Chapitre 3 : Processus Unifié de Modélisation

INSTITUT SUPERIEUR INFORMATIQUE ISI
المعهد العالي للإعلامية

Responsables du cours :
Héla Hachicha
Hatem Ben Sta

Année Universitaire : 2014 - 2015

Plan du cours

- Chapitre 1 : Introduction au génie logiciel
- Chapitre 2 : Gestion de projet informatique
- Chapitre 3 : Processus Unifié de Modélisation
- Chapitre 4 : Activités dans le Processus Unifié
- Chapitre 5 : Les Design Pattern
- Chapitre 6 : MDA : Model-Driven Architecture
- Chapitre 7 : Le langage de contraintes OCL
- Chapitre 8 : Méta-modélisation MOF
- Chapitre 9 : Les profils UML
- Chapitre 10 : MDE: Model Driven Engineering

Sommaire

Chapitre 3 : Processus Unifié de Modélisation

- Décrire le processus unifié
- Décrire le processus unifié de Rational
- 2TUP, une variante du Unified Process
- eXtreme programming, … (méthodes agiles)

Génie logiciel

De nombreux processus de développement existent

Modèle en cascade

UP - Unified Process

Modèle en V

RAD - Développement rapide d'applications

DSDM

XP - eXtreme Programming

Pas de choix universel

Le choix dépend de :

- L'organisation qui développe
- Le type de logiciel
- Les personnes impliquées

UP - Unified Process
Processus Unifié

Processus Unifié

- Le "Processus unifié de développement logiciel " peut être utilisé par toute personne prenant part au développement de logiciels.

- Il s'adresse avant tout aux membres de l'équipe de développement chargé des activités du cycle de vie qui sont la formulation des besoins, l'analyse, la conception et les tests; en d'autres termes, des activités produisant des modèles UML.

Problématique

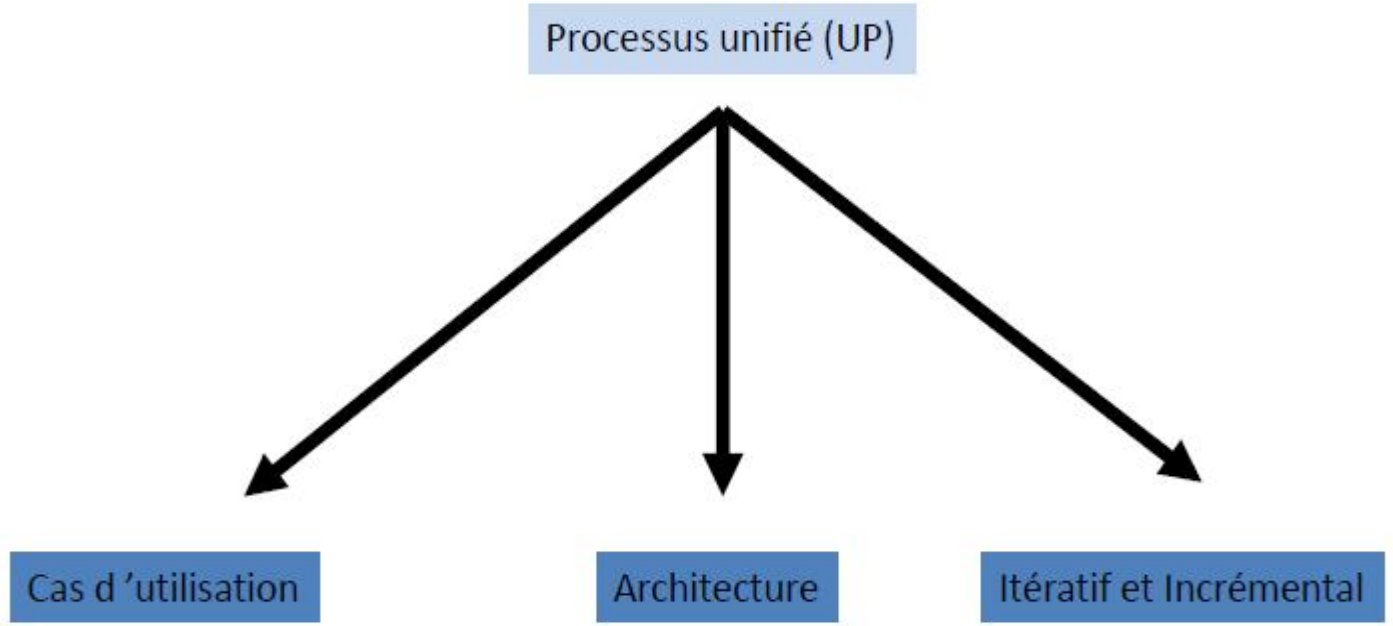
Le processus doit:

- 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

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



Le processus unifié est 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.

