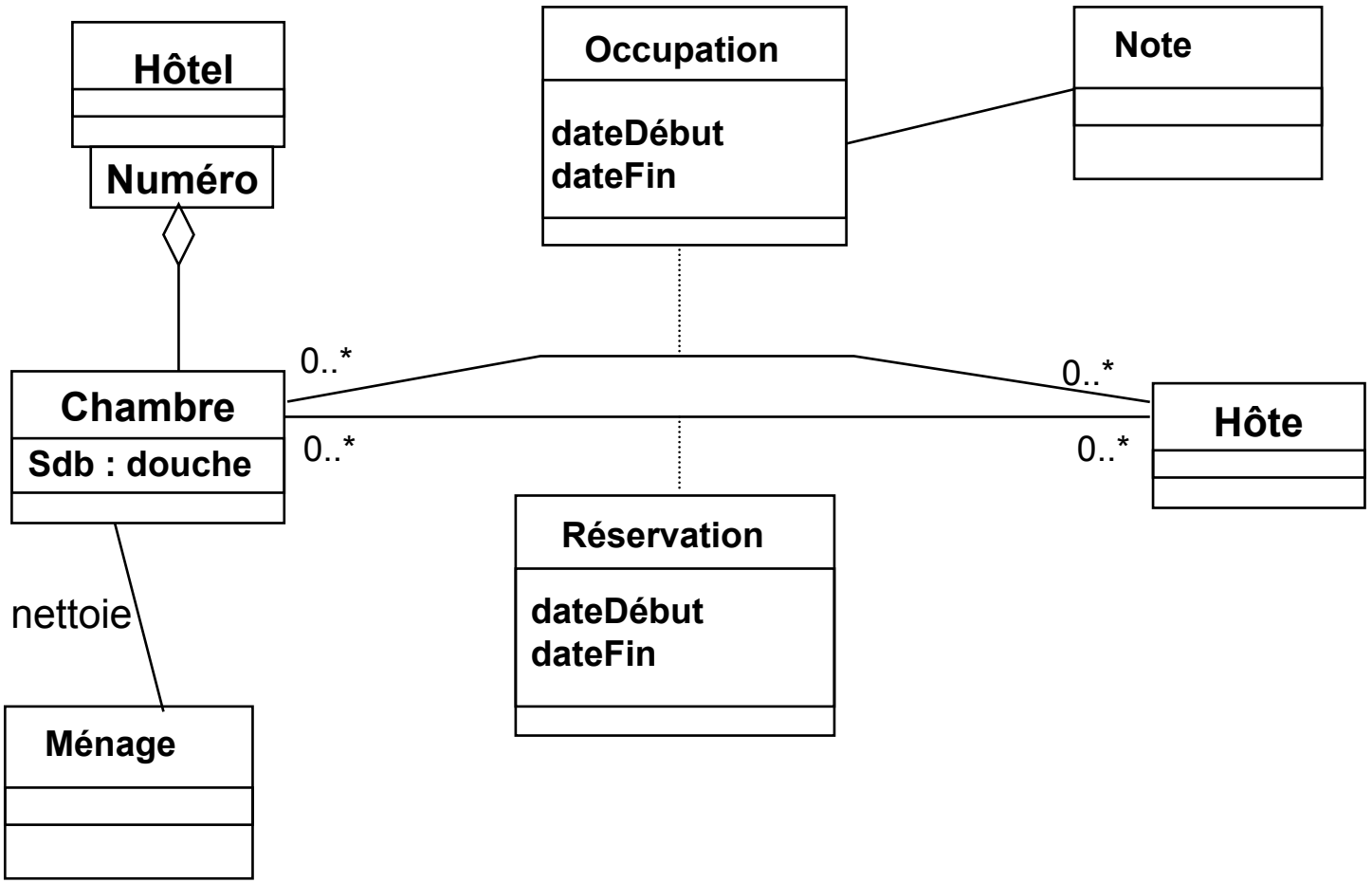
Activité *acquisition des besoins :*
Modèle du domaine



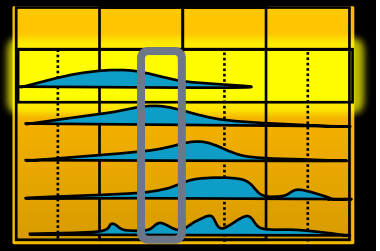
- Représentation des classes conceptuelles d'une situation réelle
  - objets métier (ex. Réservation), du monde réel (ex. Chambre), événements
  - quelques attributs, peu d'opérations
  - associations (s'il y a nécessité de conserver la mémoire de la relation)
  - les classes non retenues sont placées dans un glossaire
- « Dictionnaire visuel » du domaine construit surtout pendant la phase d'élaboration, itérativement en fonction des CU considérés
- Servira à réduire le décalage des représentation entre domaine et objets logiciels
  - inspiration pour la construction de la couche domaine de l'architecture logicielle (parfois aussi appelée modèle de domaine : ne pas mélanger)

Activité *acquisition des besoins :* modèle du domaine





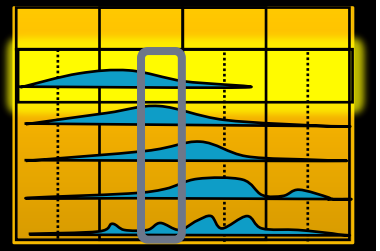
Activité *acquisition des besoins :*
Pour construire un modèle du domaine



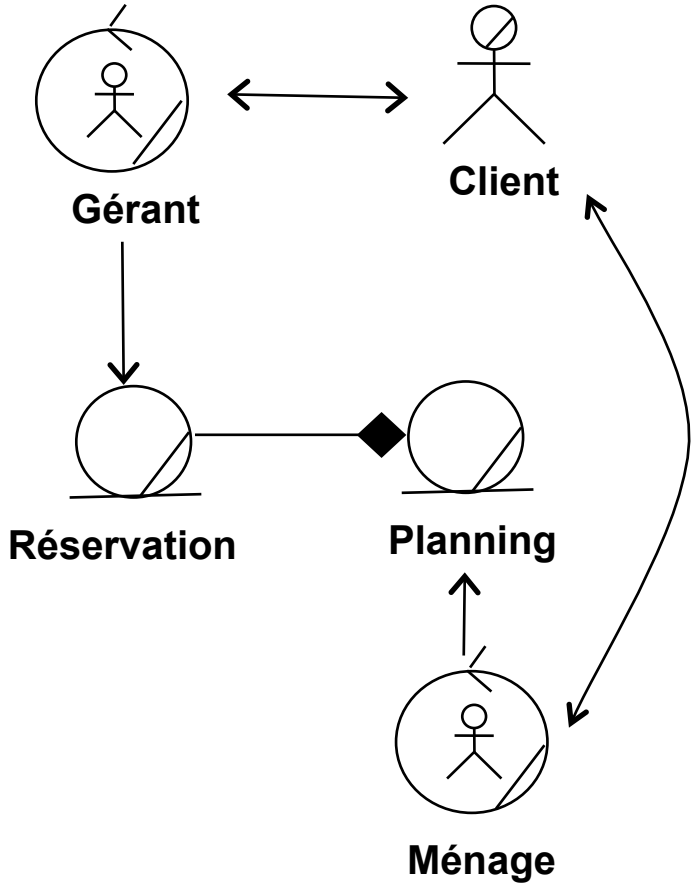
(Larman)

- Réutiliser/modifier des modèles existants
- Lister les catégories
  - classes : objets physiques, transactions, autre systèmes informatiques, organisations, documents de travail, etc.
  - associations : A est membre de B, A est une transaction liée à une transaction B, A est une description de B, etc.
- Utiliser des groupes nominaux
  - extraits par exemple des CU détaillés

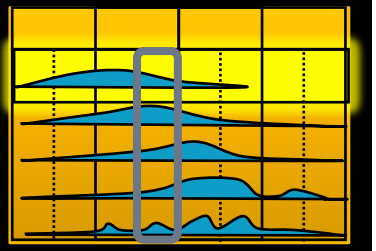
Activité *acquisition des besoins :*
Modèle du métier (facultatif)

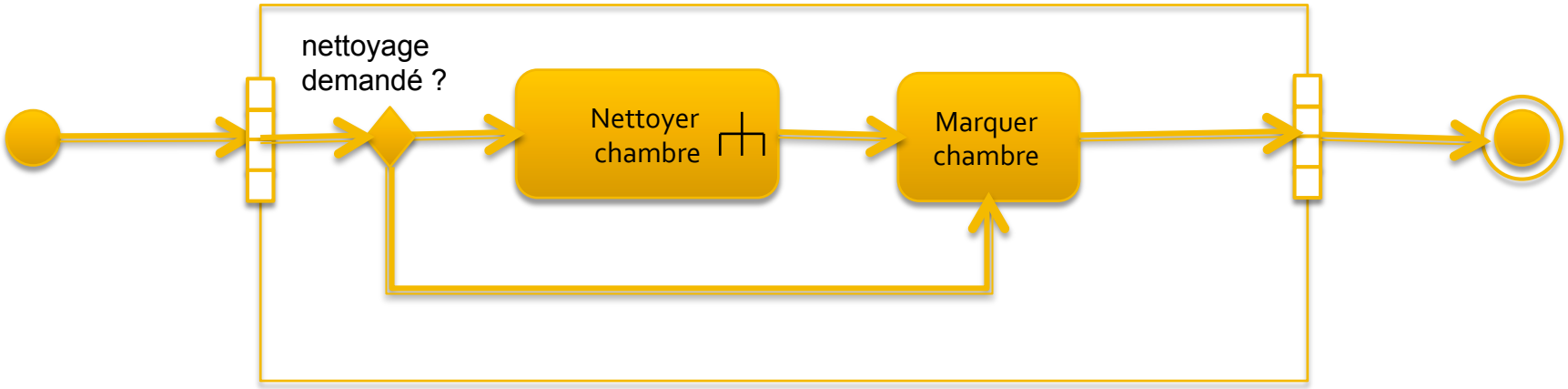


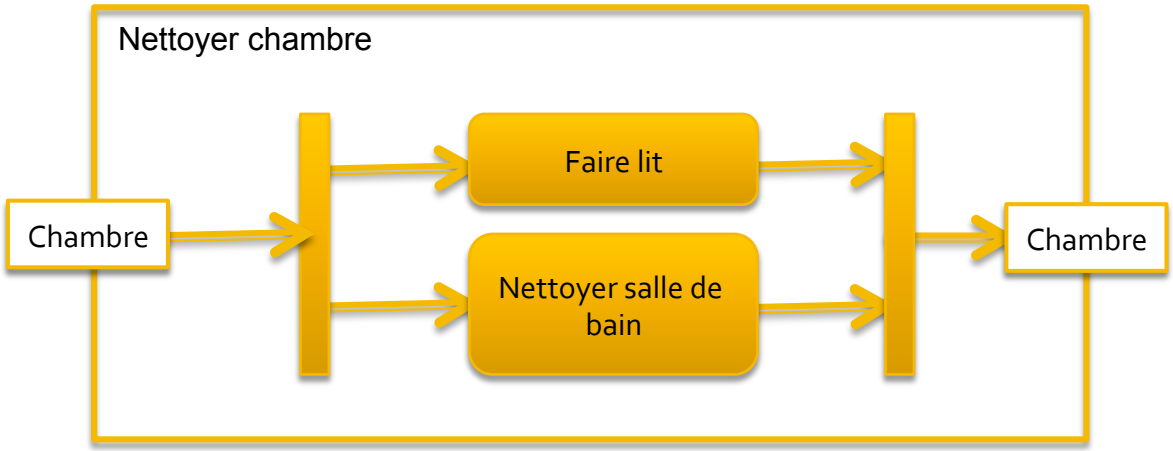
- Modèle des CU métier
  - représenter les processus
  - acteurs métier
  - CU organisation boîte blanche, objectif stratégique
- Modèle objet métier
  - montre comment les CU métier sont réalisés par
    - acteurs métier
    - travailleurs métier
    - entités métier



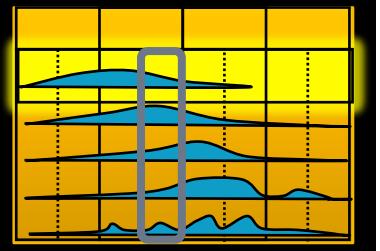
Activité *acquisition des besoins*
Modélisation de processus métiers

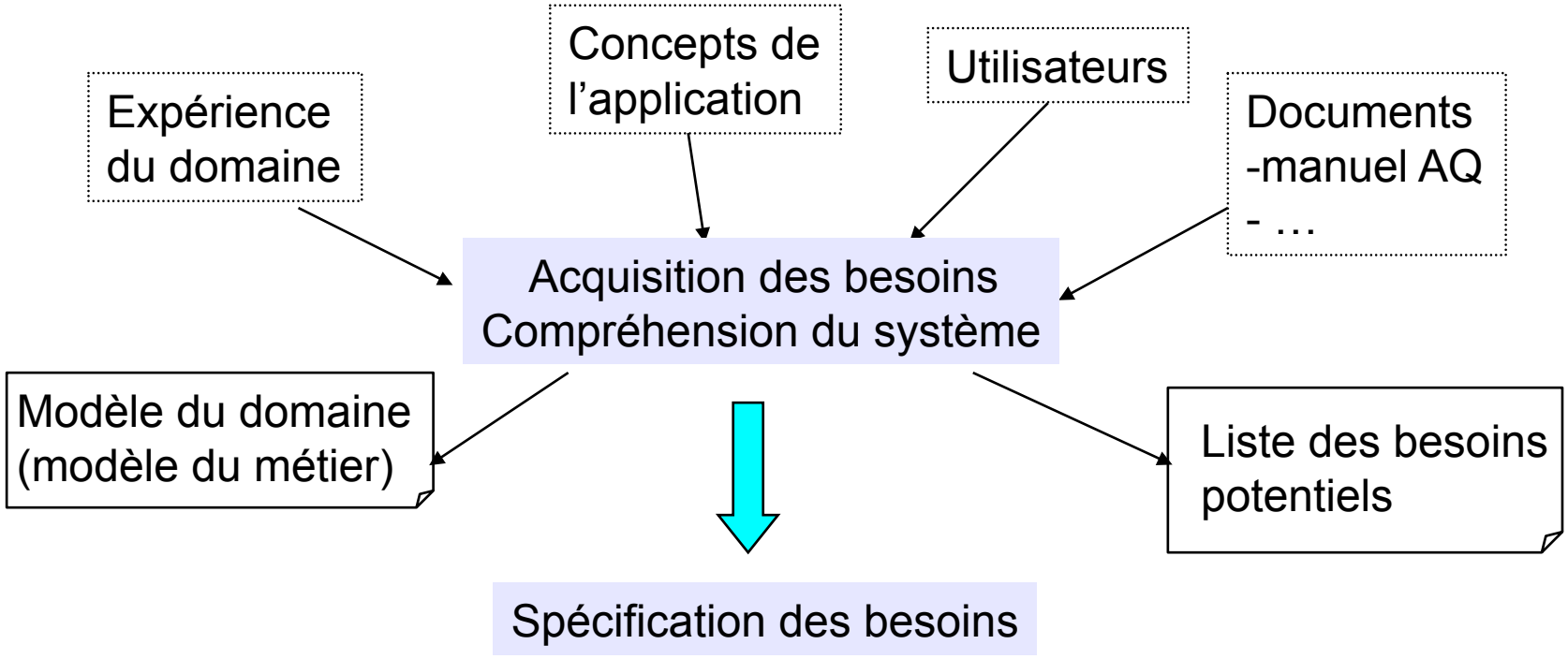




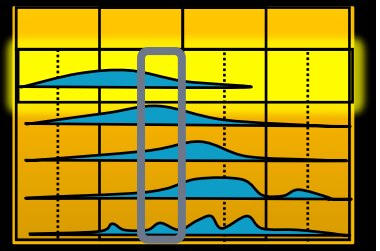


Activité *acquisition des besoins :*
Produits de l'acquisition des besoins



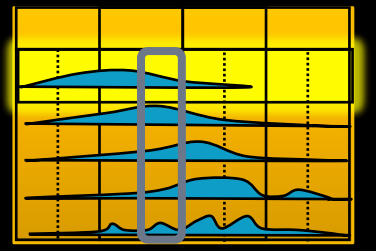


Activité *expression des besoins*
Objectifs (1/4)

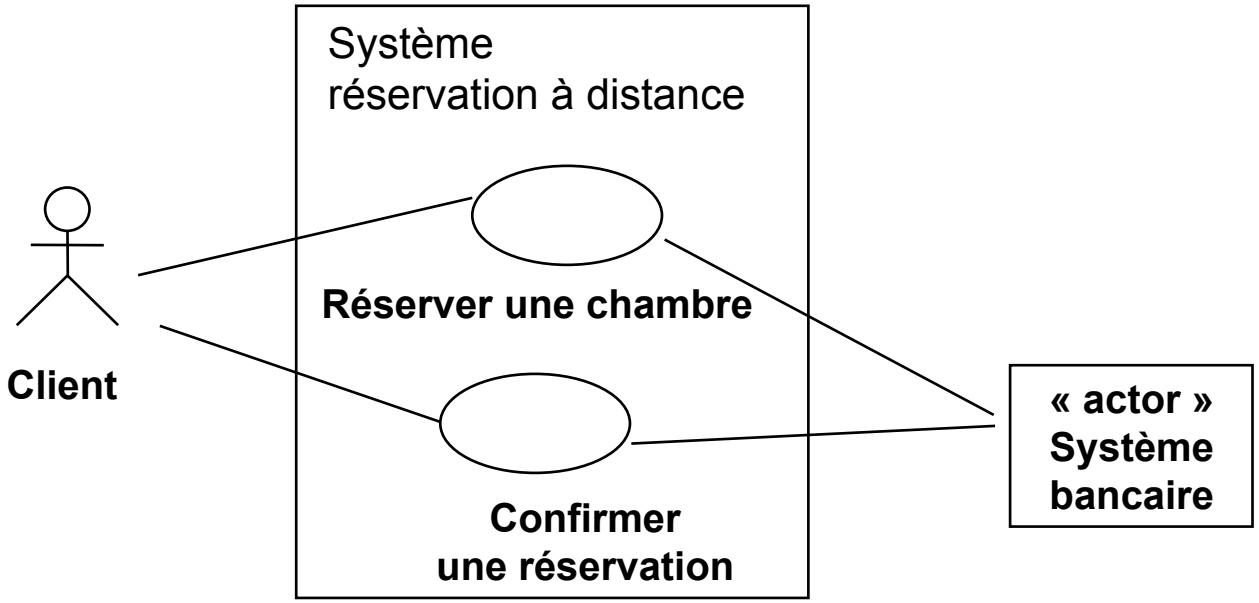


- Définir les besoins fonctionnels
  - écrire les récits d'utilisation
  - définir les acteurs et leurs objectifs
    - limites du système à construire
      - interactions avec le système
  - définir les cas d'utilisation
    - portée système / objectif utilisateur
    - description brève, puis plus détaillée
  - construire le modèle des cas d'utilisation
    - organiser les cas d'utilisation
      - utiliser des CU portée organisation

