haute école
neuchâtel berne jura

# Méthodologies de développement de logiciels de gestion

## Chapitre 6

## Le Processus unifié de développement logiciel

## Partie I

## Les concepts

Présentation réalisée par P.-A. Sunier
Professeur à la HE-Arc de Neuchâtel

http://lgl.isnetne.ch

# Références

Ce document est rédigé à partir de la partie I de l'ouvrage

[JBR-00]
Le processus unifié de développement
Traduit par V. Zaïm de
The Unified Software Development Process
I. Jacobson, G. Booch, J. Rumbaugh

c/o Eyrolles, 2000
ISBN 2-212-09142-7

haute école
neuchâtel berne jura

# Plan

Introduction

1. Le Processus unifié: piloté par les cas d'utilisation, centré sur l'architecture, itératif et incrémental
2. Les 4 « P »: personnes, projet, produit et processus
3. Un processus piloté par les cas d'utilisation
4. Un processus centré sur l'architecture
5. Un processus itératif et incrémental

# Introduction

*... l'objectif sous-jacent que poursuit une entreprise n'est évidement pas de posséder un bon système informatique, mais d'exploiter les processus métiers (ou les systèmes intégrés) pour répondre au mieux à la demande du marché et accélérer la production de biens et de services de qualité à un prix raisonnable.*

# 1. Le processus unifié

*Les traits véritablement distinctifs du processus unifié tiennent en trois expressions clés:*

- ***piloté par les cas d'utilisation***
- ***centré sur l'architecture***
- ***itératif et incrémental***

# Problématique

*Le processus doit:*

- *dicter l'organisation des activités*
- *diriger les tâches*
  - *individuelles*
  - *de groupe, équipe...*
- *spécifier les artefacts à produire*
- *proposer des critères de contrôle*
  - *des produits du projet*
  - *des activités du projet*

# Piloté par les cas d'utilisation

*L'objectif d'un système logiciel est de rendre service à ses utilisateurs. Pour réussir la mise au point d'un système, il importe, par conséquent, de bien* ***comprendre les désirs et les besoins de ses futurs utilisateurs****.*

*Un cas d'utilisation est une* ***fonctionnalité du système produisant un résultat satisfaisant*** *pour l'utilisateur.*

*Les cas d'utilisation saisissent les besoins fonctionnels et* ***leur ensemble forme le modèle des cas d'utilisation*** *qui décrit les fonctionnalités complètes du système.*

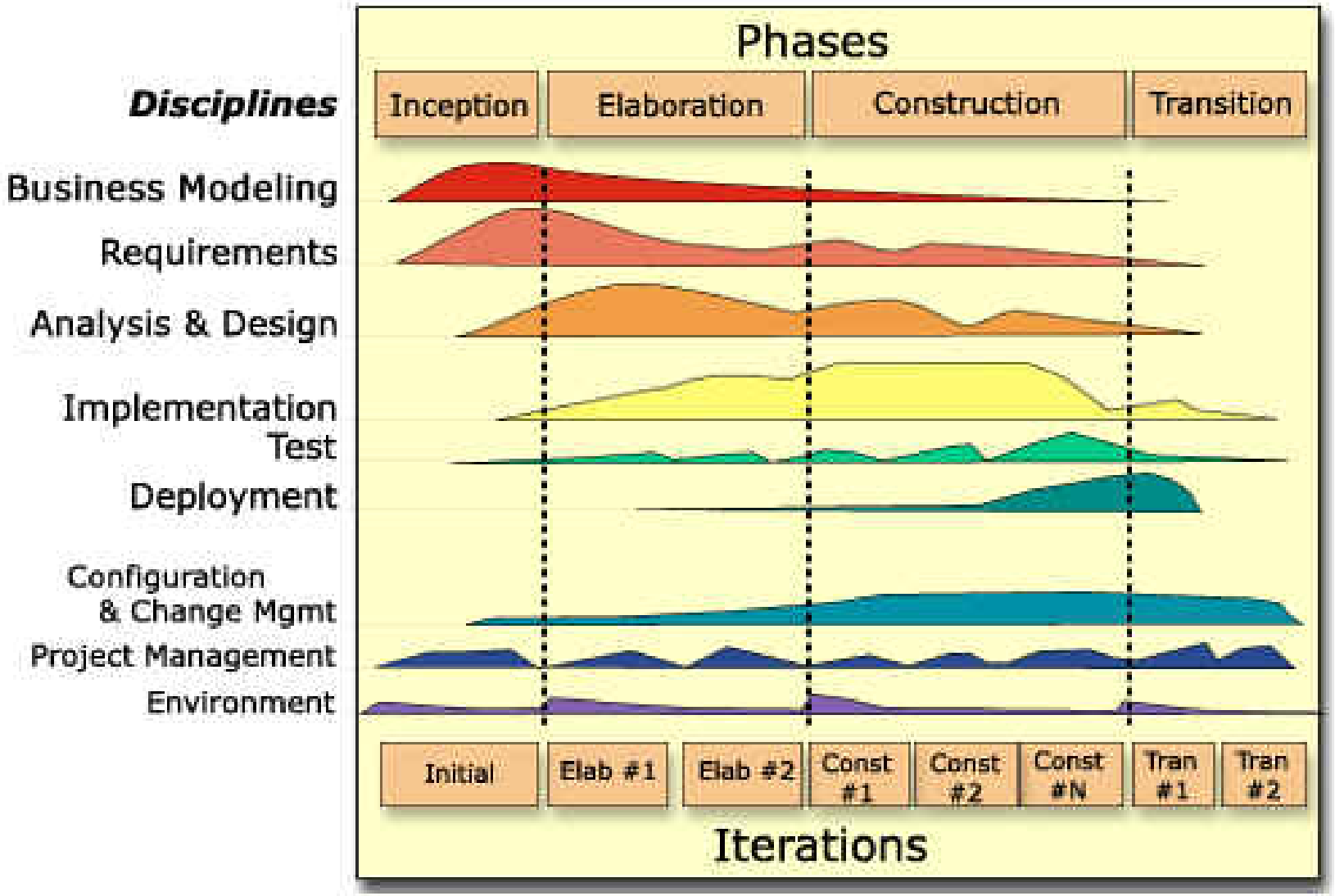
# Centré sur l'architecture

*Le rôle de l'architecture logicielle est comparable à celle que joue l'architecte dans la construction d'un bâtiment. Le bâtiment est envisagé de différents points de vue: structure, services, conduite de chauffage, plomberie, etc.* ***Ce regard multiple dessine une image complète du bâtiment avant le début de la construction****. De la même façon, l'architecture d'un système logiciel peut être décrite comme les différentes vues du système qui doit être construit.*

# Itératif et incrémental

*Le développement d'un produit logiciel destiné à la commercialisation est une vaste opération qui peut s'étendre sur plusieurs mois, voire sur une année ou plus. Il n'est pas inutile de découper le travail en plusieurs parties qui sont autant de mini-projets (Concept systémique de système et sous-systèmes).Chacun d'eux représente une itération qui donne lieu à un incrément.* ***Les itérations désignent des étapes de l'enchaînement d'activités****, tandis que* ***les incréments correspondent à des stades de développement*** *du produit.*

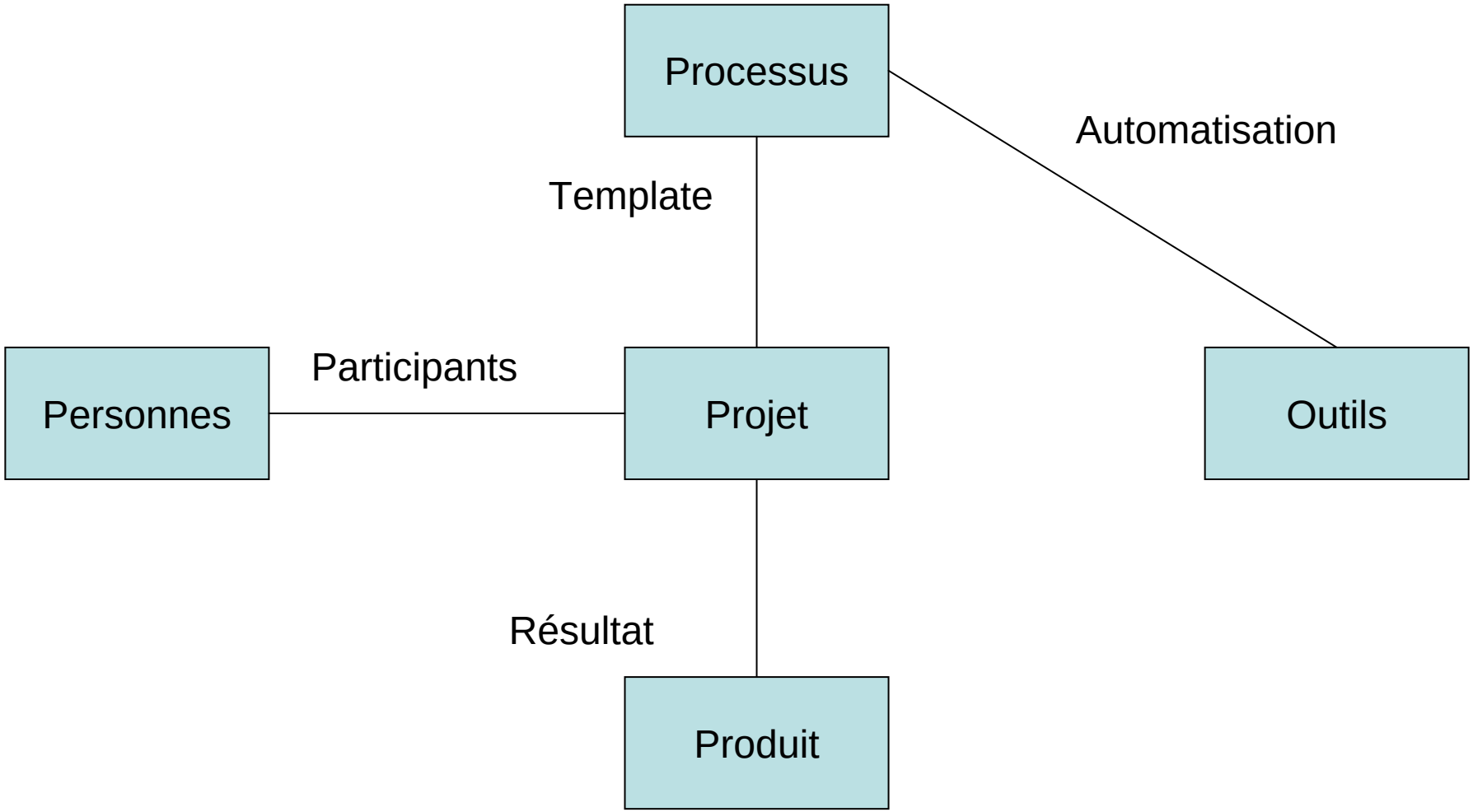
# La vie du Processus unifié



# Un processus intégré

- *basé sur les **composants***
- *utilisant la **modélisation visuelle UML***
- *reposant sur 3 notions maîtresses*
  - ***les cas d'utilisation***
  - ***l'architecture***
  - ***le développement itératif et incrémental***
- *s'appuyant sur **un cadre général** (framework) intégrant*
  - *les cycles*
  - *les phases*
  - *les enchaînements d'activités*
  - *la réduction des risques*
  - *le contrôle qualité*
  - *la gestion de configuration*

# 2. Les 4 «P» du développement logiciel



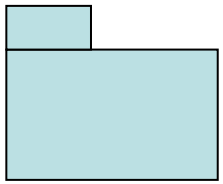
# Les personnes jouent un rôle crucial

*Diverses personnes sont impliquées dans le développement d'un produit logiciel tout au long de son cycle de vie. Certains financent le produit, tandis que d'autres le planifient, le développent, le gèrent, le testent , l'utilisent ou en bénéficient. Le processus guidant le développement doit, par conséquent, être orienté vers les personnes, c'est-à-dire convenir parfaitement à ses utilisateurs.*

