



Chapitre 4: Approches hybrides- Méthodes du processus unifié

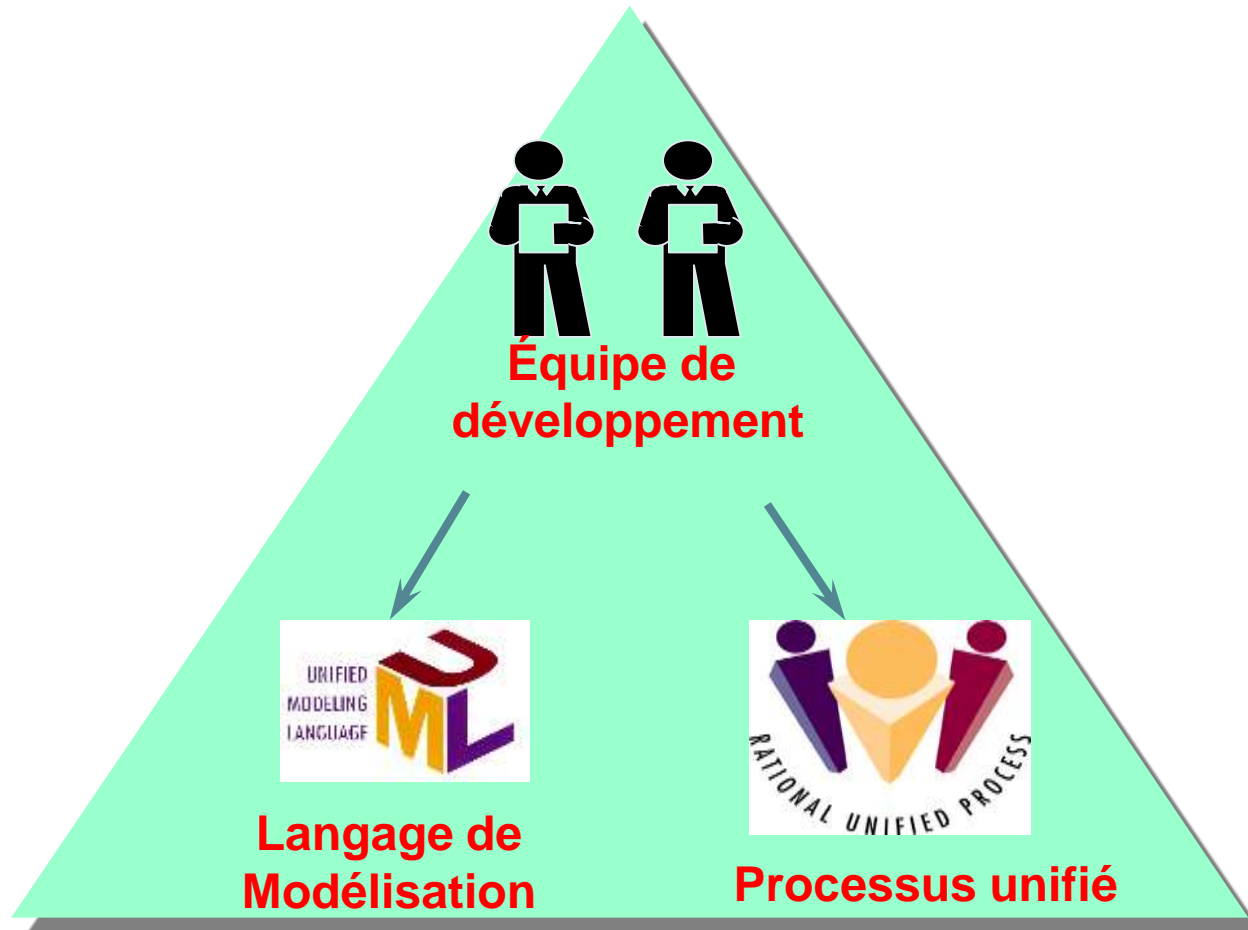


Le Processus Unifié

- ▶ Les auteurs principaux d'UML, I. Jacobson, G. Booch et J. Rumbaugh, proposent une démarche de développement pouvant être facilement associée à UML : **le Processus Unifié** (**Unified Software Development Process** - 1999).
- ▶ L'essentiel du Processus Unifié provient des méthodes **Objectory**, **Booch**, et **OMT**, auxquelles s'ajoutent quelques apports issus des travaux d'élaboration d'**UML**.
- ▶ **Processus générique**



Dans la construction d'un système, un langage ne suffit pas.



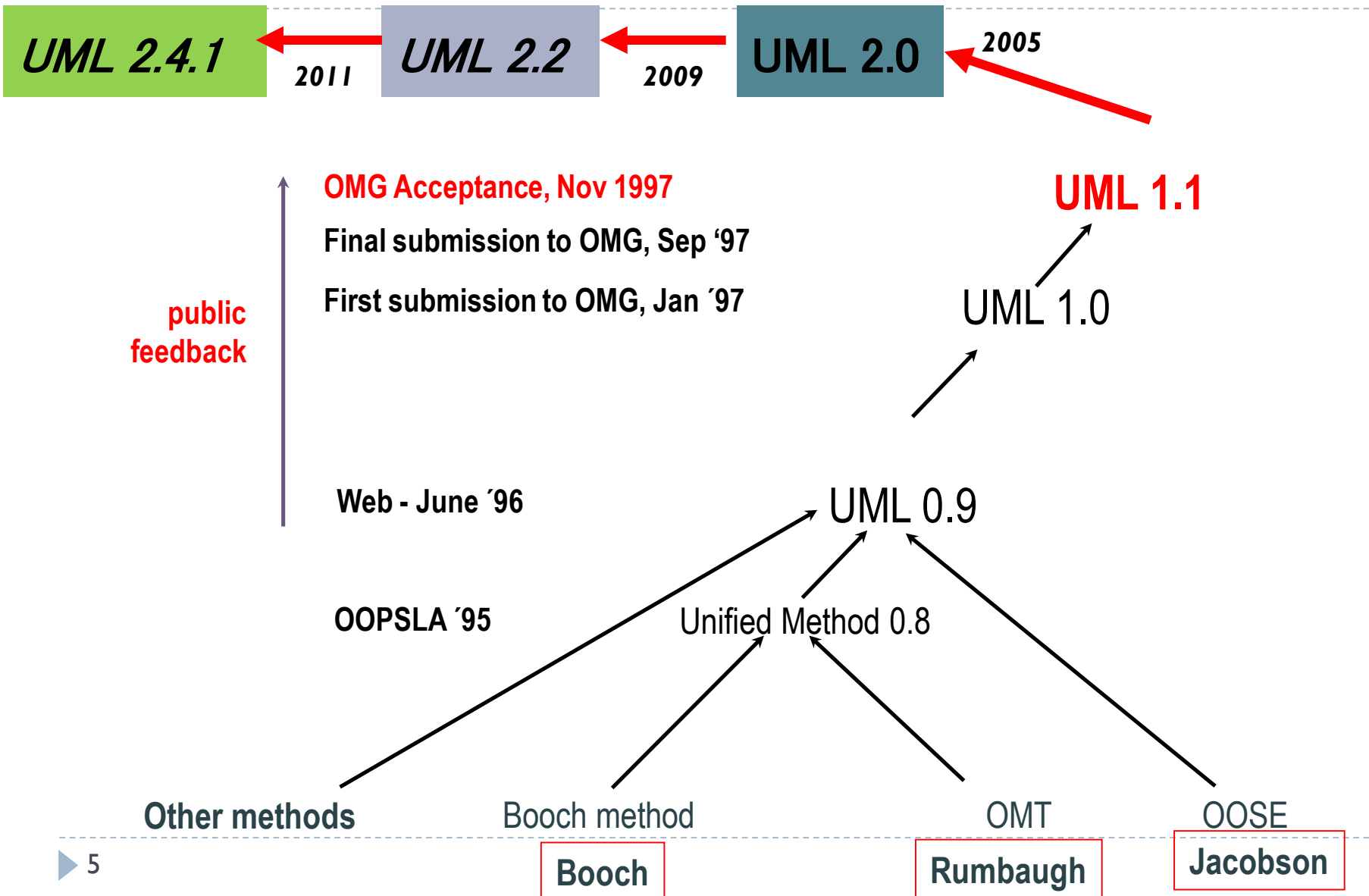
UML n'est pas un standard pour les processus de développement logiciel.



Le Processus Unifié

- **Le Processus Unifié ou UP (*Unified Process*) est une méthode générique de développement de logiciel développée par les concepteurs d'UML**
 - Générique signifie qu'il est nécessaire d'adapter UP au contexte du projet, de l'équipe, du domaine et/ou de l'organisation.
 - Il existe donc un certain nombre de méthodes issues de UP comme par exemple RUP (Rational Unified Process), 2TUP (Two Track Unified Process)

UML



UML

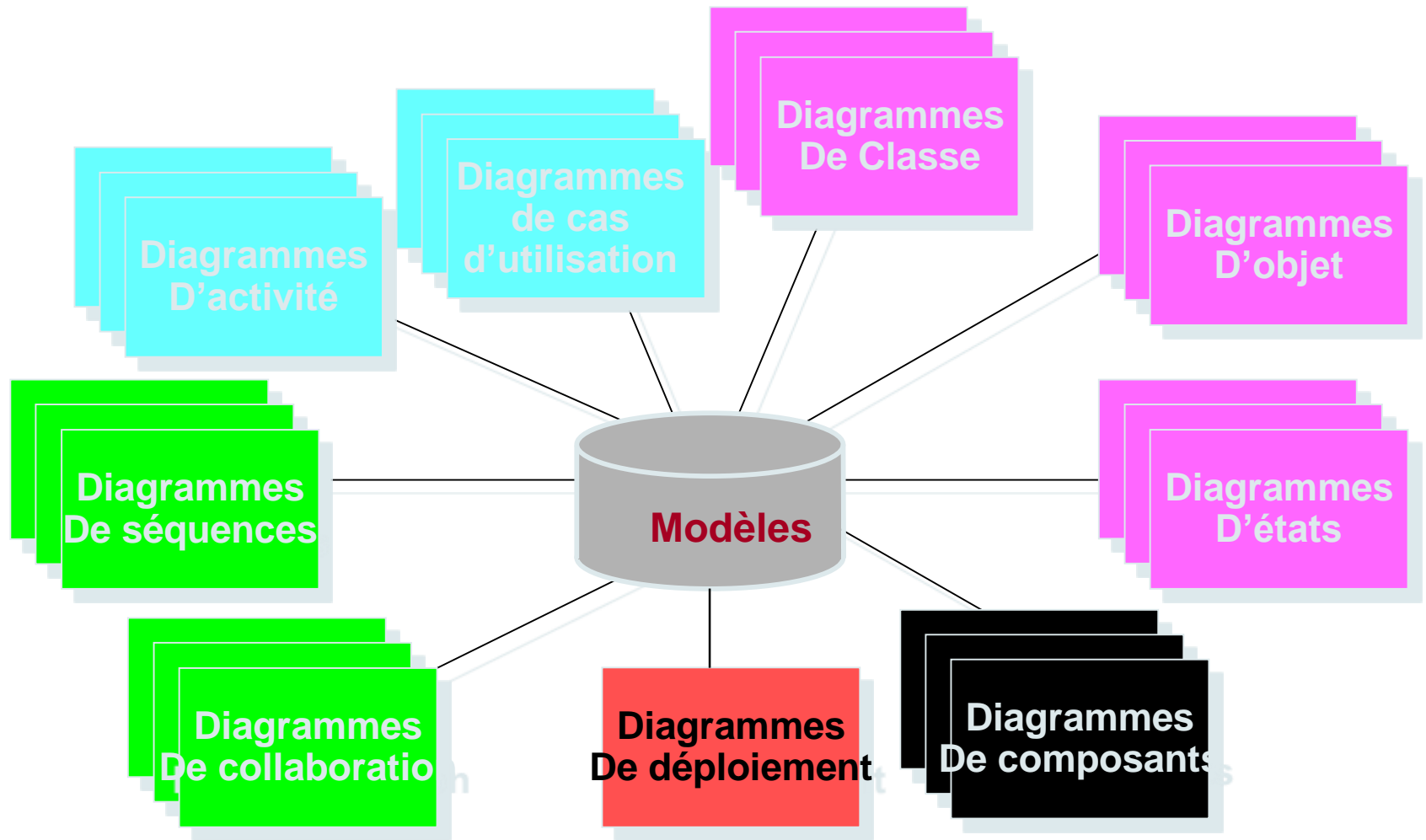
UML est aujourd'hui **le langage standard industriel de modélisation**. Son développement à été lancé par **trois leaders dans l'industrie de l'approche objet** :

- Grady Booch
- Ivar Jacobson
- Jim Rumbaugh.

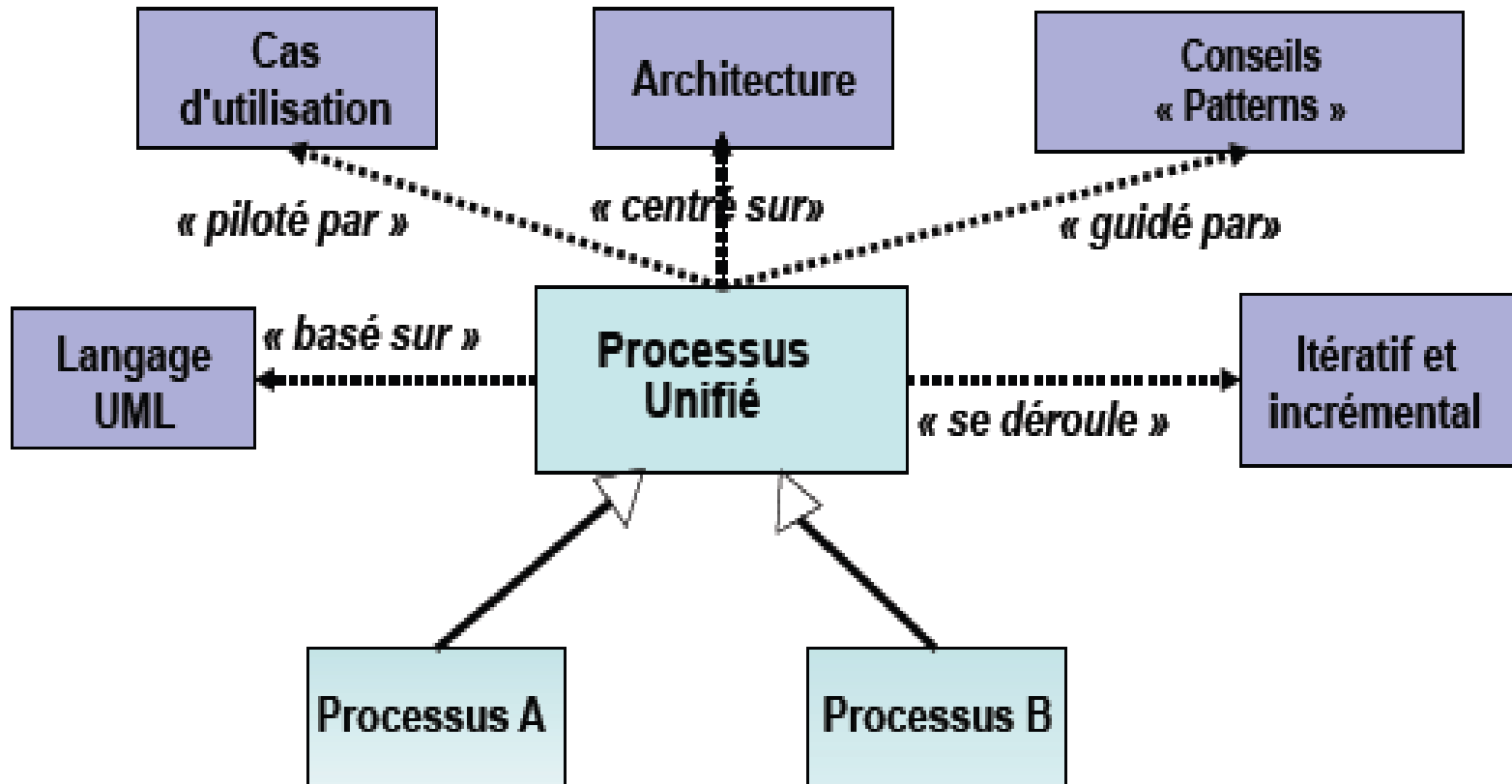
UML est en développement depuis 1990.



UML fournit des diagrammes standardisés



Processus Unifié



Processus Unifié

- ▶ **Guidé par les patterns**
- ▶ Les patterns (patrons): Modèle **générique** résolvant un problème classique et récurrent.
- ▶ Solution qui a fait ses preuves, fondée sur l'expérience.
- ▶ Solution **réutilisable** dans des contextes différents.

Processus Unifié

► Guidé par les patterns

La notion de patron (ou « pattern »)

Nom_du_Pattern

Un problème récurrent
de conception O.O.

Une solution exprimée
en UML

Les bénéfices

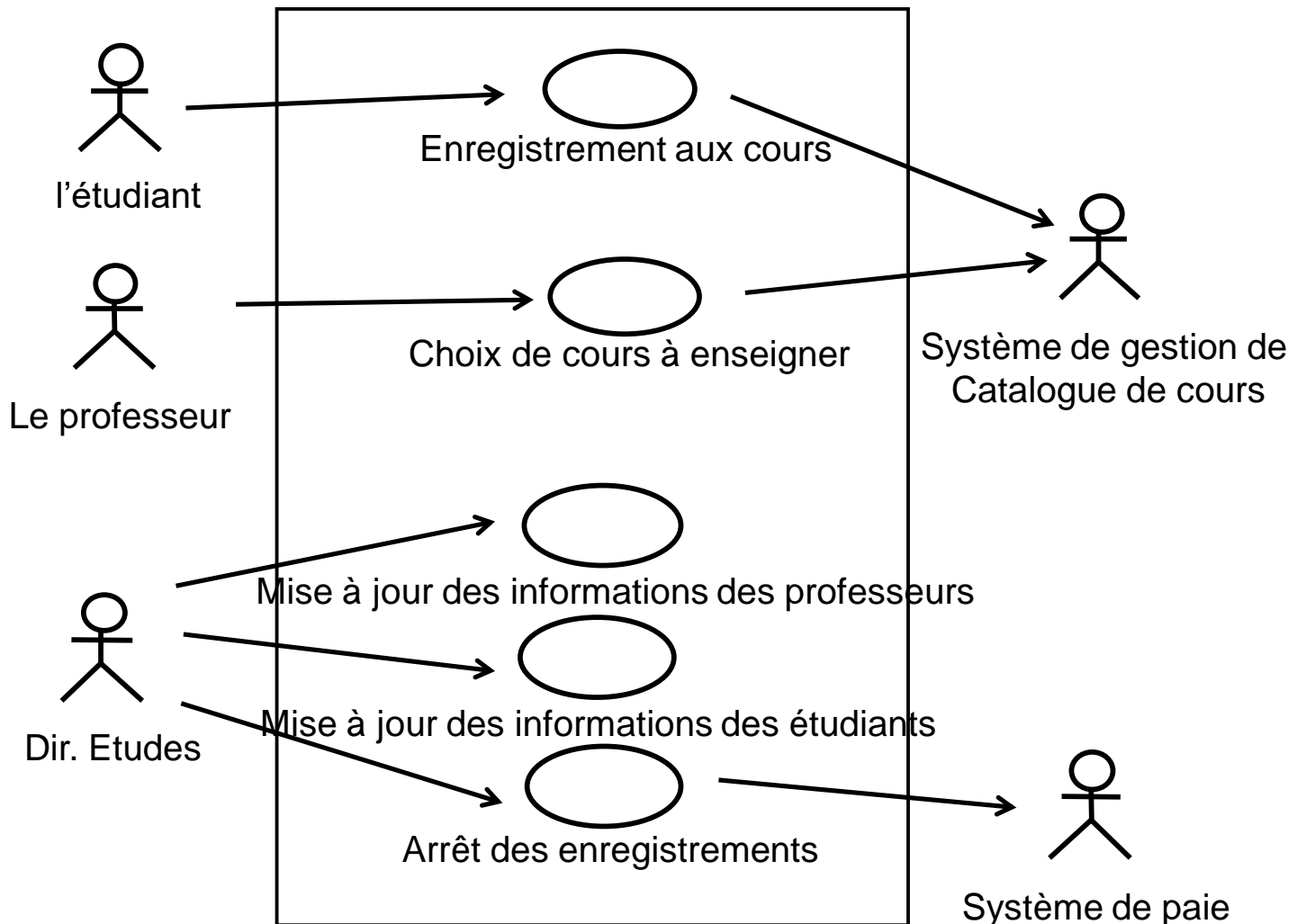
Processus Unifié

- ▶ **Piloté par les cas d'utilisation**
- ▶ **bien comprendre les besoins** de ses futurs utilisateurs.
 - ▶ 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.