Le processus unifié est centré sur l'architecture d'utilisation

- 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.

- L'architecture se dévoile peu à peu, au rythme de la spécification et de la maturation des cas d'utilisation, qui favorisent, à leur tour, le développement d'un nombre croissant de cas d'utilisation. Ce processus se poursuit jusqu'à ce que l'architecture soit jugée stable.

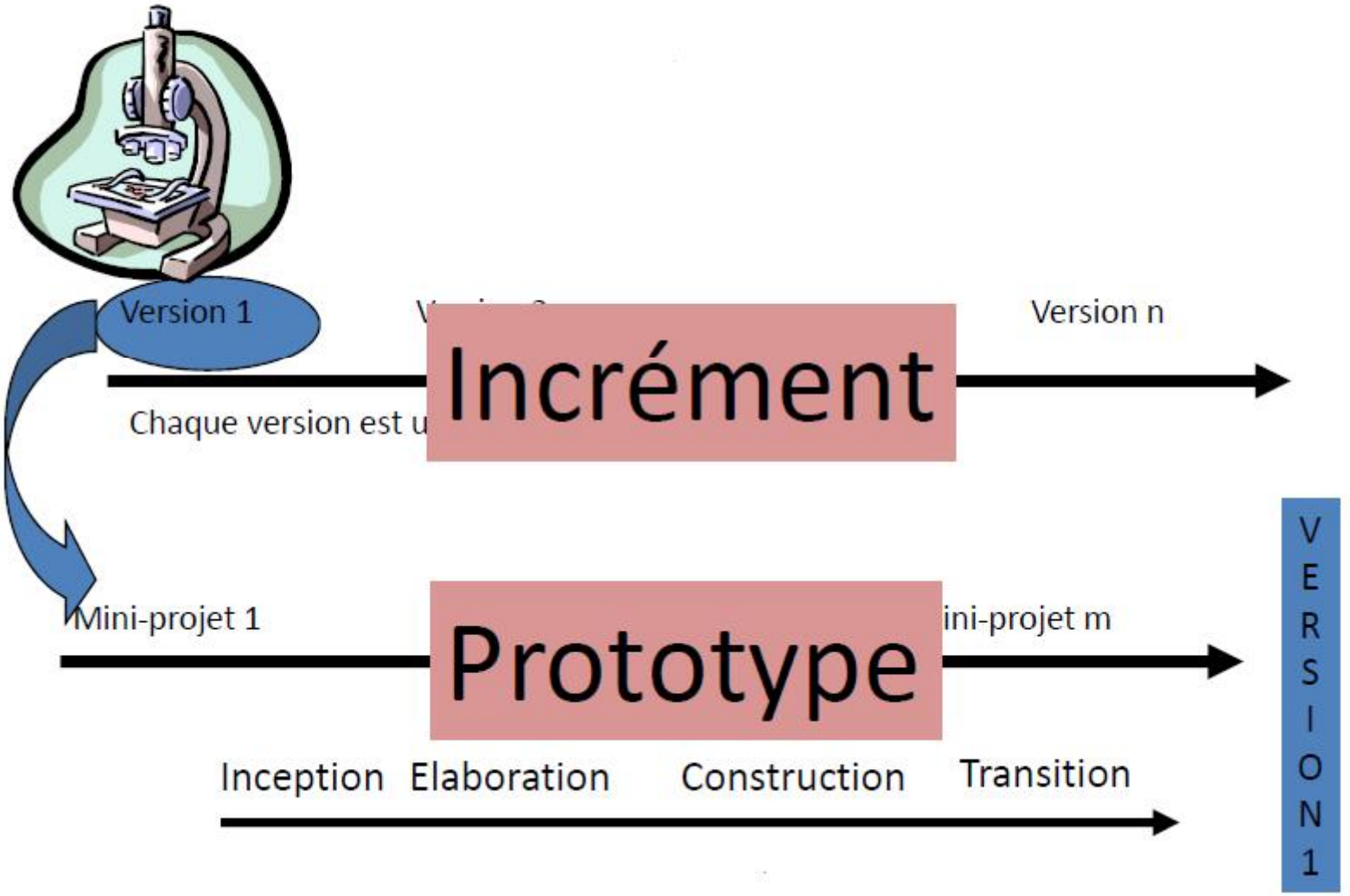
Le processus unifié est itératif et incrémental

- 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.