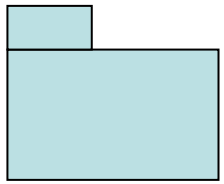
# Un projet ne se résume pas à du code

*Qu'est-ce qu'un système logiciel ?*

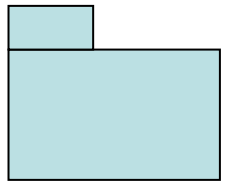
- *Artefacts*
  - *Modèles*



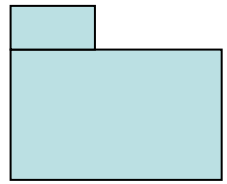
Modèle des cas d'utilisation



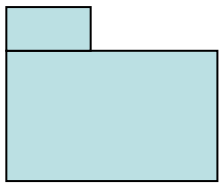
Modèle d'analyse



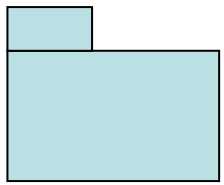
Modèle de conception



Modèle de déploiement

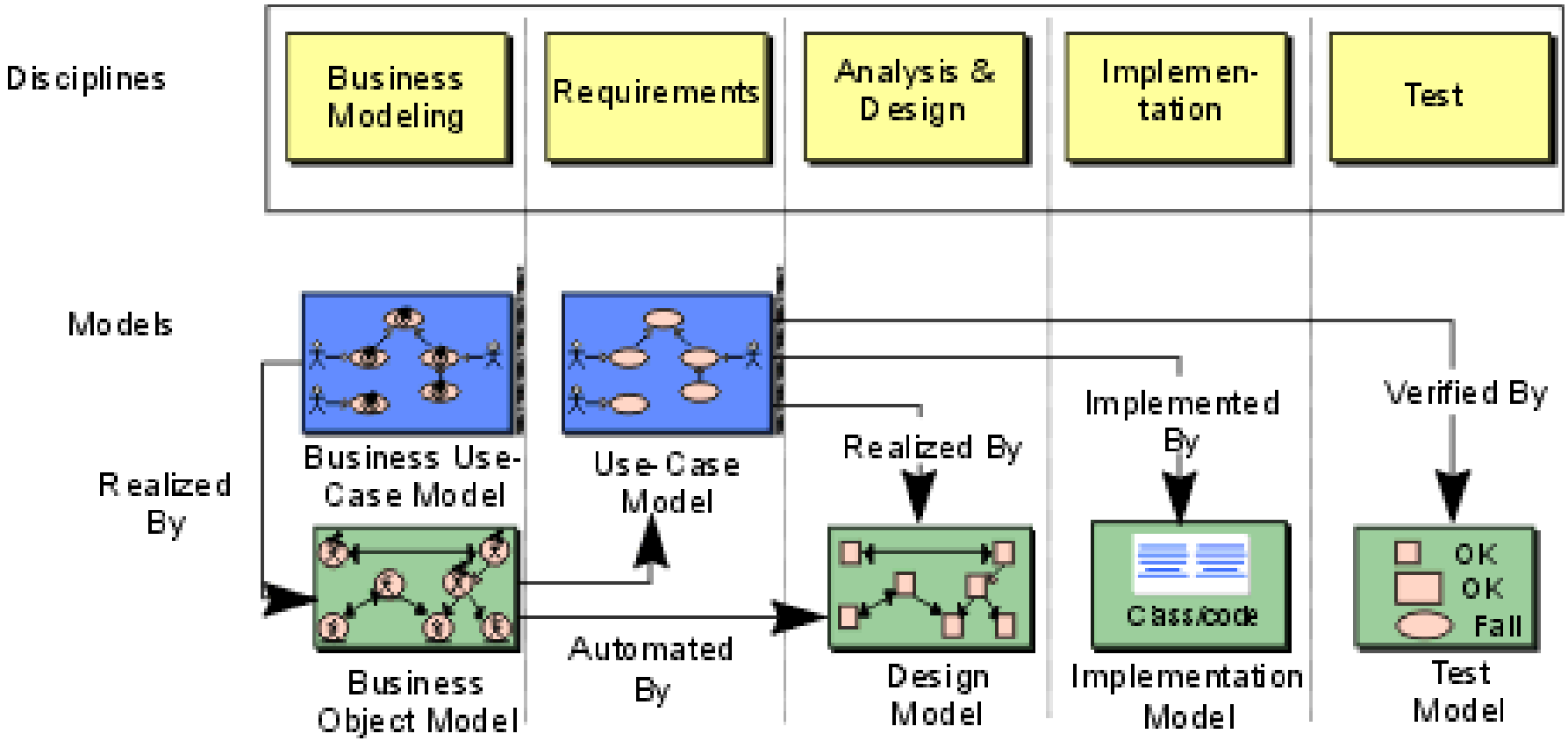


Modèle d'implémentation



Modèle des tests

# Les modèles au cœur du processus



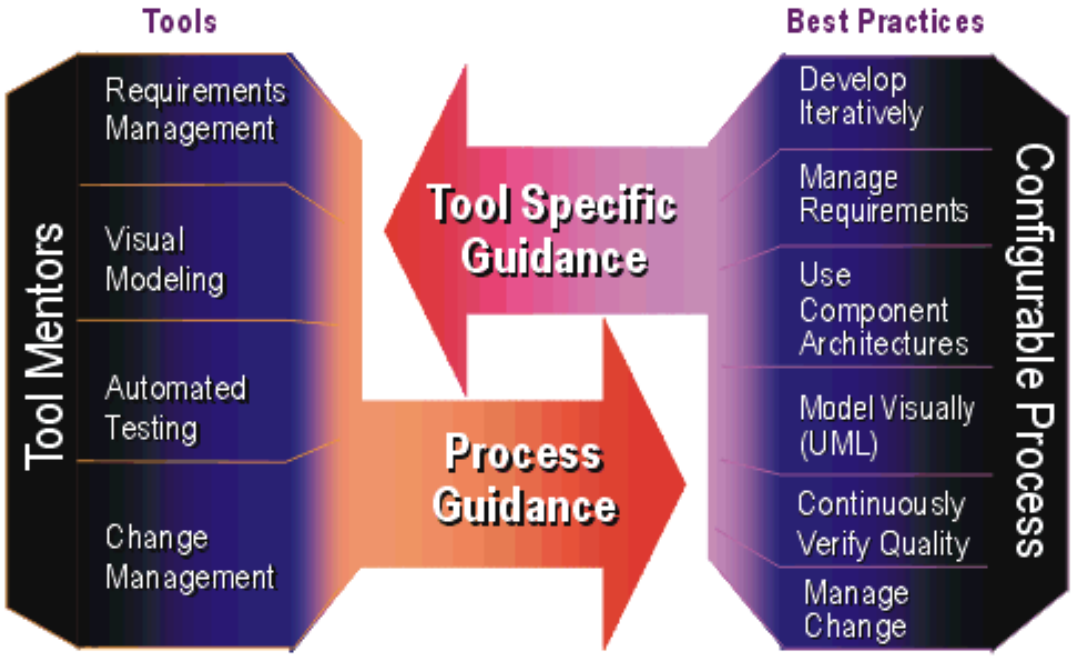
# Le processus dirige les projets

*Dans le Processus unifié, le terme processus renvoie à l'idée de cadre (« template ») pouvant être réutilisé par la création d'instances de celui-ci… L'expression instance de processus est synonyme de projet.*

*Dans cet ouvrage, un processus de développement définit les activités nécessaires à la transformation des besoins des utilisateurs en un ensemble cohérent d'artefacts constituant un produit logiciel et, plus tard, par la transformation des évolutions de ces besoins  en un ensemble d'artefacts tout aussi cohérent.*

# Les outils font partie intégrante du processus

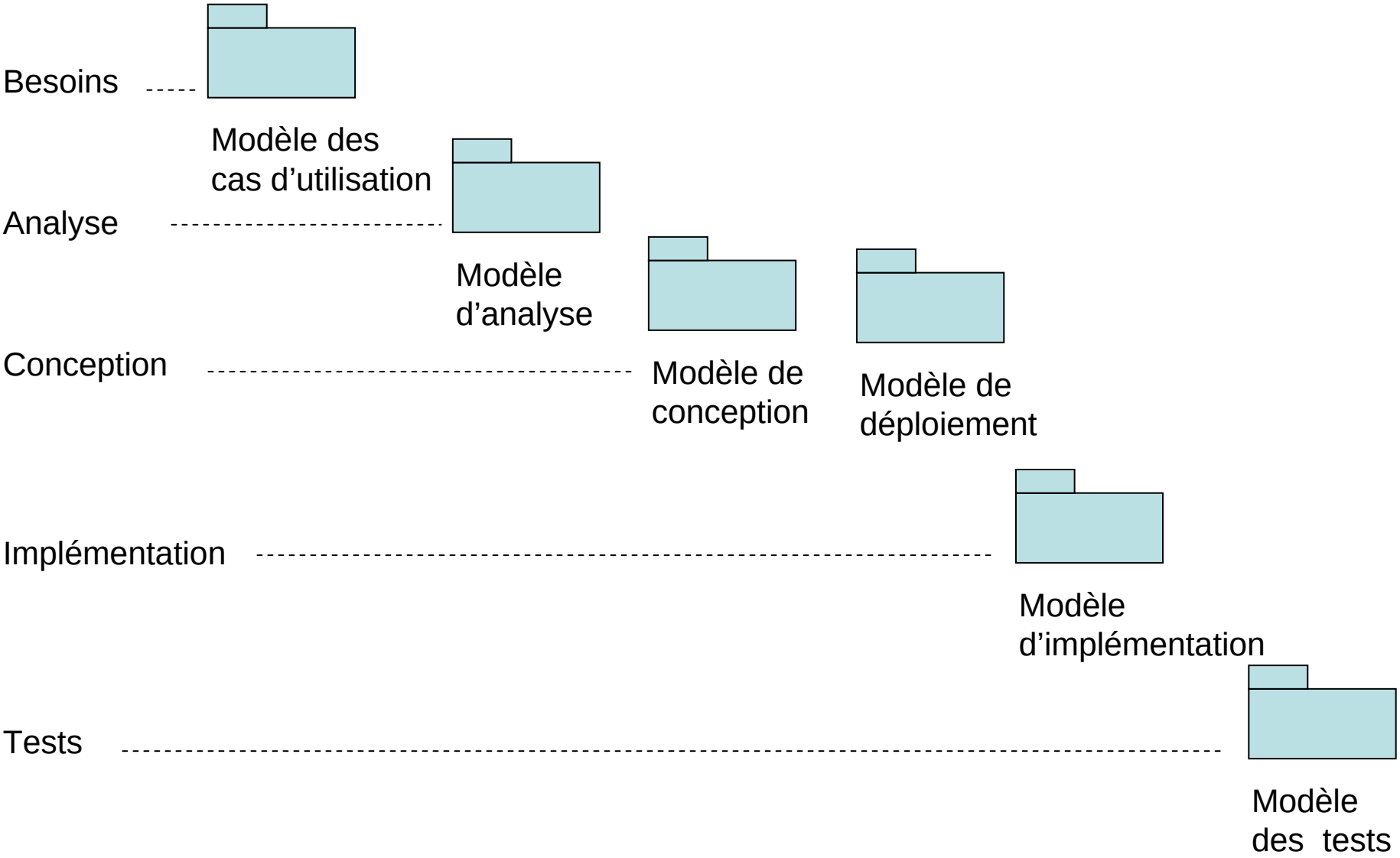
*Les processus actuels de développement logiciel sont pris en charge par des outils. Il est impensable, aujourd'hui, de mettre au point des logiciels sans recourir à un processus reposant sur des outils. Processus et outils sont livrés ensemble, les outils faisant partie intégrante du processus.*



# 3. Le processus unifié piloté par les cas d'utilisations

*L'objectif du Processus unifié est de guider les développeurs vers l'implémentation et le déploiement efficaces de systèmes répondant aux* ***besoins des clients****. Cette efficacité se mesure en termes de coûts, de qualité et de délai de fabrication. Le passage de l'estimation des besoins du client à leur implémentation est loin d'être naturel. D'abord parce que* ***les besoins des clients ne se laissent pas si facilement appréhender****. Il faut disposer d'un moyen de les communiquer de façon claire à toute personne impliquée dans le projet. Il faut, ensuite, être en mesure de concevoir une implémentation opérationnelle répondant à ces besoins. Il faut, enfin, vérifier la pleine satisfaction de ces besoins en testant le système. En raison de sa complexité, le système est décrit sous forme d'activités qui donnent peu à peu naissance à un système opérationnel.*

# Enchaînement d'activités

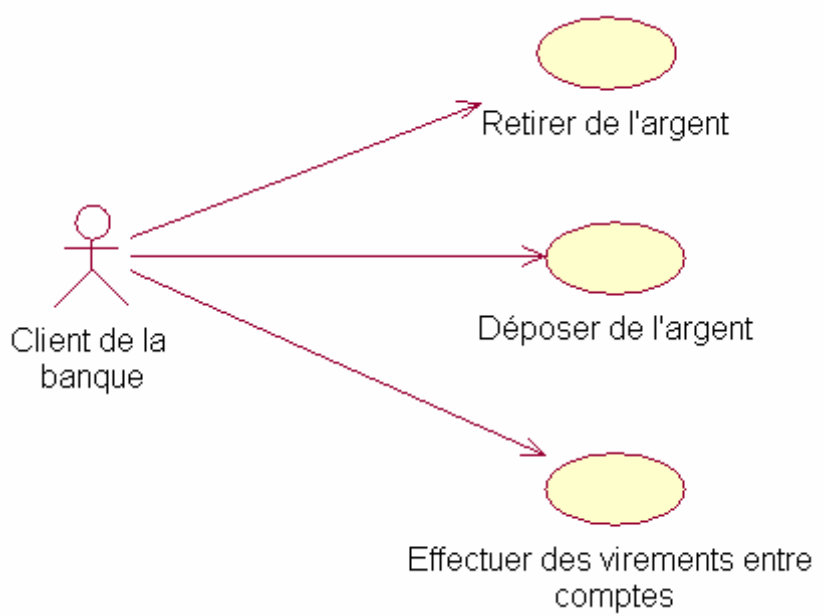


# Pourquoi?

- *Pour appréhender les vrais besoins*
  - Analyse de la valeur des fonctions envisagées
- *Pour diriger le processus*
  - Du besoin aux tests de la solution en créant tous les artefacts et documents de traçabilité nécessaires
- *Pour mettre au point l'architecture*
  - Stabilisation de l'architecture lors des premières itérations

# Besoins fonctionnels

*Le modèle des cas d'utilisation aide le client, les utilisateurs et les développeurs à* ***s'accorder sur l'utilisation du système****.*