Piloté par les cas d'utilisation

Un système d'enregistrement aux cours dans une université

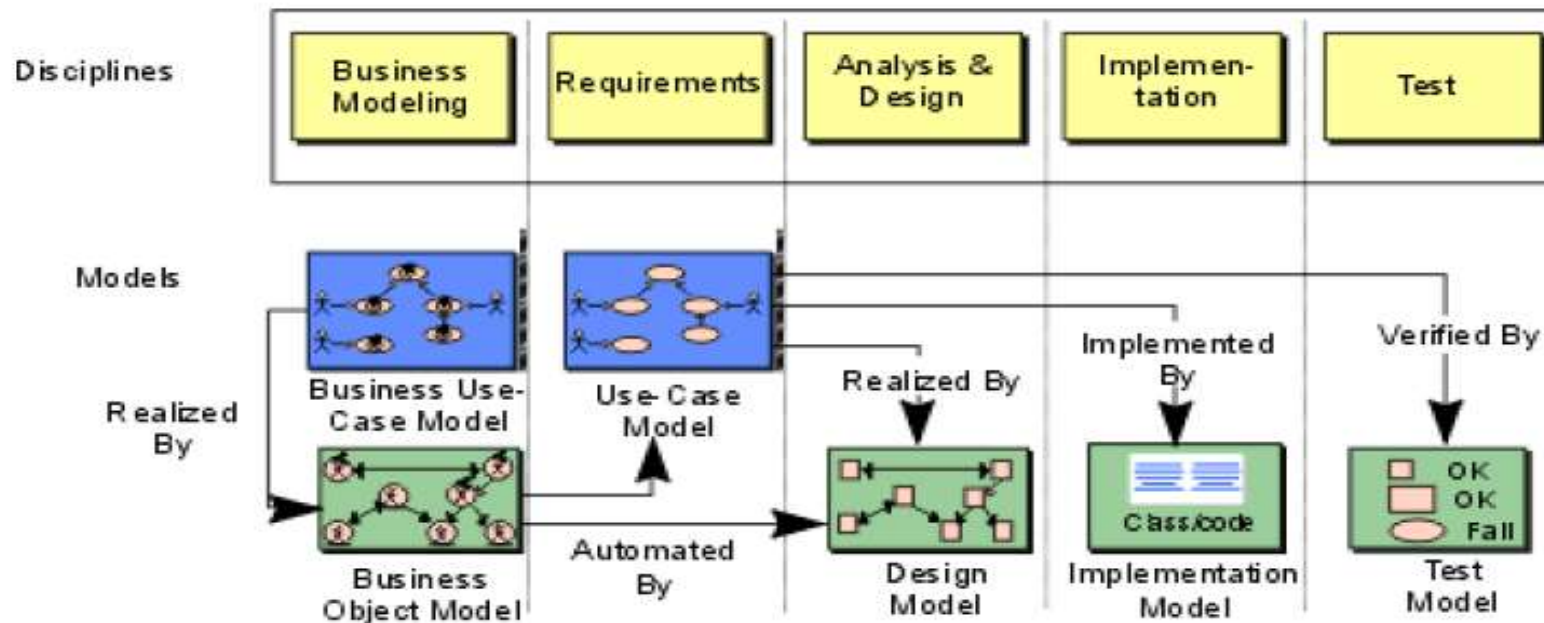




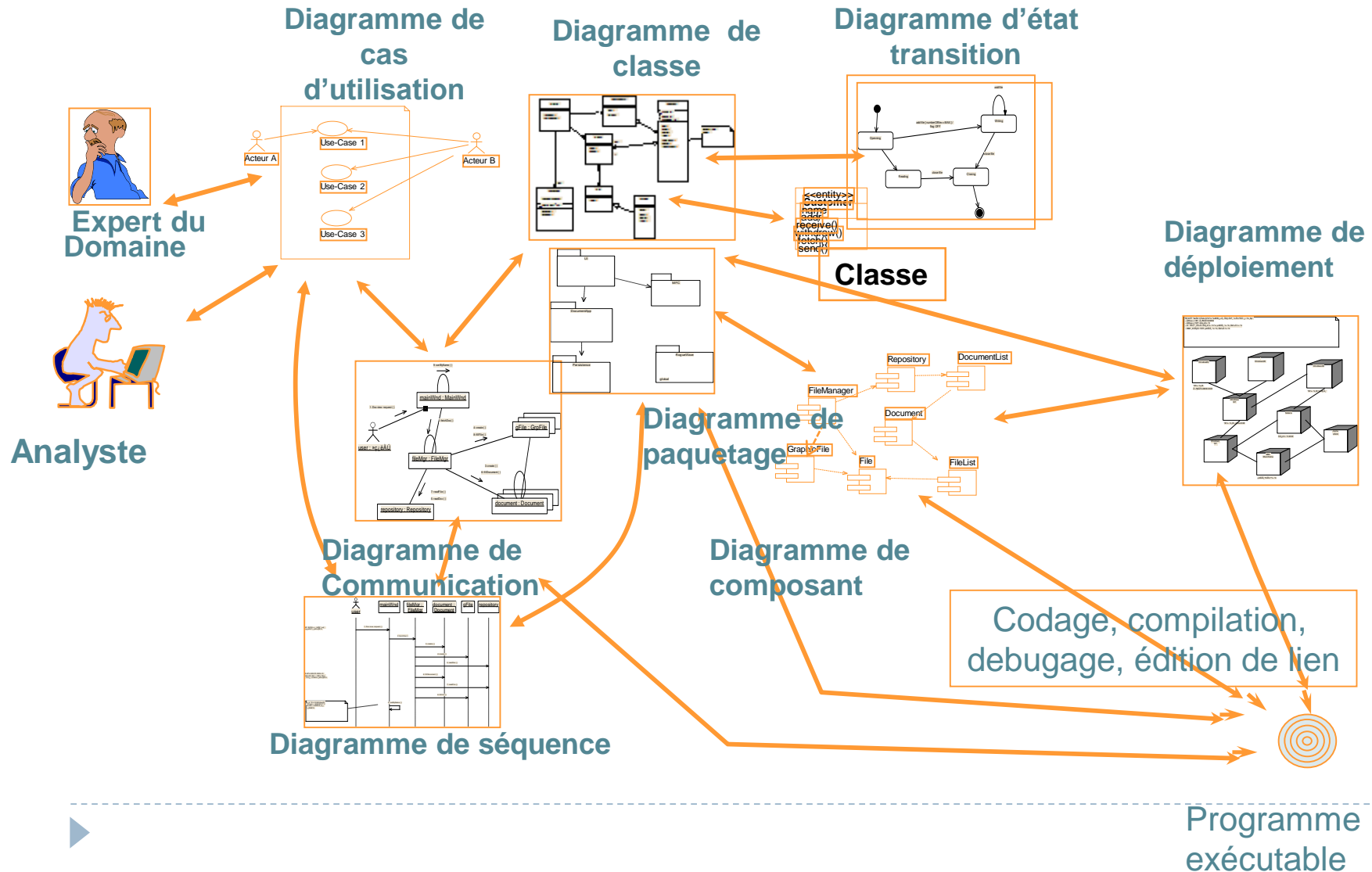
Processus Unifié

- ▶ Centré sur l'architecture (non sur le code)
- ▶ Orienté modèle

Les modèles au cœur du processus



Les diagrammes UML sont les artefacts clés de UP



Qu'est ce qu'un processus ?

Un processus définit **qui** fait quoi, **quand** et **comment** pour atteindre un **objectif donné**.

Le Processus Unifié de Rational est **un processus générique** qui utilise UML comme langage de modélisation.

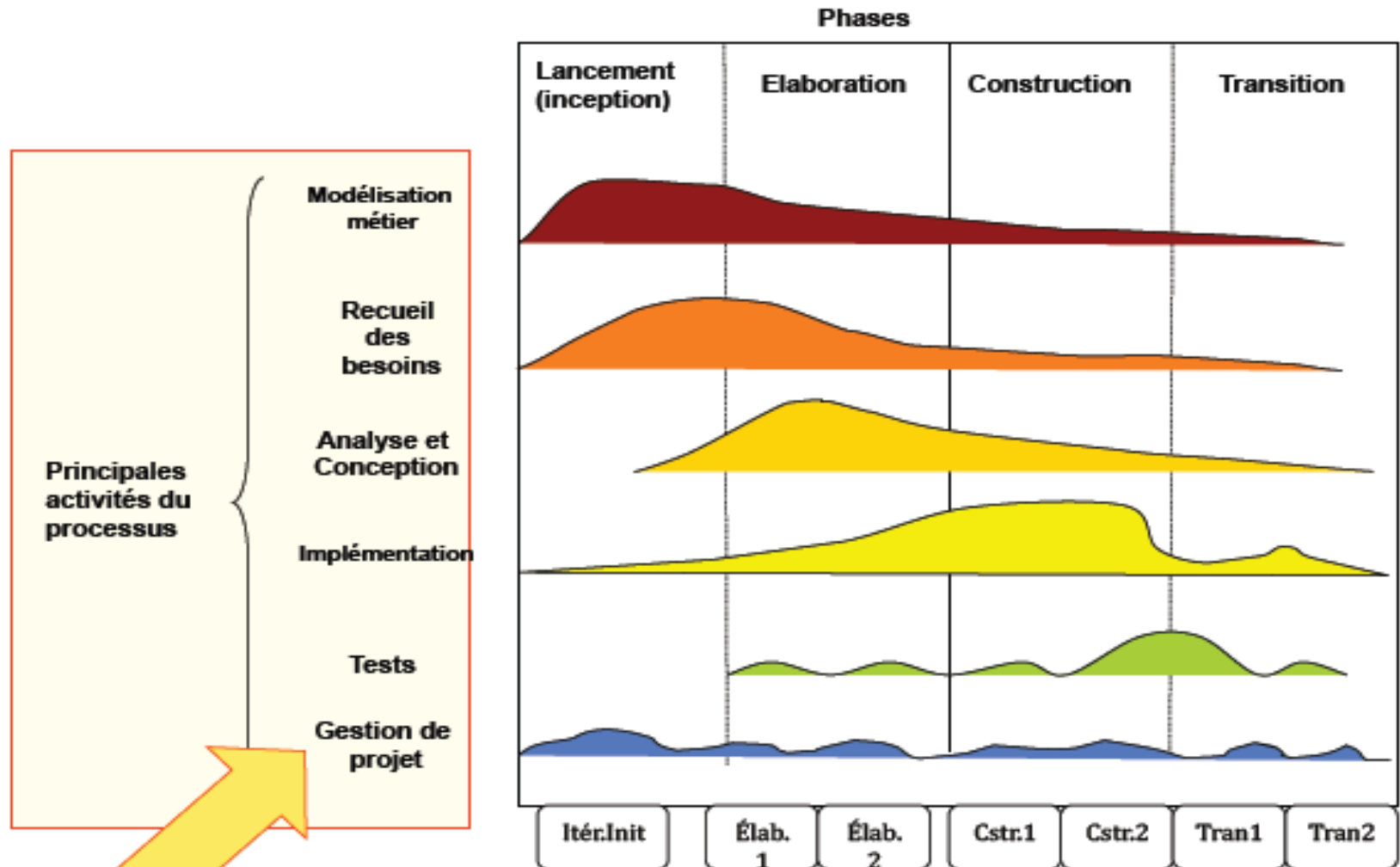


Processus unifié

- ▶ PU : processus unifié, désigne les préceptes généraux de la méthode.
- ▶ UP : *unified process*, la dénomination anglaise.
- ▶ USDP : *unified software development process*, autre dénomination courante.
- ▶ RUP : *rational unified process*, instantiation par Rational Software (IBM) des préceptes UP.
- ▶ EUP : *enterprise unified process*, instantiation de UP
- ▶ XUP : *extreme unified process*, instantiation hybride intégrant UP avec *extreme programming*.
- ▶ AUP : *agile unified process*, instantiation de la méthode mettant l'accent sur l'optimisation et l'efficacité sur le terrain plus que sur le modèle théorique.
- ▶ 2TUP : *two tracks unified process*, Instantiation de UP proposé par Valtech
- ▶ EssUP : *essential unified process*, par Ivar Jacobson, initiateur d'UML et RUP, entre autres. Ivar travaille aujourd'hui pour Microsoft afin d'intégrer EssUP aux outils de travail collaboratifs de la firme (*Visual Studio Team System*).

Méthode de développement logiciel: Rational Unified Process (RUP)

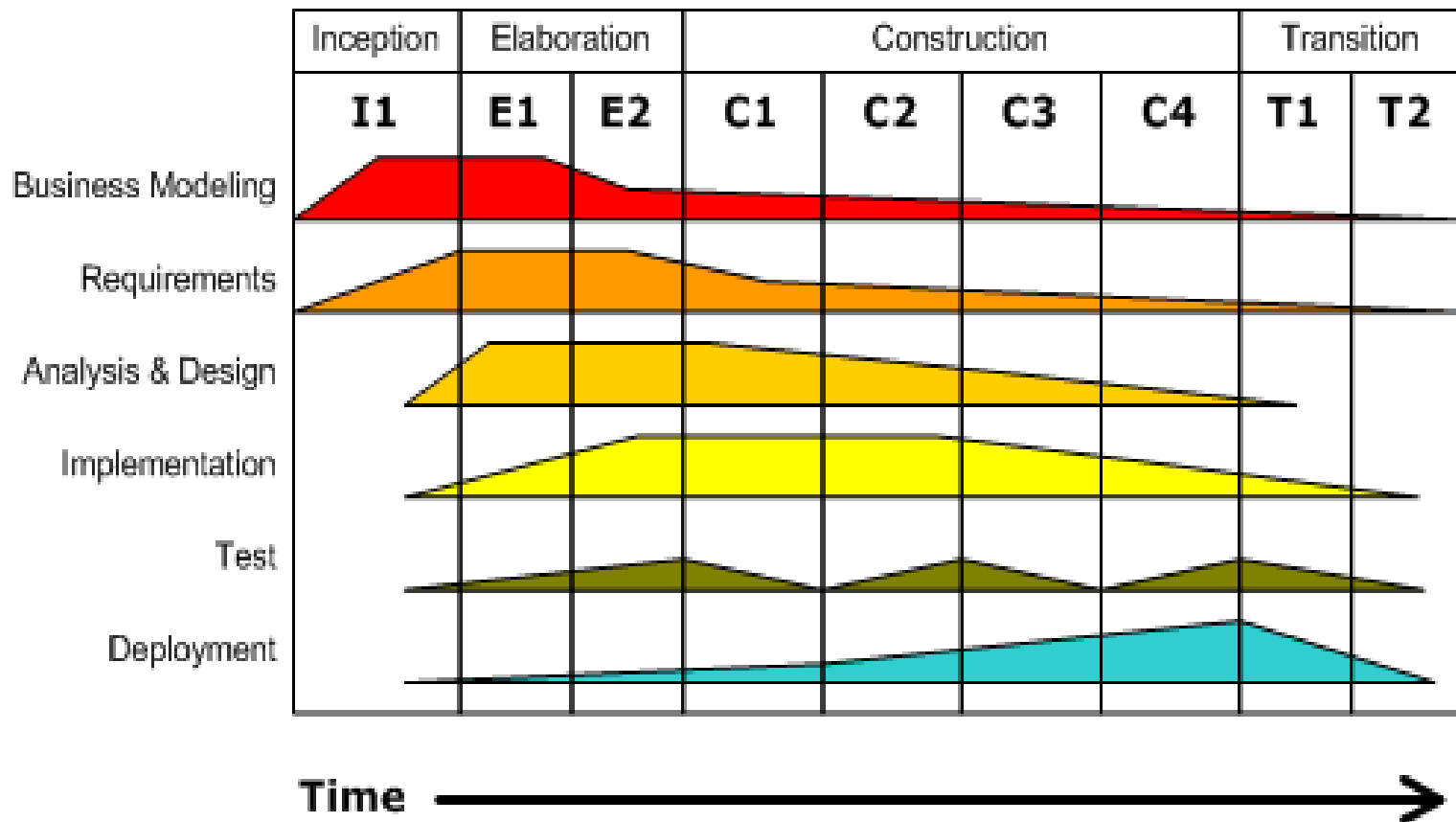
RUP: Matrice Disciplines (ou Activités) / Phases



A use-case model (at least 80% complete)

Iterative Development

Business value is delivered incrementally in time-boxed cross-discipline iterations.