Activité *expression des besoins*
Exemple : description des CU

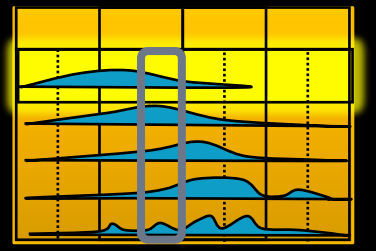


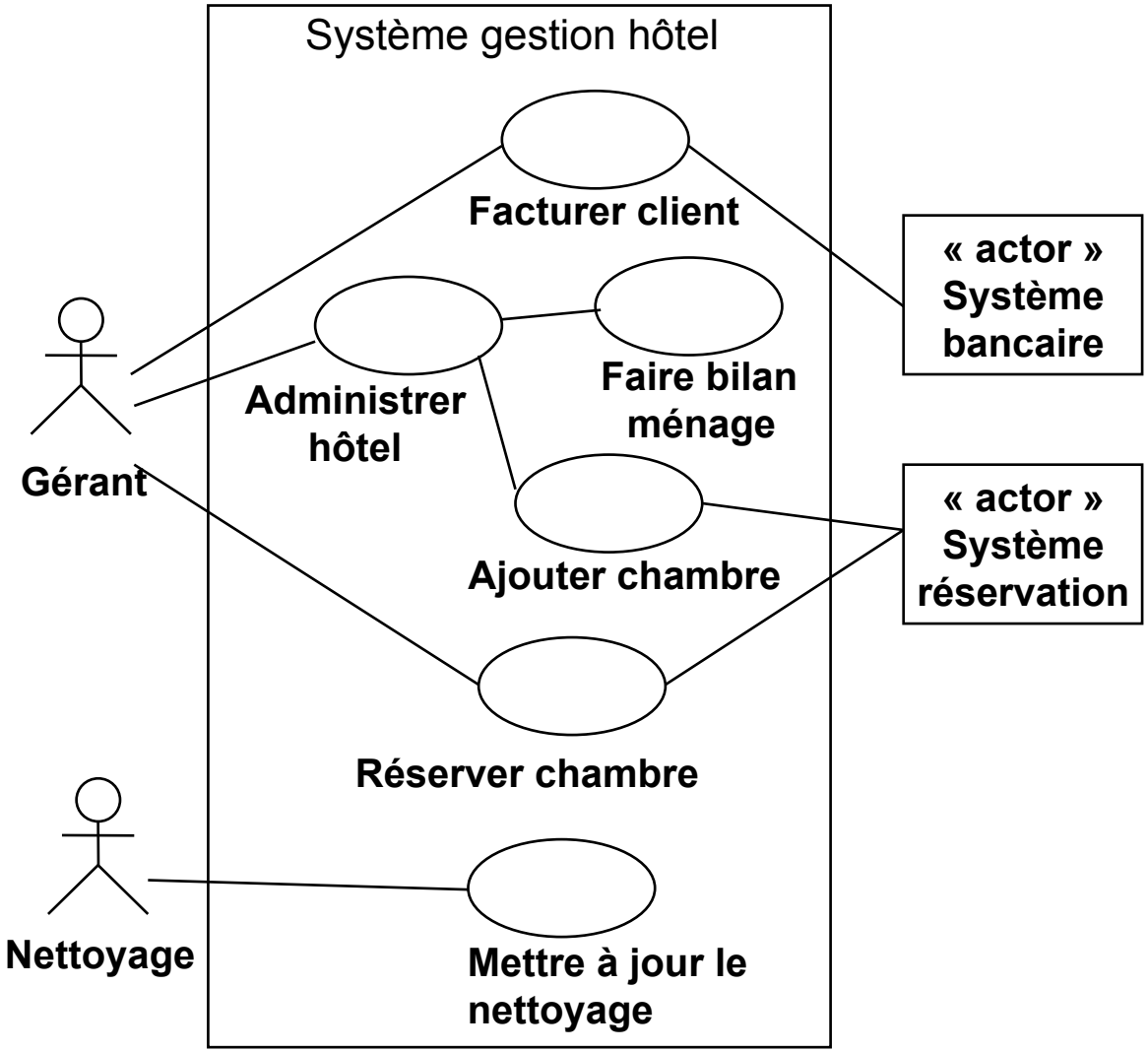
Le système à développer est divisé en deux sous-systèmes indépendants : le système de réservation à distance et le système de gestion local à l'hôtel.
La base de données des réservations est considérée comme un système externe mais non détaillé (sous-système classique).
Localement, la base de données est considérée comme interne au système et ignorée pour l'instant.



Activité *expression des besoins*

Description des CU





Administrer hôtel

Le gérant gère les chambres, leur catégorie, leur prix... Il peut déclarer qu'une chambre est momentanément non utilisable (travaux...). Il fait le bilan mensuel des employés de nettoyage. Il tient à jour une liste des services et les prix (petit déjeuner en salle, dans la chambre...)...

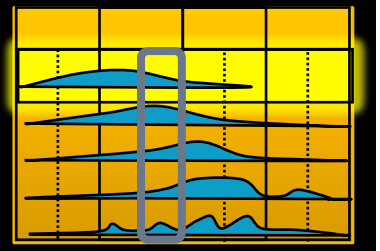
Ajouter chambre

Le gérant crée une chambre dans le système, avec toutes ses caractéristiques.

Facturer client

...

Activité *expression des besoins*
Objectifs (2/4)

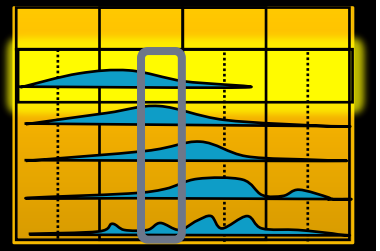


- Classer les cas d'utilisation par priorité
  - la priorité dépend des risques associés au cas d'utilisation et de leur importance pour l'architecture, des nécessités de réalisation et de tests

Les traitements très classiques de la gestion locale sont à examiner en dernier (risque faible). Les réservations (client ou gérant) mettent en jeu une architecture plus complexe et sont à examiner en priorité. Le système de gestion des chambres pour le ménage peut entraîner des questions techniques difficiles (équipement mobile).

1. Réserver une chambre sur le web
2. Réserver une chambre (locale)
3. Trouver la prochaine chambre à nettoyer
4. Administrer
5. ...

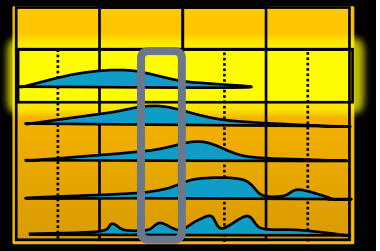
Activité *expression des besoins*
Objectifs (3/4)



- Détailler et formaliser les cas d'utilisation
  - scénarios
    - séquence d'étape nominale
    - extensions
  - compléter la description des scénarios
    - éventuellement diagrammes de séquence système, diagrammes d'activité, de machines d'états
    - maquette IHM
      - uniquement si l'interface est complexe ou nécessite une évaluation par le client
- Structurer le modèle
  - réorganiser si besoin

Utilisateurs non spécialistes, interface simple et logique. Problème classique, sans risque majeur, donc pas de maquette.

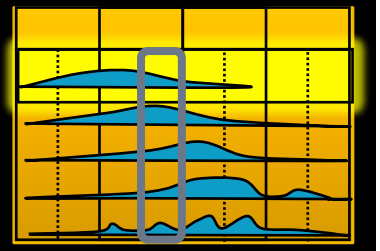
Activité *expression des besoins*
Détails des cas d'utilisation



- Description détaillée
  - scénario nominal, extensions (voir cours sur la rédaction des CU)

CU : Réserver une chambre
Portée : système de réservation
Niveau : objectif utilisateur
Acteur principal : Gérant
Intervenants et intérêts : Client, Chaîne hôtelière
Préconditions : une chambre est libre pour la période désirée
Garanties minimales : rien ne se passe
Garanties en cas de succès : la chambre est réservée
Scénario nominal
1. Le gérant demande le planning d'occupation pour la période qui vient. Le système affiche le planning sur plusieurs semaines.
2. Le gérant sélectionne une chambre libre pour une date qui l'intéresse. Le système lui présente le récapitulatif de cette chambre, et sa disponibilité quelques jours avant et après la date choisie.
…

Activité *expression des besoins*
Objectifs (4/4)



- Appréhender les besoins non fonctionnels (contraintes sur le système : environnement, plate-forme, fiabilité, vitesse…)
  - rattacher si possible les besoins aux cas d'utilisation
    - description dans les descriptions des CU (section « exigences particulières » pour UP)
  - sinon, dresser une liste des exigences supplémentaires

La chaîne possède 117 hôtels de 30 chambres en moyenne. Les appels sur le réseau sont évalués à 300 par jour (au début, prévoir des évolutions).

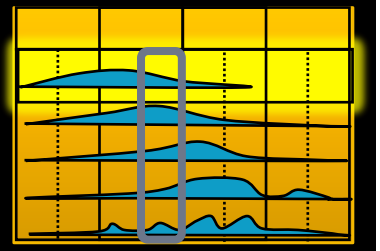
Pour des raisons d'extensibilité, de performances et de sécurité, la chaîne de traitement des réservations des clients doit être indépendante des liaisons des hôtels avec le système de réservation. Les hôtels sont reliés en permanence au système de réservation aux coupures près (ADSL) et les postes devront être fiables (reprise sur erreur après coupures de courant...).

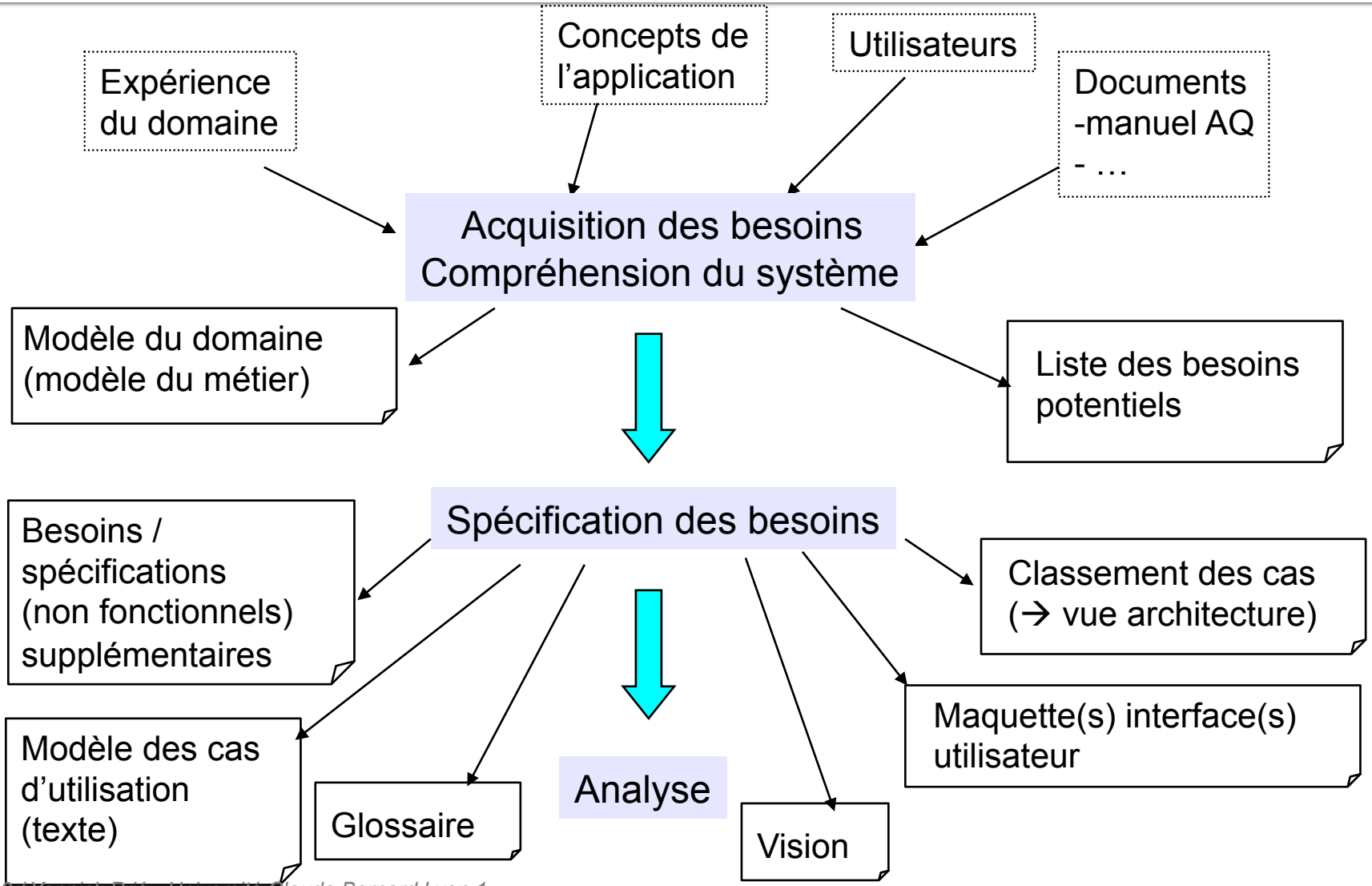
Le temps d'apprentissage du logiciel par les acteurs professionnels ne doit pas dépasser une demi-journée.

Une société tierce s'occupera de la maintenance du système et abritera les serveurs.

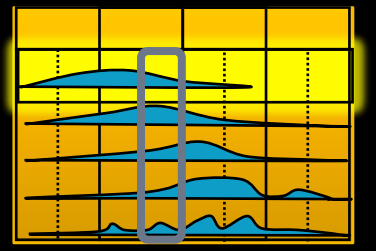
Activité *expression des besoins*

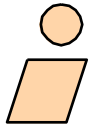
Artefacts



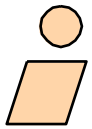


Activité *expression des besoins*
Expression des besoins : travailleurs

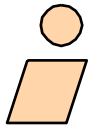




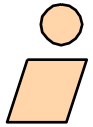
- Analyste système, du domaine
  - modèle du domaine / du métier
  - modèle des CU / acteurs
  - glossaire



- Spécificateur de cas d'utilisation
  - CU détaillés



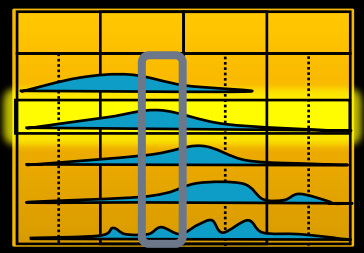
- Concepteur d'interface utilisateur
  - maquette/prototype



- Architecte
  - vue architecturale du modèle des CU

Activité *analyse*

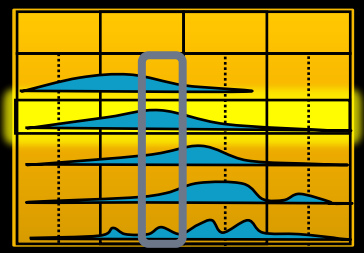
Objectif général



- Construire le modèle d'analyse pour préparer la conception
  - forme générale stable du système
  - haut-niveau d'abstraction
  - réalisation des cas d'utilisation par des objets/ classes d'analyse
  - passage du langage du client à celui du développeur

Activité *analyse*

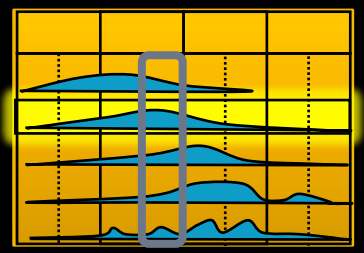
Objectifs (1/4)

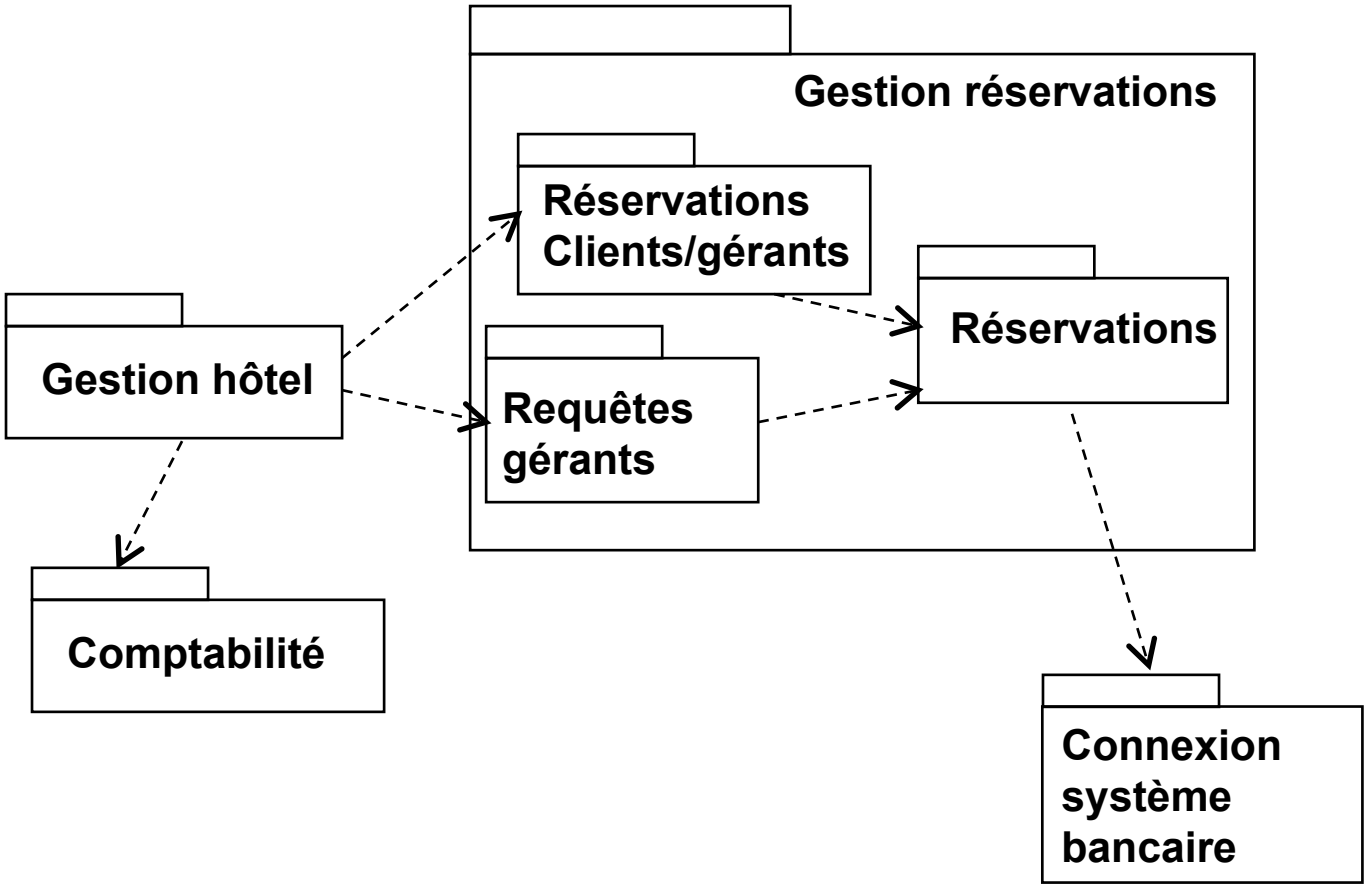


- Mener l'analyse architecturale
  - identifier les paquetages d'analyse
    - regroupement logique indépendant de la réalisation
    - relations de dépendances, navigabilité entre classes de paquetage différents
    - à partir des CU et du domaine
    - servira de point de départ pour le découpage en sous-systèmes