# Spécifications du système

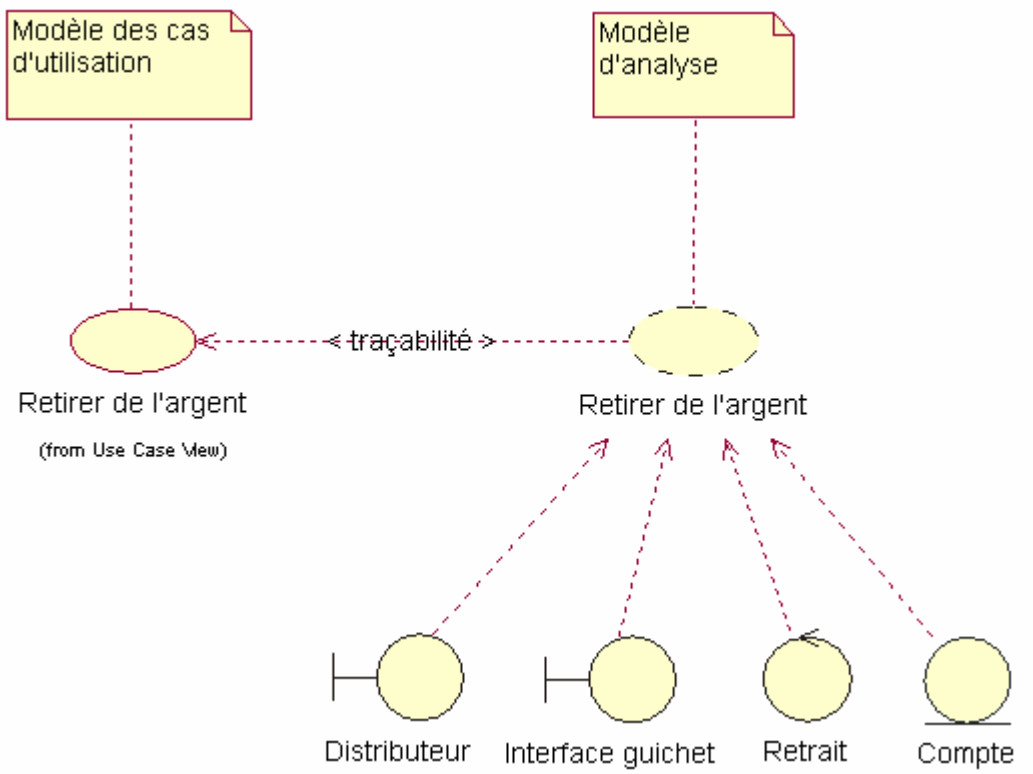
*Un cas d'utilisation spécifie une séquence d'actions, avec ses variantes, pouvant être effectuée par le système et produisant un résultat satisfaisant pour un acteur particulier.*

**Cas d'utilisation « Retirer de l'argent »**

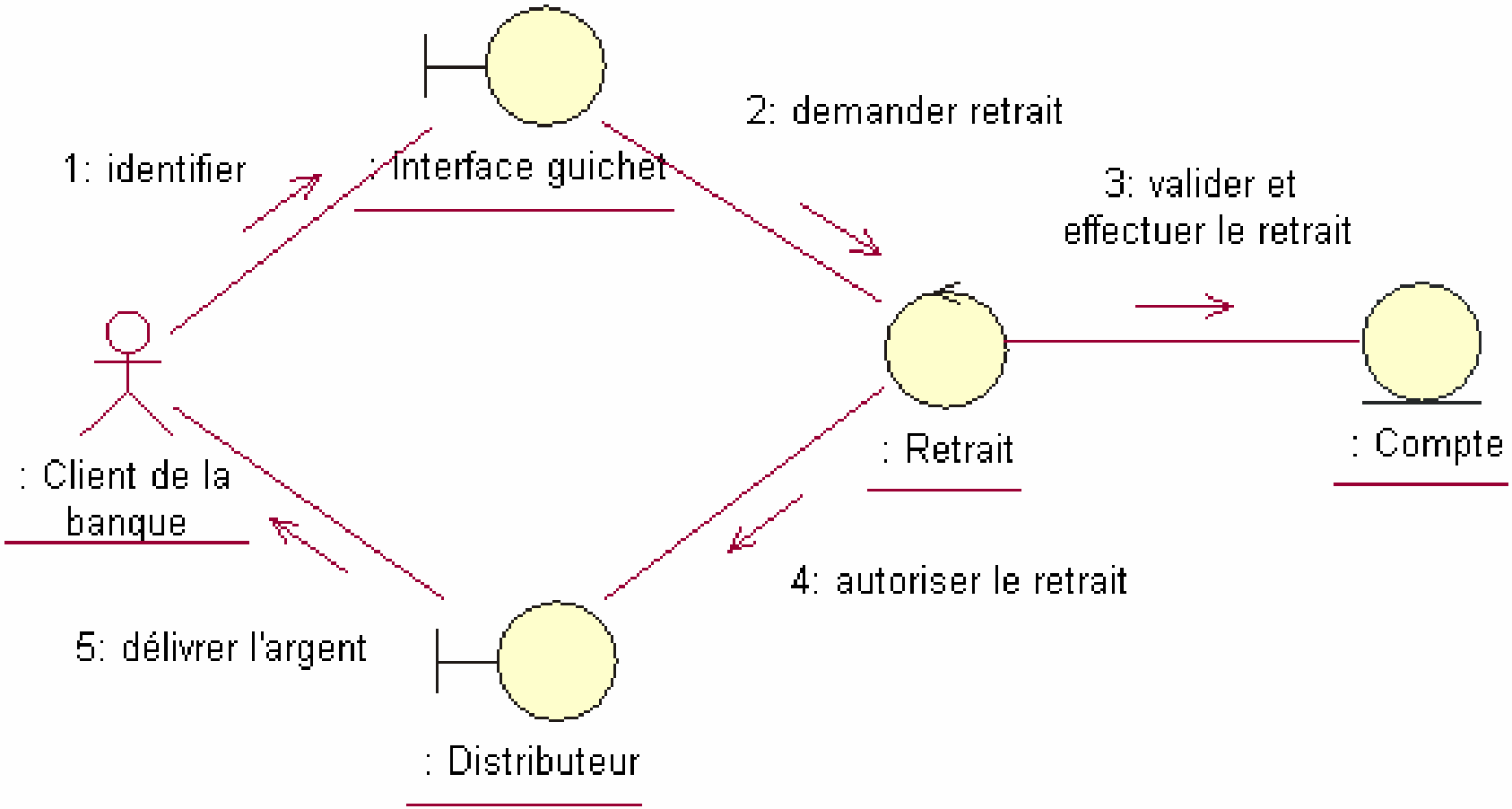
1. Le Client de la banque **s'identifie**
2. Le Client de la banque **choisit le compte duquel il veut effectuer son retrait et spécifie le montant du retrait.**
3. **Le système déduit le montant du compte et délivre l'argent.**

# Réalisation en analyse

*Un cas d'utilisation spécifie une séquence d'actions, avec ses variantes, pouvant être effectuée par le système et produisant un résultat satisfaisant pour un acteur particulier.*



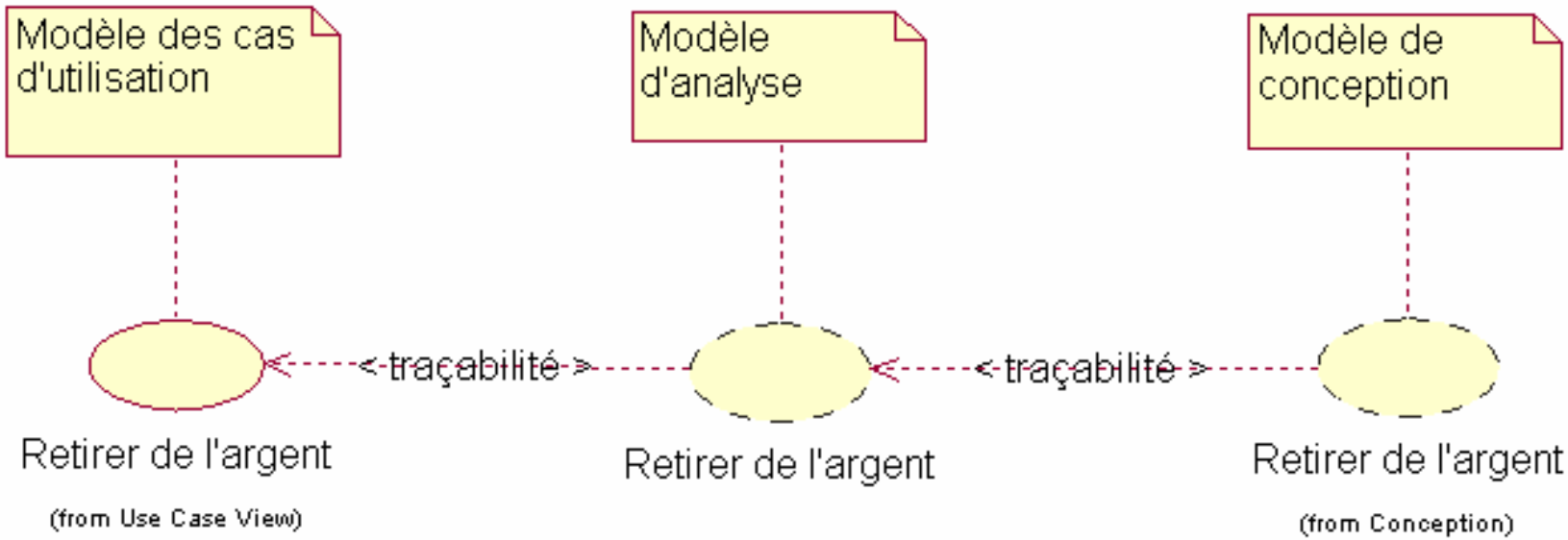
# Diagramme de collaboration en analyse



# Création du modèle de conception à partir du modèle d'analyse

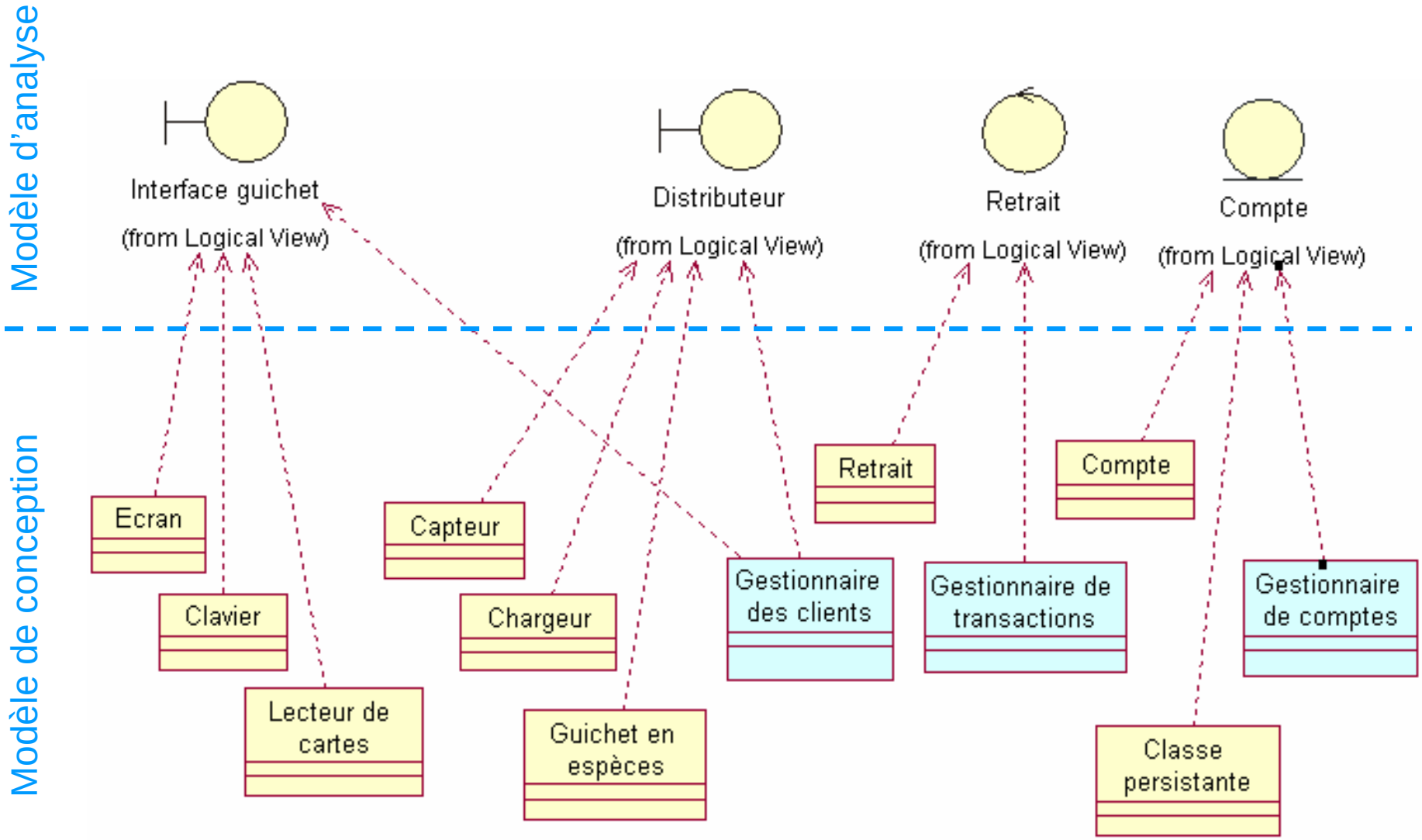


*Le modèle de conception est d'abord créé à partir du modèle d'analyse, avant d'être adapté à l'environnement d'implémentation choisi; comme un ORB, un kit de construction d'IHM ou un système de gestion de base de données.*