A use-case model (at least 80% complete)

Le Processus Unifié

- ▶ **Phases:** Lancement, élaboration, construction, transition

- ▶ On fait de tout, tout le temps

- ▶ Ce qui distingue les phases: la prépondérance des activités

- ▶ **Disciplines ou Activités**

Chaque itération consiste à enchaîner les activités suivantes:

- ☐ Analyse des besoins

- ☐ Analyse

- ☐ Conception

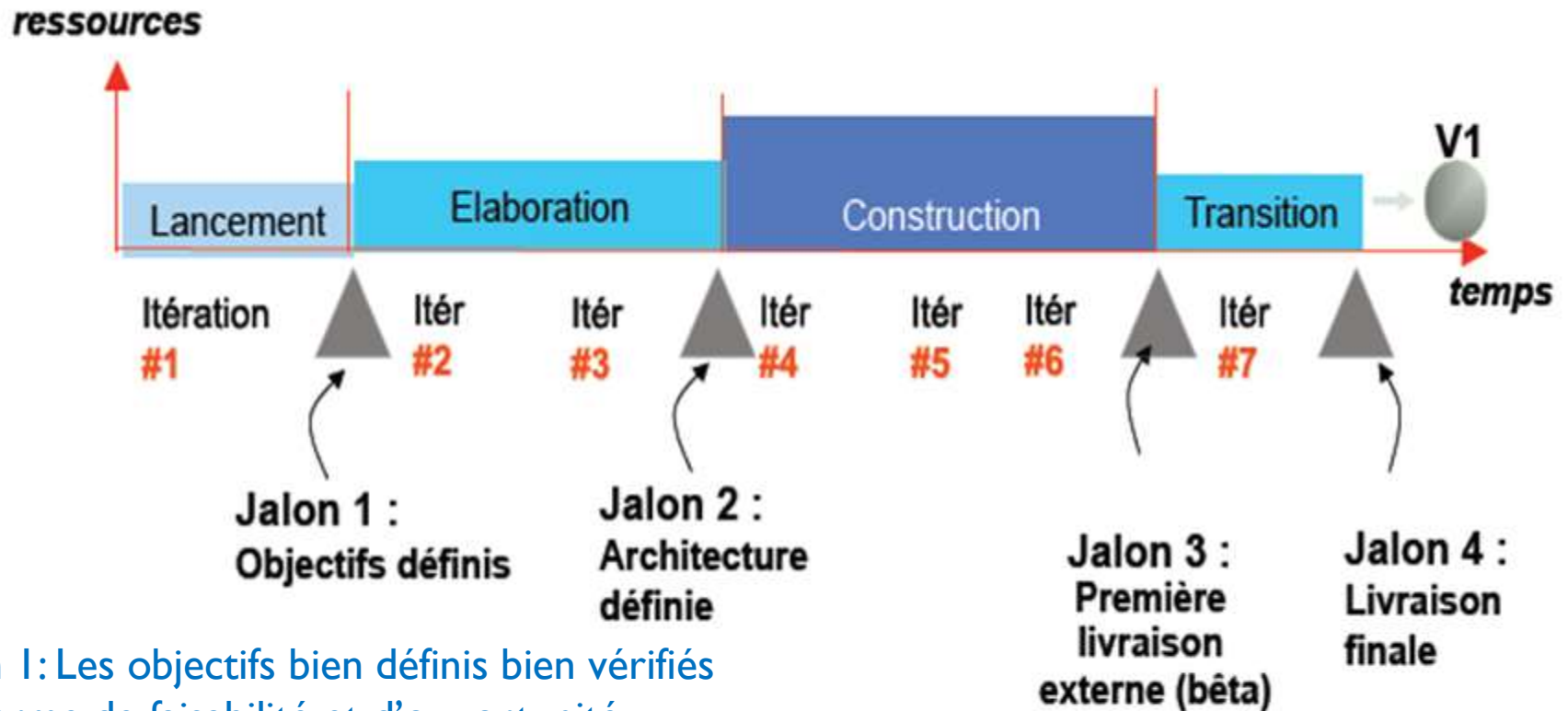
- ☐ Implémentation

- ☐ Tests

- ▶ **Itération**

Des itérations dans chaque phase

Phases, itérations, jalons



Jalon 1 : Les objectifs bien définis bien vérifiés en terme de faisabilité et d'opportunité

➡ Les Jalons correspondent à des étapes d'évaluation de la phase terminée, et de lancement de la phase suivante

RUP : Définitions et Notations

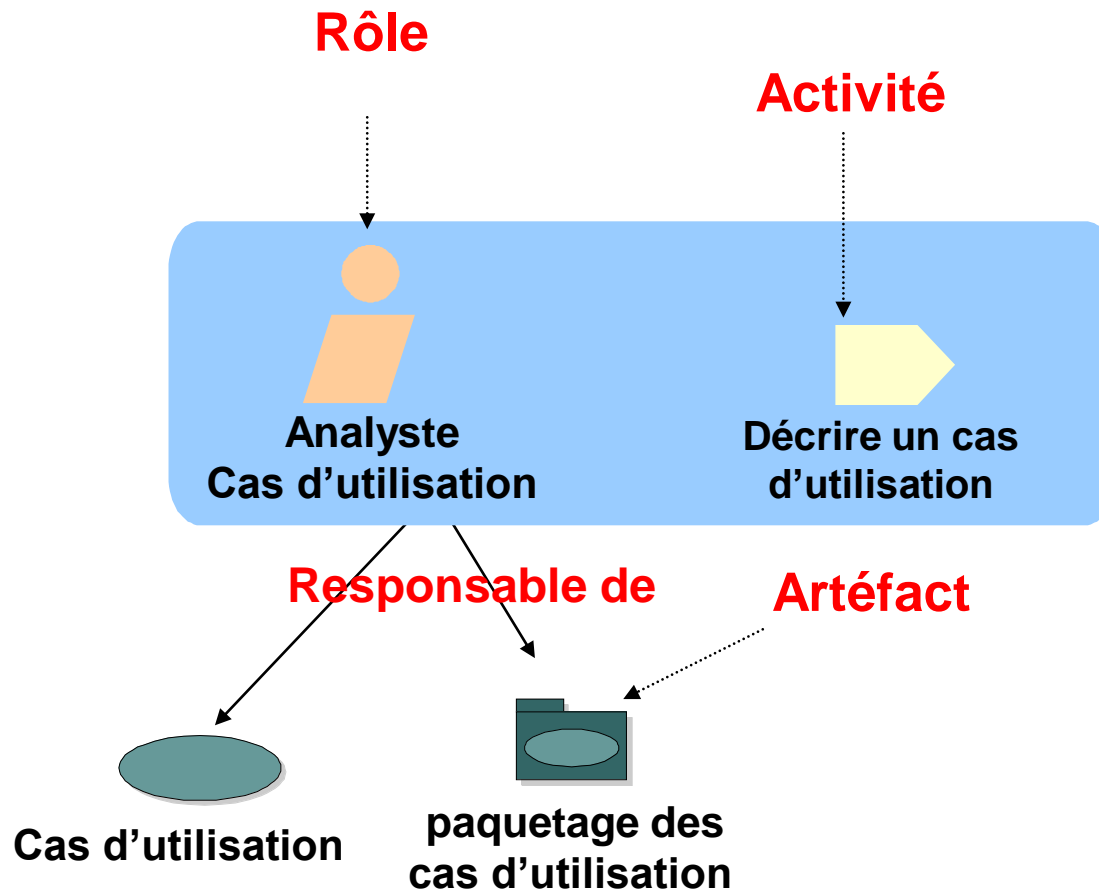
- ▶ **Artéfact** : Élément d'information, produit ou utilisé lors d'une activité de développement logiciel (modèle, source...)
- ▶ **Activité** : Opération exécutée au sein d'un état. Une activité peut être interrompue.
- ▶ **Rôle** : Comportement et responsabilités d'un ensemble de personnes.



-
- ▶ **RUP est un PRODUIT** (proposé par IBM sous forme d'un outil logiciel) contrairement aux autres méthodes.

-
- ▶ Le RUP de base définit neuf disciplines, plus de quarante rôles, et une centaine d'artefacts.

RUP : Définitions Et Notations





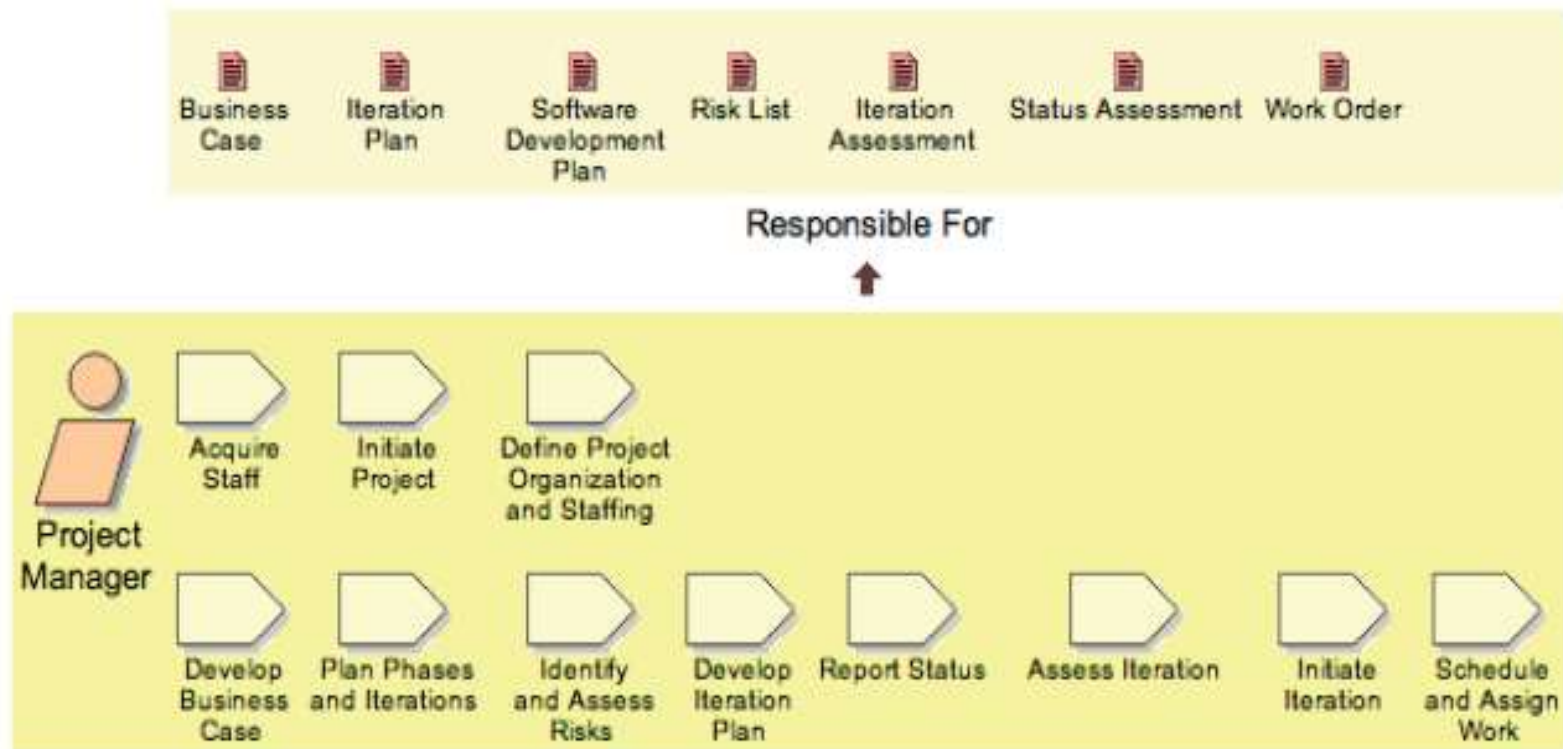
Les rôles

Le chef de projet




- Responsable de la gestion de projet
 - Plan de développement (plan de projet, plan d'itération, risques)
 - Choisir les artefacts UTILES
 - Collaborer avec tous les autres acteurs
 - Constamment focalisé sur les risques

Activités du chef de projet dans le RUP

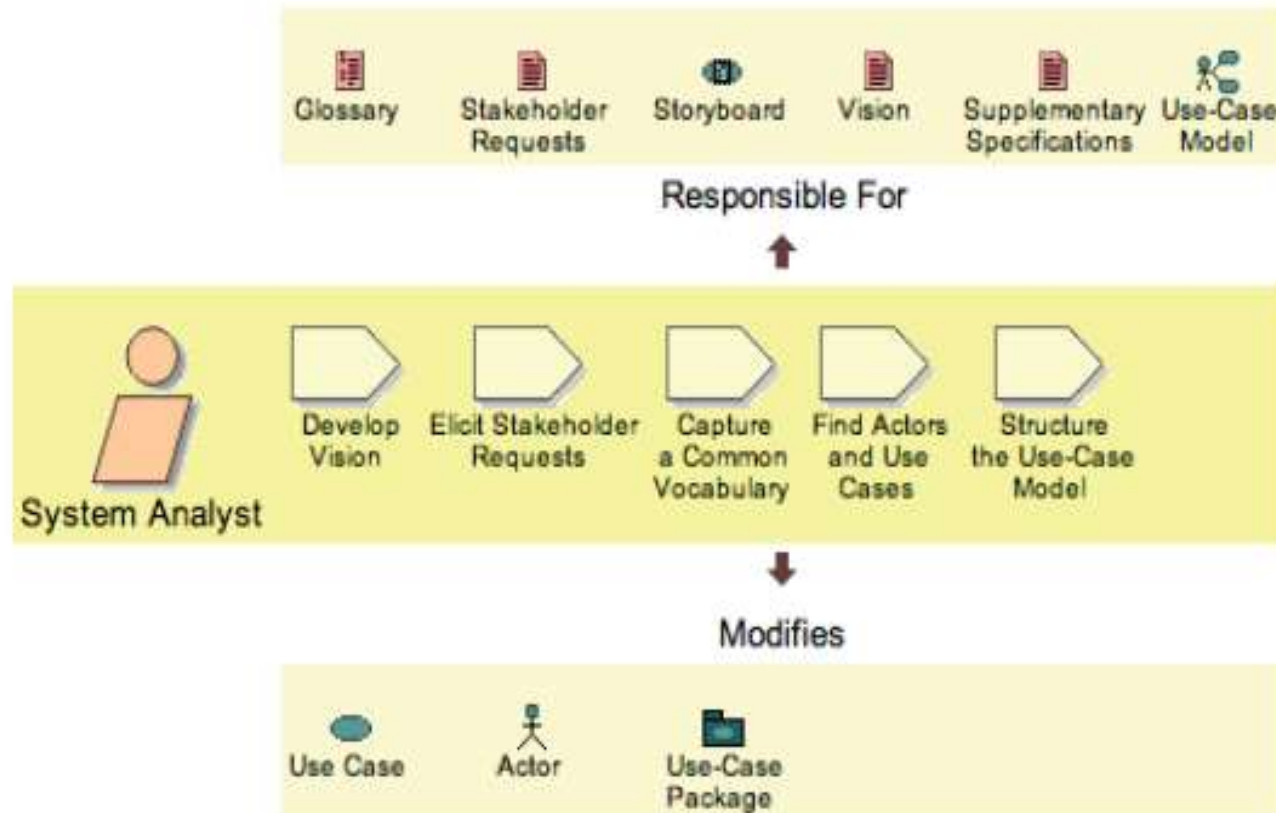


L'analyste




- Activités : Modélisation métier, Exigences
- Développer une Vision
- Liste des fonctionnalités
- Modèle de C.U., Glossaire, Prototype interface utilisateur, Spécifications Supplémentaires
- Vérifier que les exigences sont respectées et testées

Activités de l'analyste dans le RUP



L'architecte



- Communication entre les équipes de développement
- Choix d'une architecture
- Document d'architecture
- Validation par le prototype architectural

Prototype- Rappel

- ▶ Le concept de prototype permet de faire face aux risques et de lever l'incertitude sur : la viabilité commerciale du produit, son utilisation, son financement et la compréhension des exigences.
- ▶ Les types de prototypes: Les prototypes peuvent être définis selon leur finalité.
 - ▶ Les prototypes exploratoires qu'on jette une fois qu'ils ont rempli leur mission.
 - ▶ Les prototypes évolutifs qui évoluent d'une itération à l'autre pour devenir le système final.
 - ▶ Les prototypes comportementaux qui examinent un comportement spécifique du système vu de l'extérieur par l'utilisateur. Ici on ne se soucie pas de la qualité ni des normes du projet.
 - ▶ Les prototypes structurels qui explorent les aspects architecturaux et technologiques du système, vu de l'intérieur. Ce sont des prototypes en général évolutifs qui utilisent et testent l'infrastructure du système final, son ossature.

Activités de l'architecte dans le RUP


Implementation
Model


Reference
Architecture


Deployment
Model


Design Model


Software
Architecture
Document


Analysis
Model


Event


Interface

Responsible For



Software
Architect


Incorporate
Existing
Design Elements


Structure
the Implementation
Model

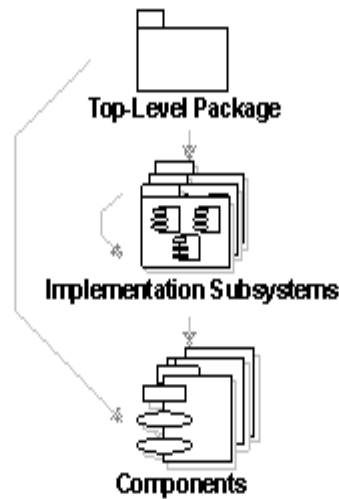

Architectural
Analysis


Prioritize
Use Cases


Identify
Design Elements

Modèle d'implémentation

- ▶ Le **modèle d'implémentation** est une collection de composants, et de sous-systèmes qui les contiennent.
- ▶ Les Composants incluent **des composants délivrables** comme les exécutables, ainsi que des composants à partir desquels les délivrables sont produits comme les fichiers de code source.



The notation in the implementation model. The arrows show possible ownership.