Activité *analyse*

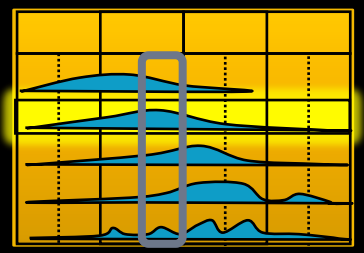
Découpage en paquetages





Activité *analyse*

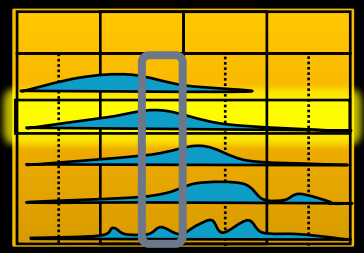
Objectifs (2/4)



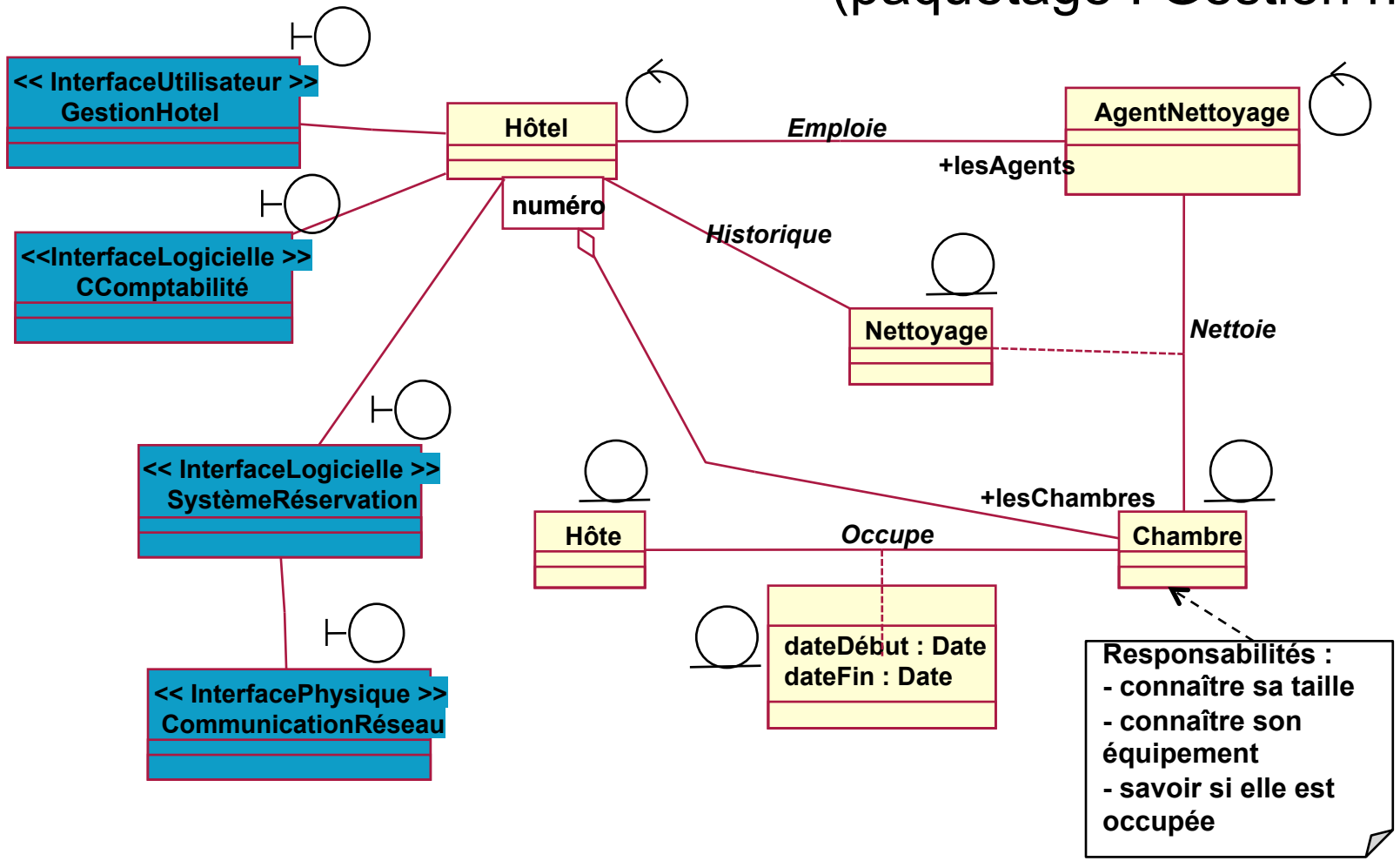
- Mener l'analyse architecturale (suite)
  - Identifier les classes entités manifestes (premier modèle structurel)
    - modèle des 10-20 classes constituant l'essence du domaine (à partir du modèle du domaine/métier)
    - 3 stéréotypes de classe : frontière, contrôle, entité
    - responsabilités évidentes
  - Identifier les exigences particulières communes
    - distribution, sécurité, persistance, tolérance aux fautes…
    - les rattacher aux classes et cas d'utilisation

Activité *analyse*

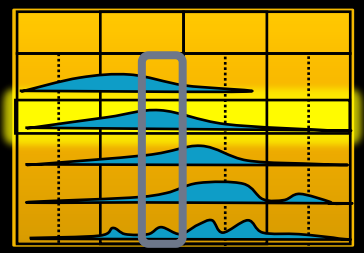
Premier modèle structurel



(paquetage : Gestion hôtel)



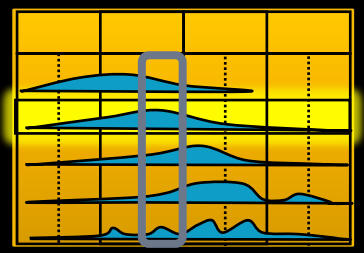
Activité *analyse*
Objectifs (3/4)



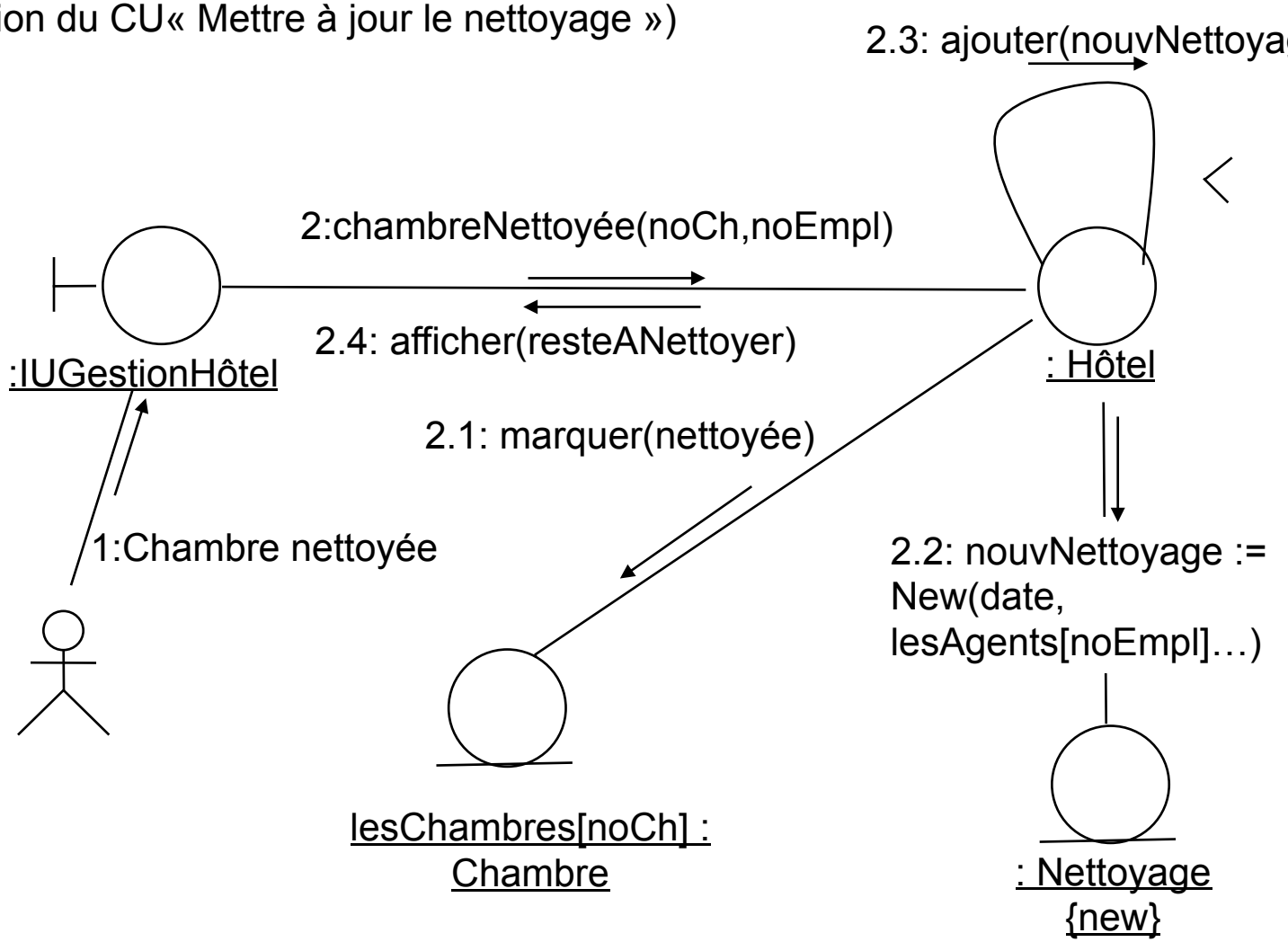
- Analyser les cas d'utilisation
  - réaliser les scénarios des cas d'utilisation
  - identifier les classes, attributs et relations
    - examiner l'information nécessaire pour réaliser chaque scénario
    - ajouter les classes isolant le système de l'extérieur (interfaces physiques, vues externes des objets…)
    - éliminer les classes qui n'en sont pas : redondantes, vagues, de conception, etc.
  - décrire les interactions entre objets
    - diagrammes de séquence
    - diagrammes de communication
    - si scénario trop complexe, modéliser par parties (enchaînements), et indiquer les branchements
- Le modèle structurel sera construit pour supporter l'union des collaborations et interactions exprimées

Activité *analyse*

Diagramme de communication

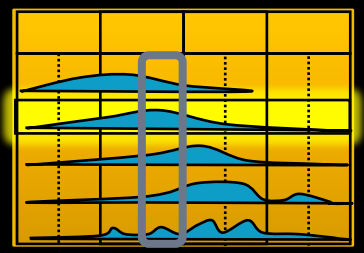


(Réalisation du CU« Mettre à jour le nettoyage »)



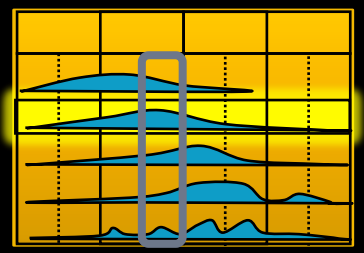
Activité *analyse*

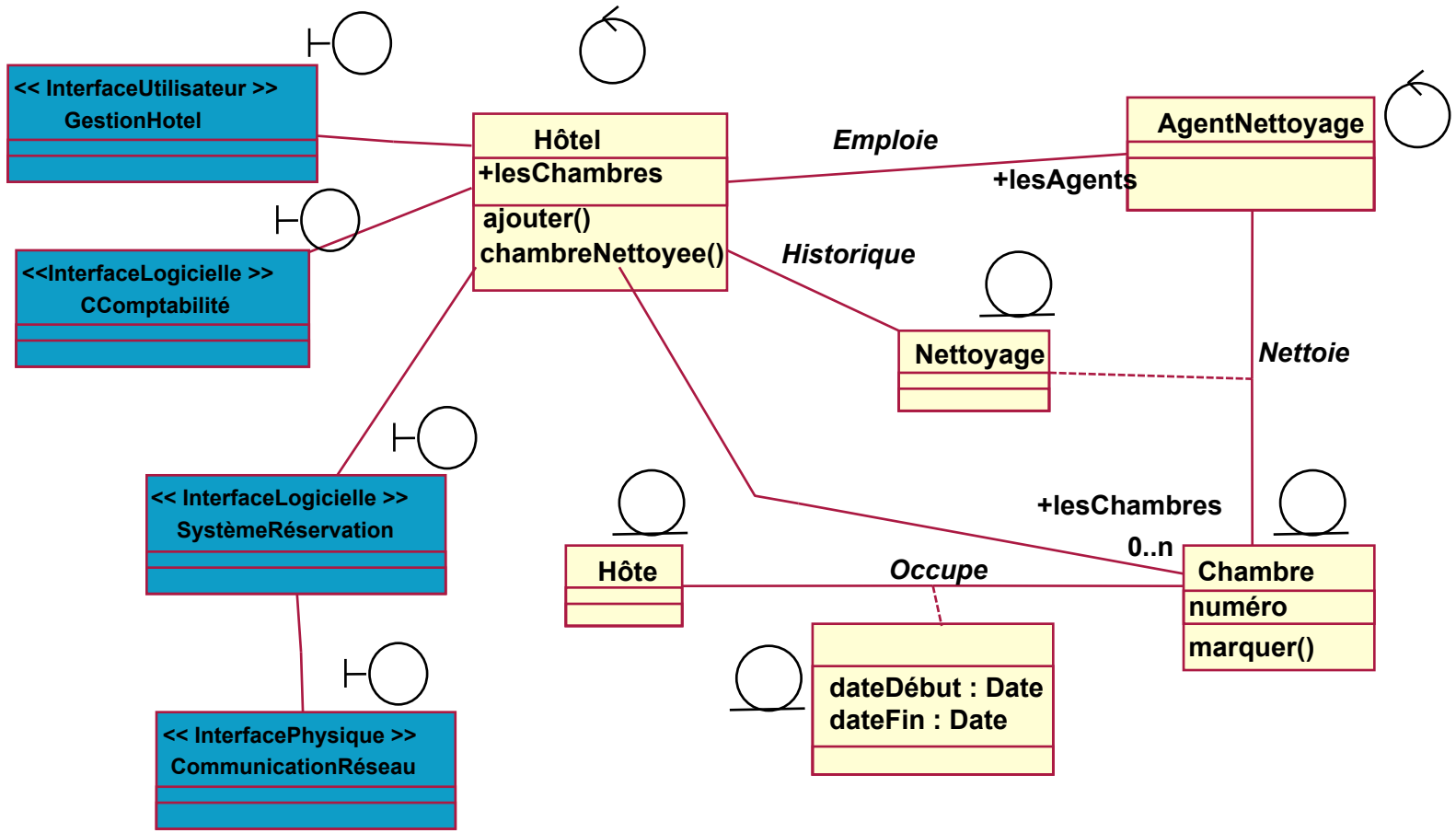
Objectifs (4/4)



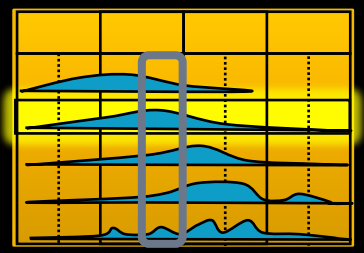
- Préciser les classes d'analyse
  - faire le bilan des responsabilités à partir des collaborations
    - responsabilité d'une classe = union des rôles dans tous les cas (négliger en analyse les opérations implicites)
  - identifier les attributs
    - rester simple, pas choix de conception à ce niveau
  - identifier les associations
  - identifier les relations d'héritage
  - identifier les besoins spéciaux des classes
- Vérifier les paquetages d'analyse
  - dépendances, couplages
  - ils seront à la base des composants / sous-systèmes
- Prendre un nouveau cas d'utilisation et recommencer

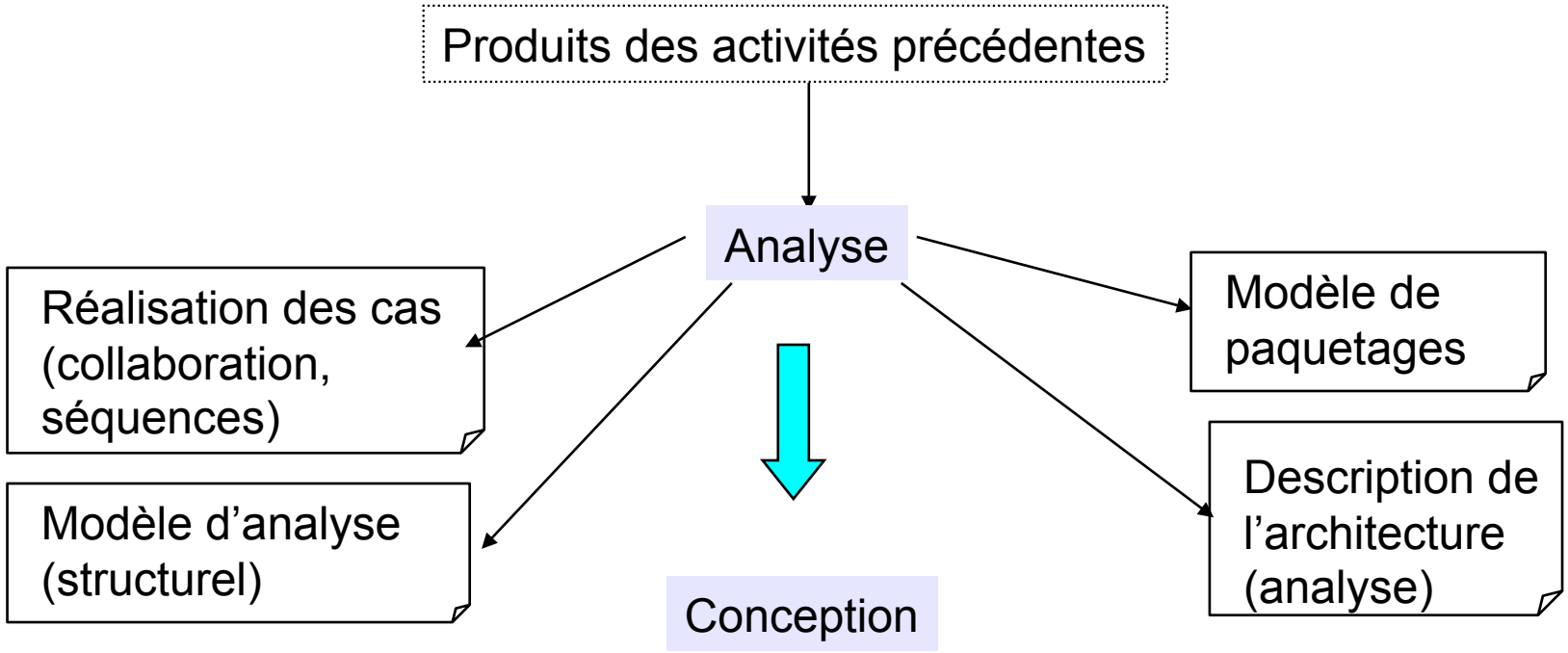
Activité *analyse*
Diagramme de classes complété