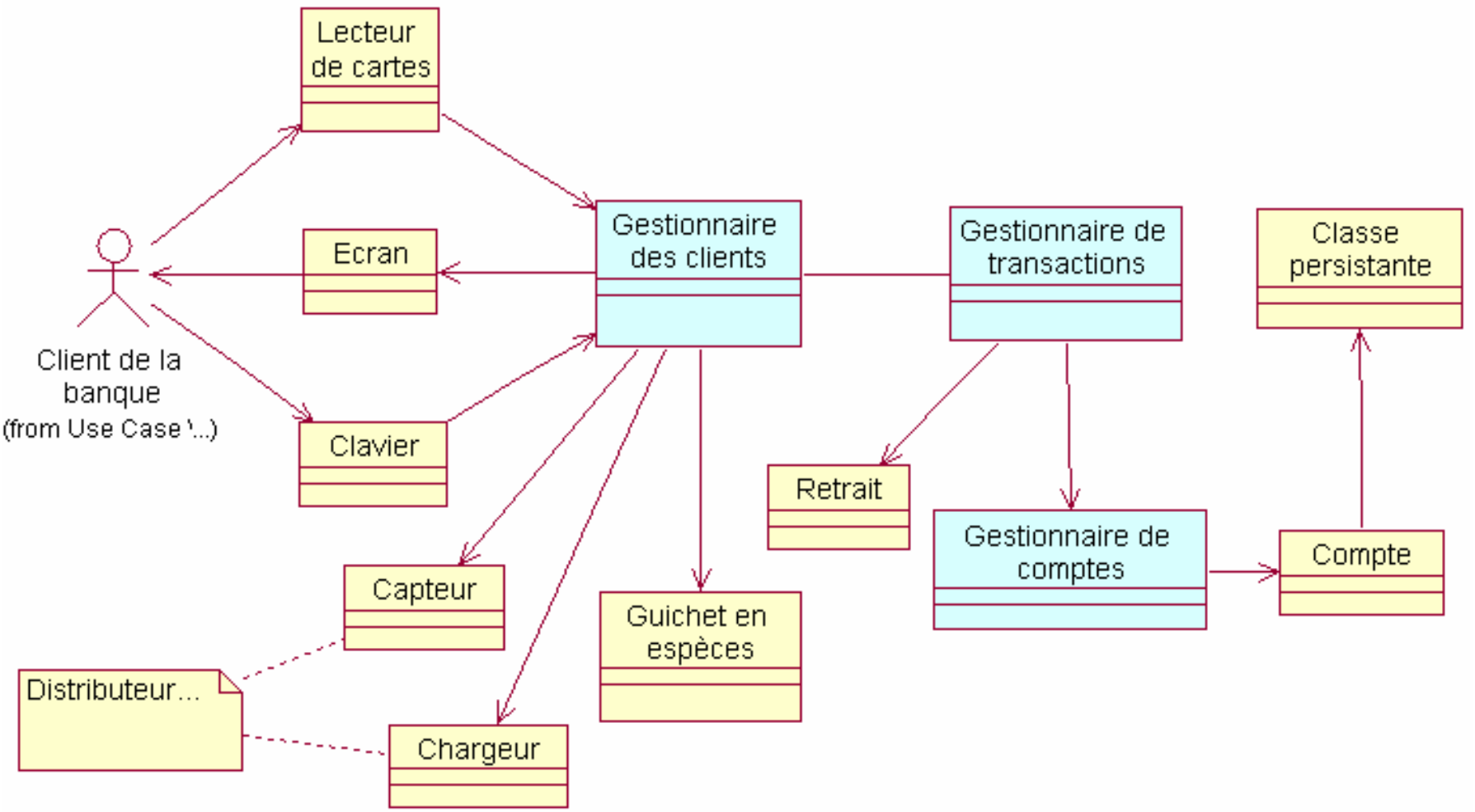
# Dépendances entre classes de conception et classes d'analyse





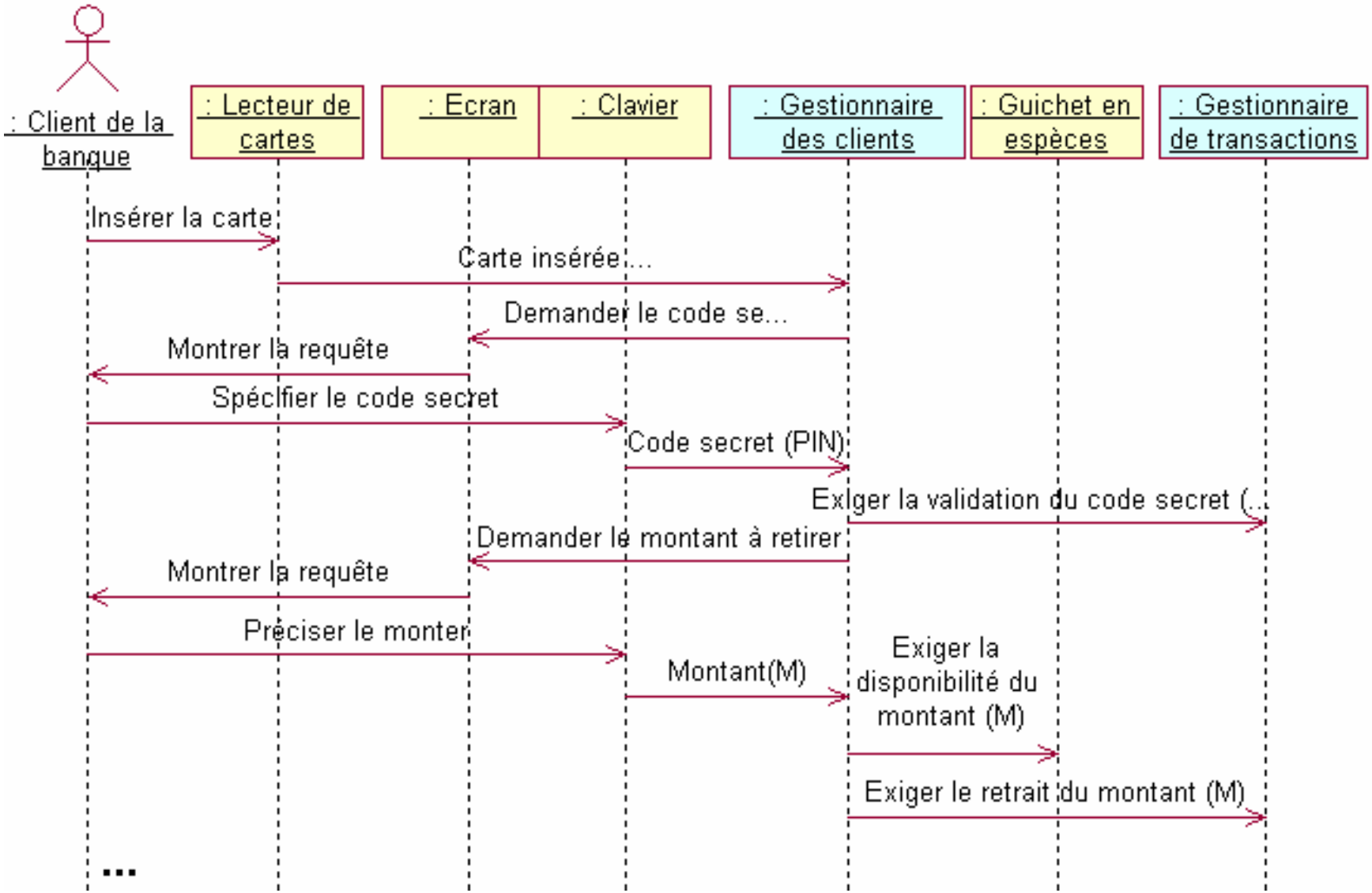

Remarque: Les classes actives sont montrées par un fond bleu

# Diagramme de classes de conception réalisant un cas d'utilisation (Retirer de l'argent)



# Diagramme de séquence, instance d'un cas d'utilisation (Retirer de l'argent)



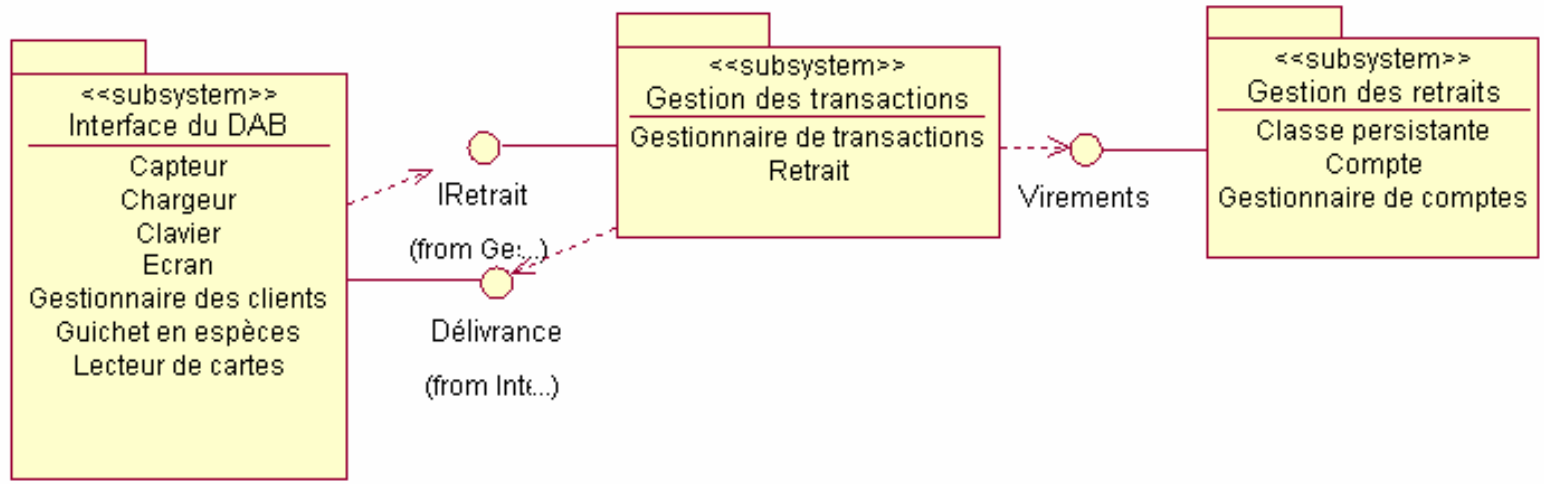





[JBR-00 p65]

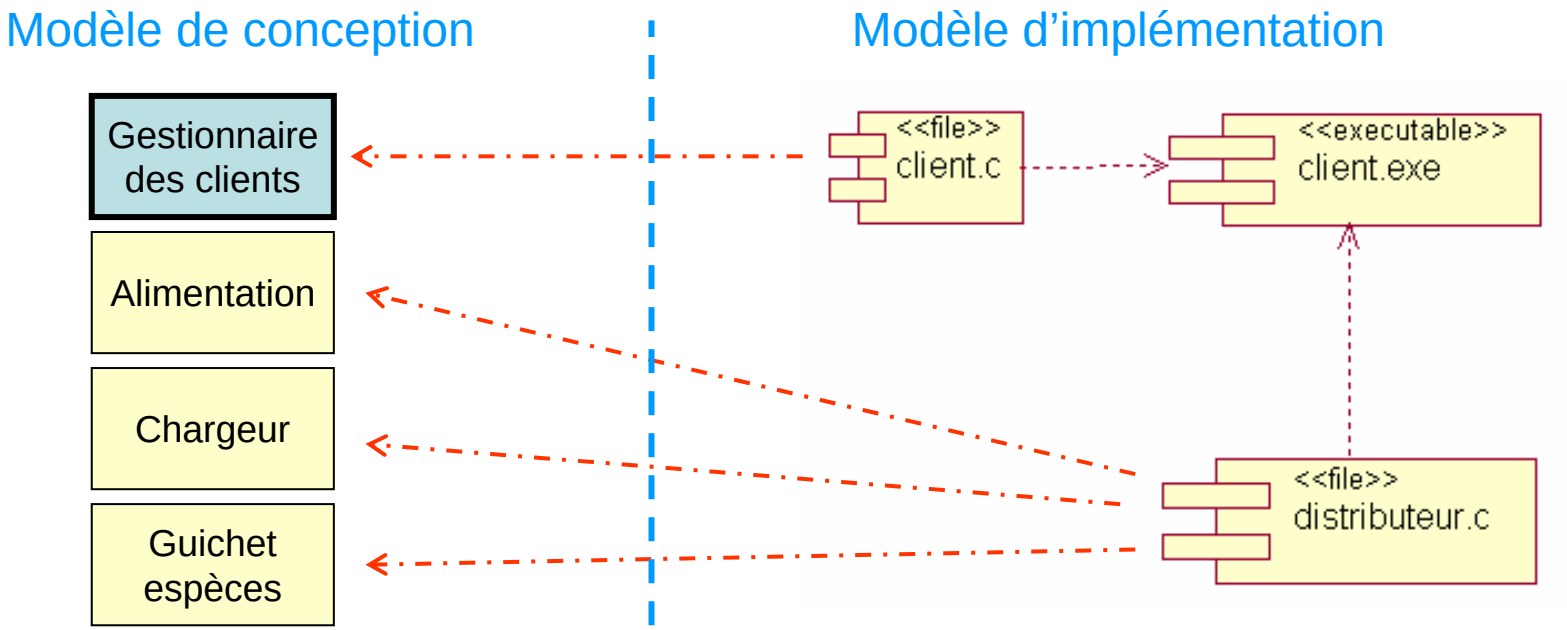
# Les sous-systèmes regroupent les classes

*Les classes sont réparties en sous-systèmes qui permettent le regroupement sémantique de classes et d'autres sous-systèmes. Chaque sous-système fournit et utilise lui-même un ensemble d'interfaces qui en définissent le contexte.*