Le Développeur

- ▶ **Réalisation des Cas d'Utilisation**
- ▶ **Réalisation des Tests unitaires et Intégration**

Le testeur

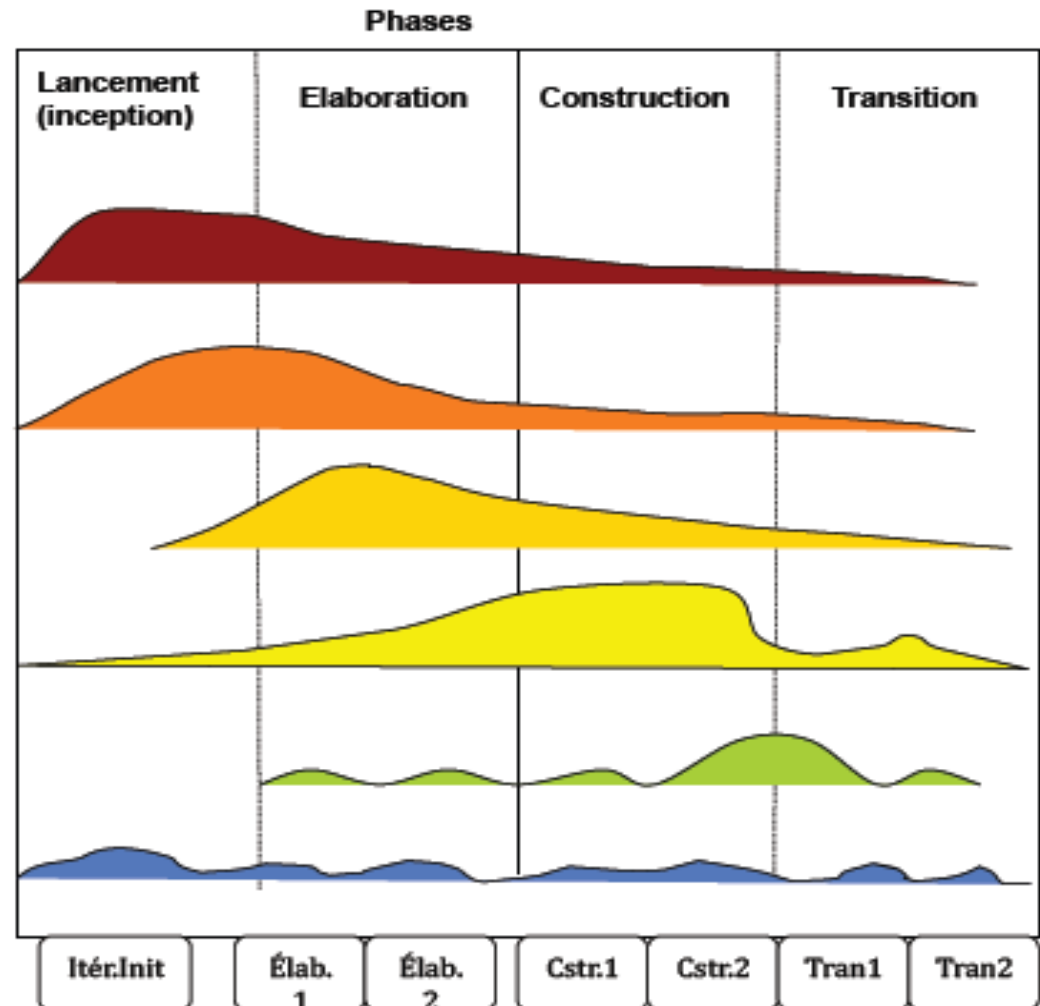
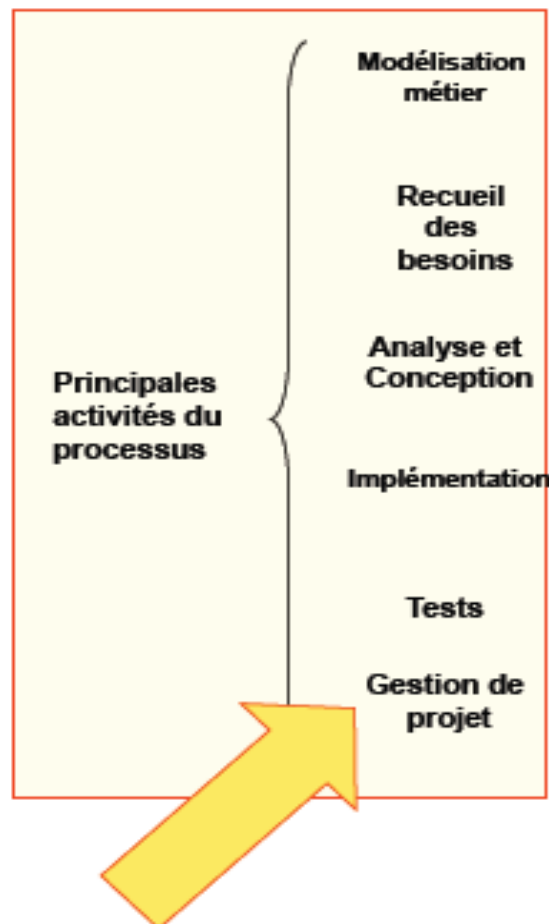


- Activités en fonction des phases :
 - Lancement : test des idées des technologies possibles
 - Élaboration : test architecture (notamment performance, évolutivité, etc.)
 - Construction : test fonctionnel, robustesse, ergonomie, etc.
 - Transition : test fonctionnel, robustesse, ergonomie, etc.



Activités

Activités



Gestion de projet

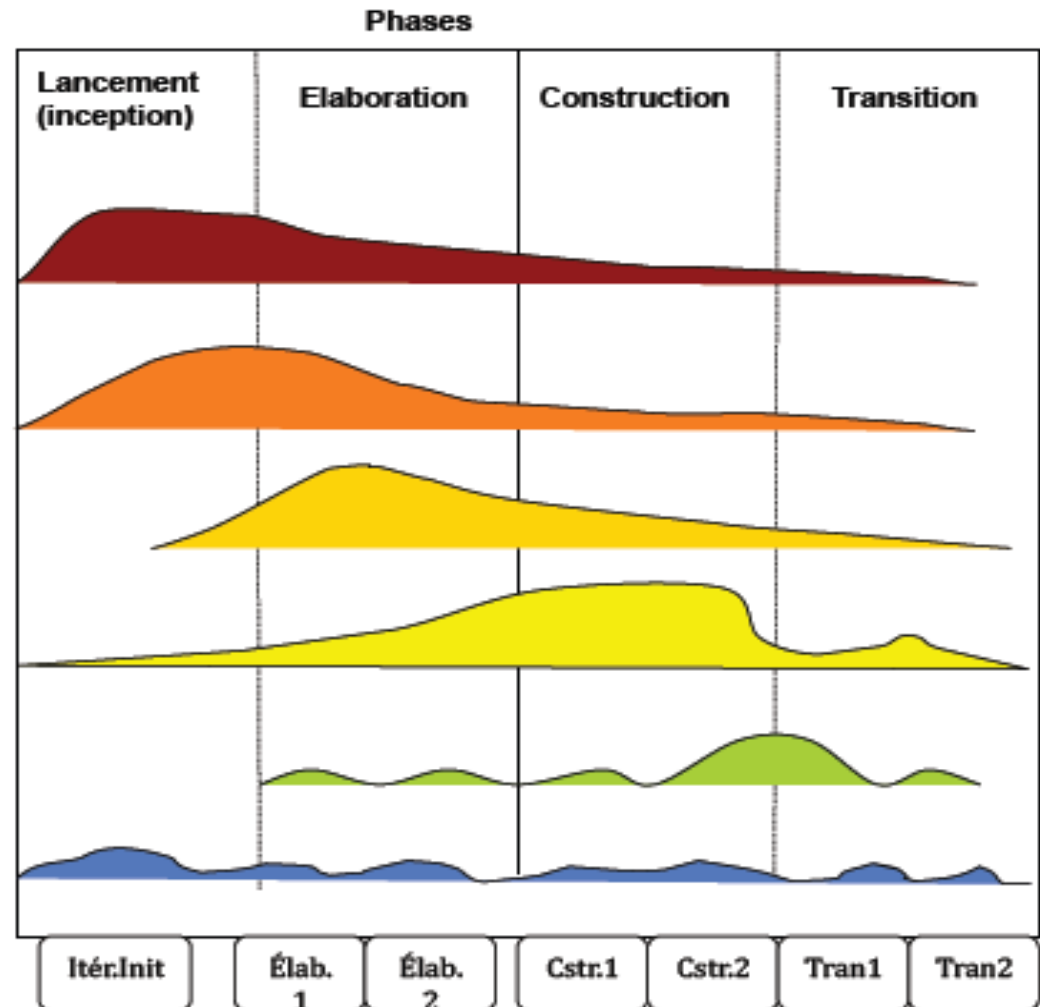
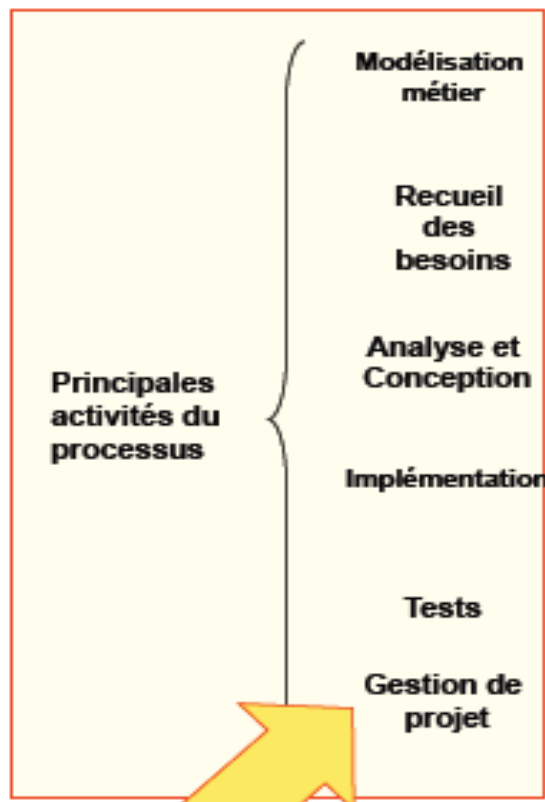
- Objectifs
 - Planifier/évaluer un projet itératif
 - Gérer les risques
 - Contrôler les progrès (délais, coûts, qualité, efforts, satisfaction client, productivité, etc.)

Gestion De Projet : artéfacts

- ▶ Liste des risques, plan de projet
- ▶ La planification des itérations
- ▶ L'estimation des itérations
- ▶ Evaluations des itérations, des phases et du projet



Activités (gr 2 B)

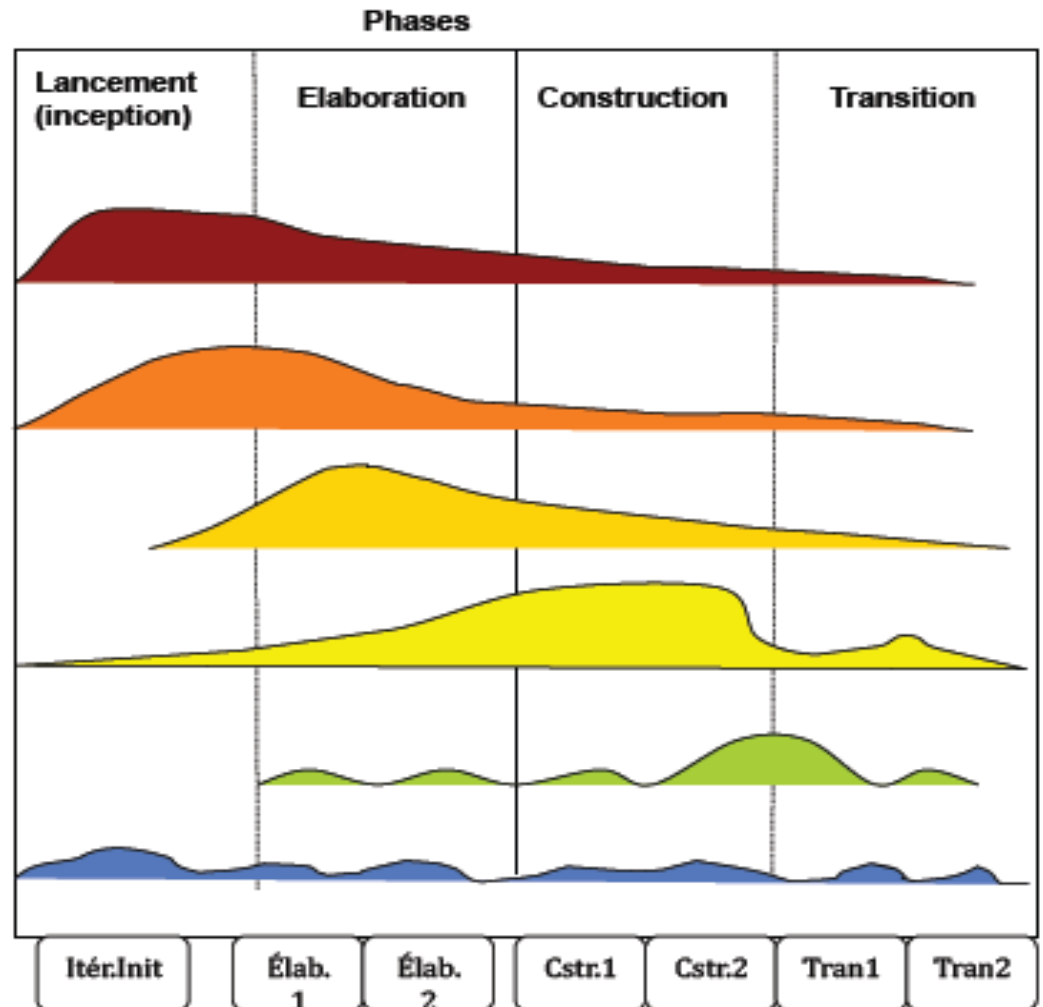
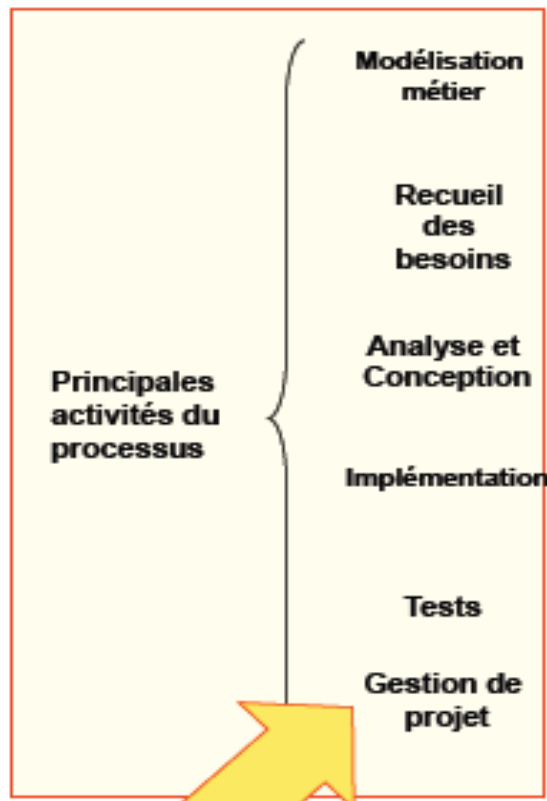


Modélisation métier

- Compréhension de la structure et la dynamique de l'organisation
- Comprendre les problèmes posés dans le contexte de l'organisation
- Conception d'un glossaire

- Vérifier que les clients, les utilisateurs finaux, et l'équipe ont une vision commune exacte de l'organisation.

Activités



Recueil des besoins

- L'une des étapes les plus importantes
 - déterminante pour la suite
- L'une des étapes les plus difficiles
 - intervenants multiples : client, utilisateurs, développeurs...
- Règle d'or à respecter : Les informaticiens sont aux services du client, pas l'inverse

Recueil et expression des besoins

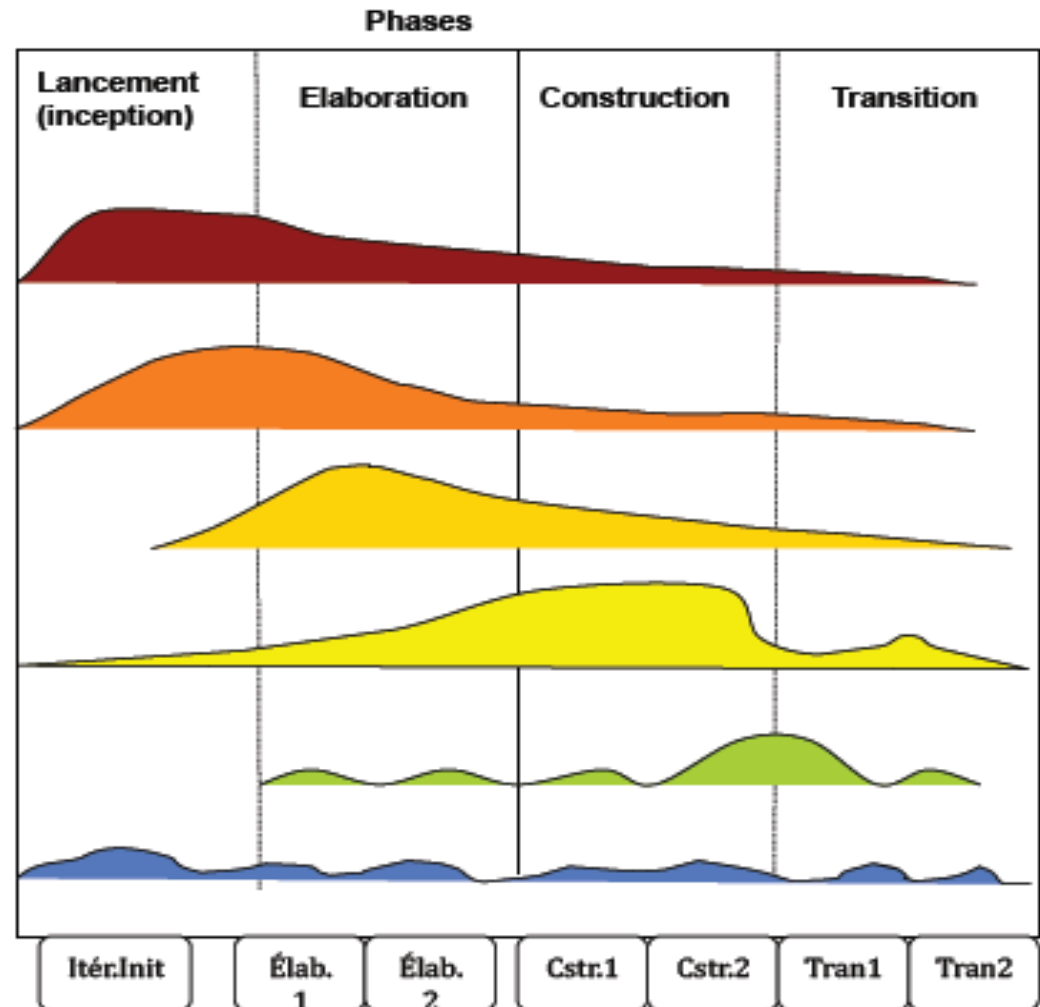
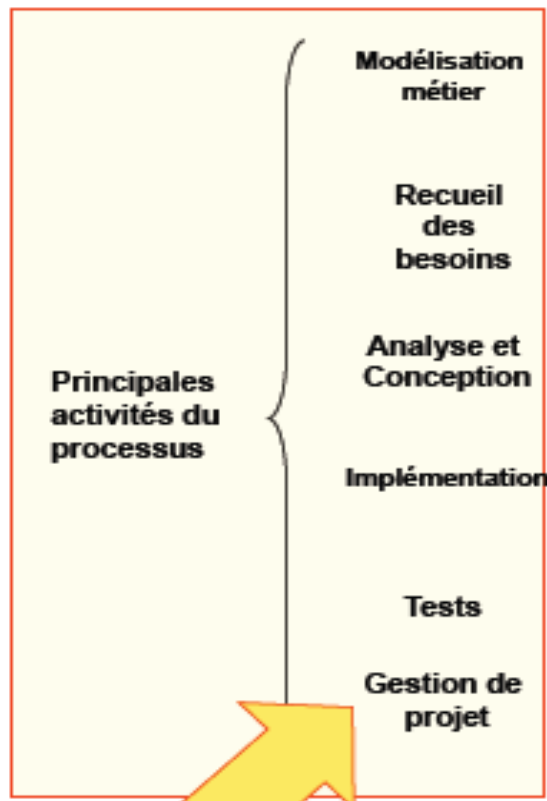
- Auprès des clients et parties prenantes du projet
- Ce que le système doit faire
- Expression des besoins dans les cas d'utilisation (exigences fonctionnelles)
- Spécification des cas d'utilisation en scénarios
- Limites fonctionnelles du projet et exigences non fonctionnelles (utilisabilité, fiabilité, performances)
- Ebauche de l'IHM (prototypage et validation par l'utilisateur)
- Etablir des priorités entre les cas d'utilisation.