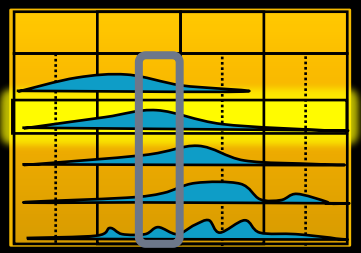


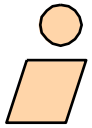
Activité *analyse*
Produits de l'analyse



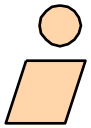


Activité *analyse*
Travailleurs

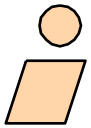




- Architecte
  - responsable de l'intégrité du modèle d'analyse
  - description de l'architecture
    - extrait du modèle d'analyse

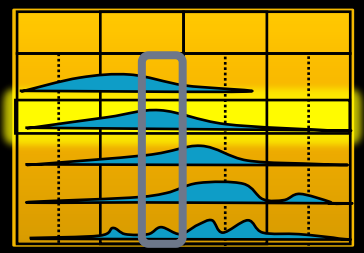


- Ingénieur des CU
  - réalisation/analyse des CU



- Ingénieur des composants
  - classes d'analyse
  - paquetages d'analyse

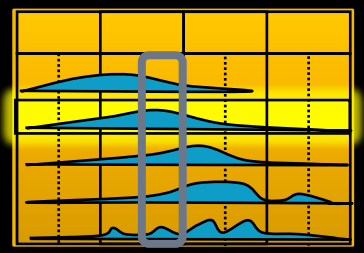
Activité *analyse*
Comparaison des modèles CU et analyse



Modèle des cas d'utilisation	Modèle d'analyse
Langage du client	Langage du développeur
Vue externe du système	Vue interne du système
Structuré par les cas (vue externe)	Structuré par les classes et paquetages (vue interne)
Utilisé comme "contrat" avec le client	Utilisé pour comprendre le système
Informel	Cohérent, non redondant...
Capture les fonctions du système et ce qui conditionne l'architecture	Esquisse la manière de réaliser les fonctions dans le système et leur répartition dans des classes

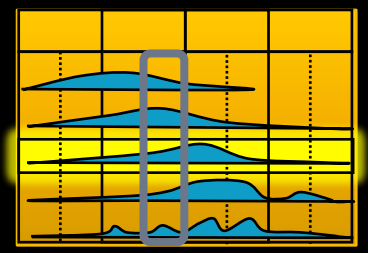
Remarque :

Différentes sortes de classes



- Classe conceptuelle
  - objets du monde réel
- Classe d'analyse
  - 3 stéréotypes de Jacobson : frontière, contrôle, entité
- Classe logicielle
  - composant logiciel du point de vue des spécifications ou de l'implémentation
  - classe de conception
- Classe d'implémentation
  - classe implémentée dans un langage OO

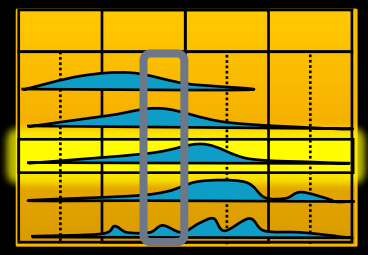
Activité *conception*
Objectifs

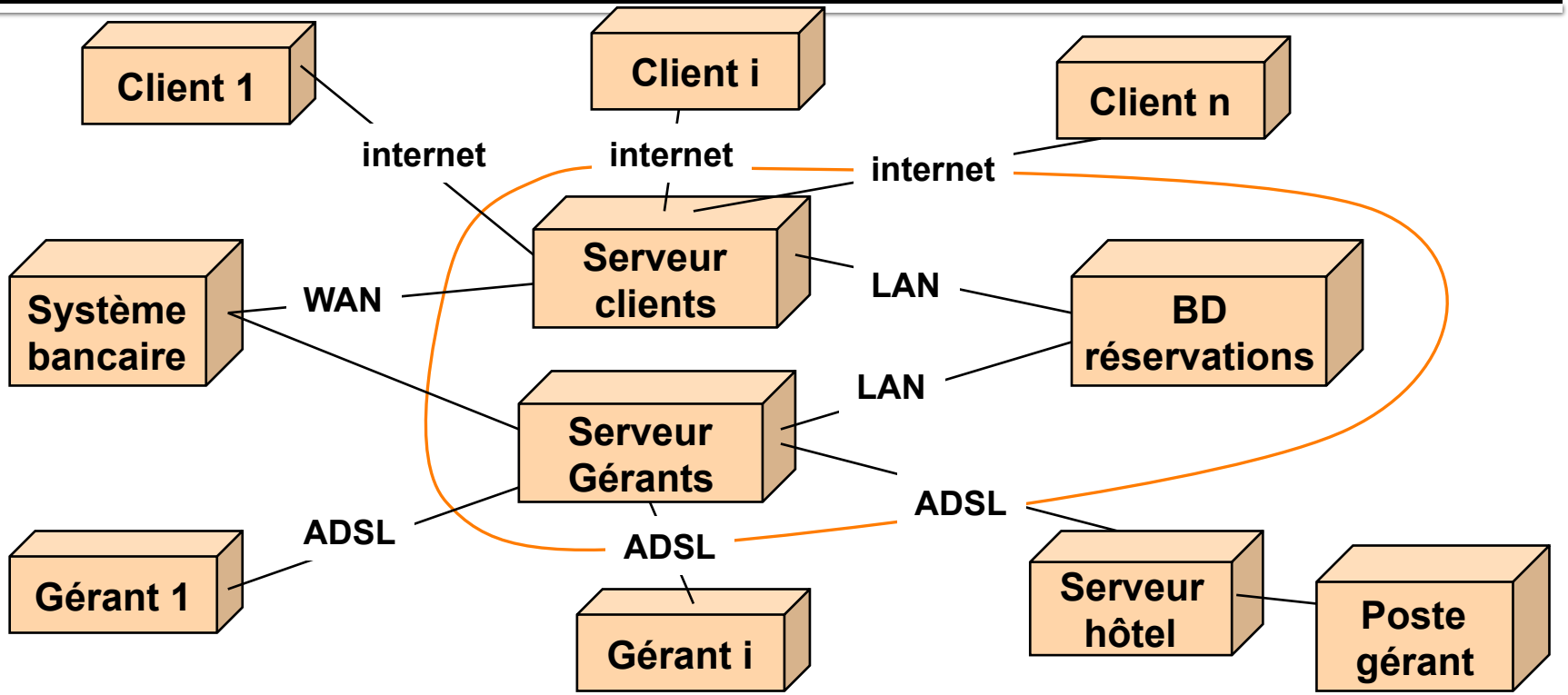


- Proposer une réalisation de l'analyse et des cas d'utilisation en prenant en compte toutes les exigences
- Mener la conception architecturale
  - identifier les nœuds et la configuration du réseau (déploiement), les composants/sous-systèmes et leurs interfaces (modèle en couche), les classes significatives de l'architecture
- 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…
  - composants de service liés à l'appli, de middleware…
  - permettra de distribuer le travail aux développeurs

Activité *conception*

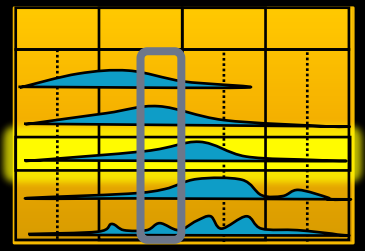
Diagramme de déploiement

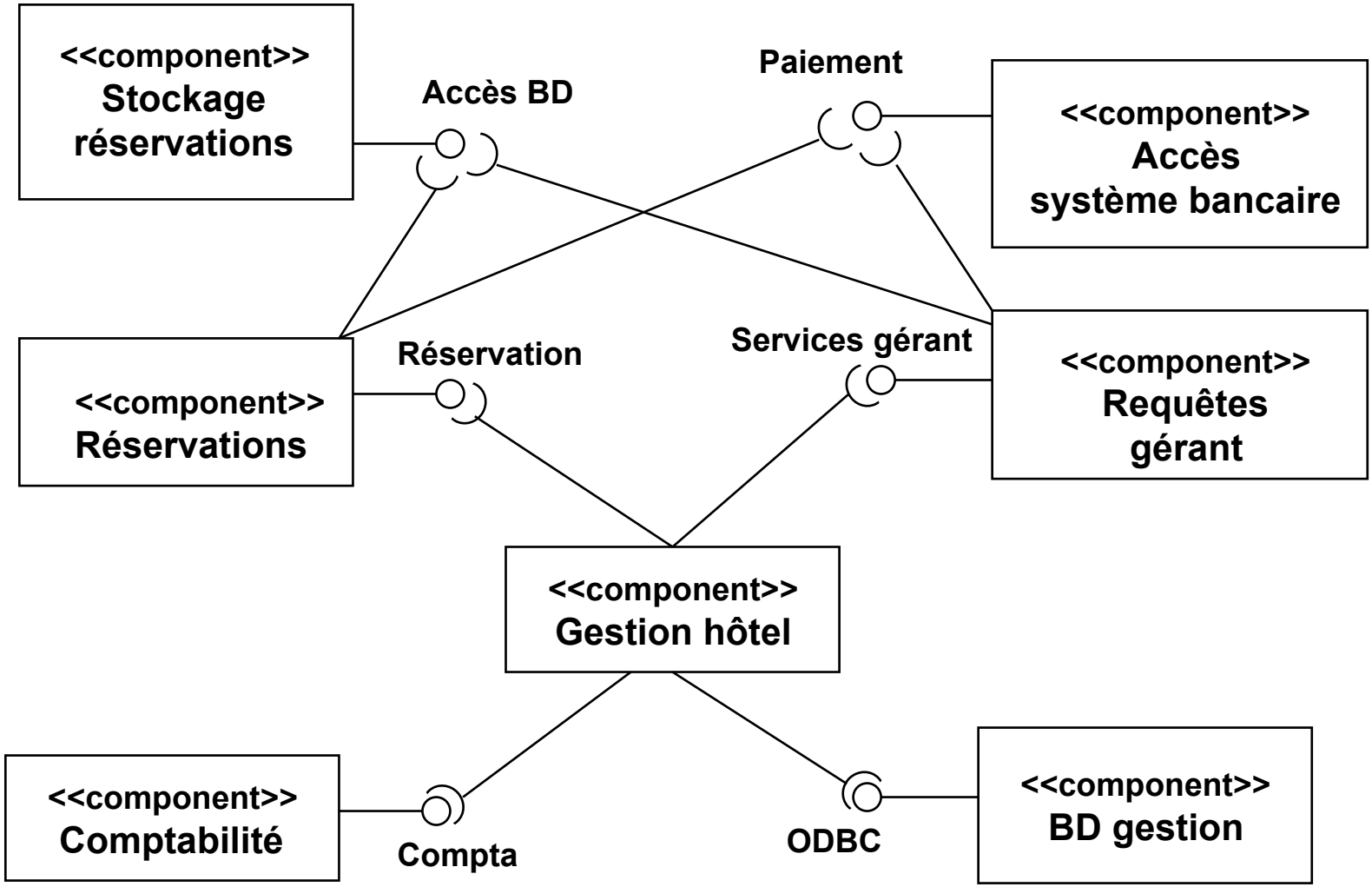




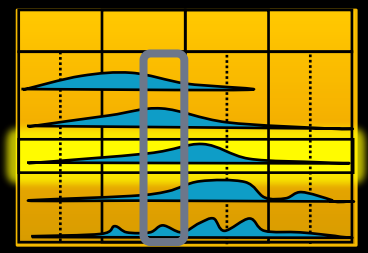
- Les deux serveurs et la BD des réservations peuvent être sur un même nœud tant que les liaisons clients ne pénalisent pas les liaisons gérants
- Les liaisons gérant-serveur sont en ADSL
- Possibilité que les hôtel aient un unique poste, ou bien un serveur local
- ...

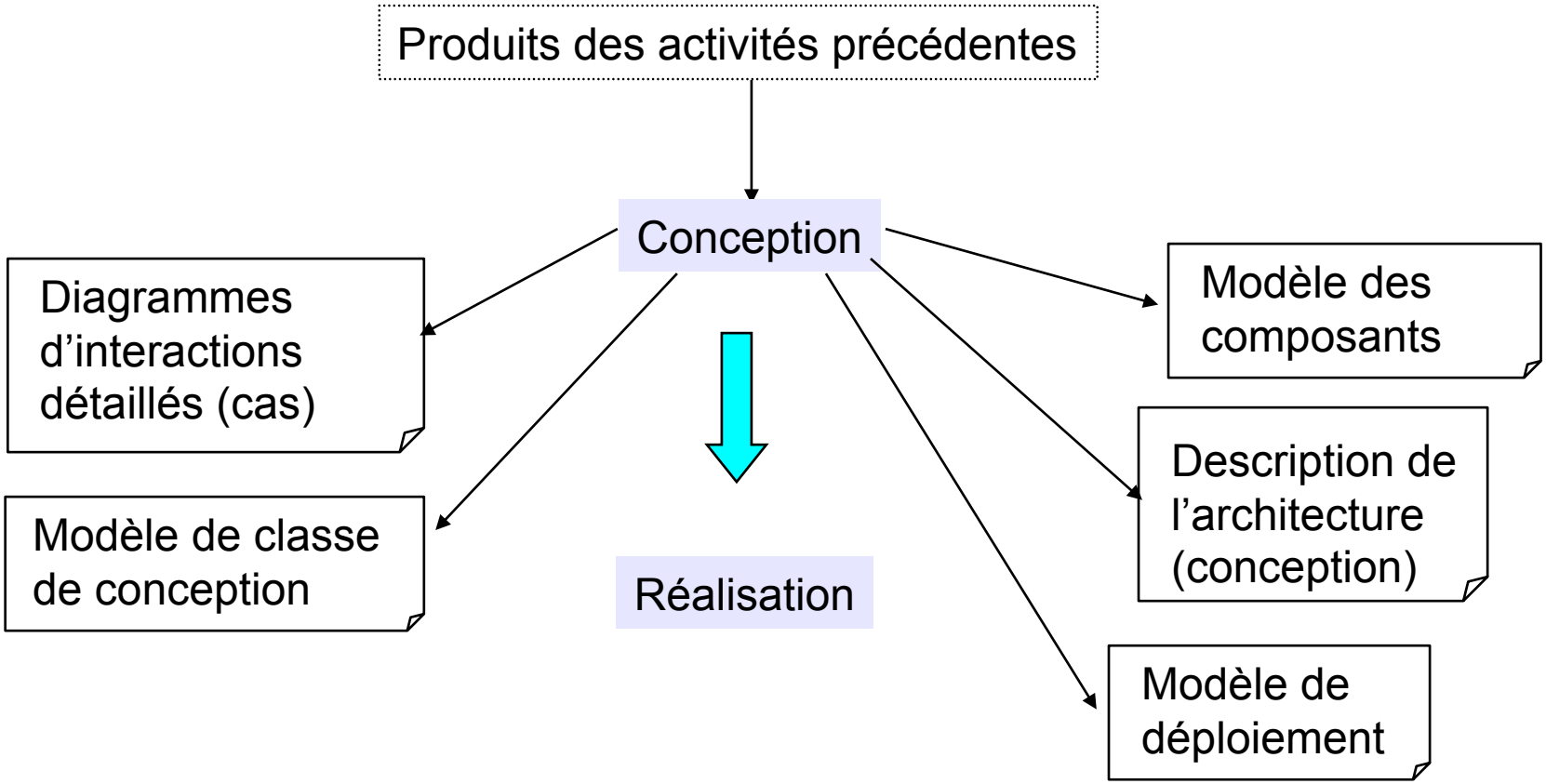
Activité *conception*
Découpage en composants



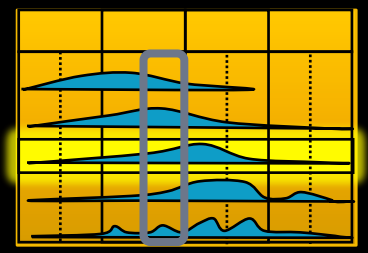


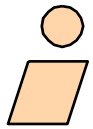
Activité *conception*
Produits de la conception



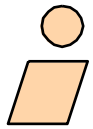


Activité *conception*
Travailleurs

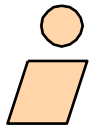




- Architecte
  - intégrité des modèles de conception et de déploiement
  - description de l'architecture



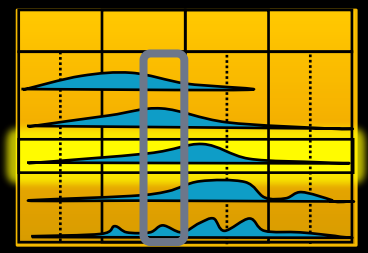
- Ingénieur de CU
  - réalisation/conception des CU



- Ingénieur de composants
  - classes de conception
  - composants/sous-systèmes de conception
  - interfaces

Activité *réalisation*

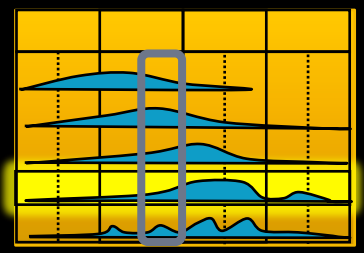
Comparaison des modèles analyse et conception



Modèle d'analyse	Modèle de conception
Modèle conceptuel, abstraction du système	Modèle physique qui sera mis en oeuvre
Générique vis à vis de la conception	Spécifique
Moins formel	Plus formel
Donne les grandes lignes de la conception, dont l'architecture. Définit une structure essentielle pour le système	Réalise cette conception. Façonne le système en essayant de conserver la structure définie
Représente 1/ 5ème du coût de la conception	
Éventuellement non maintenu durant le cycle	Nécessairement maintenu durant le cycle (UML mode plan)