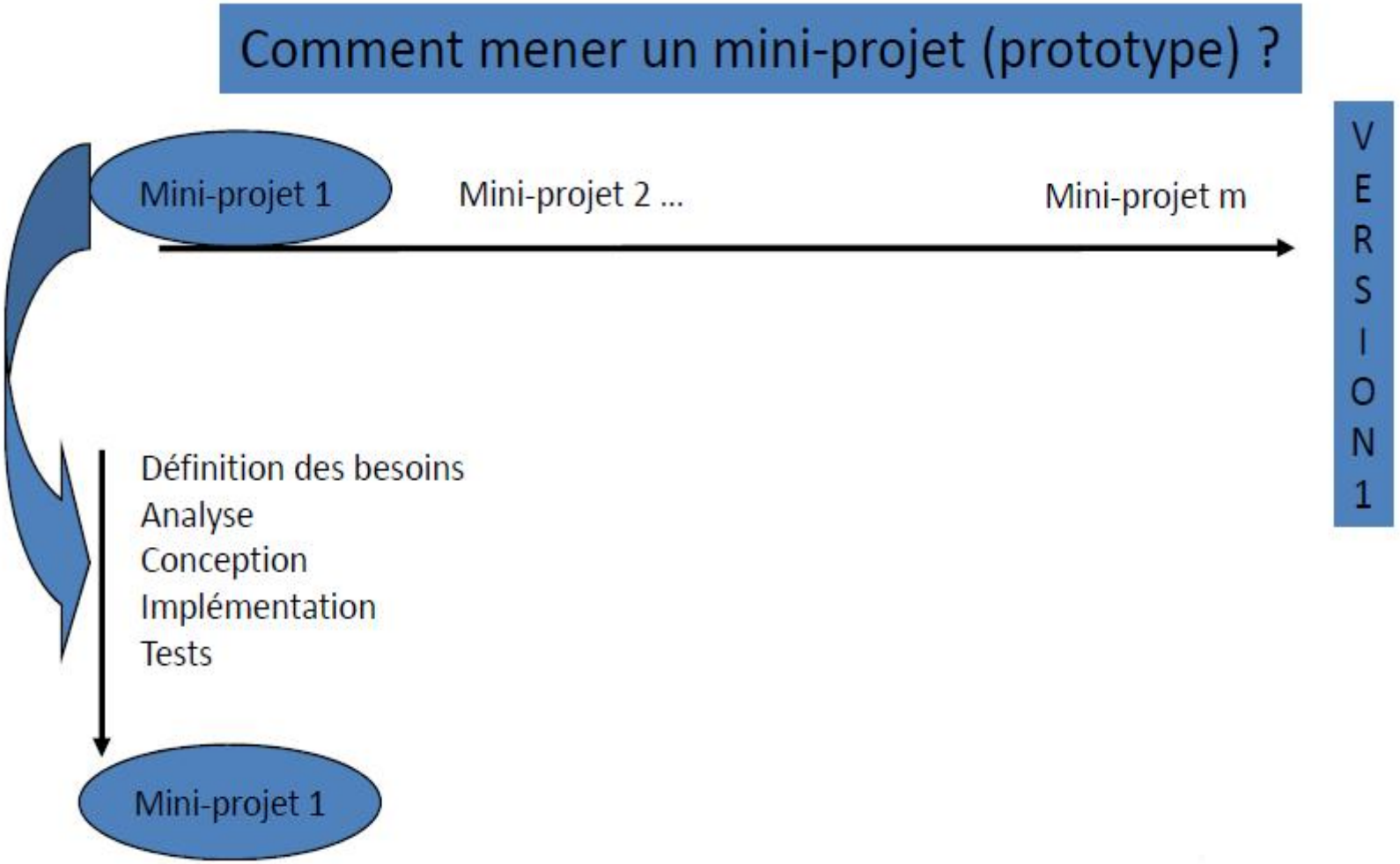
Itération et Incrément

- Une itération est un mini-projet, c'est à dire un déroulement plus ou moins complet des principaux enchaînements d'activités, aboutissant à une version livrée en interne.
- Un incrément est la différence entre la version interne d'une itération et la version interne de l'itération suivante.
- Un incrément est la différence entre deux références successives (état d'avancement).

Itération et Incrément



Itération et Incrément



Itérations

- A chaque itération, les développeurs spécifient les cas d'utilisations pertinents, créent une conception en se laissant guider par l'architecture choisie, implémentent cette conception sous forme de composants et vérifient que ceux ci sont conformes aux cas d'utilisation.