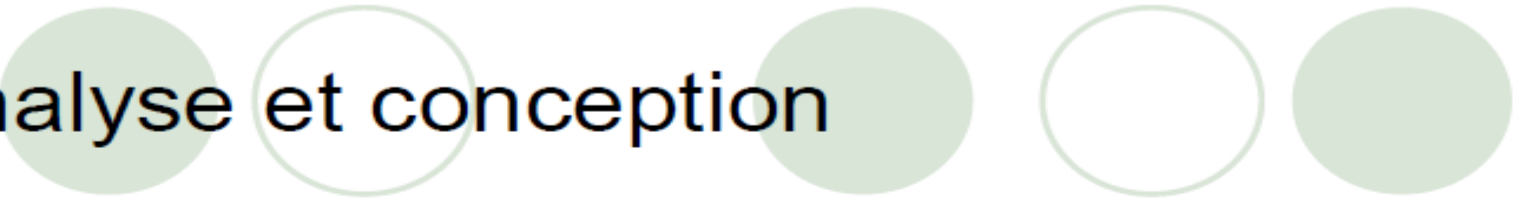
Exemple

Se servir des cas d'utilisation pour définir les itérations :

Cas d'utilisation	Risque	Priorité	Itération
Traiter une commande	Moyen	Moyenne	5
Gérer les infos clients	Bas	Moyenne	6
Consulter les en-cours	Haut	Moyenne	3
Gérer la facturation	Bas	Basse	8
Suivre les règlements	Bas	Basse	8
Planifier une mission	Haut	Haute	2
Suivre une mission	Moyen	Haute	4
Réaliser l'inventaire	Bas	Moyenne	7
Manipuler les colis	Bas	Moyenne	7
Définir le plan de transport	Haut	Haute	1
Gérer les ressources	Moyen	Haute	3
Gérer les profils	Bas	Moyenne	9
S'authentifier	Bas	Moyenne	9

Activités

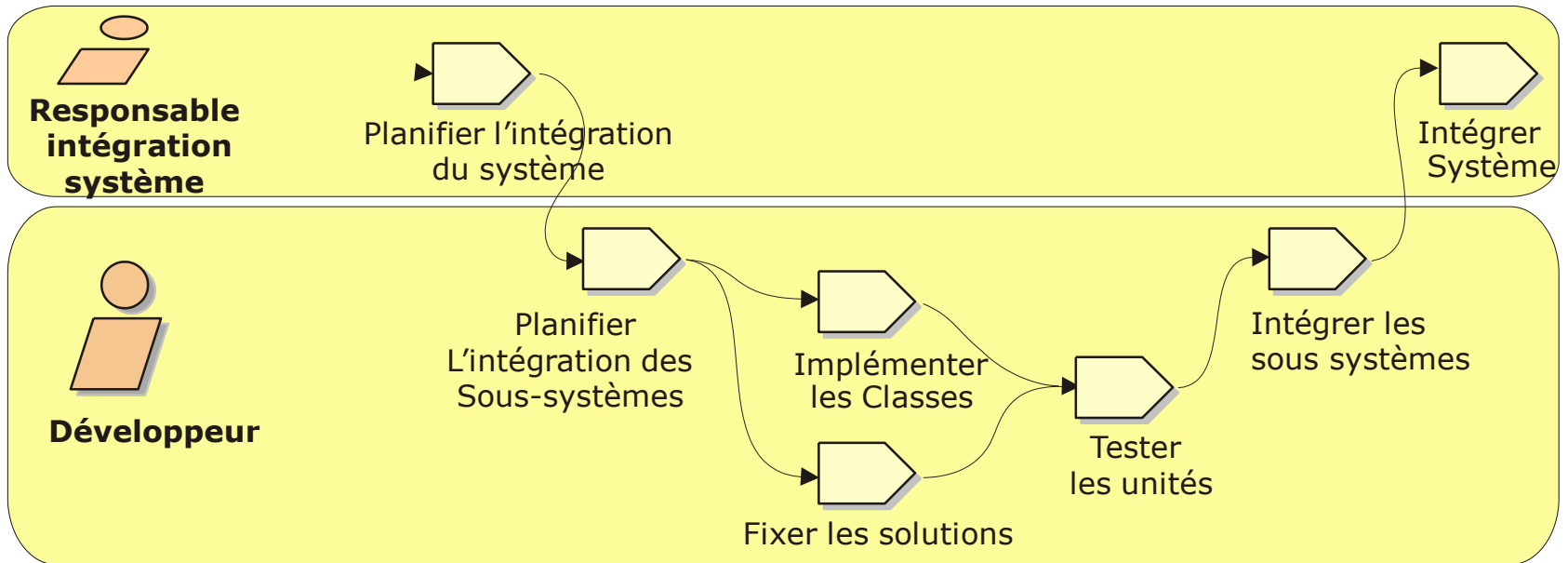




Analyse et conception

- Transformation des besoins utilisateurs en modèles UML
- Unified Modeling Language (pas une méthode mais un langage)

Implémentation



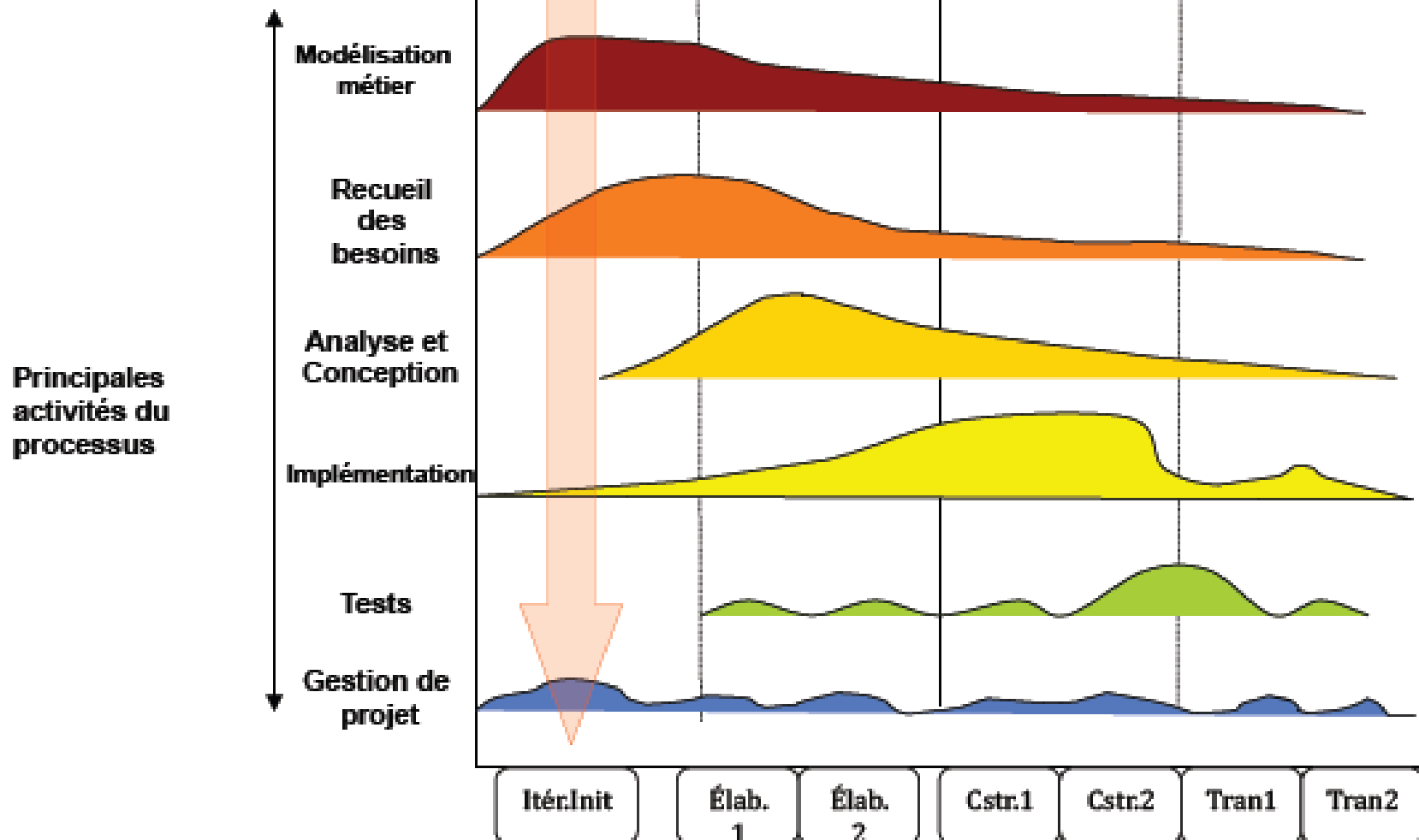
Tests

- Etapes (unitaire, d'intégration, système, acceptation)
- Types :
 - De « benchmark » (comparaison)
 - De configuration (différentes config matérielles et logicielles)
 - De fonctionnement (vérification des CU)
 - D'installation
 - D'intégrité (fiabilité, robustesse, résistance)
 - De charge (conditions opérationnelles plus lourdes = nb utilisateurs, transactions,...)
 - De performance (en modifiant les configurations)
 - De stress (conditions anormales opérationnelles)

Enchaînement des phases et des itérations (gr 1 A)

Phase de lancement

Phases




Phase de lancement

- Comprendre les objectifs et la portée du projet
- Objectifs :
 1. Comprendre le système à construire
 2. Identifier la fonctionnalité principale du système
 3. Déterminer au moins une solution possible
 4. Décider du processus à appliquer et des outils à utiliser

Nb : en général une seule itération, sauf pour projets importants, innovants, ou très risqués

Objectif 1 : Comprendre le système

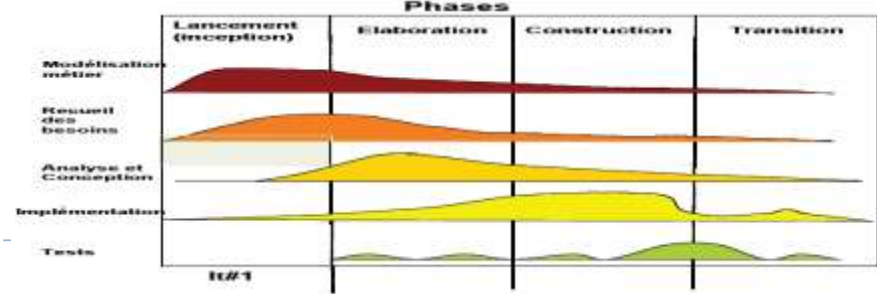
- Produire une « vision » (artefact:document de vision), i.e. opportunités apportées par l'application, les utilisateurs cibles, quelques cas d'utilisation clés (un ou deux), certains besoins non fonctionnels (artefact:spécifications supplémentaires)
- Générer une description large et superficielle (brainstorming, identification des acteurs et des C.U. principaux (artefact: modèle des C.U.), décrire les C.U., créer un artefact:glossaire développer des proto d'interface jetables)



Les artefacts

- Éléments produits lors du développement (documents, modèles, fichiers compilés, composants, etc.)

I. Comprendre le système



- **Modélisation métier:** Compréhension du métier à travers un *document de Vision*.
 - Activités
 - Acteurs
 - Opportunité: **le business Case** (*Décrire le business Case: Justification, Retour sur investissement, Opportunité, Faisabilité*)
- **Recueillir les besoins de l'utilisateur:**
 - Spécifier un ou deux cas d'utilisation clés
 - Portée ou périmètre fonctionnel



Objectif 2 : Identifier la fonctionnalité essentielle du système

- Collaboration chef de projet, architecte, et client (artefact: demandes des intervenants) pour déterminer les C.U. les plus critiques
 - La fonctionnalité est le noyau de l'application
 - Elle DOIT être livrée

Objectif 3 : Déterminer au moins une solution possible

- Architecture (client-serveur, centralisée, distribuée, etc.)
 - Technologies utilisées (éventuellement faire des tests d'implémentation pour estimer les risques liés à une technologie)
- Il est possible de réaliser des prototypes (jetables) pour évaluer le risque de faisabilité de la solution

Objectif 4 : Comprendre les coûts, les délais et les risques

- Examiner la faisabilité du projet (étude de rentabilité, artefact: liste des risques)
- Établir le plan du projet (artefact : plan de développement du logiciel)

- ▶ Décider de la faisabilité du projet
- ▶ Lancer le projet?

Objectif 5 : Décider du processus à appliquer et des outils à utiliser

- Faire des choix :
 - Processus de développement
 - Outils de développement
 - Compléter le plan de développement avec les choix

Revue de projet : Jalon fin de lancement

- Les différents intervenants valident:
 - le périmètre du projet, coûts et délais
 - la liste des exigences
- Les risques initiaux sont identifiés et il existe une stratégie de réduction pour chacun d'eux
- L'ensemble des artefacts produit est complet, à jour et cohérent



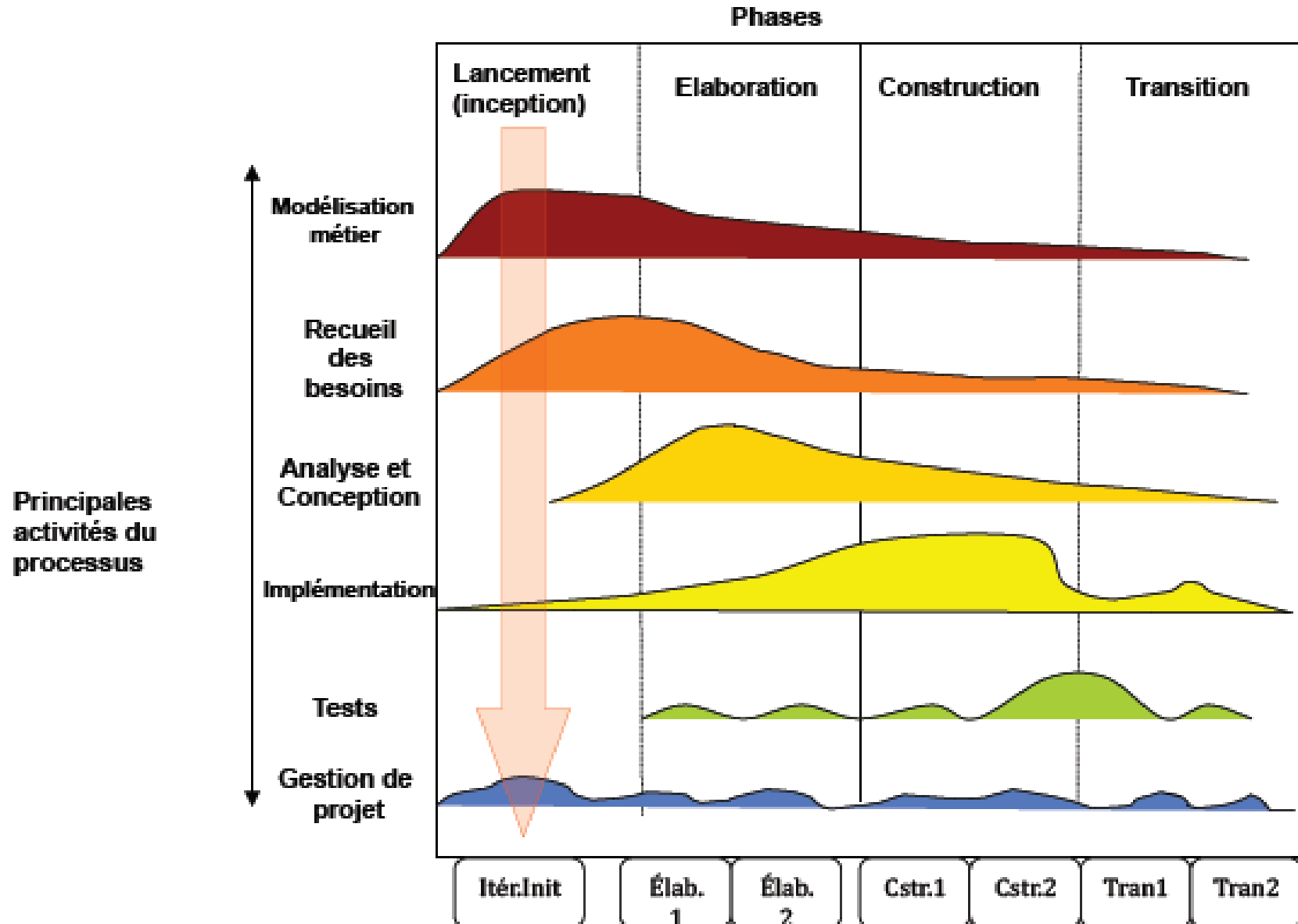
Exemple de Document de Vision

Table des matières

1.Introduction	3
1.1 Domaine d'application	3
1.2 Terminologie utilisée	3
1.3 Références	3
2.Positionnement	3
2.1 Opportunité d'affaire	3
2.2 Compte rendu des problèmes	3
3. Description des acteurs: utilisateurs et intervenants	3
4.Vue d'ensemble du produit	3
4.1 Perspective du produit	3
4.2 Principaux avantages	3
4.3 Coûts et prix prévus	
5. Caractéristiques du produit	3
6.Contraintes	3
7.Autres exigences du produit	3
7.1 Standards applicables	3
7.2 Exigences de qualité	3
7.3 Exigences en performance	3
7.4 Exigences environnementaux	3
8.Exigences en documentation	3
8.1 Manuel de l'utilisateur	3
8.2 Aide en-ligne	3
8.3 Guides d'installation, de configuration, fichier «Read Me»	3
9.Risques	3

Exercice : Lister ci dessous l'ensemble des artefacts produits pendant la phase de lancement

Phase d'élaboration



Phase d'élaboration

1. Spécification détaillée du produit: **80% des cas d'utilisation**

- ▶ Explorer les scénarios **avec les utilisateurs et les experts du domaine** afin de réduire les risques d'incompréhension.