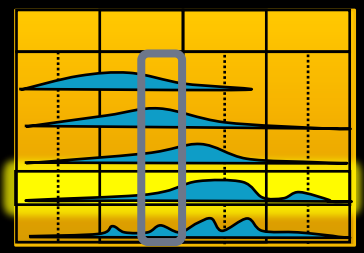
Activité *réalisation*

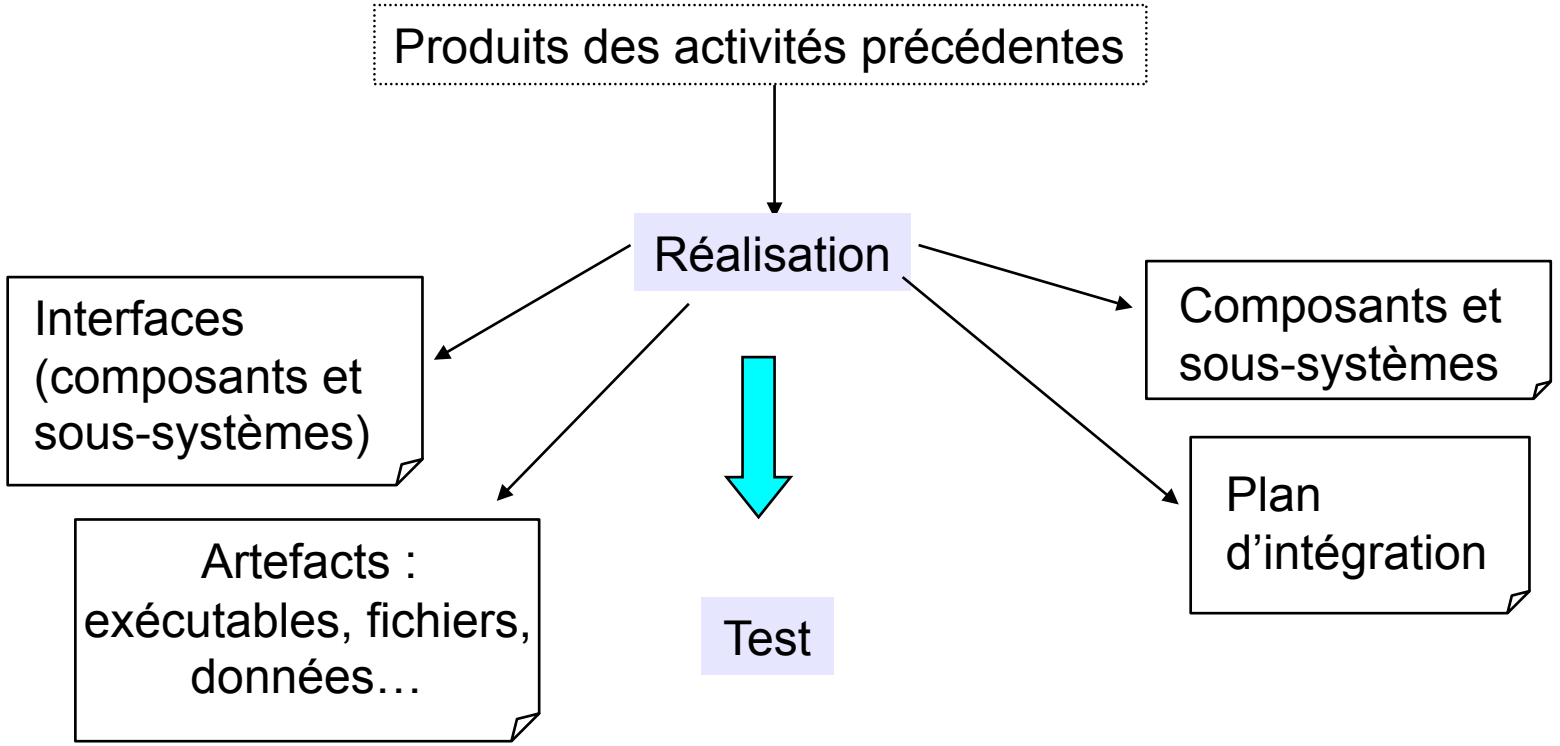
Objectifs



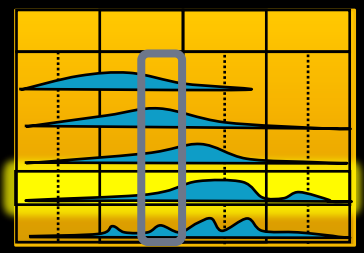
- Faire la mise en œuvre architecturale
  - identifier les artefacts logiciels et les associer à des nœuds
- Intégrer le système
  - planifier l'intégration, intégrer les incréments réalisés
- Réaliser les composants/sous-systèmes
- Réaliser les classes
- Mener les tests unitaires
  - tests de spécification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

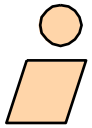
Activité *réalisation*
Produits de la réalisation



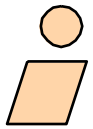


Activité *réalisation*
Travailleurs

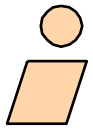




- Architecte
  - modèles d'implémentation et de déploiement
  - description de l'architecture



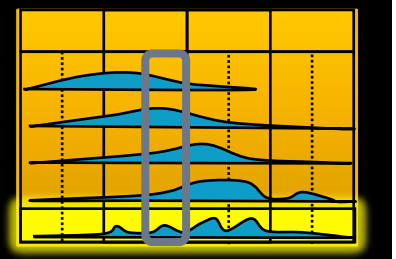
- Ingénieur de composants
  - artefacts logiciels, sous-systèmes et composants d'implémentation, interfaces



- Intégrateur système
  - plan de construction de l'intégration

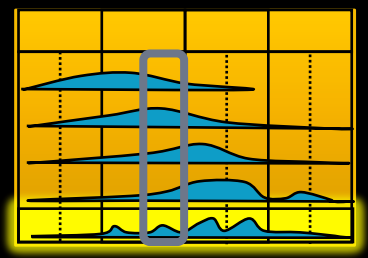
Activité *test*

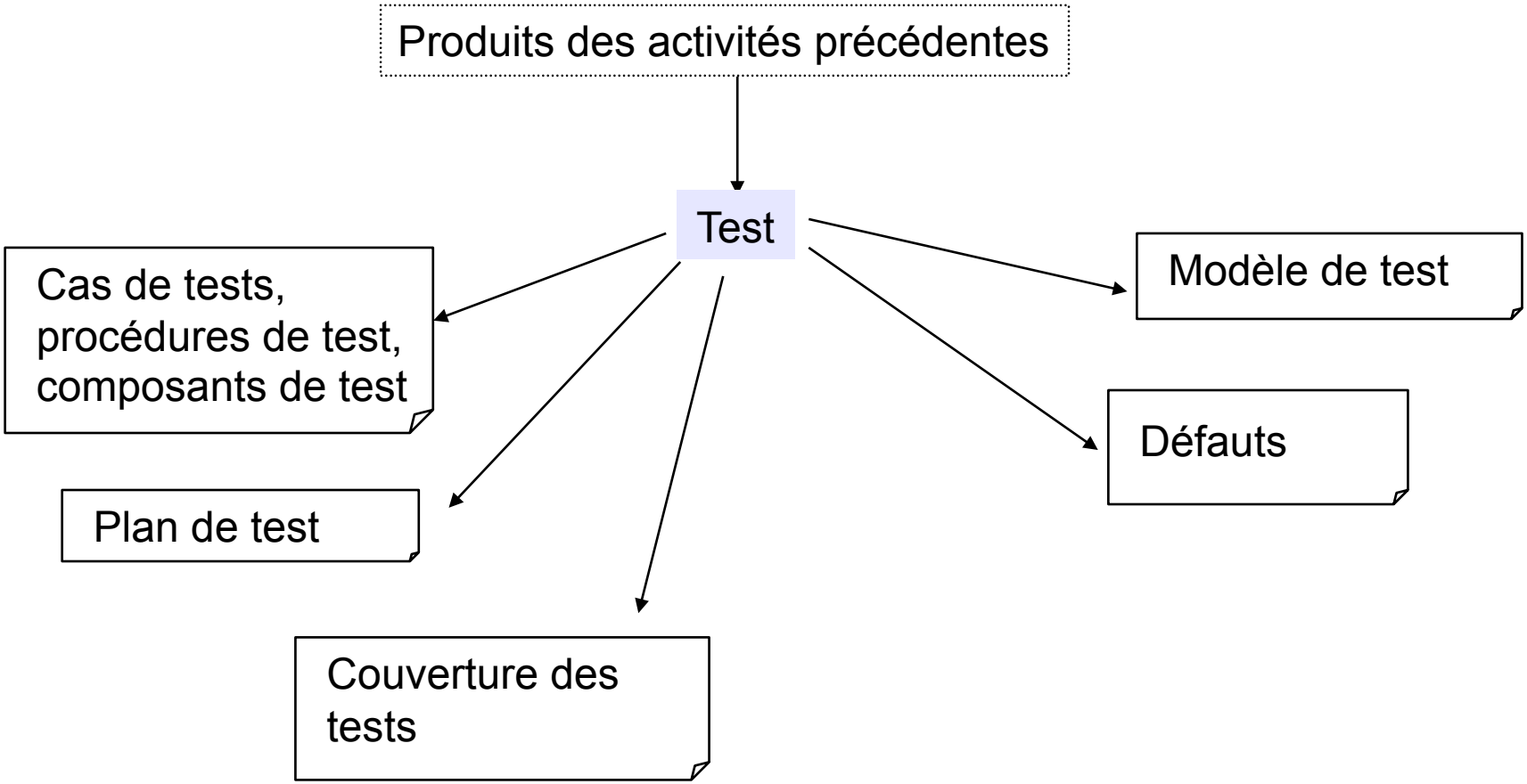
Objectifs



- Rédiger le plan de test
  - décrire la stratégie de test, estimer les besoins pour l'effort de test, planifier l'effort dans le temps
- Concevoir les tests
- Automatiser les tests
- Réaliser les tests d'intégration
- Réaliser les tests du système
- Évaluer les tests

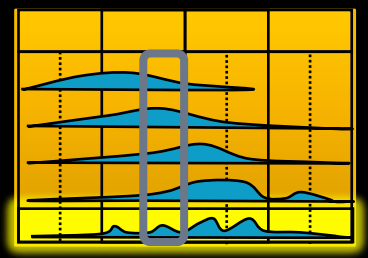
Activité *test*
Produits de l'activité de test

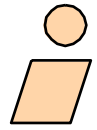




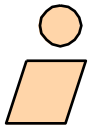
Activité *test*

Travailleurs

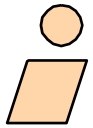




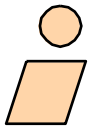
- Concepteur de tests
  - modèle de tests, cas de test, procédures de test, évaluation des tests, plan de tests



- Ingénieur de composants
  - test unitaires



- Testeur d'intégration
  - tests d'intégration



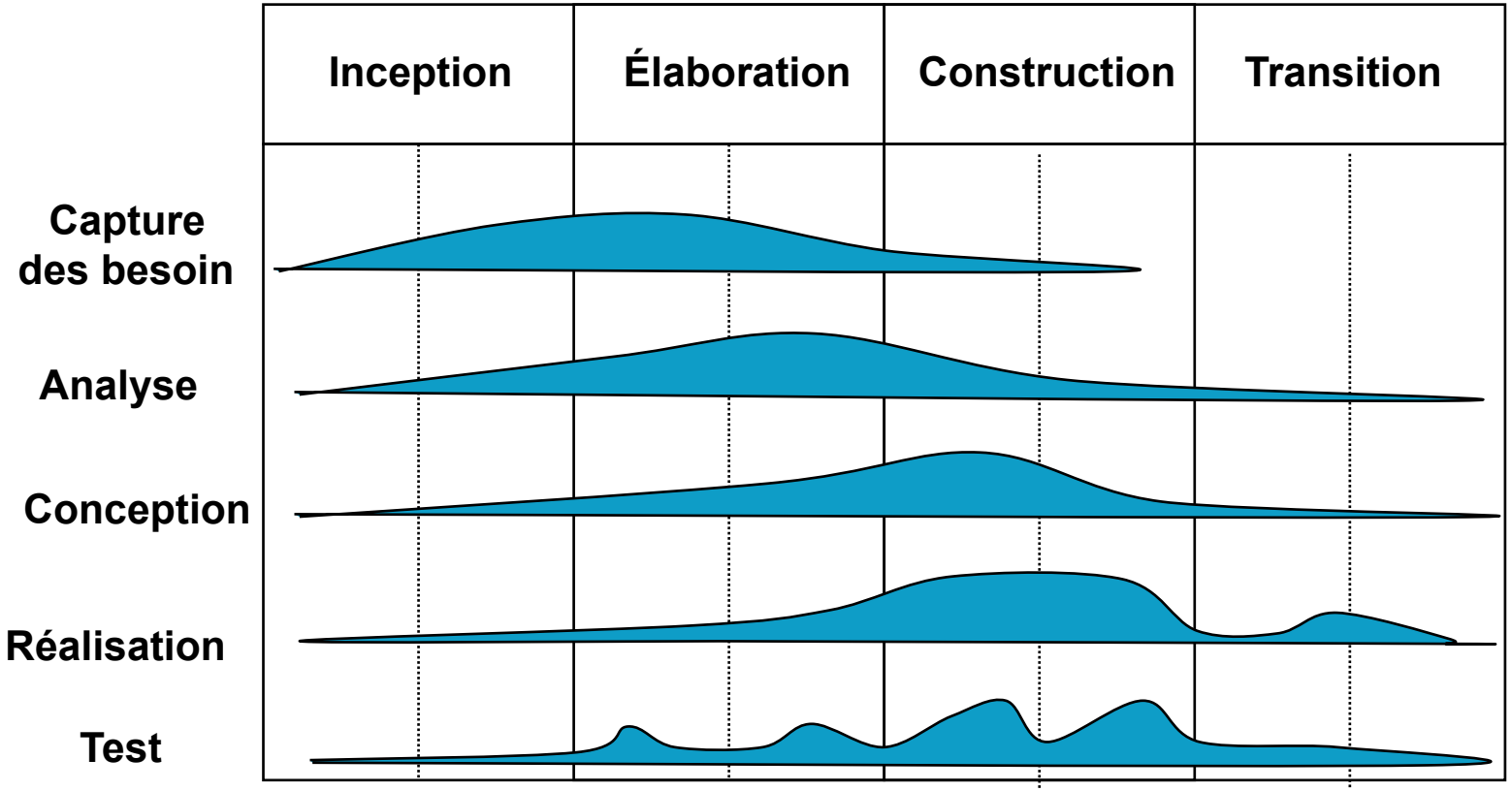
- Testeur système
  - vérification du système dans son ensemble

Plan

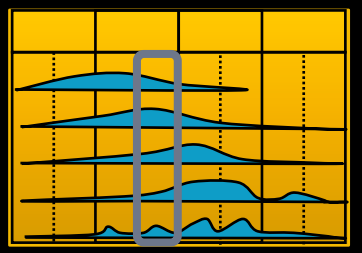
- Trame du processus unifié
- Processus unifié : caractéristiques essentielles
- **Description du processus unifié**
  - Activités, travailleurs, artefacts et modèles
  - Différentes activités pour passer des besoins au code
  - **Différentes phases pour piloter les activités**
  - Focus sur quelques points importants
- Illustration : deux déclinaisons

Généralités sur les phases

- Déroulement du projet par phases
- Chaque phase spécifie les activités à effectuer

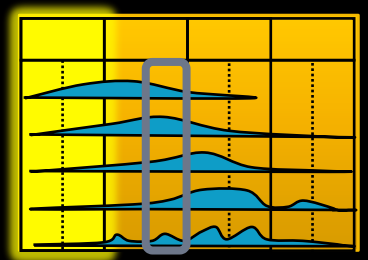


Gestion des phases

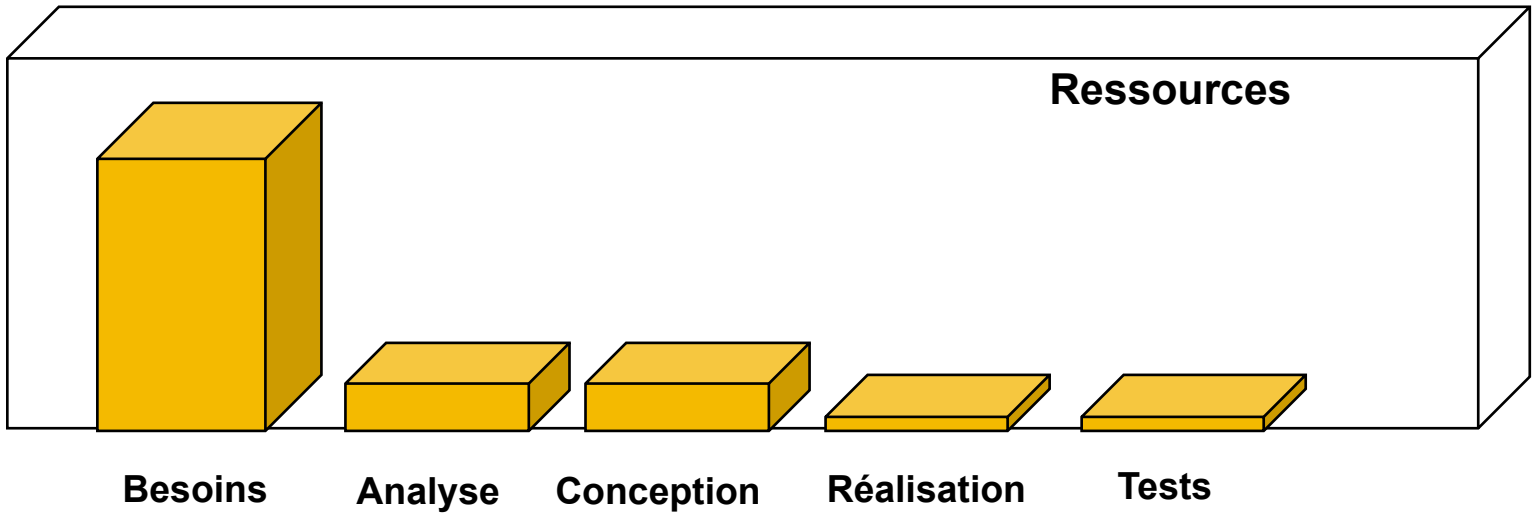


- Planifier les phases
  - allouer le temps, fixer les points de contrôle de fin de phase, les itérations par phase et le planning général du projet
- Dans chaque phase
  - planifier les itérations et leurs objectifs de manière à réduire
    - les risques spécifiques du produit
    - les risques de ne 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...)