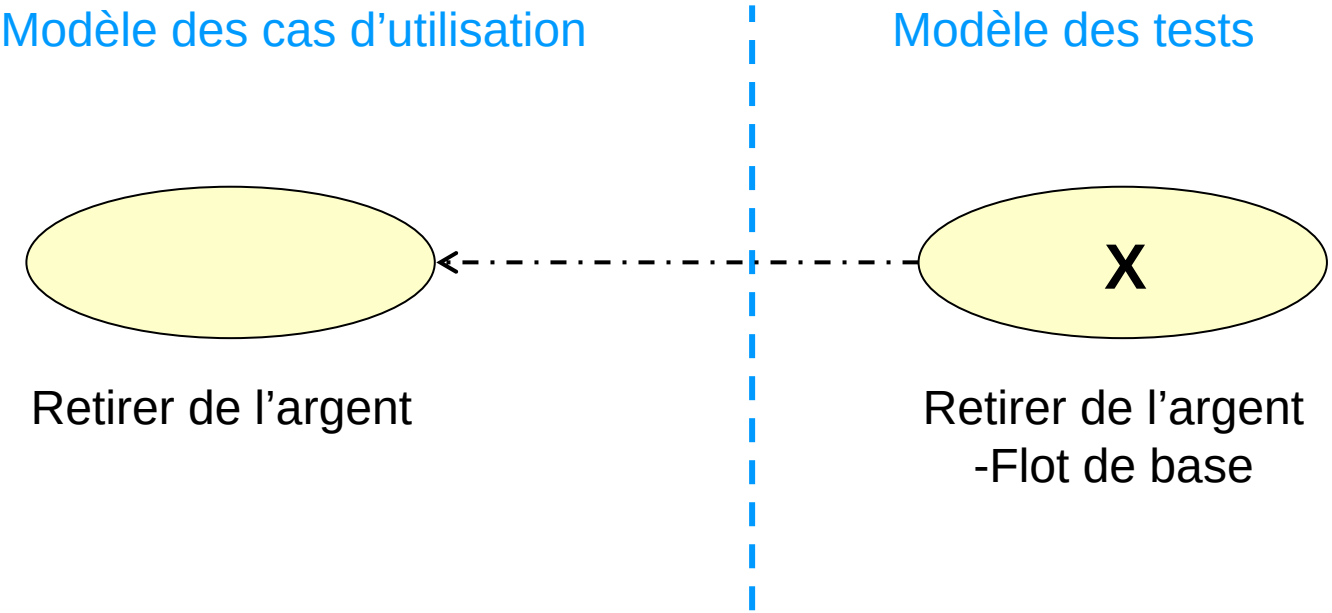
# Les sous-systèmes regroupent les classes

*Au cours de l'enchaînement d'activités de l'implémentation, sont développés tous les éléments exécutables nécessaires à la production d'un système exécutable: les composants, les composants fichiers (code source, shell scripts, etc.), les composants tables (éléments de base de données) et ainsi de suite.* ***Un composant est une partie physique et remplaçable d'un système, conforme à la réalisation d'un ensemble d'interfaces qu'elle doit fournir****.*



# Tests des cas d'utilisation

*Les tests permettent de vérifier que le système implémente correctement sa spécification... Un cas de test est un ensemble d'entrées de test, de conditions d'exécution et de résultats attendus déterminé dans un objectif précis...*



# Instance d'un cas de test pour le cas d'utilisation « Retirer de l'argent »

**Entrées:**

*Le compte 12-121-1211 du Client de la banque montre un solde de 3'500FS.*
*Le Client de la banque s'identifie correctement.*
*Le Client de la banque demande à retirer 2'000FS du compte 12-121-1211.*
*Il y a assez d'argent (au moins 2'000FS) dans le DAB.*

**Résultat:**

*Le solde du compte 12-121-1211 du Client de la banque descend à 1'500FS.*
*Le Client de la banque reçoit 2'000Fs du DAB.*

**Conditions:**

*Aucun autre cas d'utilisation (instance) n'est autorisé à accéder au compte 12-121-1211 pendant ce cas de test.*

# 4. Le processus unifié centré sur l'architecture

*Les cas d'utilisation ne suffisent pas.* ***Pour échafauder un système opérationnel****, il faut quelque chose de plus. Ce "plus" c'est l'architecture. On peut se représenter l'architecture d'un système comme la vision commune sur laquelle doivent s'accorder tous les travailleurs (c'est-à-dire les développeurs et les autres intervenants), ou qu'ils doivent au moins accepter. L'architecture offre une perspective claire de tout le système, indispensable pour en maîtriser le développement.*

# Définition de l'architecture

*Ensemble des **décisions significatives concernant l'organisation d'un système logiciel**, la sélection des éléments structurels dont est composé le système et de leurs interfaces, ainsi que leur comportement tel qu'il est spécifié dans les collaborations entre ces éléments, la composition de ces éléments structurels et comportementaux en sous-systèmes, progressivement plus importants, et le style architectural guidant cette organisation: ces éléments et leurs interfaces, leurs collaborations, et leur composition. L'architecture logicielle ne se soucie pas seulement de structure et de comportement, mais aussi d'utilisation, de fonctions, de performances, de capacité à réagir aux changements, de réutilisation, de clarté, de contraintes et de compromis économiques et technologiques, enfin de préoccupations esthétiques.*

[JBR-00 p468]

# Pourquoi une architecture?

*Il nous faut une architecture pour:*

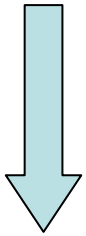
- *comprendre le système*
  - *Complexité, complication, pluridisciplinarité*
- *organiser le développement*
  - *Besoin de communication de l'équipe*
- *favoriser la réutilisation*
  - *Normalisation, standardisation des composants logiciels*
- *faire évoluer le système*
  - *Réaction aux changements de l'environnement*

# Cas d'utilisation et architecture?

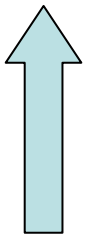
*… **l'architecture est influencée par les cas d'utilisation** que nous voulons faire prendre en charge par le système. Ces cas d'utilisation agissent, en effet, comme des **pilotes** pour l'architecture. Après tout, on veut obtenir une architecture adaptée à l'implémentation des cas d'utilisation retenus. On sélectionne, dans les premières itérations, quelques cas d'utilisation jugés indispensables à la création de l'architecture. Parmi ces cas d'utilisation pertinents pour l'architecture, figurent ceux dont les clients ont le plus besoin pour la version à venir et peut-être pour les versions suivantes.*

# Facteurs d'influence de choix architecturaux

Cas d'utilisation



***Architecture***



Contraintes et facteurs favorables:

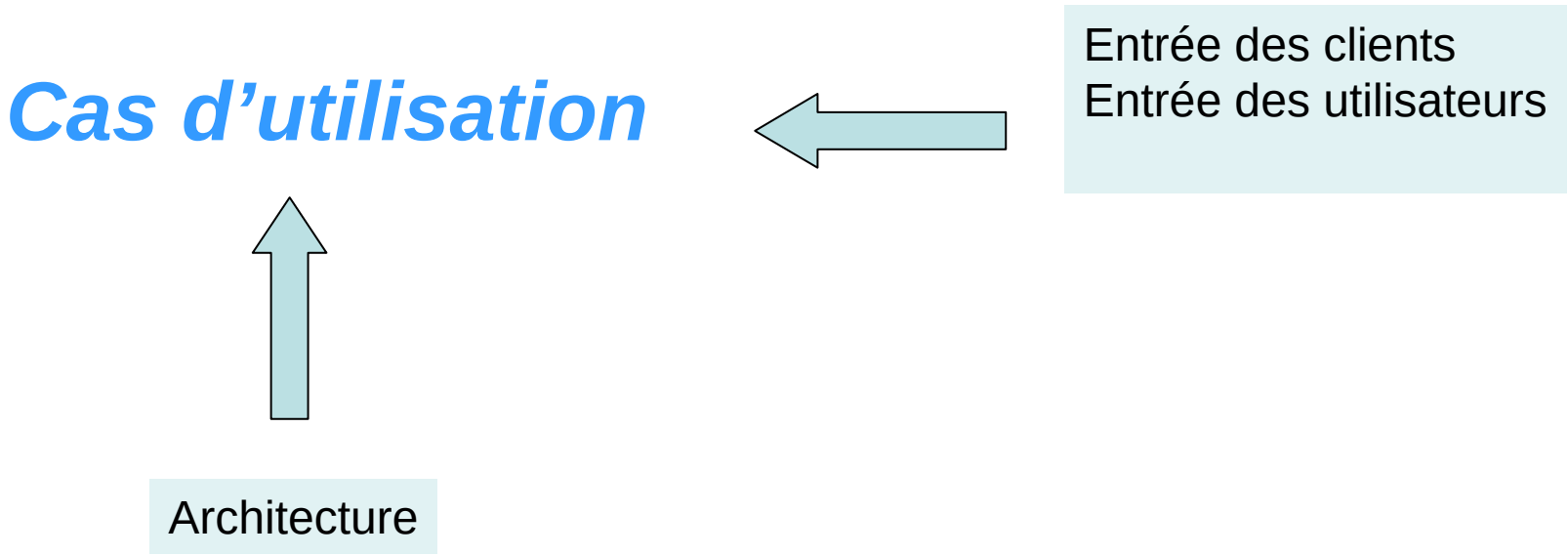
- Logiciel système
- Middleware (y-compris les frameworks)
- Systèmes existants
- Standards et politiques
- Besoins non fonctionnels
- Besoins de distribution

Expérience:

- Architectures précédentes
- Patterns (Modèles) d'architecture

# Influence de l'architecture sur l'élaboration de cas d'utilisation

*Les cas d'utilisation peuvent être élaborés à partir des entrées fournies par les clients et les utilisateurs* (concepteurs et développeurs). *Cependant, ils sont également influencés par l'architecture déjà en place.*

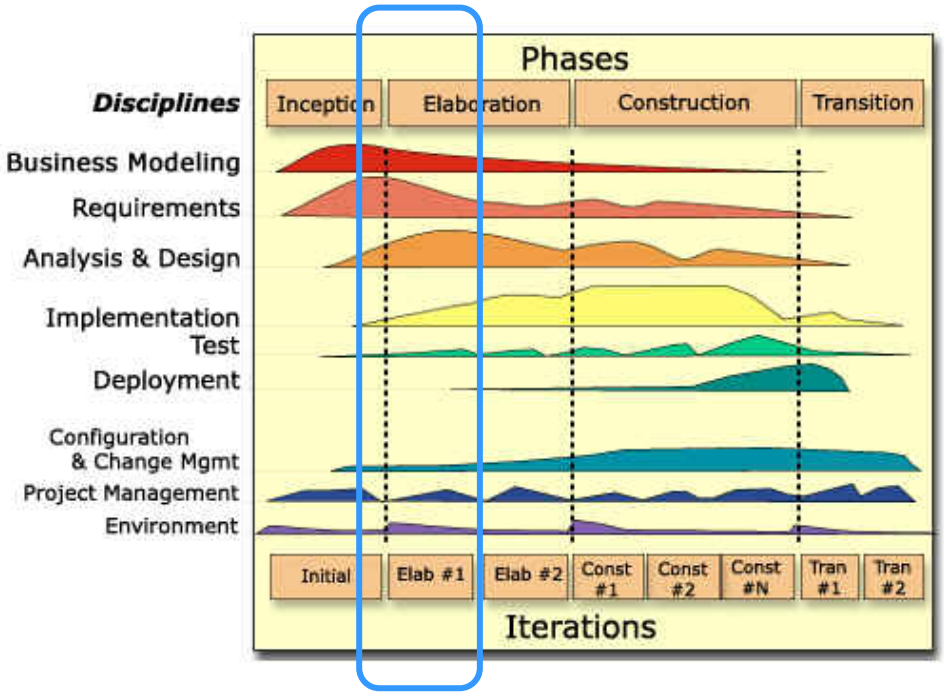


# Influence réciproque de l'architecture et des cas d'utilisation

*Les cas d'utilisation orientent le développement de l'architecture, tandis que l'architecture guide la réalisation des cas d'utilisation*

# Mise au point de l'architecture

*Il est fascinant de constater que l'on peut parfaitement mettre au point une architecture stable pendant la phase d'élaboration du premier cycle de vie, alors même que seuls quelque 30% de la première version du produit ont été investis.*