2. Mise à jour des estimations (charge, coûts)

3. Planification détaillée des activités et détermination des ressources.

4. La liste des risques est revue



Exemple

Se servir des cas d'utilisation pour définir les itérations :

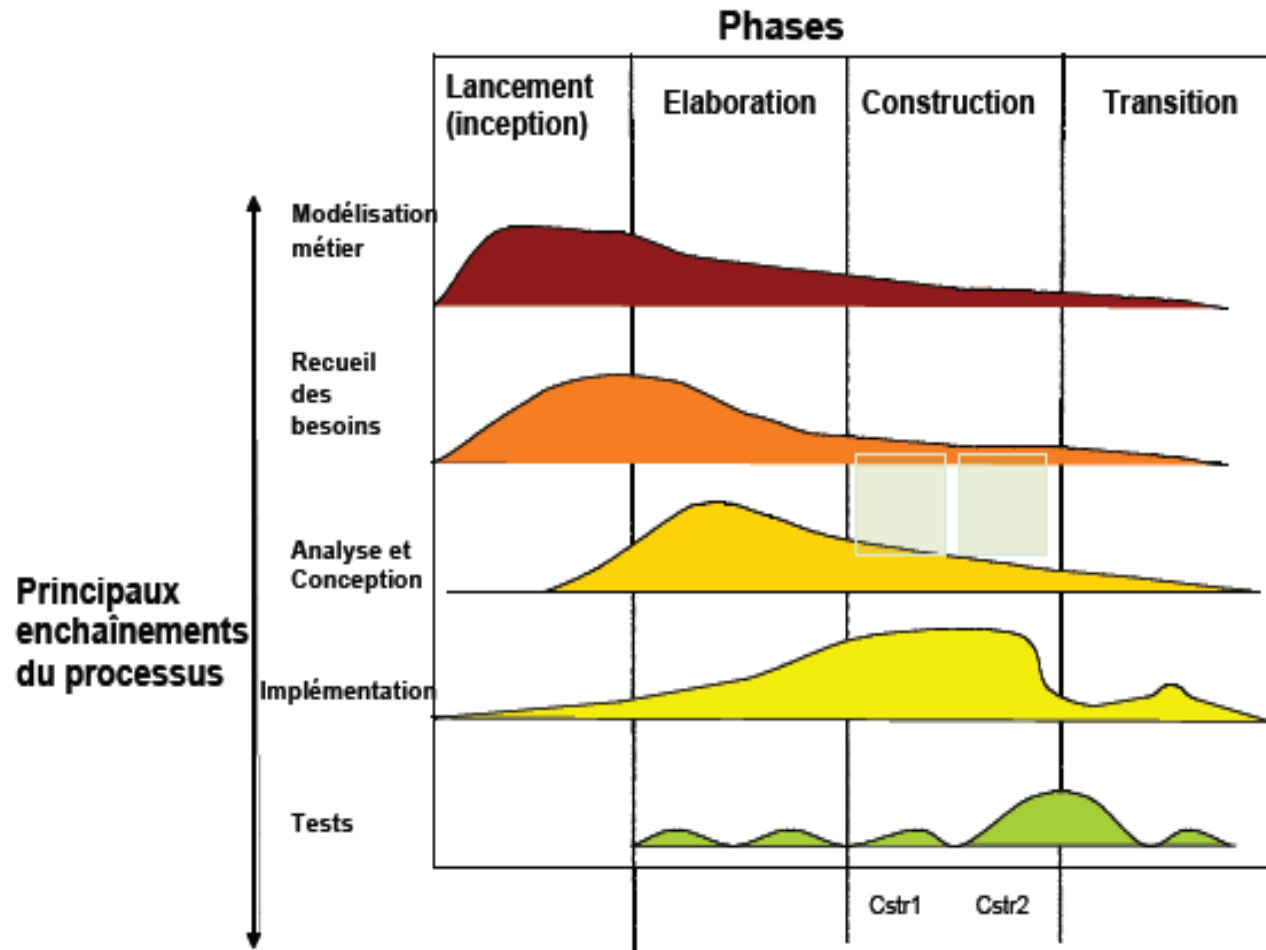
Cas d'utilisation	Risque	Priorité	Itération
Traiter une commande	Moyen	Moyenne	5
Gérer les infos clients	Bas	Moyenne	6
Consulter les en-cours	Haut	Moyenne	3
Gérer la facturation	Bas	Basse	8
Suivre les règlements	Bas	Basse	8
Planifier une mission	Haut	Haute	2
Suivre une mission	Moyen	Haute	4
Réaliser l'inventaire	Bas	Moyenne	7
Manipuler les colis	Bas	Moyenne	7
Définir le plan de transport	Haut	Haute	1
Gérer les ressources	Moyen	Haute	3
Gérer les profils	Bas	Moyenne	9
S'authentifier	Bas	Moyenne	9

Phase d'élaboration

5. Analyse et conception: Définir et valider l'architecture du logiciel.

- ▶ L'architecture est raffinée par les itérations successives:
 - ▶ Chaque itération prend en compte un certain nombre de cas d'utilisation.
 - ▶ les UC (Use Case) les plus prioritaires avant et les + risqués.
 - ▶ Les risques majeurs sont traités en priorité.
 - ▶ *(exemple: risque le plus haut à cause de leurs complexités de mise en œuvre).*
- ▶ Valider l'architecture du logiciel à travers des prototypes qui implémentent des fonctionnalités complexes.
- La prise en charge des risques principaux est adressée par le(s) prototype(s).

Phase de construction

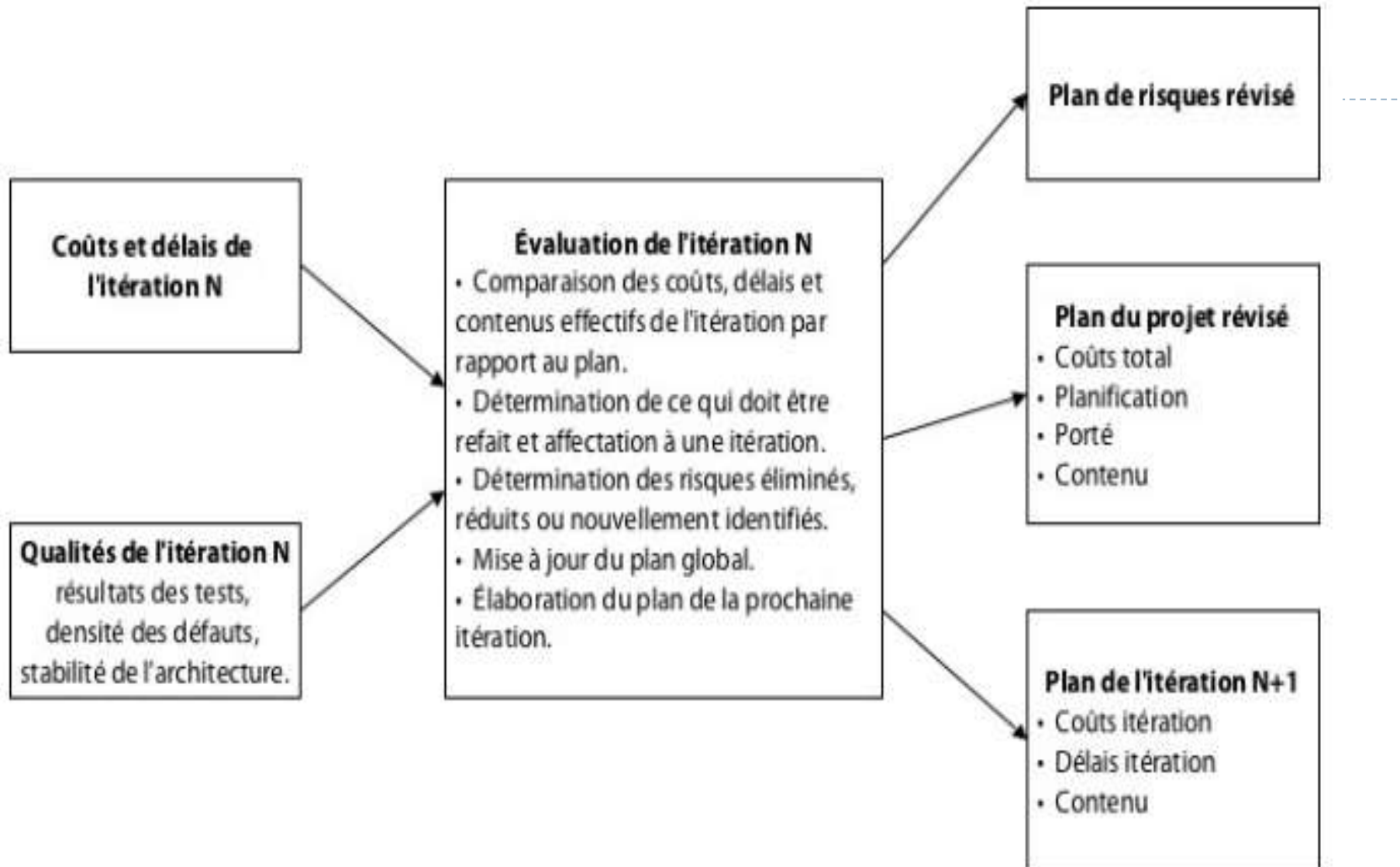


Phase de construction

- ▶ fabrication **incrémentale** de l'application.
- ▶ Chaque itération produit **la livraison d'un prototype exécutable** donc testable.
- ▶ **Evaluation à la fin de chaque itération**



Evaluation de l'itération



- ❑ Au-delà de 25 % de changements à l'issue des livraisons de construction, l'architecture est considérée comme instable.
- ❑ C'est généralement le signe d'une phase d'élaboration qui n'a pas donné une place suffisante à l'expérimentation.

Mesures du nombre d'erreurs

- Nombre d'erreurs du code (NCE) vs. nombre d'erreurs pondérés du code (WCE).

	Calculation of NCE	Calculation of WCE	
Error severity class	Number of Errors	Relative Weight	Weighted Errors
a	b	c	$D = b \times c$
low severity	42	1	42
medium severity	17	3	51
high severity	11	9	99
Total	70	---	192
NCE	70	---	---
WCE		---	192

Classification des erreurs

On classifie la sévérité des erreurs en trois niveaux:

- **Fatale** : *l'application est inopérable complètement (Crash).*
- **Grave** : *l'application fonctionne, mais certaines fonctions sont inopérables ou certains résultats sont erronés.*
- **Mineure** : l'impact est mineur sur l'utilisation du système:

Exemple:

- Certains **formats** sont erronés, bien que les résultats soient corrects,
- **Délais** sont supérieurs à ce qu'on attend d'une telle application,
- **Apparence légèrement** incorrecte d'un élément d'interface graphique.



Métriques de densité d'erreurs

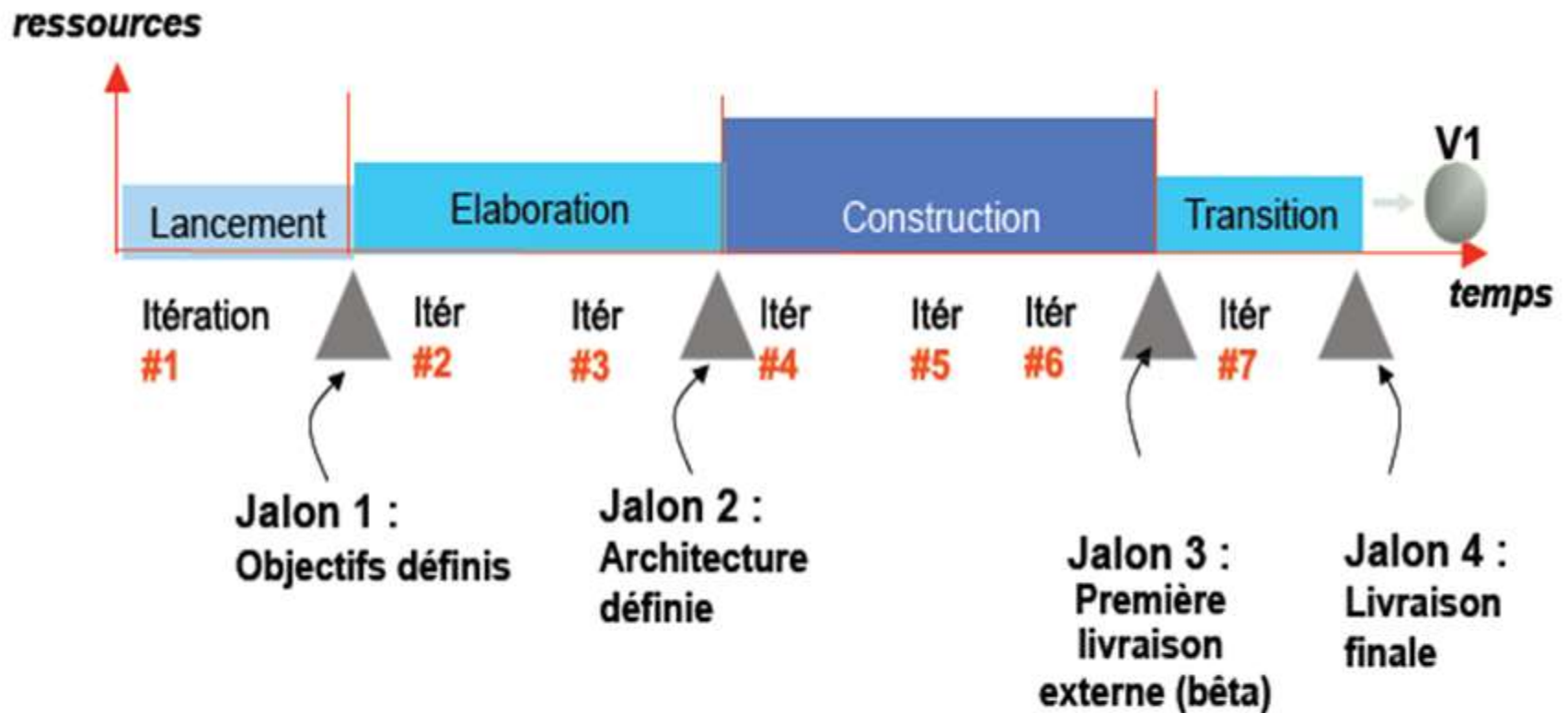
Code	Nom	Formule de Calcul
CED	Code Error Density	$CED = \frac{NCE}{KLOC}$
DED	Development Error Density	$DED = \frac{NDE}{KLOC}$
WCED	Weighted Code Error Density	$WCDE = \frac{WCE}{KLOC}$
WDED	Weighted Development Error Density	$WDED = \frac{WDE}{KLOC}$

- **NCE** = nombre total d'erreurs du code détectés par inspection de code et tests.
- **NDE** = nombre total **d'erreurs de développement** (spec; conception et code) détectés pendant le processus de développement.
- **WCE** = nombre total d'erreurs **pondérés** du code détectés par inspection de code et tests.
- **WDE** = nombre total d'erreurs pondérés.

Phase de construction

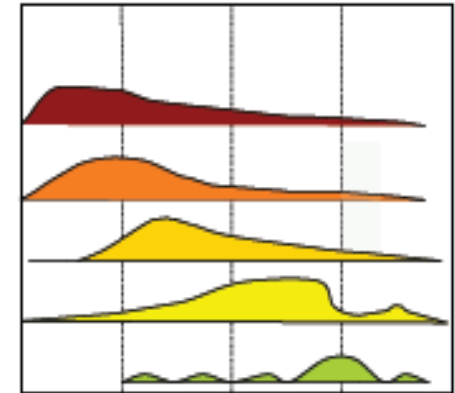
- ❑ Le logiciel est ramené à un état de maturité suffisant pour être **livré à ses utilisateurs**.
- ❑ La phase de construction **se termine par la livraison de la version **béta**** du logiciel qui est placée chez les utilisateurs pour être testée dans l'environnement de déploiement.

Phases, itérations, jalons



➡ Les Jalons correspondent à des étapes d'évaluation de la phase terminée, et de lancement de la phase suivante

Phase de transition (4/4)



- Déployer
- Tester
- Valider (réponse aux attentes des utilisateurs)
- Accompagner l'utilisateur final (packaging, documentation, formations, manuels ...)

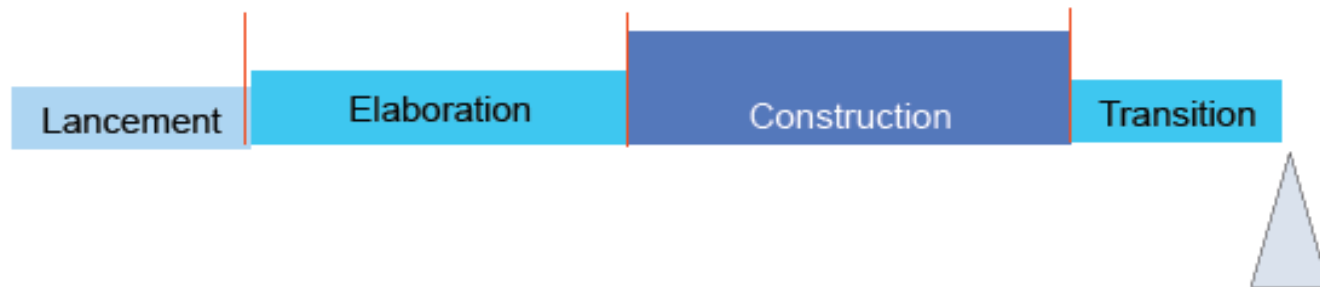
phase de transition

- ▶ Installation des versions bêta sur site.
- ▶ Des livraisons de versions viennent **corriger les défauts détectés par les utilisateurs.**
- ▶ **La phase de transition s'arrête** lorsque le logiciel est arrivé à sa version de production.



Revue de projet : Jalon Fin de la Transition

- Évaluation :
 - Satisfaction des utilisateurs
 - Bilan sur les ressources réellement consommées



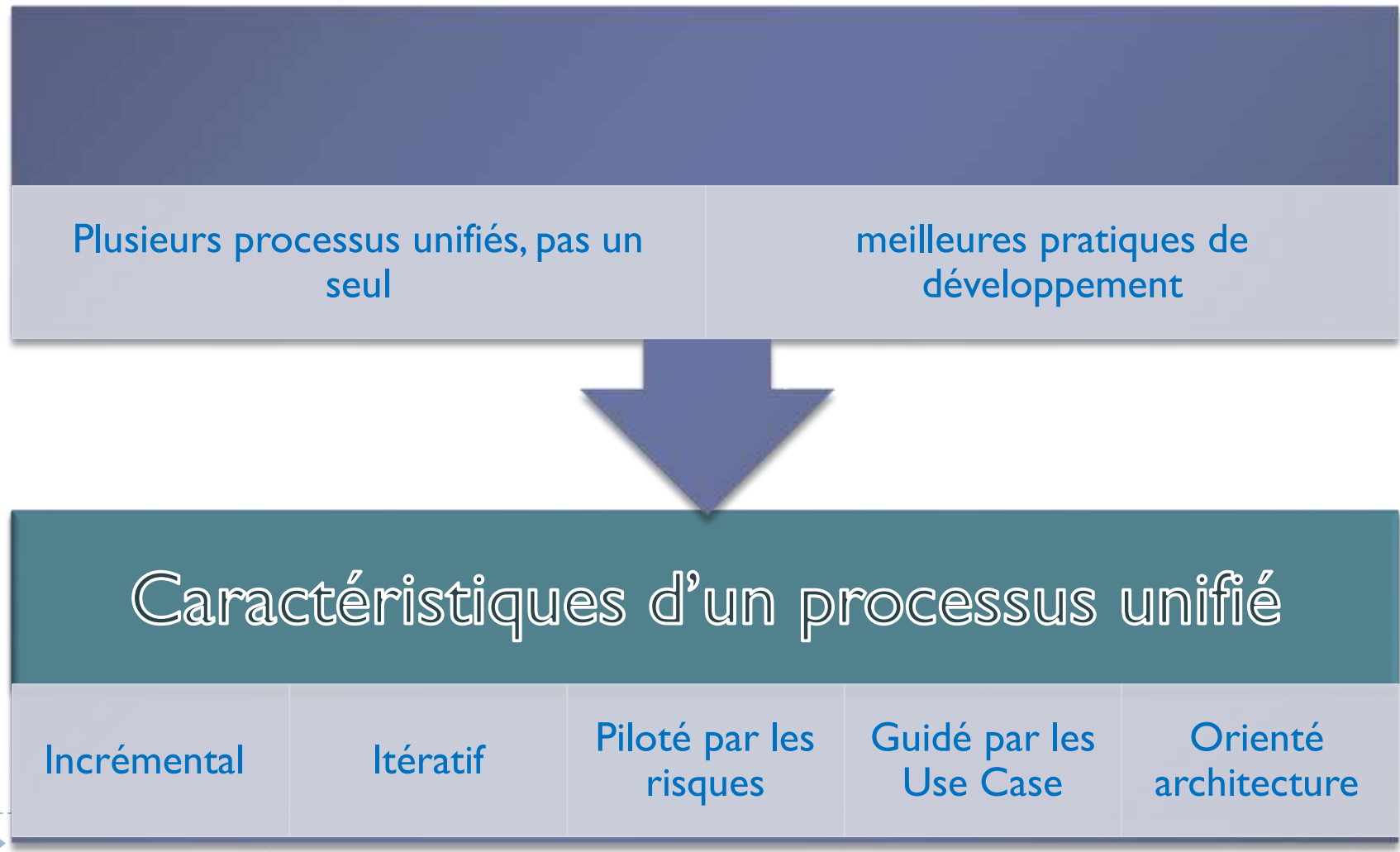
Méthode de développement logiciel: 2TUP (gr 2 A – IB- IA)

Méthode 2TUP

- ▶ Le processus 2TUP (Two Track Unified Process) est un processus unifié.
- ▶ Il gère **la complexité technologique** en donnant part à la technologie dans son processus de développement.
- ▶ Permet de réduire **les risques** liés à **la complexité technologiques**
- ▶ Processus créé par Valtech
 - ▶ un groupe français de conseil en technologies, e-business, management, et de formation professionnelle.