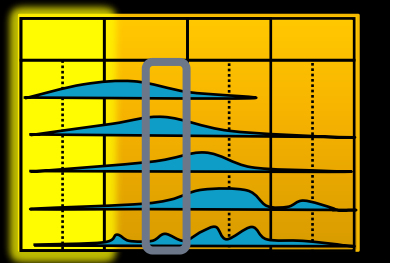
Phase d'étude préliminaire (inception)



- 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
  - à la fin de cette phase, on décide de continuer ou non
  - attention à ne pas définir tous les besoins, à vouloir des estimation fiables (coûts, durée), etc.
    - on serait dans le cadre d'une cascade

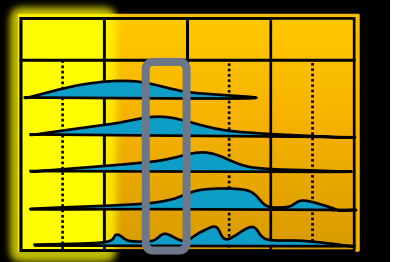


Activités principales de l'étude préliminaire



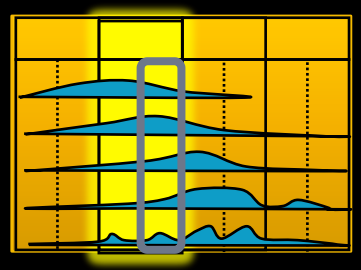
- Capture des besoins
  - comprendre le contexte du système (50-70% du contexte)
  - établir les besoins fonctionnels et non fonctionnels (80%)
  - traduire les besoins fonctionnels en cas d'utilisation (50%)
  - détailler les premiers cas par ordre de priorité (10% max)
- Analyse
  - analyse des cas 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 : sous-systèmes, nœuds, réseau, couches logicielles
  - examen des aspects importants et à plus haut risque

Livrables de l'étude préliminaire

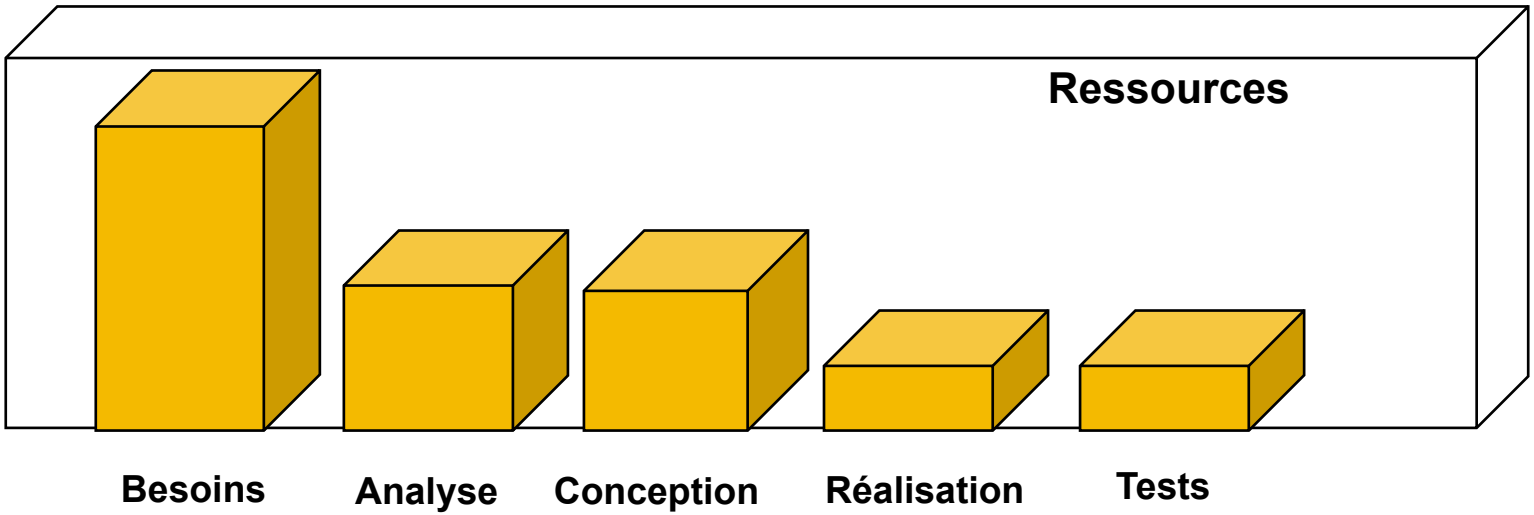


- Première version du modèle du domaine ou de contexte de l'entreprise
  - parties prenantes, utilisateurs
- Liste des besoins
  - fonctionnels et non fonctionnels
- Ébauche des modèles de cas, d'analyse et de conception
- Esquisse d'une architecture
- Liste ordonnée de risques et liste ordonnée de cas
- Grandes lignes d'un planning pour un projet complet
- Première évaluation du projet, estimation grossière des coûts
- Glossaire

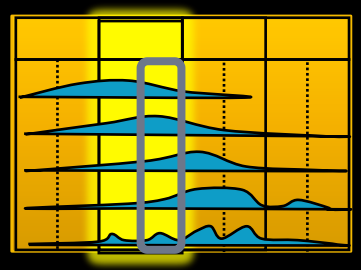
Phase d'élaboration



- 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

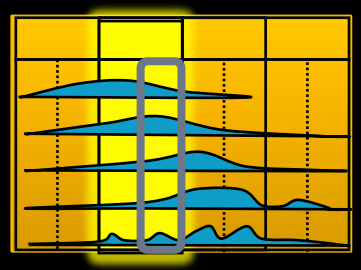


Activités principales de l'élaboration



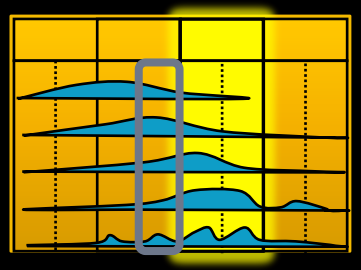
- Capture des besoins
  - terminer la capture des besoins et en détailler de 40 à 80%
  - faire un prototype de l'interface utilisateur (éventuellement)
- Analyse
  - analyse architecturale complète (paquetages…)
  - raffinement des cas d'utilisation (pour l'architecture, < 10%)
- Conception
  - terminer la conception architecturale
  - effectuer la conception correspondant aux cas sélectionnés
- Réalisation
  - limitée au squelette de l'architecture
  - faire en sorte de pouvoir éprouver les choix
- Test
  - du squelette réalisé

Livrables de l'élaboration

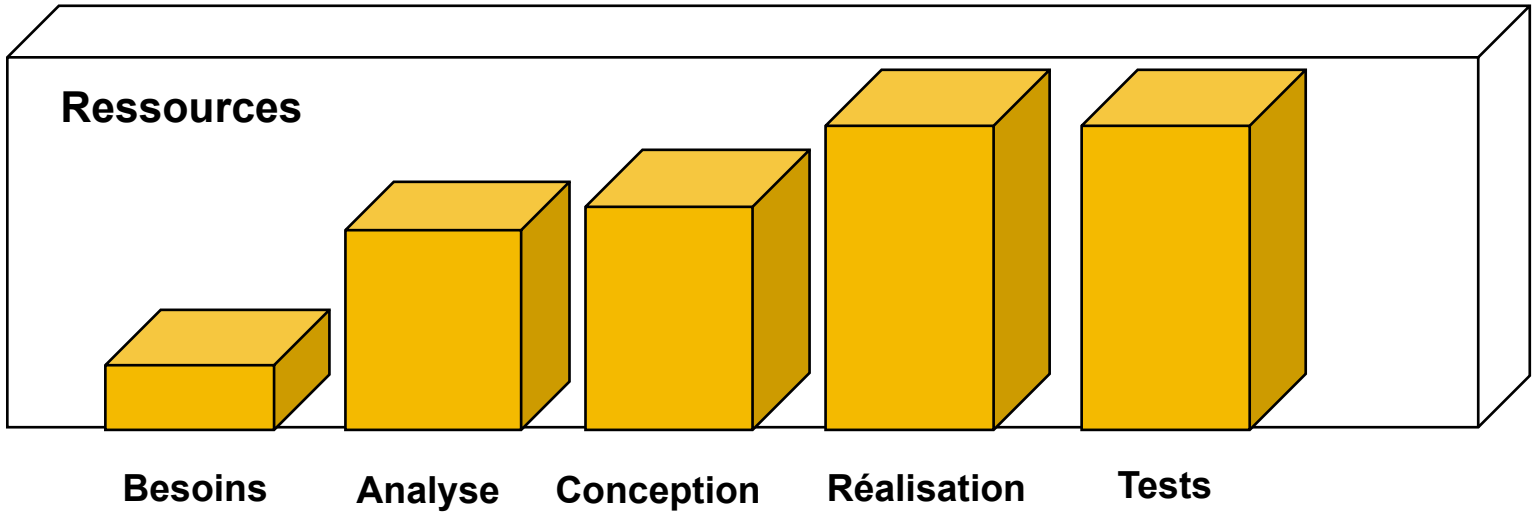


- Un modèle de l'entreprise ou du domaine complet
- Une version des modèles : cas, analyse et conception, (<10%), déploiement, implémentation (<10%)
- Une architecture de base exécutable
- La description de l'architecture (extrait des autres modèles)
  - document d'architecture logicielle
- Une liste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

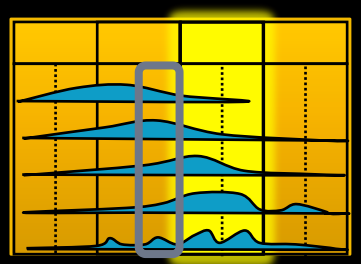
Phase de construction



- Objectif : Réaliser une version beta

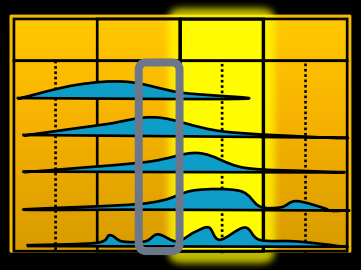


Activités principales de la construction



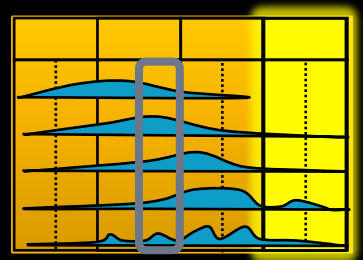
- Capture des besoins
  - spécifier l'interface utilisateur
- Analyse
  - terminer l'analyse de tous les cas d'utilisation, la construction du modèle structurel d'analyse, le découpage en paquetages…
- Conception
  - l'architecture est fixée et il faut concevoir les sous-systèmes dans l'ordre de priorité (itérations de 30 à 90j, max. 9 mois)
  - concevoir les cas puis les classes
- Réalisation
  - réaliser, faire des tests unitaires, intégrer les incréments
- Test
  - toutes les activités de test : plan, conception, évaluation…

Livrables de la construction

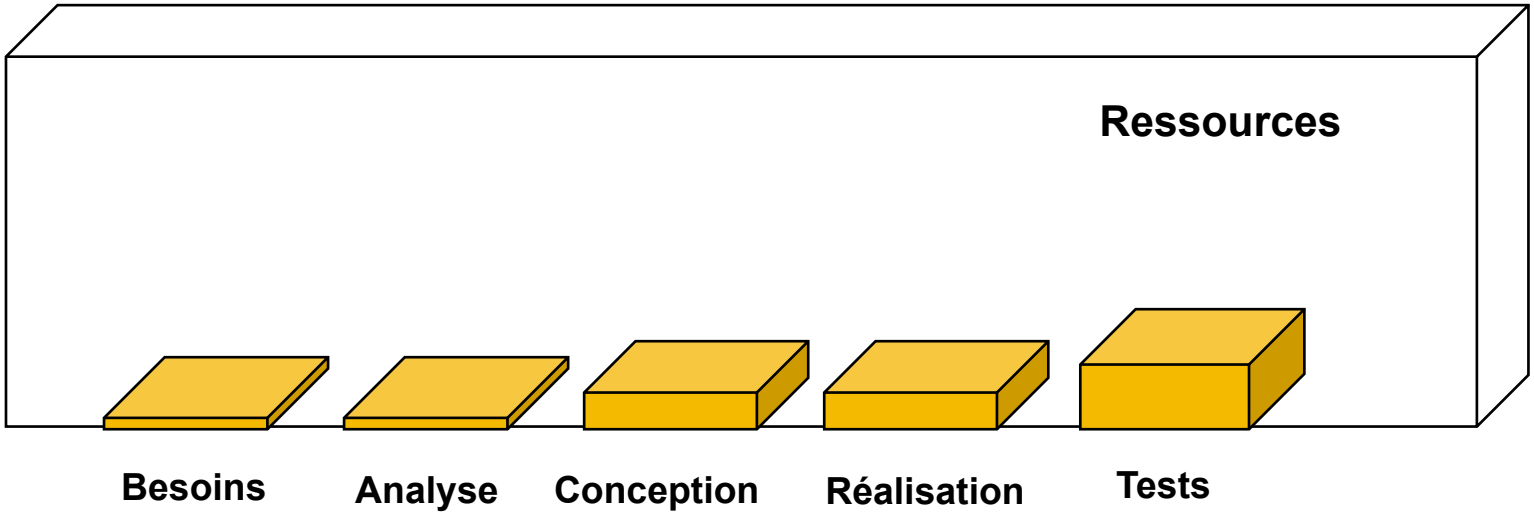


- Un plan du projet pour la phase de transition
- L'exécutable
- Tous les documents et les modèles du système
- Une description à jour de l'architecture
- Un manuel utilisateur suffisamment détaillé pour les tests

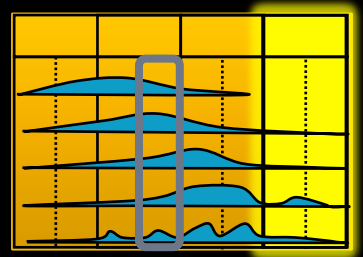
Phase de transition



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

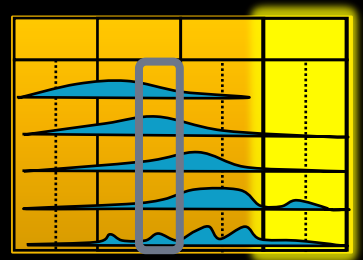


Activités principales de la phase de transition (déploiement)



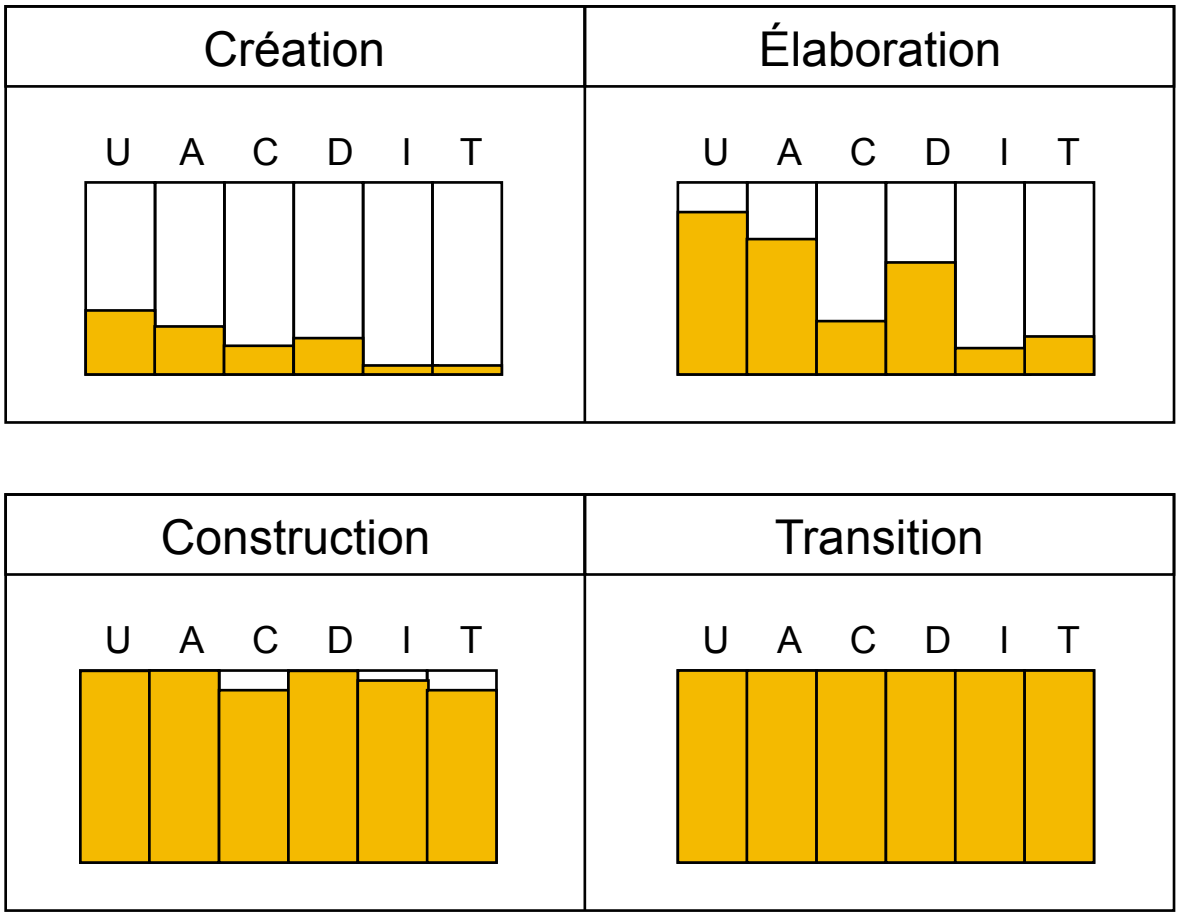
- Préparer la version beta à tester
- Installer la version sur le site, convertir et faire migrer les données nécessaires…
- Gérer le retour des sites
- Le système fait- il ce qui était attendu ? Erreurs découvertes ?
- Adapter le produit corrigé aux contextes utilisateurs (installation…)
- Terminer 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'exécutable 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…)

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

Plan

- Trame du processus unifié
- Processus unifié : caractéristiques essentielles
- Description du processus unifié
  - Activités, travailleurs, artefacts et modèles
  - Différentes activités pour passer des besoins au code
  - Différentes phases pour piloter les activités
  - **Focus sur quelques points importants**
    - **Analyse architecturale**
    - **Planification des itérations**
    - **Les tentations de la cascade**
    - **Personnes, environnement et outils**
- Illustration : deux déclinaisons

Description de l'architecture (1)

- L'architecture doit être une vision partagée
  - sur un système très complexe
  - pour guider le développement
  - tout en restant compréhensible
- Il faut donc mettre en place une description (ou documentation) explicite de l'architecture
  - qui servira de référence jusqu'à la fin du cycle (et après)
  - qui doit rester aussi stable que l'architecture de référence

Description de l'architecture (2)

(Larman, 2004)

- Analyse architecturale
  - identifier et traiter les besoins non fonctionnels dans le contexte des besoins fonctionnels
  - identifier les points de variation et d'évolution les plus probables
    - points de variation : variations dans le système existant ou dans les besoins
    - points d'évolution : points de variation spéculatifs, actuellement absents des besoins
- Comment décrire l'architecture ?
  - décrire les facteurs qui influencent l'architecture
    - **facteurs architecturaux**
  - décrire les choix qui ont été faits
    - **mémos techniques**
  - décrire l'architecture
    - **document d'architecture du logiciel**

Description de l'architecture : Facteurs architecturaux

(Larman, 2004)

- Facteurs qui ont une influence significative sur l'architecture
  - fonctionnalités, performance, fiabilité, facilité de maintenance, implémentation et interface, etc.
- Un facteur architectural doit être identifié et décrit dans une fiche

- Nom
  ex. : « fiabilité, possibilité de récupération »
- Mesures et scénarios de qualité
  ce qu'il doit se passer et comment le vérifier
  ex. : « si problème, récupération dans la minute »
- Variabilité
  souplesse actuelle et évolutions futures
  ex. : «pour l'instant service simplifiés acceptables en cas de rupture, évolution : services complets »
- Impacts
  pour les parties prenantes, l'architecture…
  ex. : « fort impact, rupture de service non acceptable »
- Priorité
  ex. : élevée
- Difficulté ou risque
  ex. : moyen

Description de l'architecture
Mémos techniques

(Larman, 2004)

- Les choix architecturaux doivent prendre en compte les facteurs architecturaux
- Il est important de décrire les solutions choisies et leur motivation
  - assurer la traçabilité des décisions architecturales
    - raisons des choix
    - alternatives étudiées, *etc.*
- Les « mémos techniques » décrivent les choix architecturaux
  - texte + diagrammes