# Utilisation de patterns d'architecture

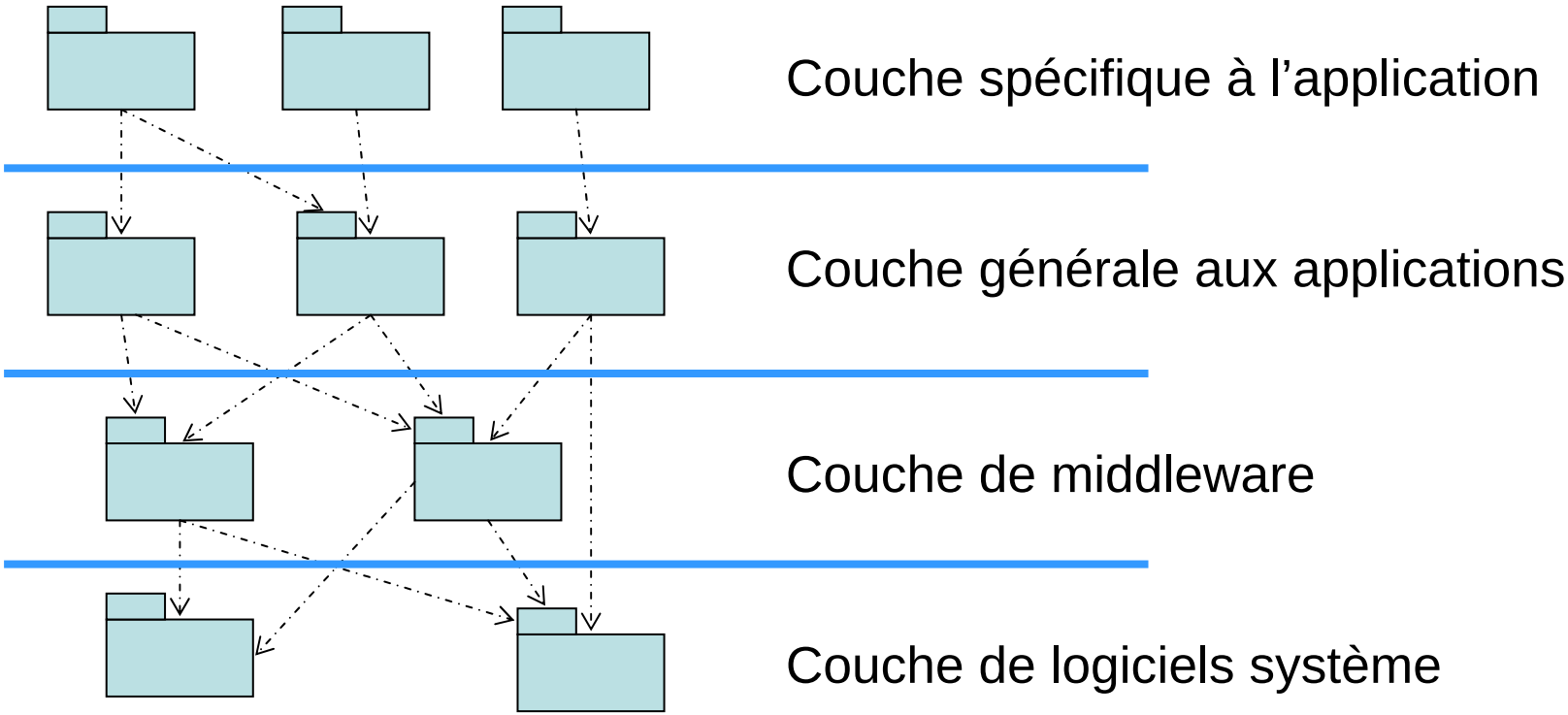
*Les spécialistes définissent un pattern comme* ***une solution à un problème récurrent de conception****.*

*… nous définirons un pattern comme une collaboration générique pouvant être spécialisée*

*… les patterns de conception deviennent , ainsi, des collaborations entre classes et instances, dont le comportement est expliqué par les diagrammes d'interaction*

# Pattern Couches (Layers)

*… Les composants d'une couche ne peuvent faire référence qu'aux composants de la couche située immédiatement au-dessous.*




[JBR-00 p85]

# Description de l'architecture

*...la description de l'architecture présente les vues des modèles et comprend ainsi des cas d'utilisation, des sous-systèmes, des interfaces, certains composants, des nœuds et des collaborations. Elle intègre également les besoins significatifs sur le plan architectural qui ne sont pas décrits par les cas d'utilisation. Les autres besoins, non fonctionnels, sont formulés en tant qu'exigences supplémentaires... La description de l'architecture doit, en outre, contenir une brève description de la plate-forme, des systèmes existants et des logiciels commerciaux à utiliser.*

# C'est l'architecte qui crée l'architecture

*C'est l'architecte entouré de quelques développeurs qui créée l'architecture...*
*C'est lui qui choisit parmi les patterns d'architecture, sélectionne les produits existants et organise les dépendances de sous-systèmes de façon à sérier les problèmes. "Sérier les problèmes" signifie, dans ce cas, créer une conception évitant qu'un changement intervenant sur un sous-système ne se répercute sur plusieurs autres sous-systèmes.*

*...Un architecte puise à deux types de sources:*

- *la connaissance du domaine*
- *la connaissance du développement logiciel*

*L'expérience acquise ... se révélera également précieuse.*

# Vue architecturale du modèle des cas d'utilisation

*La vue architecturale du modèle des cas d'utilisation présente les principaux acteurs et cas d'utilisation.*

## *Vue architecturale du modèle des cas d'utilisation du DAB*

Retirer de l'argent ***est le cas d'utilisation le plus important…***

***Pour définir l'architecture, l'architecte suggère donc de procéder é l'implémentation intégrale de ce seul cas d'utilisation pendant la phase d'élaboration..***

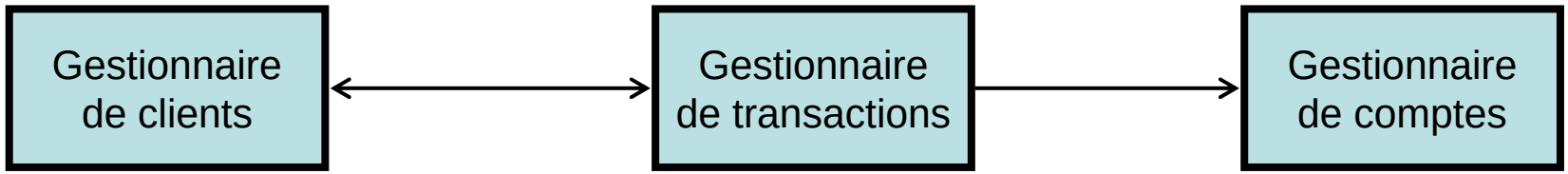
***La vue architecturale du modèle des cas d'utilisation devra donc montrer la description complète du cas d'utilisation*** Retirer de l'argent.

# Vue architecturale du modèle de conception (1)

*La vue architecturale du modèle de conception présente les classificateurs de ce modèle ayant une importance cruciale pour l'architecture: les principaux sous-systèmes et interfaces, ainsi que quelques-unes des classes primordiales, essentiellement les classes actives.*

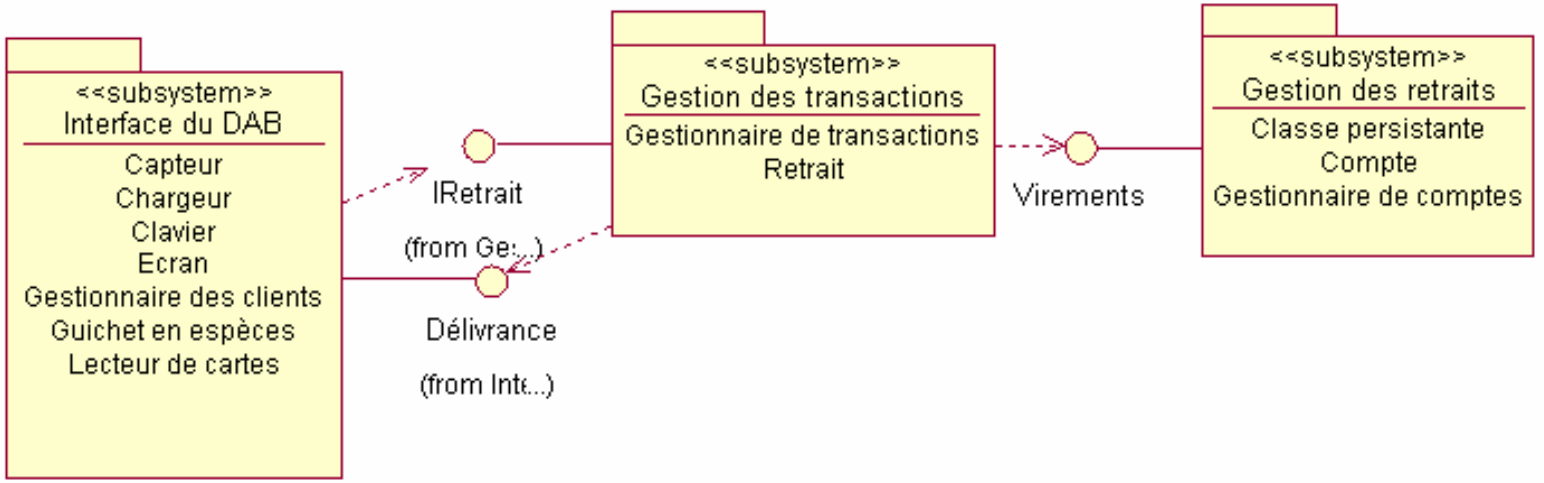
***Vue architecturale du modèle de conception du DAB***

***…nous avons identifié 3 classes actives (pour réaliser le cas d'utilisation*** Retirer de l'argent***)***



# Vue architecturale du modèle de conception (2)

***Diagramme de classes (pour réaliser le cas d'utilisation** Retirer de l'argent**) décrivant les sous-systèmes et les interfaces qui les relient.***



# Vue architecturale du modèle de conception (3)

***Diagramme des sous-systèmes collaborant à l'exécution du cas d'utilisation*** Retirer de l'argent

***Pré-condition: le*** Client de la banque ***dispose d'un compte en banque opérationnel pour le DAB.***

1. ***L'acteur*** Client de la banque ***choisit de retirer de l'argent et s'identifie auprès de l'Interface du DAB, par exemple en utilisant une carte magnétique ayant un numéro et un code secret. Le Client de la banque indique également le montant du retrait et le compte à partir duquel il souhaite l'effectuer…***

Remarque: A notre connaissance, il n'est pas possible de montrer les paquetages dans un diagramme de collaboration avec Rational Rose que nous avons utilisé pour les illustrations pratiques

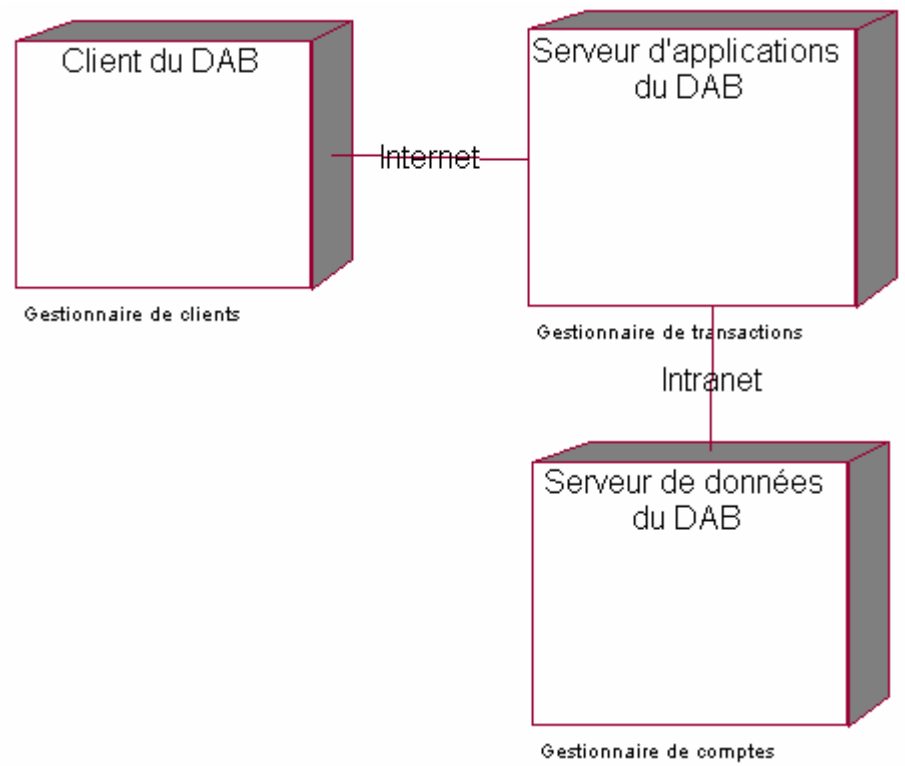
# Vue architecturale du modèle de déploiement (1)

*Le modèle de déploiement définit l'architecture physique en termes de nœuds connectés. Ces nœuds sont des unités matérielles sur lesquels s'exécutent les composants logiciels. Il est fréquent que l'on sache à quoi ressemblera l'architecture physique du système avant de commencer à développer le système. Les nœuds et connexions peuvent alors être modélisés dans le modèle de déploiement dès l'enchaînement d'activités des besoins.*

# Vue architecturale du modèle de déploiement (2)

***Le modèle de déploiement définit trois nœuds:***

- ***Client du DAB***
- ***Serveur d'application du DAB***
- ***Serveur de données du DAB***