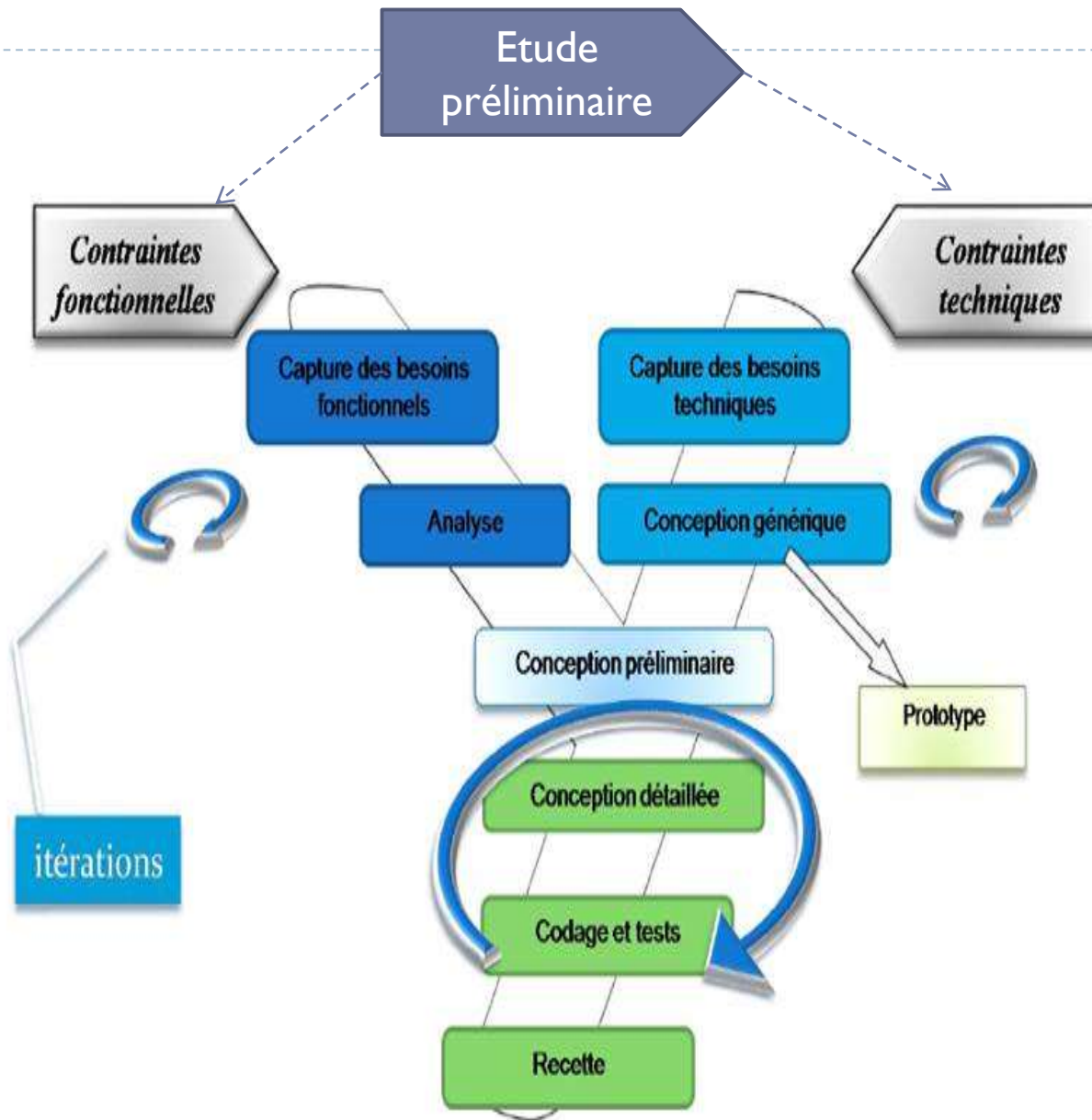
Présentation de 2TUP



- Pourquoi 2TUP ?
-



Présentation de 2TUP



- **RESUME: Une forme en Y**
- **Du côté de la branche fonctionnelle :**
 - **Capture des besoins fonctionnels** : elle aboutit à un modèle des besoins focalisé sur le métier des utilisateurs. Elle minimise le risque de produire un système inadéquat avec les besoins des utilisateurs. De cette capture, la MOE consolide les spécifications et en vérifie la cohérence et l'exhaustivité.
 - **Analyse** : étude des spécifications afin de savoir ce que le système va réellement réaliser en termes de métier. Découpage en composants.
- **Du côté de la branche technique :**
 - **Capture des besoins techniques** : recensement des outils, des matériels et des technologies à utiliser ; des contraintes (temps de réponse maximal, contraintes d'intégration avec l'existant) -> tout cela va aboutir à une première conception de l'architecture technique
 - **Conception générique** : Découpage en composants nécessaires à la construction de l'architecture technique. Il est généralement conseillé de réaliser un prototype pour assurer la validité de l'architecture. Cette étape permet de minimiser l'incapacité de l'architecture technique à répondre aux contraintes opérationnelles
- **Enfin, la branche du milieu :**
 - **Conception préliminaire** : étape délicate durant laquelle on intègre le modèle d'analyse dans l'architecture technique. Le but ici est de savoir dans quel composant technique on met nos fonctionnalités issues de l'analyse.
 - **Conception détaillée** : conception de chaque fonctionnalité
 - **Etape de codage** : phase de programmation de ces fonctionnalités, avec des tests au fur et à mesure
 - **Etape de recette** : phase de validation des fonctions du système développé

Exemple

1. une agence de tourisme passe des accords avec une compagnie aérienne de sorte que le calcul des commissions change. En l'occurrence, les résultats issus de la branche fonctionnelle qui évoluent pour prendre en compte la nouvelle spécification ;
2. Cette même entreprise décide d'ouvrir la prise de commande sur le Web. Si rien ne change fonctionnellement, en revanche, l'architecture technique du système évolue ;

Étude préliminaire

Modélisation du contexte

Services de bases

Risques



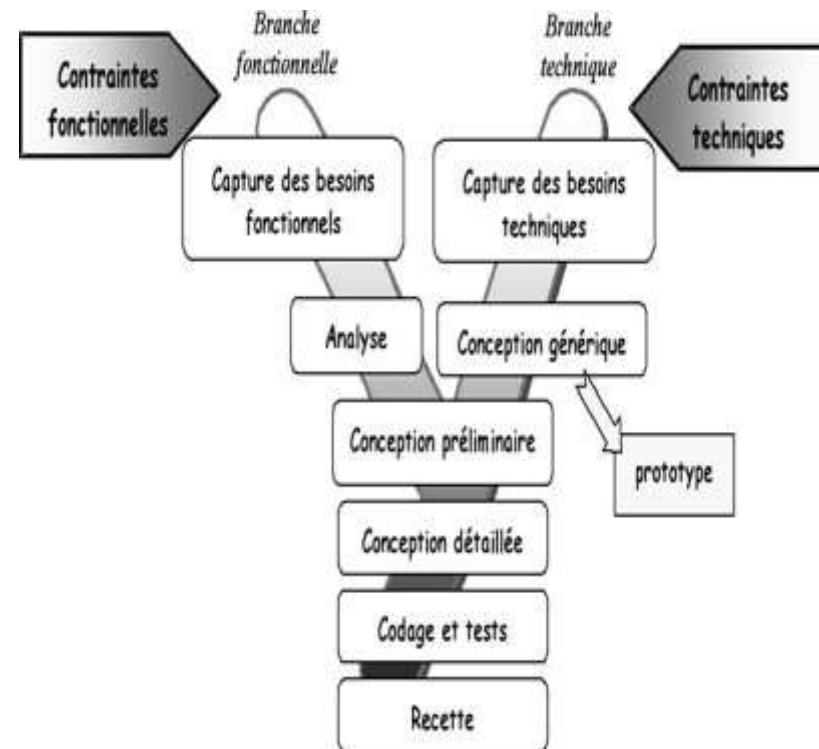
2TUP

► Une forme en Y

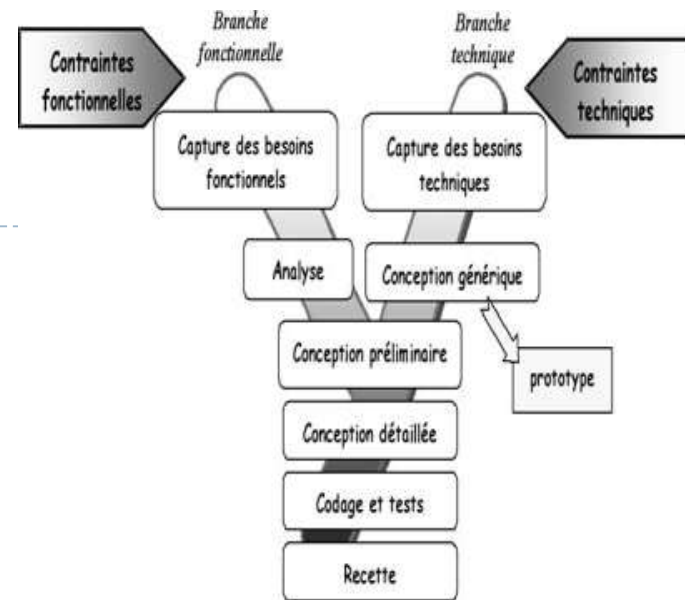
► La branche fonctionnelle :

- Capture des besoins fonctionnels :

- Elle aboutit à un modèle des besoins focalisé sur le métier des utilisateurs.
- Elle minimise le risque de produire un système inadéquat avec les besoins des utilisateurs.



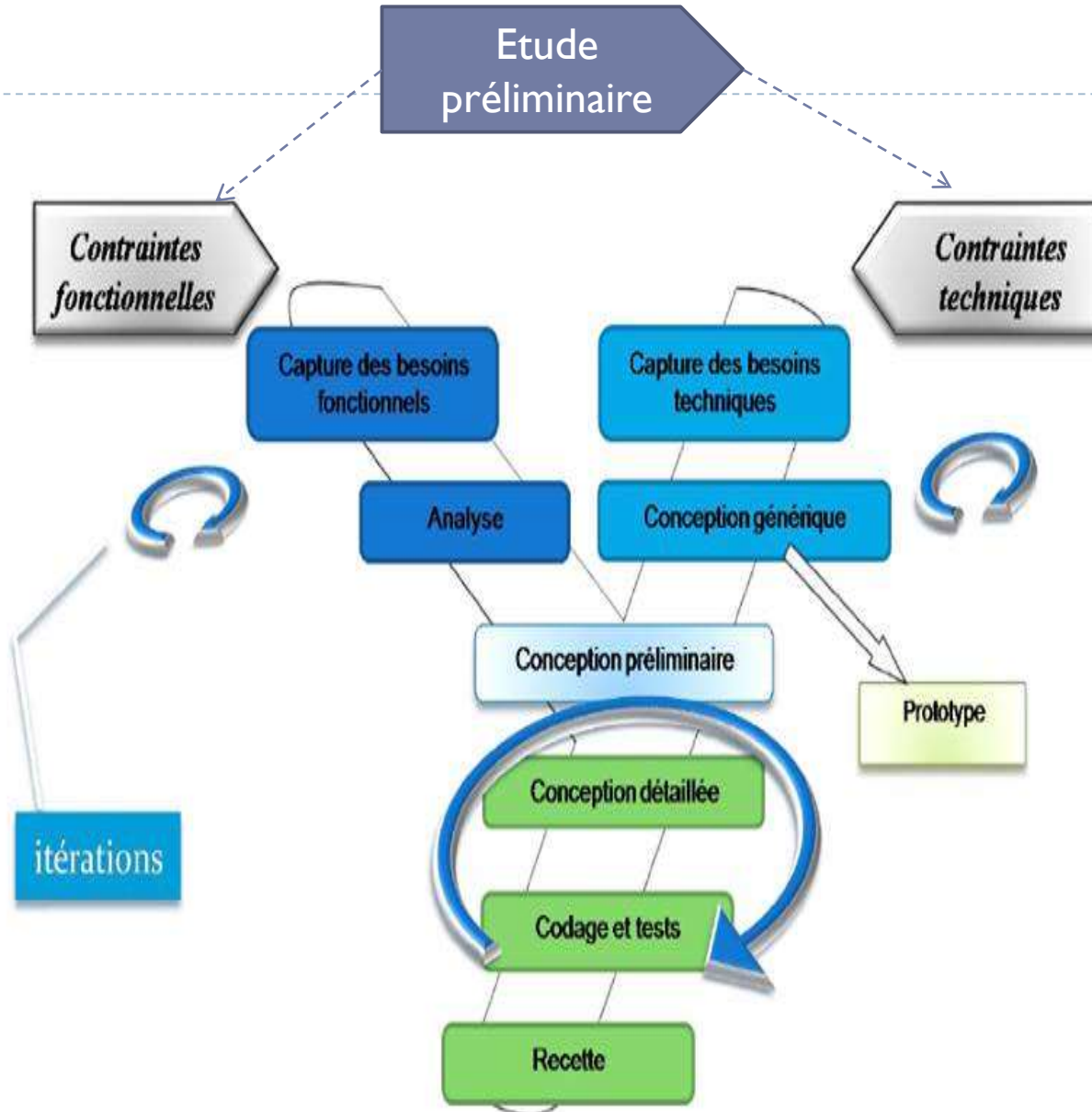
2TUP



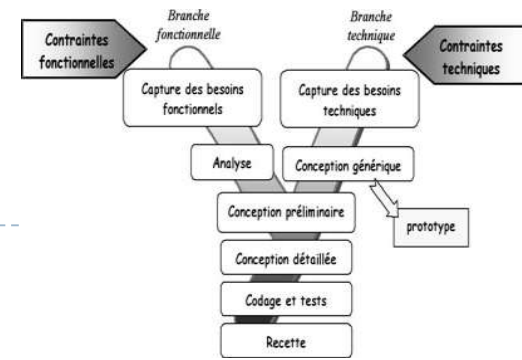
► La branche fonctionnelle :

- **Analyse** : étude des spécifications afin de savoir ce que **le système va réellement réaliser en termes de métier.**
- Découpage en composants logiques: **Architecture logique.**

Présentation de 2TUP

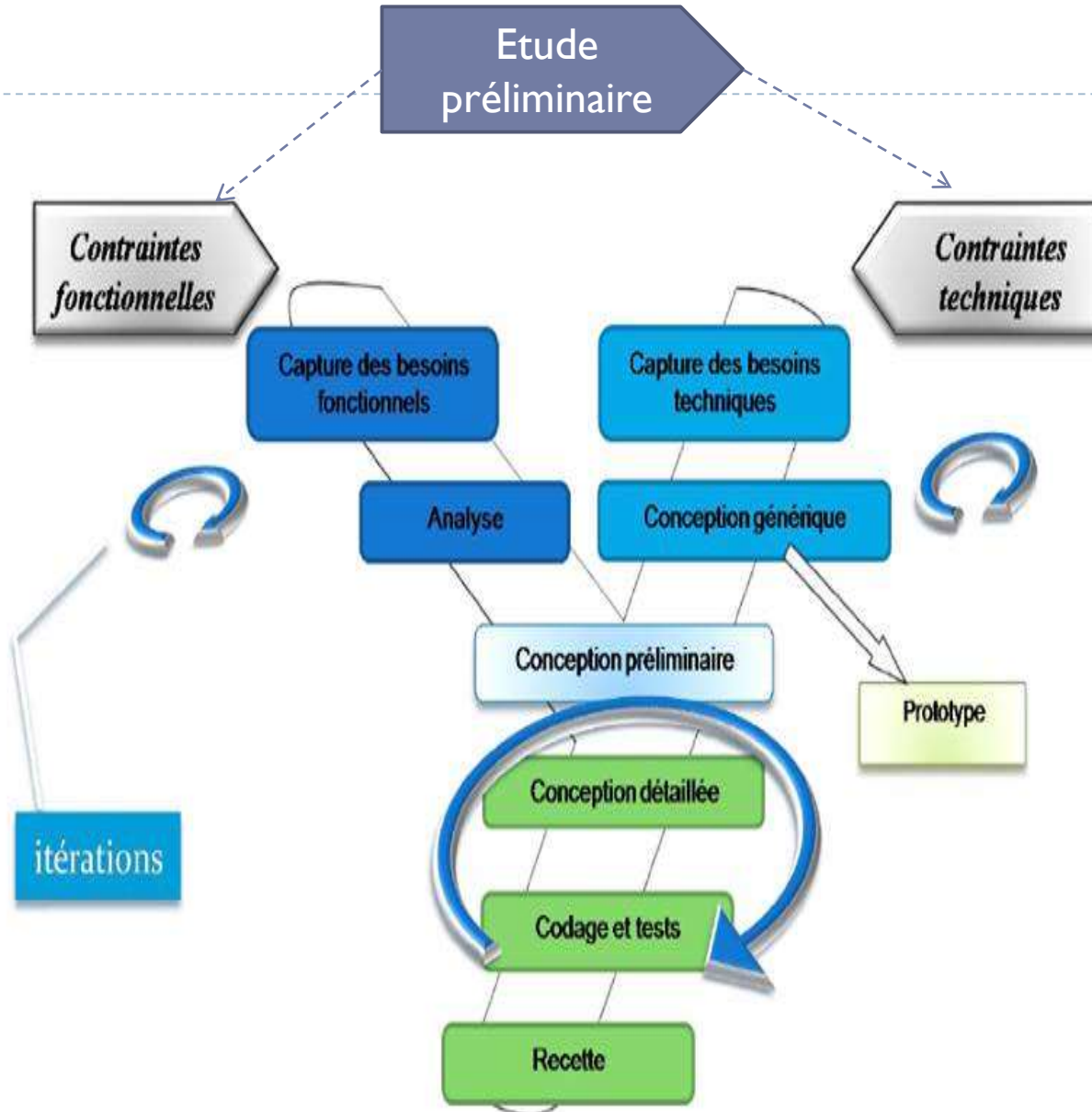


2TUP

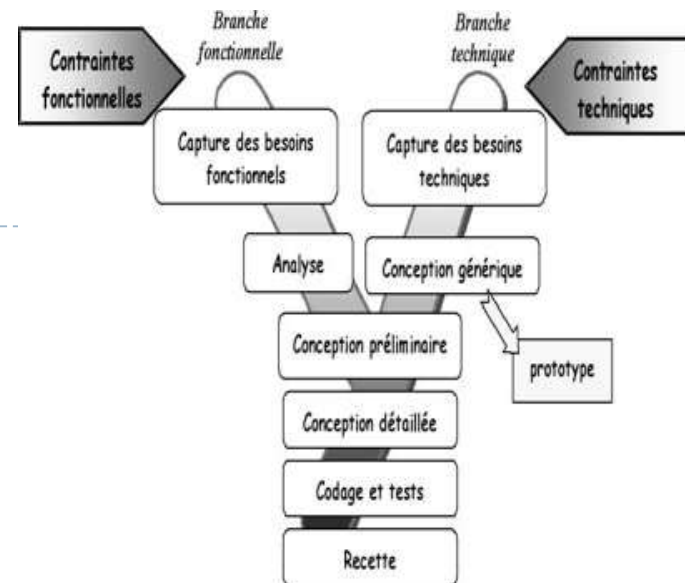


- La branche technique :
- Capture des besoins techniques :
 - recensement des outils, des matériels et des technologies à utiliser;
 - des contraintes (temps de réponse maximal, contraintes d'intégration avec l'existant, attributs de qualité)