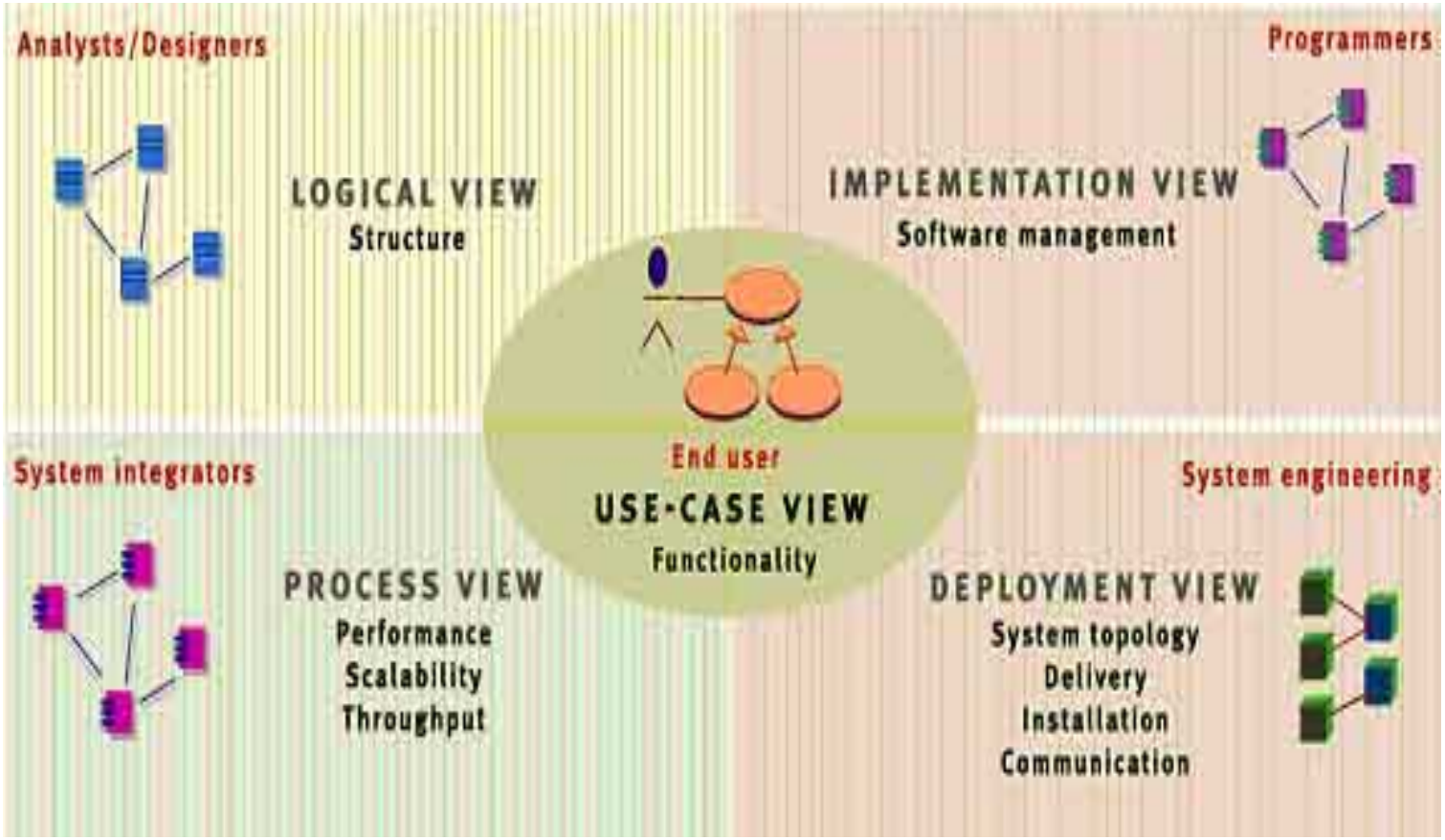
Mémo technique :
• Nom du problème
• Résumé de la solution
• Facteurs architecturaux concernés
• Solution
• Motivation
• Problèmes non résolus
• Autres solutions envisagées

Description de l'architecture
Document d'architecture du logiciel

(Larman, 2004)

- Vue architecturale
  - vue de l'architecture du système depuis un point de vue particulier
    - texte + diagrammes
  - se concentre sur les informations essentielles et documente la motivation
    - « ce que vous diriez en 1 minute dans un ascenseur à un collègue »
  - description *a posteriori*
- DAL : récapitulatif des décisions architecturales
  - introduction
  - facteurs architecturaux
  - mémos techniques
  - ensemble de vues architecturales
- Modèle N+1 vues : permettent aux différents intervenants de se concentrer sur les problèmes de l'architecture du système qui les concernent le plus
  - logique, processus, déploiement, données, sécurité, implémentation, développement, etc.
  - + la vue des cas d'utilisation pour fédérer les autres vues

Les 4+1 vues de Philip Kruchten



La description de l'architecture comme 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
- Besoins significatifs sur le plan architectural non décrits par les CU, exigences non fonctionnelles (ex. sécurité)
- Description de la plateforme, des systèmes, des middleware utilisés
- Description des frameworks avec mécanismes génériques (qui pourront être réutilisés)
- Patterns architecturaux utilisés

Architecture : en résumé

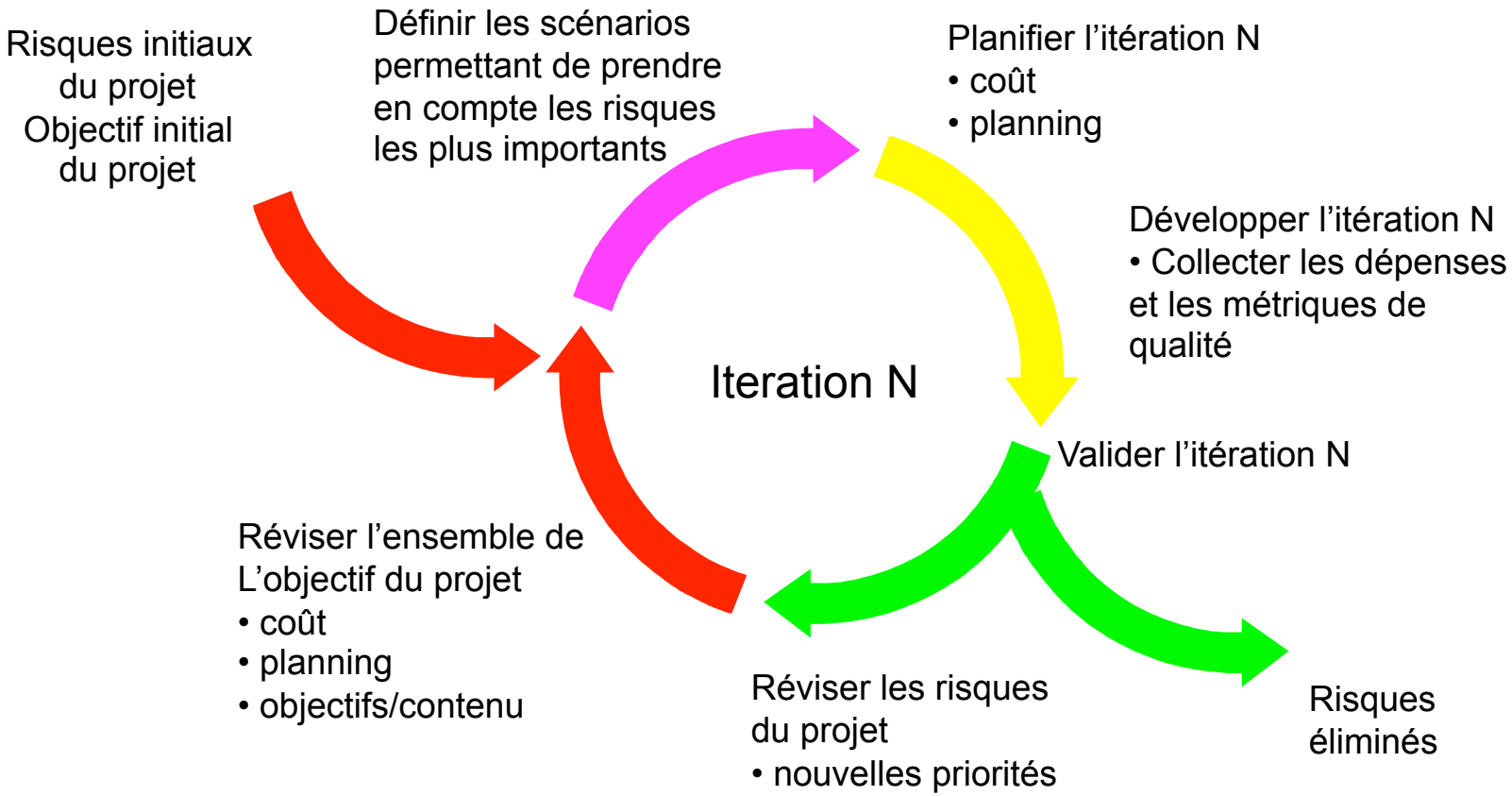
(USDP, 2000)

- Qu'est ce que l'architecture ?
  - C'est ce que l'architecte spécifie dans une description d'architecture. La description de l'architecture laisse à l'architecte la maîtrise technique du développement du système. L'architecture logicielle s'intéresse à la fois aux éléments structuraux significatifs du système, tels que les sous-systèmes, les classes, les composants et les nœuds, et aux collaborations se produisant entre ces éléments par l'intermédiaire des interfaces.
  - Les cas d'utilisation orientent l'architecture de telle sorte que le système offre les usages et les fonctionnalités désirés tout en satisfaisant des objectifs de performance. Outre son exhaustivité, l'architecture doit montrer assez de souplesse pour accueillir de nouvelles fonctions et permettre la réutilisation de logiciels existants.
- Comment l'obtient-on ?
  - L'architecture est développée de façon itérative au cours de la phase d'élaboration au travers [des différentes activités]. Les CU signifiants sur le plan de l'architecture, ainsi que certaines entrées d'une autre type, permettent d'implémenter l'architecture de référence (« squelette ») du système. Cet ensemble d'entrées supplémentaires comprend les besoins logiciels du système, les middleware, les systèmes existants à réutiliser, les besoins non fonctionnels…
- Comment la décrit-on ?
  - La description de l'architecture est une vue des modèles du système […]. Elle décrit les parties du système qu'il est important, pour les développeurs et les autres intervenants, de comprendre.

Planification des itérations

- Planification des itérations dès l'élaboration
  - précise pour l'itération suivante, imprécise pour la fin
  - la planification générale est adaptée en fonction des risques identifiés/traités et des résultats obtenus dans les itérations
- Planification adaptative (vs. prédictive)
  - fixer des butées temporelles
  - gérer l'adaptation avec le client
- Itération
  - toutes les activités des besoins aux tests, résultats différents en fonction de la phase dans laquelle on se trouve
  - mini-projet : planning, ressources, revue
  - premières itérations : suivre une méthode
  - à la fin d'une itération : retrospective
    - éléments à conserver, problèmes, éléments à essayer

Itération pilotées par la réduction des risques



Les tentations de la cascade

- Les « principes cascade » ne doivent pas envahir le processus
  - on ne peut pas tout modéliser avant de réaliser
- Exemples de situations problématiques
  - « nous avons une itération d'analyse suivie de deux itérations de conception »
  - « le code de l'itération est très bogué, mais nous corrigerons tout à la fin »
  - « la phase d'élaboration sera bientôt finie : il ne reste que quelques cas d'utilisation à détailler »

Attention aux personnes

- Le processus a un impact sur les personnes
  - changements de rôles
  - on passe des « ressources » aux « travailleurs »
- Mérites UP par rapport aux personnes
  - favorise le travail d'équipe
    - chaque membre comprend son rôle
    - les développeurs appréhendent mieux l'activité des autres développeurs
    - les dirigeants comprennent mieux
      - diagrammes d'architecture
  - si tout le monde le connaît
    - meilleure productivité de l'entreprise (passage d'un projet à l'autre, formation)
- Nécessités de la communication
  - les artefacts soutiennent la communication dans le projet
  - il faut également des moyens de communications

Personnes, environnement et outils

- Un mixte de facteurs à prendre en compte dans la gestion du projet
  - équilibre entre outils et processus (gérer leur influence réciproque)
  - nécessaire gestion de tous les artefacts et de la communication
    - site web du projet, wiki, gestion de versions, etc.
  - organisation physiques des personnes et artefacts physiques
    - tableaux, outils de projection et de capture, pièces et circulation, etc.

Plan

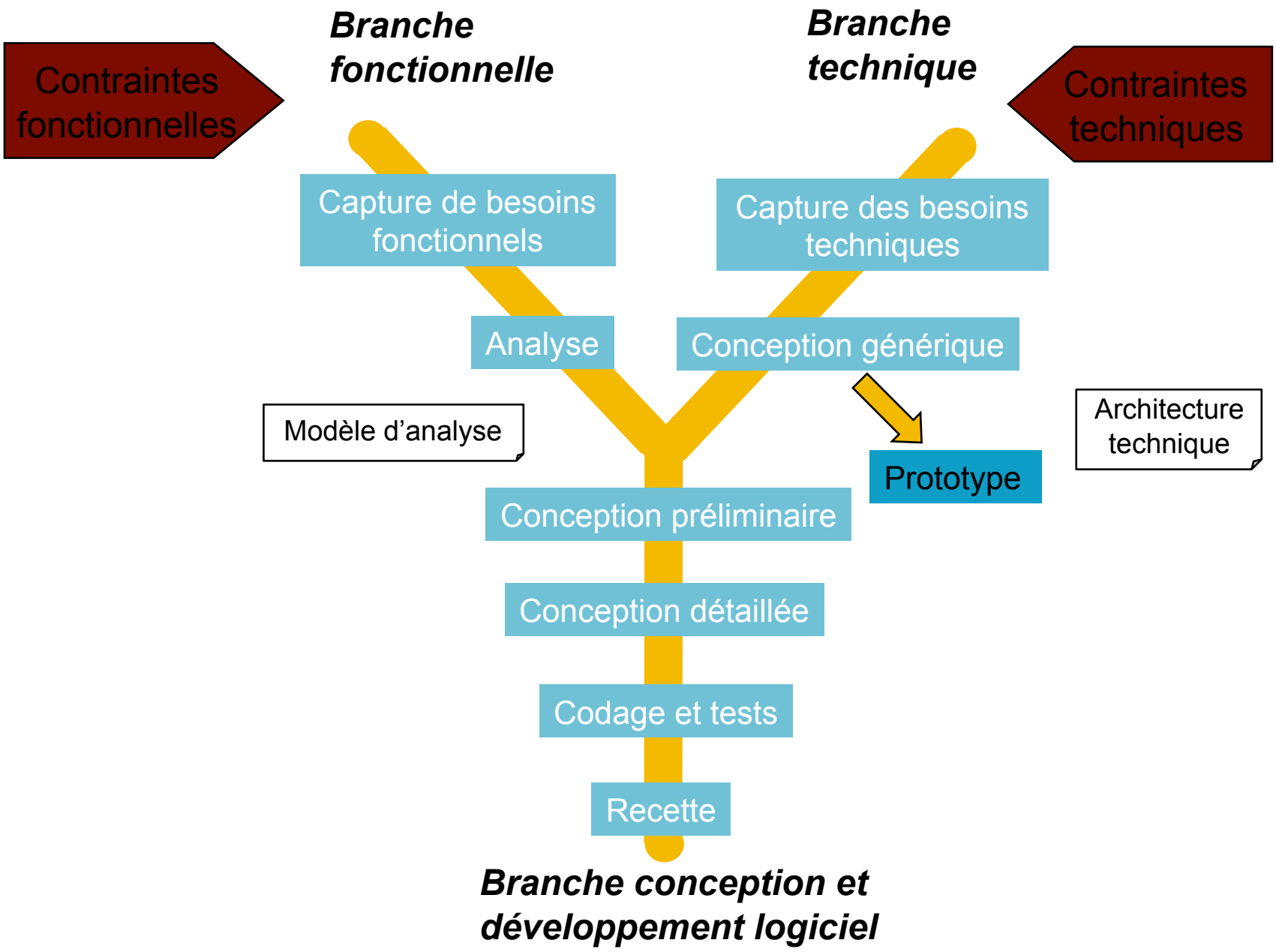
- Trame du processus unifié
- Processus unifié : caractéristiques essentielles
- Description du processus unifié
- **Illustration : deux déclinaisons**

Two Tracks Unified Process

- Processus proposé par Valtech (consulting), présenté dans
  - Pascal Roques, Franck Vallée (2004) UML2 en action (3ème édition), Eyrolles, 386 pp.
- Objectif
  - prendre en compte les contraintes de changement continuel imposées aux systèmes d'information des organisations
- Création d'un nouveau SI
  - deux grandes sortes de risques
    - imprécision fonctionnelle : inadéquation aux besoins
    - incapacité à intégrer les technologies : inadaptation technique
- Evolution du SI
  - deux grandes sortes d'évolutions
    - évolution fonctionnelle
    - évolution technique

Two Tracks Unified Process

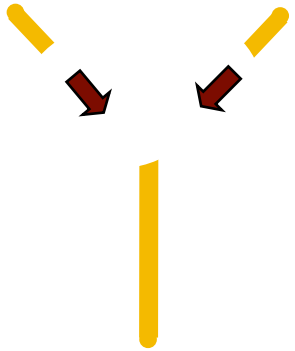
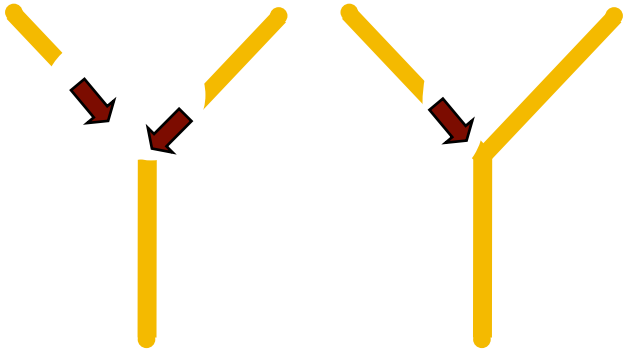
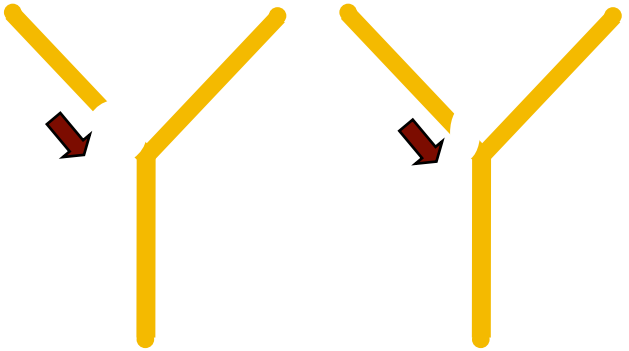
- Principe général
  - toute évolution imposée au système d'information peut se décomposer et se traiter parallèlement, suivant un axe fonctionnel et un axe technique
- TTUP
  - processus unifié (itératif, centré sur l'architecture et piloté par les CU)
  - deux branches
    - besoins techniques : réalisation d'une architecture technique
    - besoins fonctionnels : modèle fonctionnel
  - réalisation du système
    - fusionner les résultats des deux branches du processus


Contraintes fonctionnelles
Branche fonctionnelle
Branche technique
Contraintes techniques
Capture de besoins fonctionnels
Capture des besoins techniques
Analyse
Conception générique
Modèle d'analyse
Prototype
Architecture technique
Conception préliminaire
Conception détaillée
Codage et tests
Recette
Branche conception et développement logiciel

TTUP : commentaires

- Côté fonctionnel
  - relativement classique, indépendant des technologies
  - modèle d'analyse réutilisable
- Côté technique
  - insistance sur l'architecture technique indépendamment des besoins fonctionnels
    - c'est un problème de conception en soi
      - notion de CU techniques
    - dépend de l'existant
    - architecture à base de composant
  - architecture technique réutilisable
- Branche commune
  - conception préliminaire : le plus délicat