# 5. Un processus itératif et incrémental

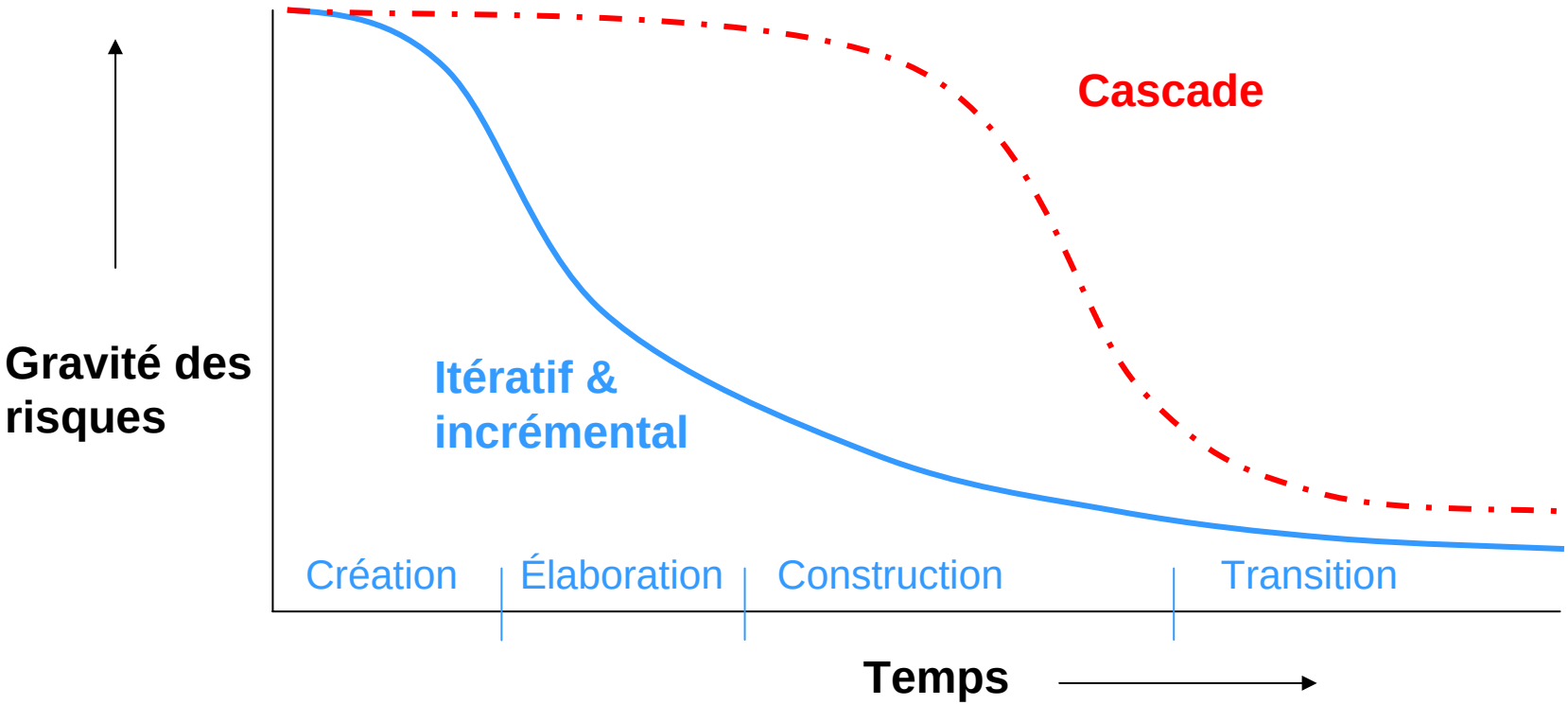
*Parvenir à un juste équilibre entre cas d'utilisation et architecture revient, grosso modo, à harmoniser forme et fond dans le développement d'un produit… Savoir ce qui vient en premier nous ramène au problème de l'œuf et de la poule… Ce sont d'interminables itérations, survenues tout au cours du long processus d'évolution qui ont donné naissance à la poule et à l'œuf. De la même façon, au fil du processus de développement logiciel, les développeurs recherchent consciencieusement cet équilibre (entre cas d'utilisation et architecture) à travers une série d'itérations. L'approche itérative et incrémentale du développement constitue bien, par conséquent, le troisième pivot du Processus unifié.*

# Pourquoi un développement itératif et incrémental

- *Prendre en main très tôt les risques importants*
- *Proposer une architecture qui guidera le développement logiciel*
- *Fournir une infrastructure préfabriquée (framework) capable de mieux prendre en compte les exigences incontournables et les autres changements*
- *Élaborer progressivement le système, de façon incrémentale, au lieu de tout faire d'un coup, à la fin, lorsque les changements coûtent chers*
- *Offrir un processus de développement favorisant la productivité en équipe*

# Réduire les risques

*« Le risque est inhérent à l'engagement de ressources actuelles vers des attentes futures » selon le prophète du management Peter F. Drucker.*



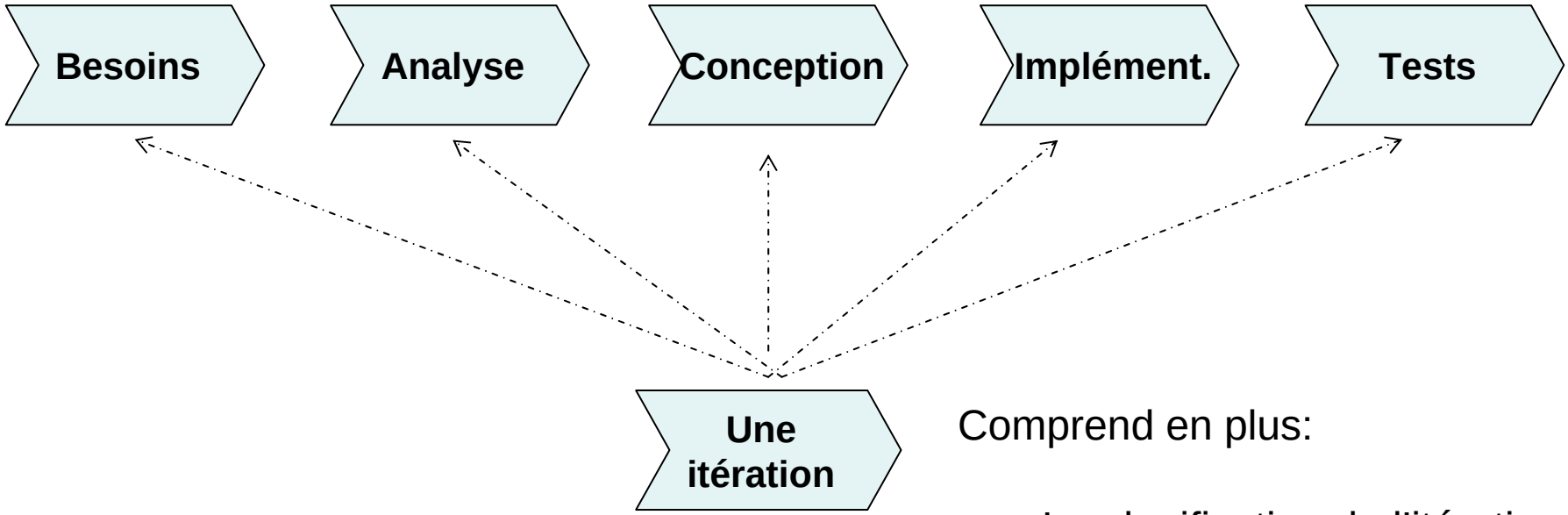
# L'approche itérative est guidée par la réduction des risques

*Un risque est une variable d'un projet qui met en danger, voire compromet totalement, la réussite du projet en question. C'est « la probabilité selon laquelle un projet peut être confronté à des événements indésirables, comme des retards de calendrier, des dépassements de budget ou une annulation pure et simple ».*

*… on organise les itérations de façon à réduire au maximum les risques.*

*La réduction des risques est cruciale pour les itérations menées dans les phases de création et d'élaboration. Dans la phase plus tardive de construction, les risques ont, dans l'ensemble, été ramenés à un niveau normal…*

# L'itération générique




Comprend en plus:

- La planification de l'itération
- L'évaluation de l'itération
- …

# Planifier les itérations

*… le cycle de vie itératif réclame plus de planification et de réflexion que l'approche en cascade.*
*Dans le modèle en cascade tout est planifié à l'avance, souvent avant que les risques aient été réduits et l'architecture établie. Les plans qui en résultent reposent donc sur une grande part d'incertitude…*
*L'approche itérative, en revanche, ne planifie pas tout le projet en détail au cours de la phase de création; elle s'engage simplement sur les premières étapes.*

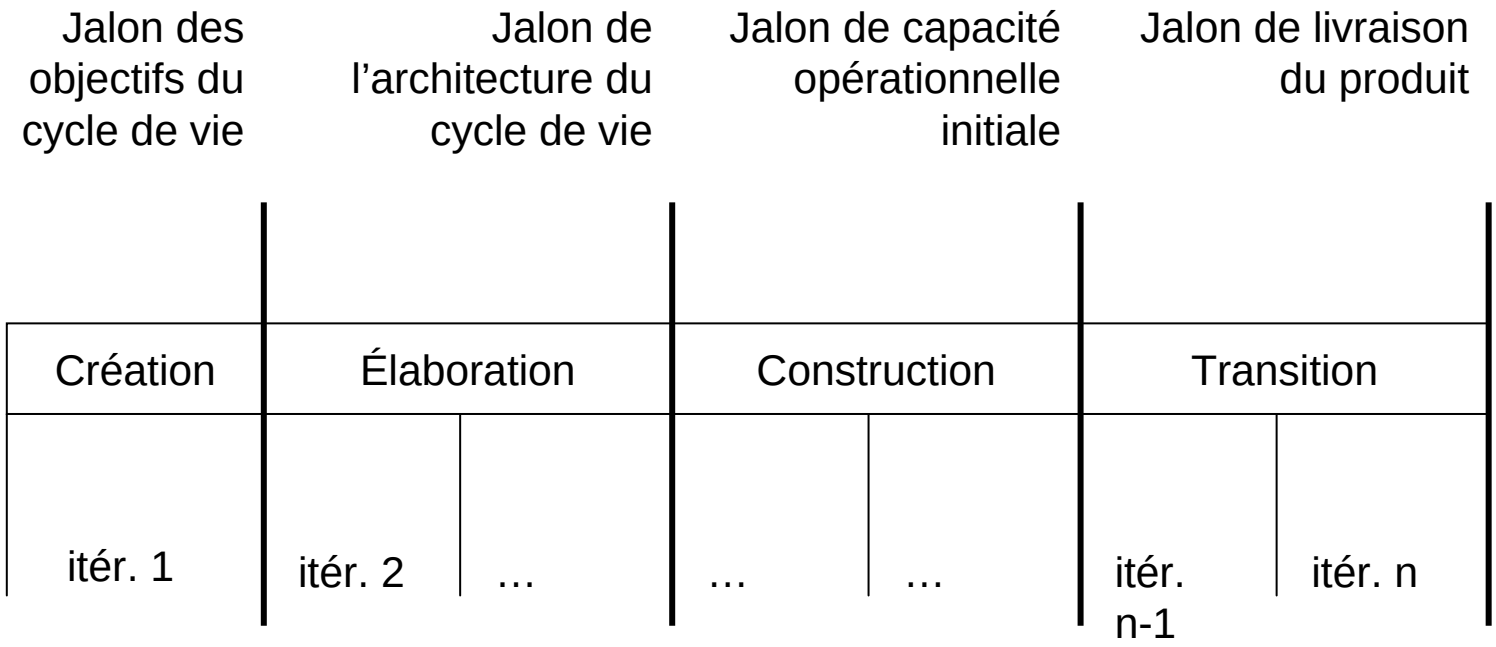
# Une itération se traduit par un incrément

*Un incrément est la différence  entre la version interne d'une itération et la version interne de l'itération suivante.*

*A la fin d'une itération, l'ensemble des modèle représentant le système se trouve dans un état particulier. Cet état, ou état d'avancement, est appelé référence.*

# Itérations dans le cycle de vie

*Chacune des quatre phases se conclut par un jalon majeur.*

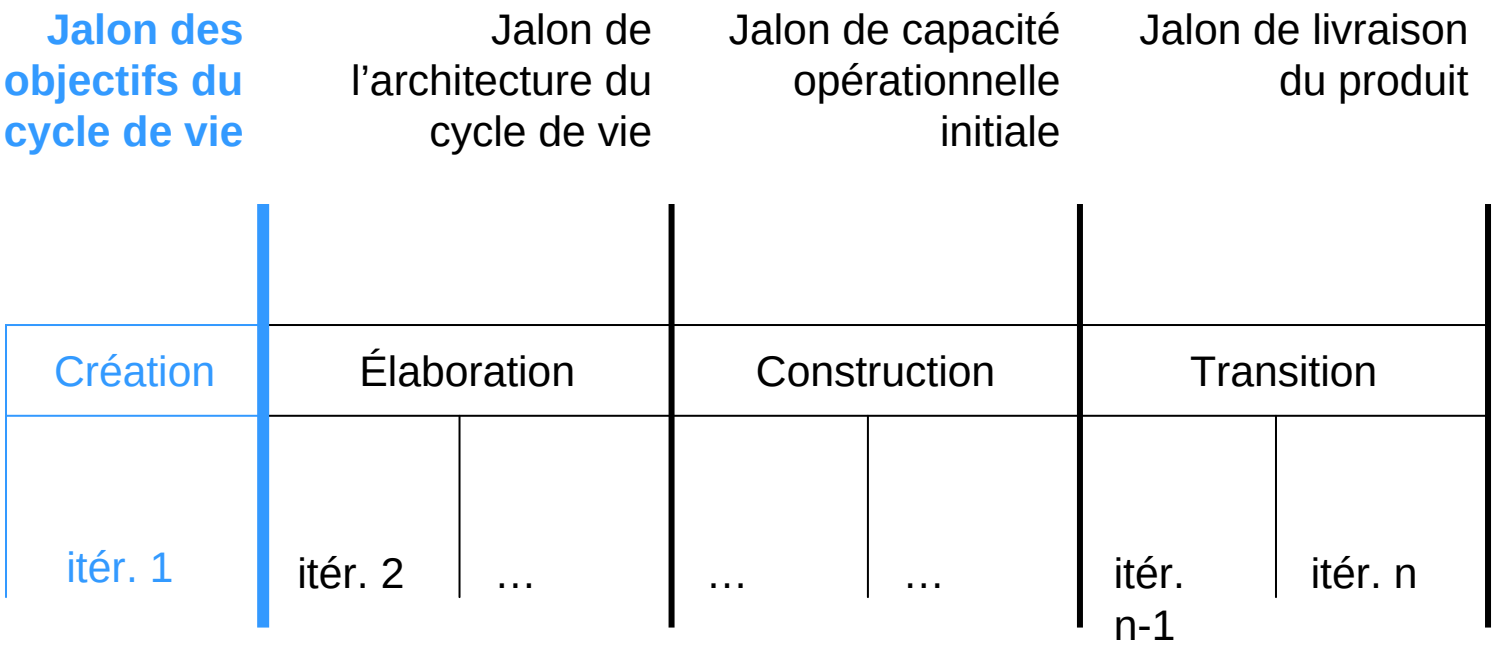


# Jalons et décisions

*L'objectif de chaque jalon majeur est de s'assurer que les modèles des différents enchaînements d'activités évoluent de façon équilibrée. Par « équilibrée », nous entendons que les décisions essentielles influant sur ces modèles et celles concernant les risques,les cas d'utilisation et l'architecture doivent être prises au début du cycle de vie.*