Présentation de 2TUP



2TUP



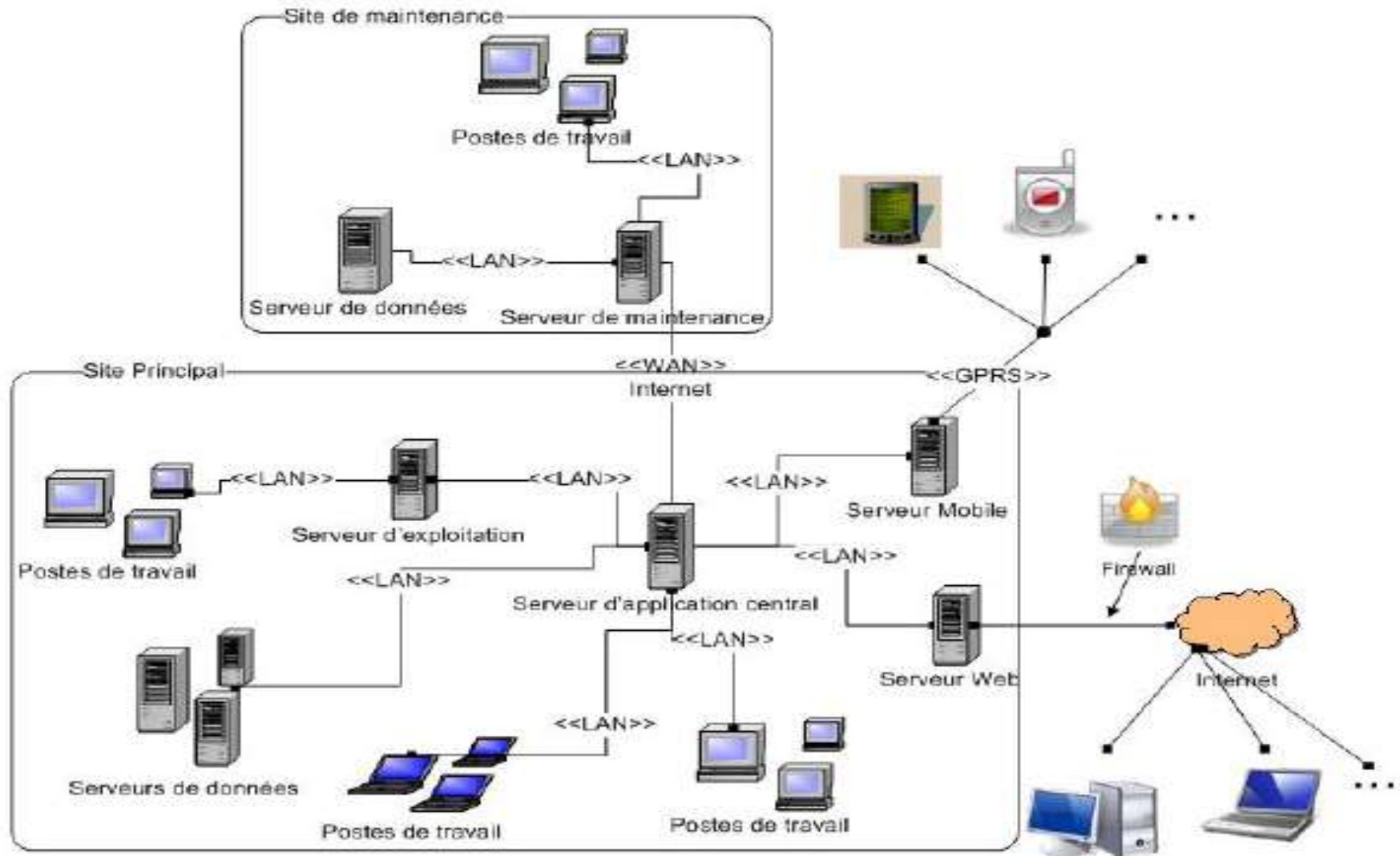
- La branche technique

Conception générique :

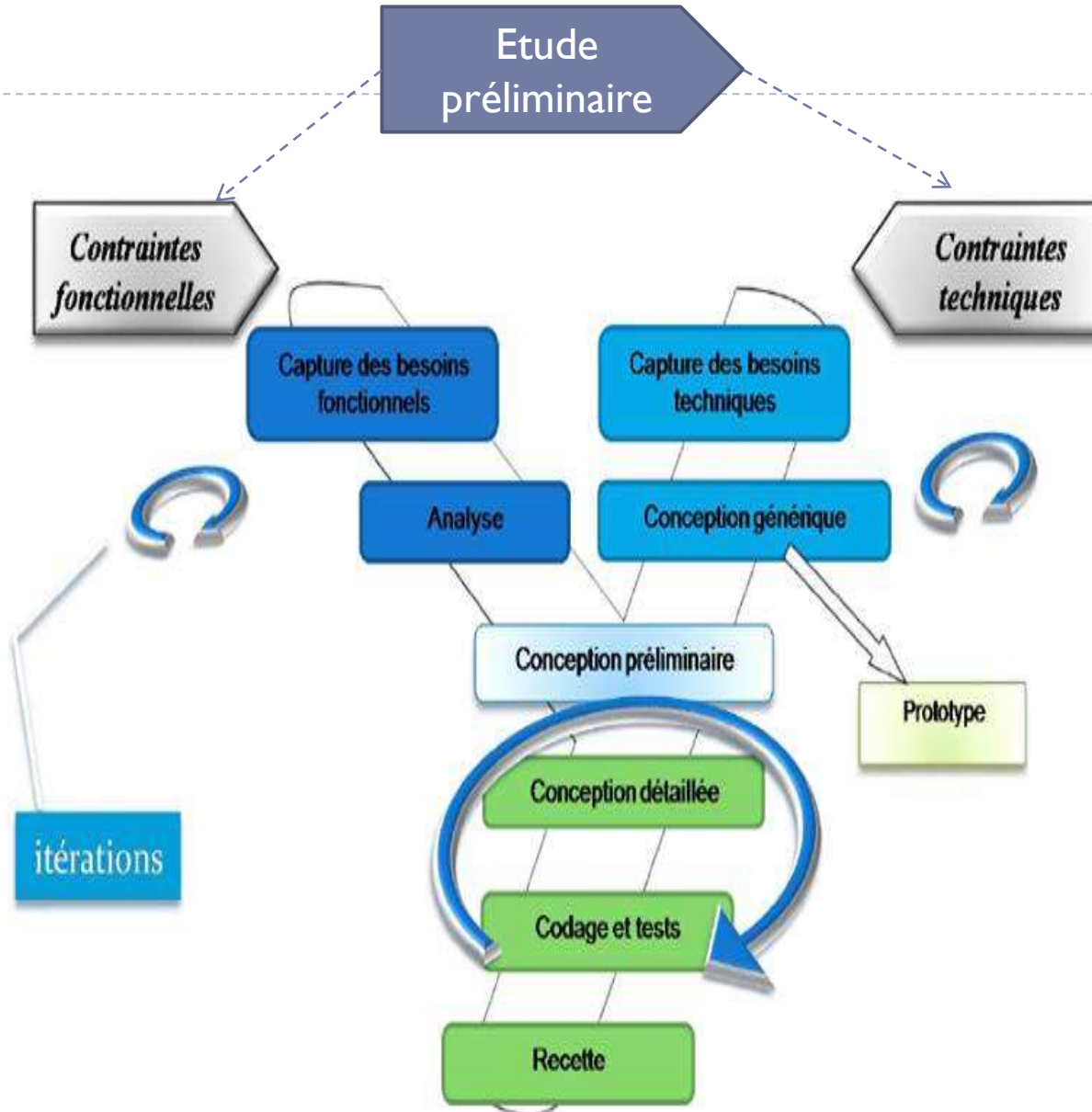
- Découpage en composants nécessaires à la construction de l'architecture technique.
- Composants **matériels**, logiciels et technologiques.
- Il est généralement conseillé de réaliser un prototype pour assurer la validité de l'architecture.

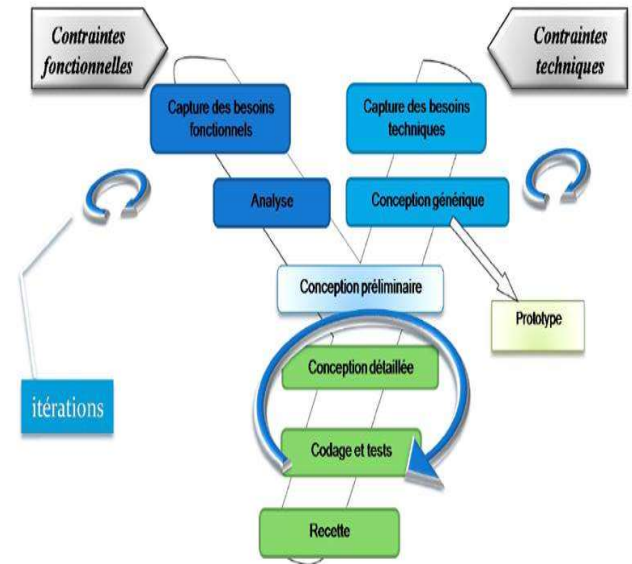
Cette étape permet de minimiser l'incapacité de l'architecture technique à répondre aux contraintes opérationnelles.

Architecture technique



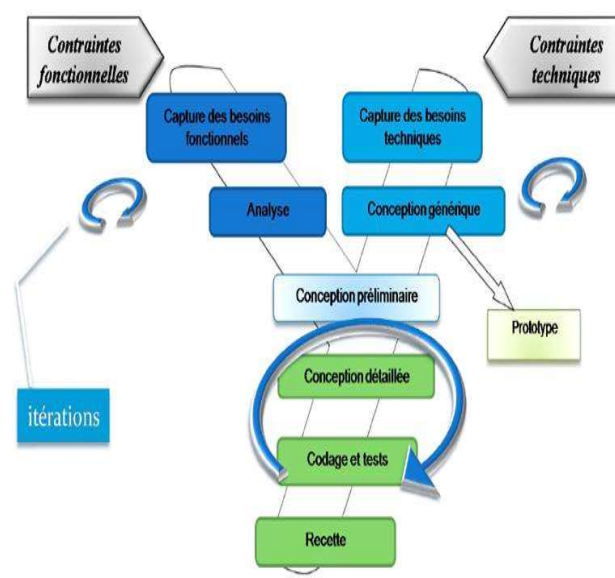
Présentation de 2TUP





La branche du milieu :

- ▶ **Conception préliminaire** : étape délicate durant laquelle **on intègre le modèle d'analyse dans l'architecture technique.**
- ▶ Le but ici est de savoir dans quel composant technique on met nos fonctionnalités issues de l'analyse.



La branche du milieu

- **Conception détaillée** : conception de chaque fonctionnalité.
- Etape de codage : phase de programmation de ces fonctionnalités, avec des tests au fur et à mesure
- Etape de recette : phase de validation des fonctions du système développé

2TUP, un processus UP

Deux types d'acteurs

L'utilisateur
consommateur
des fonctions du
système

L'utilisateur
exploitant le
système

La branche gauche
est chargée de
capturer les
besoins
fonctionnels auprès
des utilisateurs
consommateurs

La branche droite
est chargée de
capturer les
besoins techniques
auprès des
utilisateurs
exploitants

▶ L'utilisateur consommateur des fonctions du système :

- ▶ Il correspond à un rôle ou un métier dans l'entreprise.
- ▶ Il attend du système qu'il lui apporte une plus-value dans l'exercice de sa profession.

▶ L'utilisateur exploitant le système :

- ▶ rôle technique et opérationnel commun à tous les SI, exploitant/administrateur.
- ▶ attend du système des performances, une tenue à la charge, une sécurité d'accès...
- ▶ *Cet utilisateur est souvent négligé dans les autres méthodes de gestion de projets.*
- ▶ La capture de ses besoins se fera grâce à l'établissement des cas d'utilisation techniques, qui aboutiront ensuite à des spécifications d'architecture.

2TUP et UML

Capture des besoins fonctionnels

- Diagramme des cas d'utilisation,
- Diagrammes de séquence,
- Diagrammes de collaboration

Analyse

- Diagramme de classes,
- Diagrammes d'états transition

Capture des besoins techniques

- Diagramme des cas d'utilisation

Conception générique

- Diagramme de déploiement

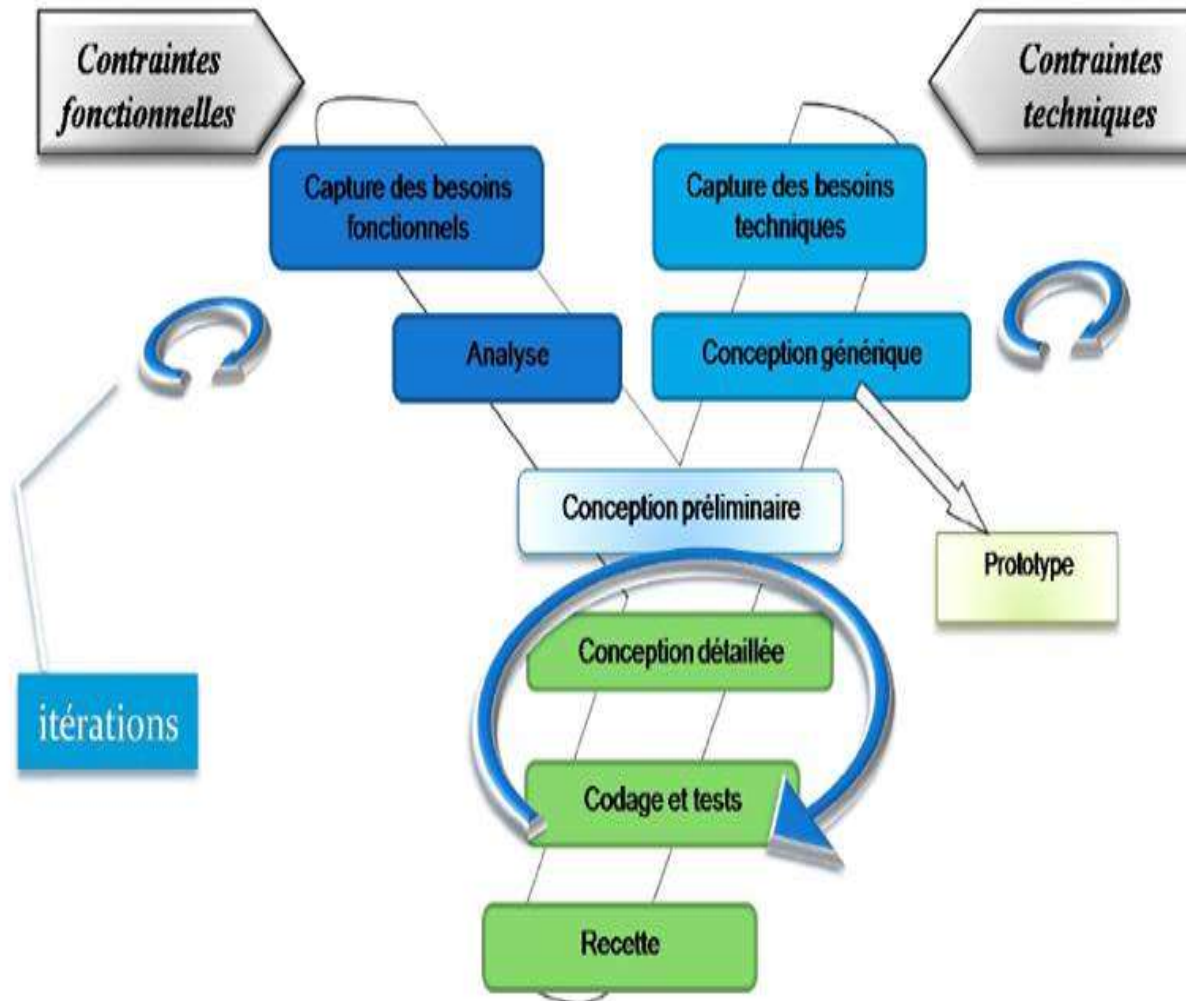
Conception préliminaire

- Diagramme de composants,
- Diagramme de déploiement

Conception détaillée

- Diagramme de classes,
- Diagramme de séquence,
- Diagramme de collaboration,
- Diagramme d'états,
- Diagramme d'activités,
- Diagramme de composants

2tup



Exemple

Se servir des cas d'utilisation pour définir les itérations :

Cas d'utilisation	Risque	Priorité	Itération
Traiter une commande	Moyen	Moyenne	5
Gérer les infos clients	Bas	Moyenne	6
Consulter les en-cours	Haut	Moyenne	3
Gérer la facturation	Bas	Basse	8
Suivre les règlements	Bas	Basse	8
Planifier une mission	Haut	Haute	2
Suivre une mission	Moyen	Haute	4
Réaliser l'inventaire	Bas	Moyenne	7
Manipuler les colis	Bas	Moyenne	7
Définir le plan de transport	Haut	Haute	1
Gérer les ressources	Moyen	Haute	3
Gérer les profils	Bas	Moyenne	9
S'authentifier	Bas	Moyenne	9

2tup

- Gestion des risques : prise en charge de deux axes du projet
- Processus unifié : itératif, incrémental, orienté modèle, guidé par les cas d'utilisation,..