Itérations et TTUP

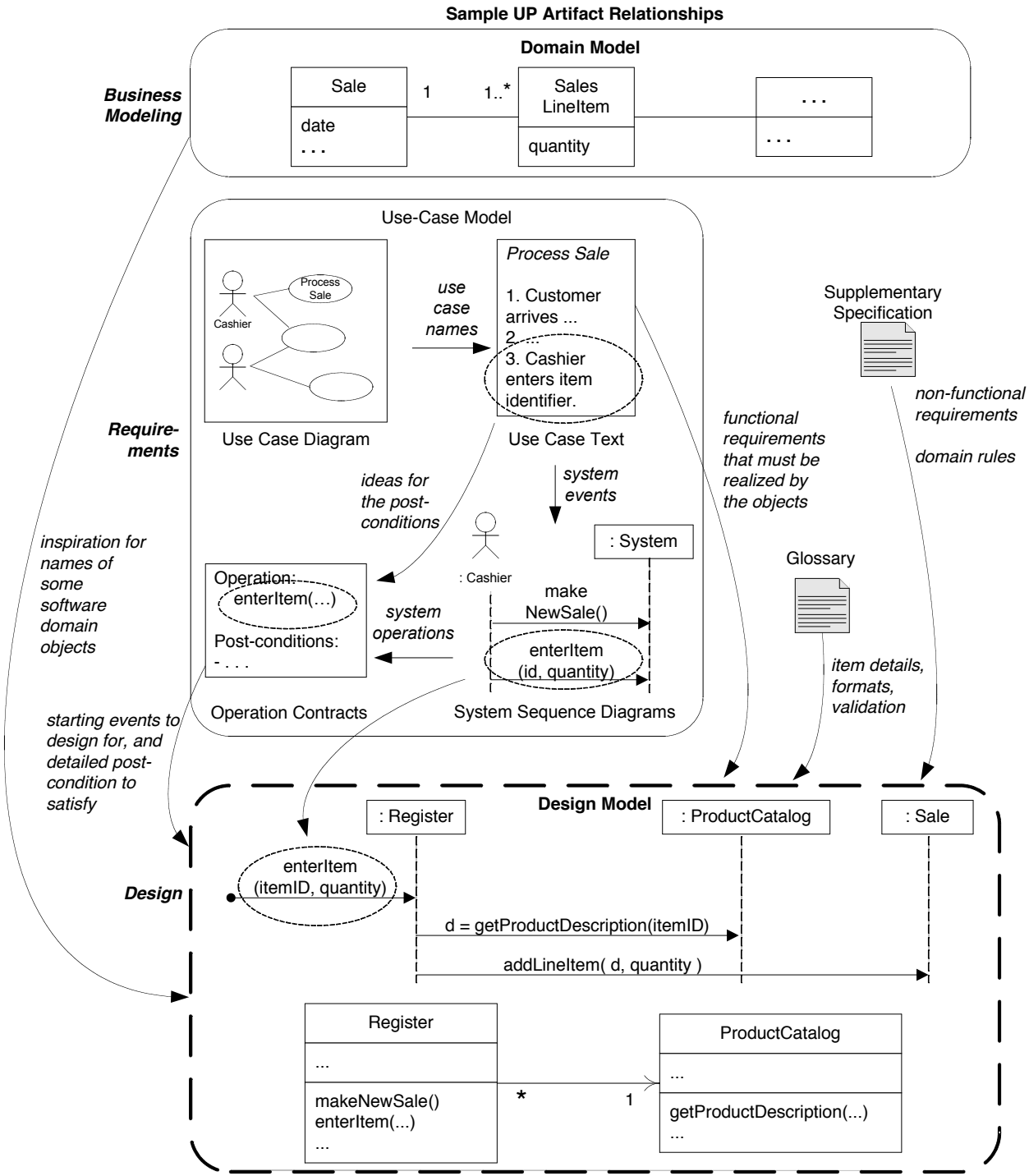
Inception	Elaboration		Construction		
					
Incrément 1	Incr.2	Incr.3	Incr.4	Incr.5	…

UP Agile (Larman)

- Proposé par Craig Larman, présenté dans
  - Craig Larman (2005) *UML 2 et les Design Patterns* (3e édition), Pearson Education, 655 p.
- Parce que UP est souvent considéré comme complexe, formel et lourd
  - dans sa description générale,
    - essaye de prendre en compte toutes les options possibles, pour toutes les entreprises, et toutes les tailles de projets
      - nombreux artefacts, activités, workflows
    - doit être adapté à chaque projet
      - ce dont tout le monde ne se rend pas forcément compte
  - dans son utilisation
    - application « à l'ancienne »
      - suivi strict des activités : rigidité
    - tendance à la cascade
      - les mauvaises habitudes sont difficiles à perdre

UP agile = les bonnes pratiques de UP

- Itérations courtes (3 semaines max)
- Mode organisationnel léger
  - petit ensemble d'activités et d'artefacts
- Fusion analyse / conception
- Utilisation de UML pour comprendre et concevoir
  - plus que pour générer du code
- Planification adaptative
- Bref, application de UP dans l' « esprit Agile »
  - vs. esprit cascade
  - en considérant que UP est agile naturellement dans sa conception (et pour ses concepteurs), mais ne l'est pas dans ses applications
- Transparents suivants
  - liens entre artefacts dans UP agile
  - mise en œuvre physique des réunions


Sample UP Artifact Relationships
Domain Model
Business Modeling
Sale
date
. . .
1
1..*
Sales LineItem
quantity
. . .
. . .
Use-Case Model
Requirements
Process Sale
Cashier
use case names
Process Sale
1. Customer arrives ...
2. ...
3. Cashier enters item identifier.
Use Case Diagram
Use Case Text
Supplementary Specification
non-functional requirements
domain rules
functional requirements that must be realized by the objects
ideas for the post-conditions
system events
: System
: Cashier
Glossary
inspiration for names of some software domain objects
Operation:
enterItem(...)
Post-conditions:
- . . .
system operations
make NewSale()
enterItem (id, quantity)
item details, formats, validation
Operation Contracts
System Sequence Diagrams
starting events to design for, and detailed post-condition to satisfy
Design Model
Design
: Register
: ProductCatalog
: Sale
enterItem (itemID, quantity)
d = getProductDescription(itemID)
addLineItem(d, quantity)
Register
...
makeNewSale()
enterItem(...)
...
*
1
ProductCatalog
...
getProductDescription(...)
...

Organisation physique : conception de CU

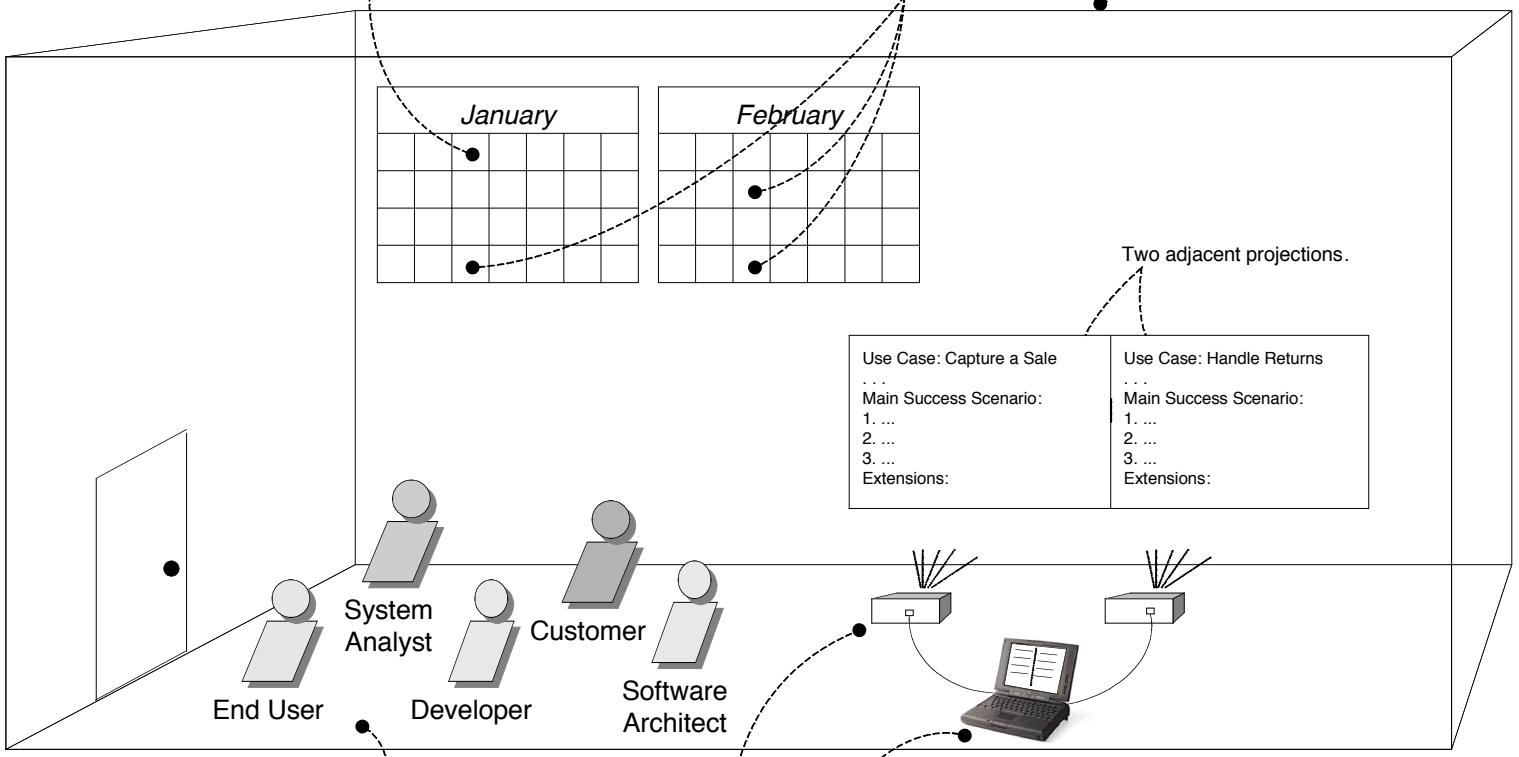
When

Once during inception. Short; do not try to define or polish all requirements.

Several times during elaboration iterations.

Where

At a requirements workshop.



Who

Many, including *end users* and *developers,* will play the role of *requirements specifier*, helping to write use cases.

Led by *system analyst* who is responsible for requirements definition.

How: Tools

Software:

- For use case text, use a web-enabled requirements tool that integrates with a popular word processor.

For use case diagrams, a UML CASE tool.

Hyperlink the use cases; present them on the project website.

Hardware: Use two projectors attached to dual video cards and set the display width double to improve the spaciousness of the drawing area or display 2 adjacent word processor windows .

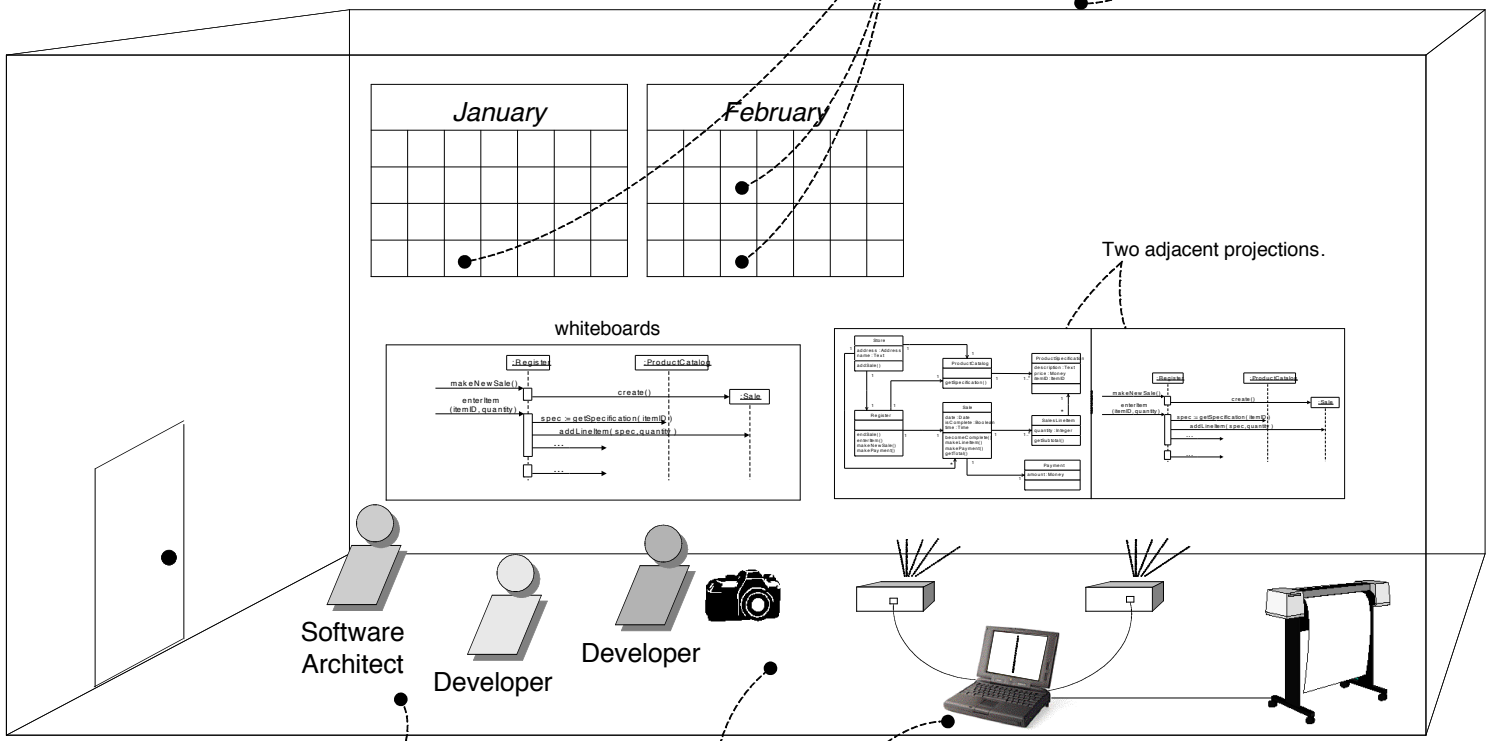
Organisation physique : modélisation objet

When

Near the beginning of each iteration, for a "short" period before programming.

Where

In a project room with lots of support for drawing and viewing drawings.



Who

Perhaps developers will do some design work in pairs. The software architect will collaborate, mentor, and visit with different design groups.

How: Tools

Software: A UML CASE tool that can also reverse engineer diagrams from code.

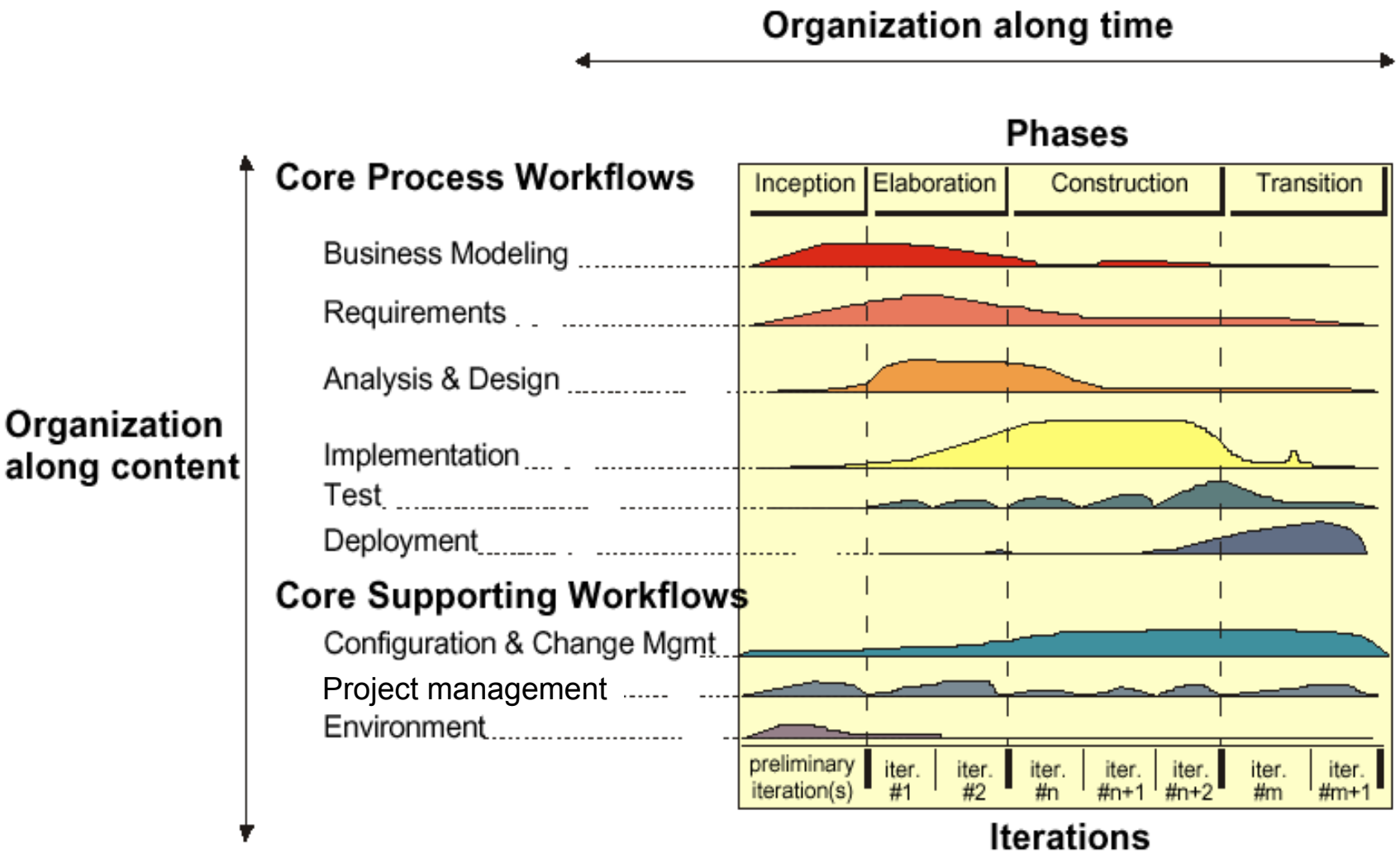
Hardware:
- Use two projectors attached to dual video cards.
- For whiteboard drawings, perhaps a digital camera.
- To print noteworthy diagrams for the entire team, a plotter for large-scale drawings to hang on walls.

Conclusion partielle

- UP décrit un ensemble de processus applicables
  - il faut adapter aux besoins du projet en cours
- **Les principes fondamentaux de UP sont valables pour toute conception orientée-objets**
  - y compris dans les méthode dites « Agiles » (à suivre)

ANNEXE

R(ational)UP quelques disciplines de plus



Annexe : gestion des composants et bibliothécaire

- Qualité
  - niveau plus élevé que celui d'une application (surcoût minimum 50%, en pratique 100%)
  - une confiance absolue est nécessaire pour que la réutilisation soit effective
- Une fonction : bibliothécaire
  - participe au choix des produits à généraliser
  - contribue ou procède à la généralisation
  - établit des normes, règles, niveaux de qualité, protocoles de mise en bibliothèque
  - classifie les produits de la bibliothèque (critères)
  - diffuse, informe, facilite l'accès (outils)