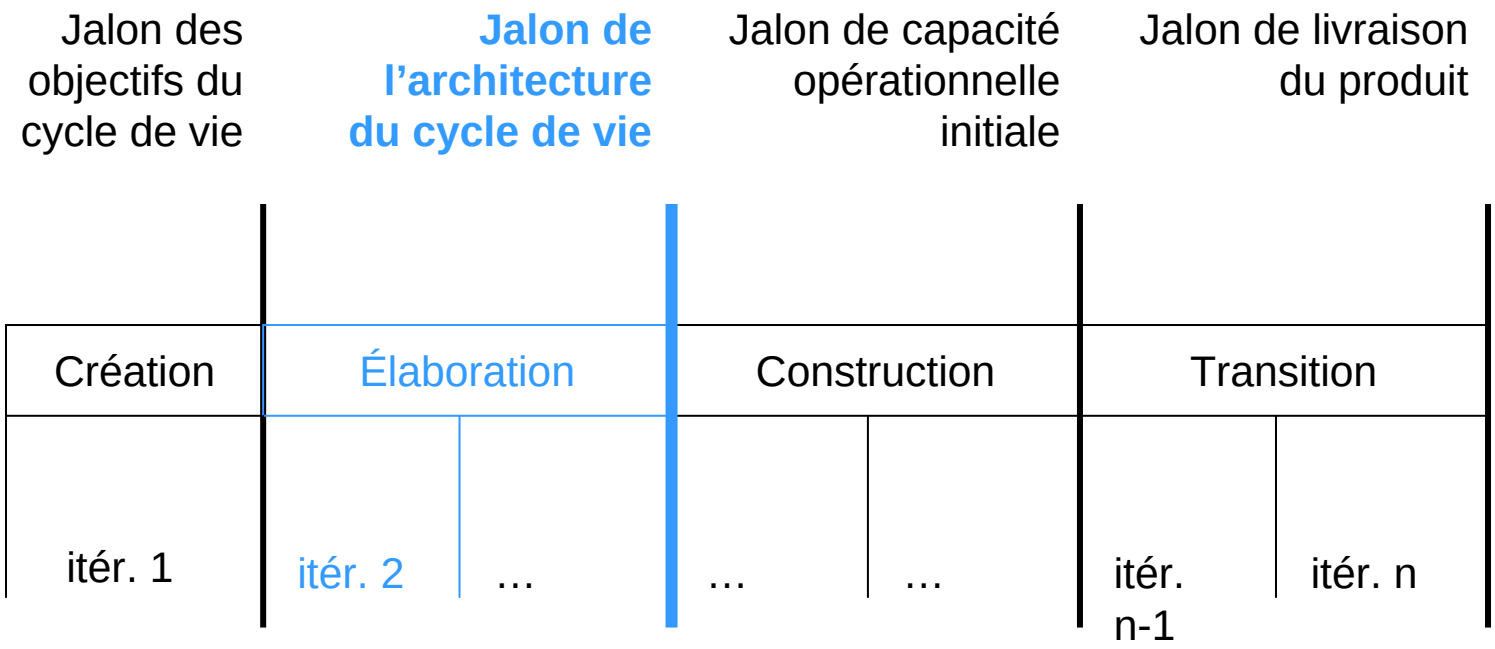
# Objectifs de la phase de création

- *Portée et rôle du produit*
- *Réduire les risques les plus importants*
- *Etude rentabilité, viabilité commerciale*



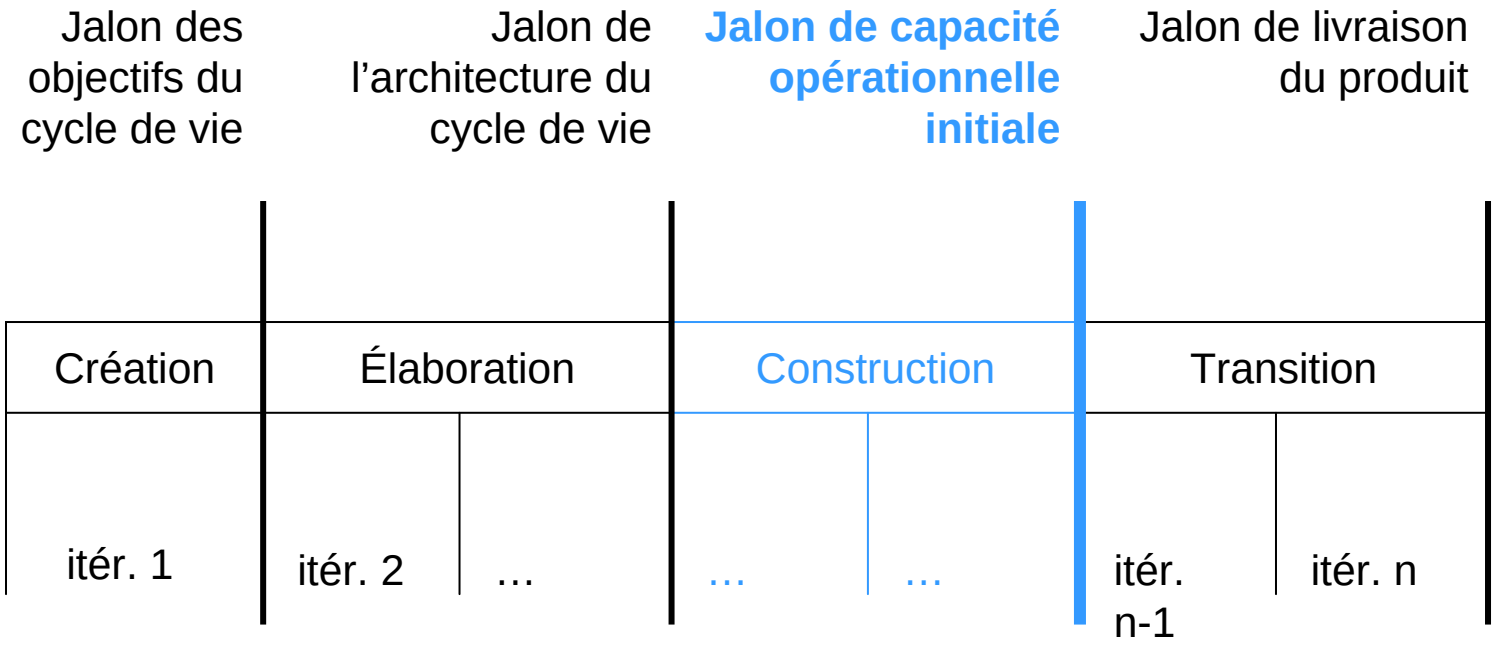
# Objectifs de la phase d'élaboration

- *Créer l'architecture de référence*
- *Saisir l'essentiel des besoins*
- *Réduire les risques de moindre gravité*



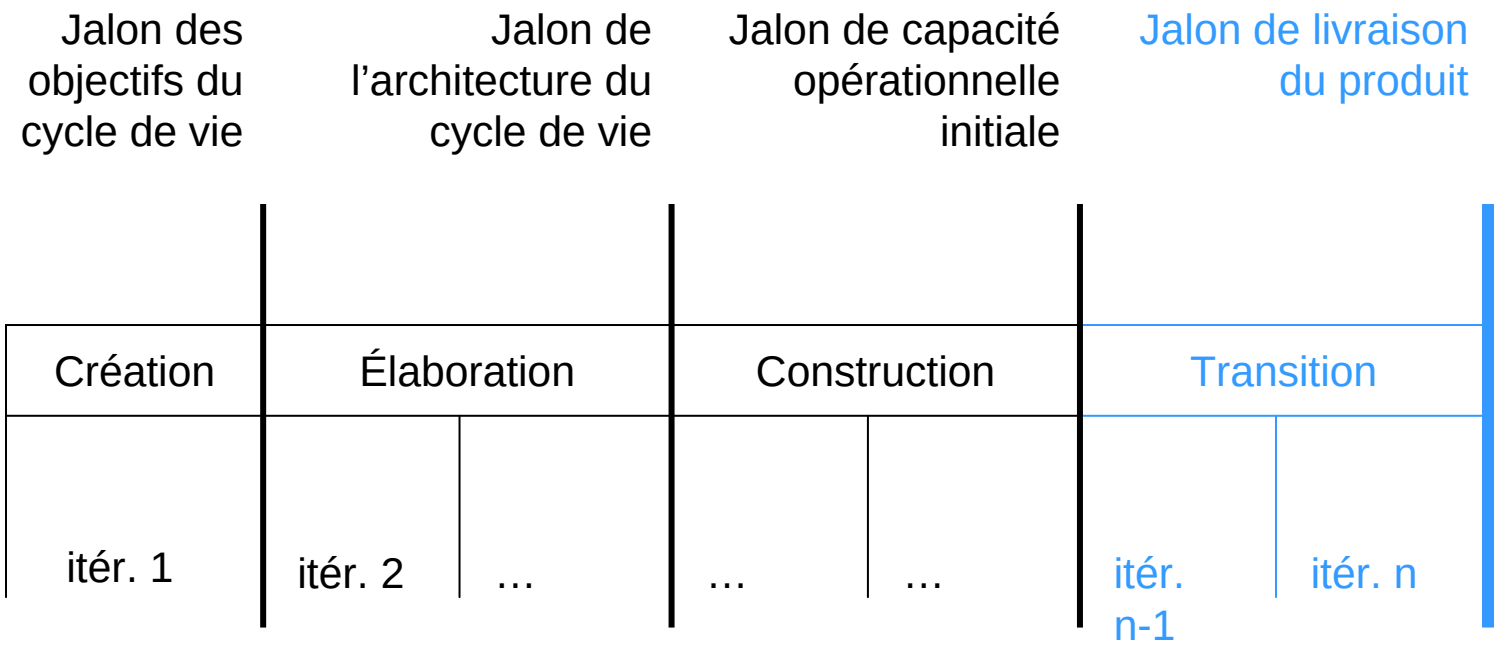
# Objectifs de la phase de construction

- *Développer le système complet*
- *S'assurer que le produit peut être utilisé par les clients*



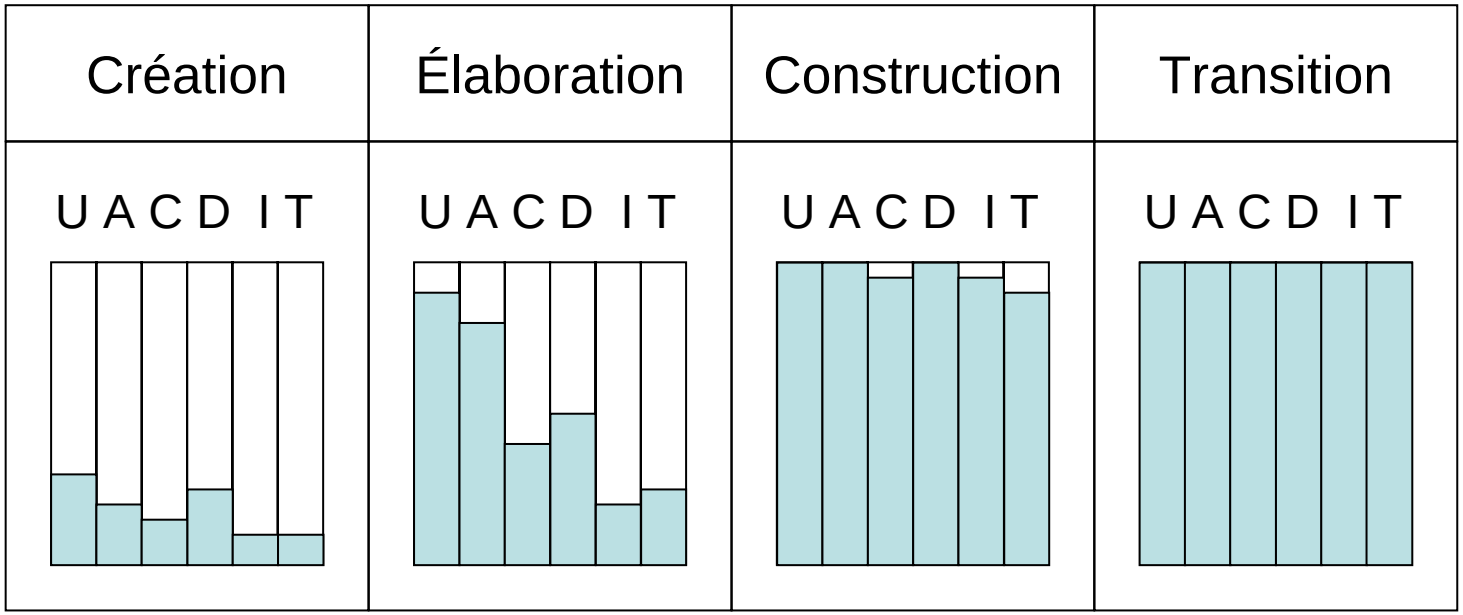
# Objectifs de la phase de transition

- *S'assurer que l'on dispose d'un produit prêt à être livré à l'ensemble des utilisateurs*
- *Former les utilisateurs*



# Les modèles évoluent à partir des itérations

*Incrément par incrément, les itérations construisent les modèles qui en résultent.*




Modèles:

U – Cas d'utilisation, A – Analyse, C – Conception,
D – Déploiement, I – Implémentation, T - Test

# Les itérations remettent en question l'organisation

*Les équipes de développement souvent ne résistent pas à la tentation de passer directement à l'écriture de lignes de code, facilement quantifiables par les chefs de projet…*
*Le passage à un développement itératif remet en question ces pratiques… L'attention doit, en effet, se détourner du nombre de lignes de code pour se porter sur la réduction de risques et la création de fonctions architecturales de référence.*