- Dès qu'une itération répond aux objectifs fixés le développement passe à l'itération suivante.

- Pour rentabiliser le développement il faut sélectionner les itérations nécessaires pour atteindre les objectifs du projet. Ces itérations devront se succéder dans un ordre logique. Un projet réussi suivra un déroulement direct, établi dés le début par les développeurs et dont ils ne s'éloigneront que de façon très marginale.

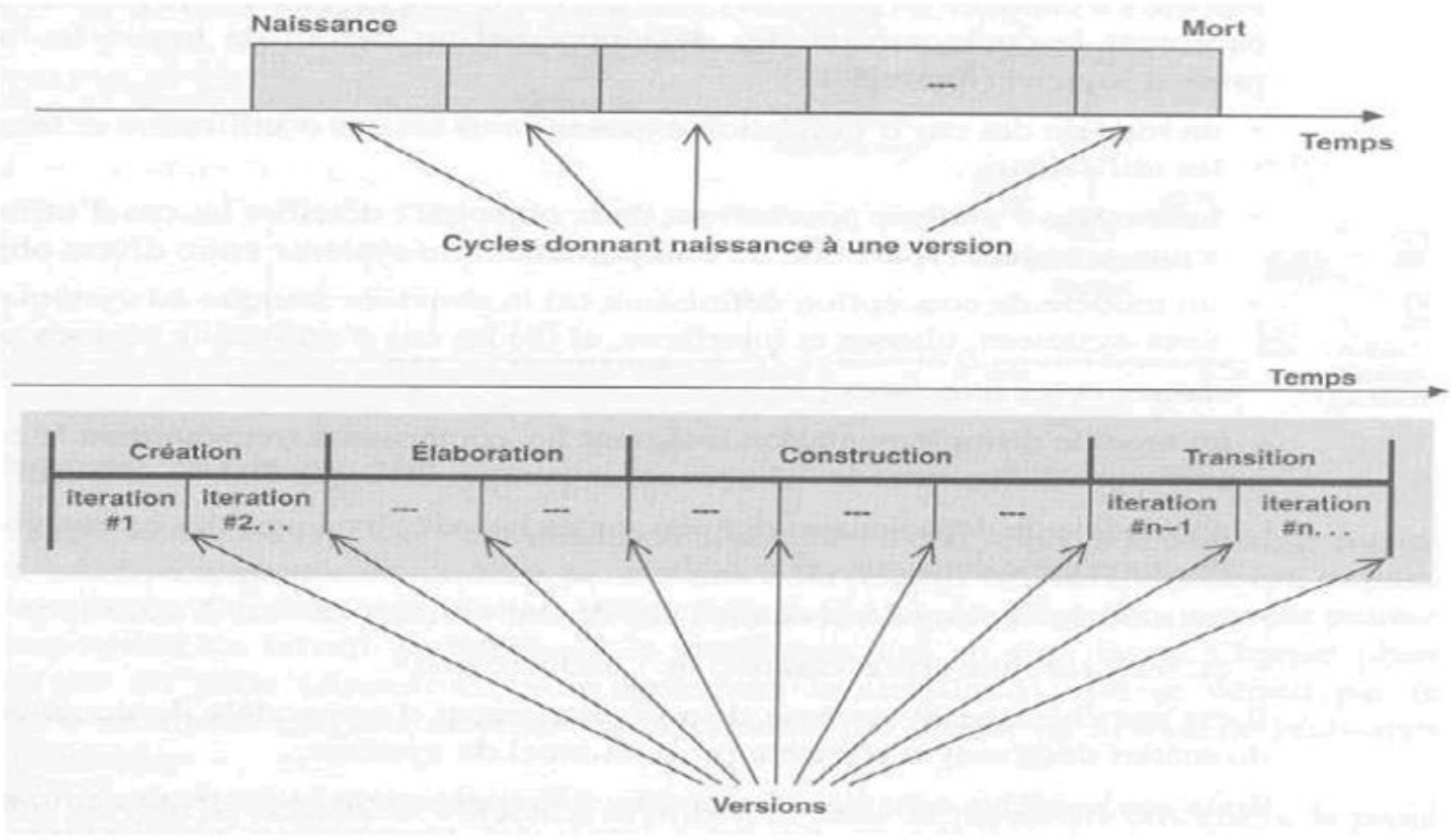
- L'élimination des problèmes imprévus fait partie des objectifs de réduction des risques.

Bénéfice attendu des itérations

Une itération contrôlée :

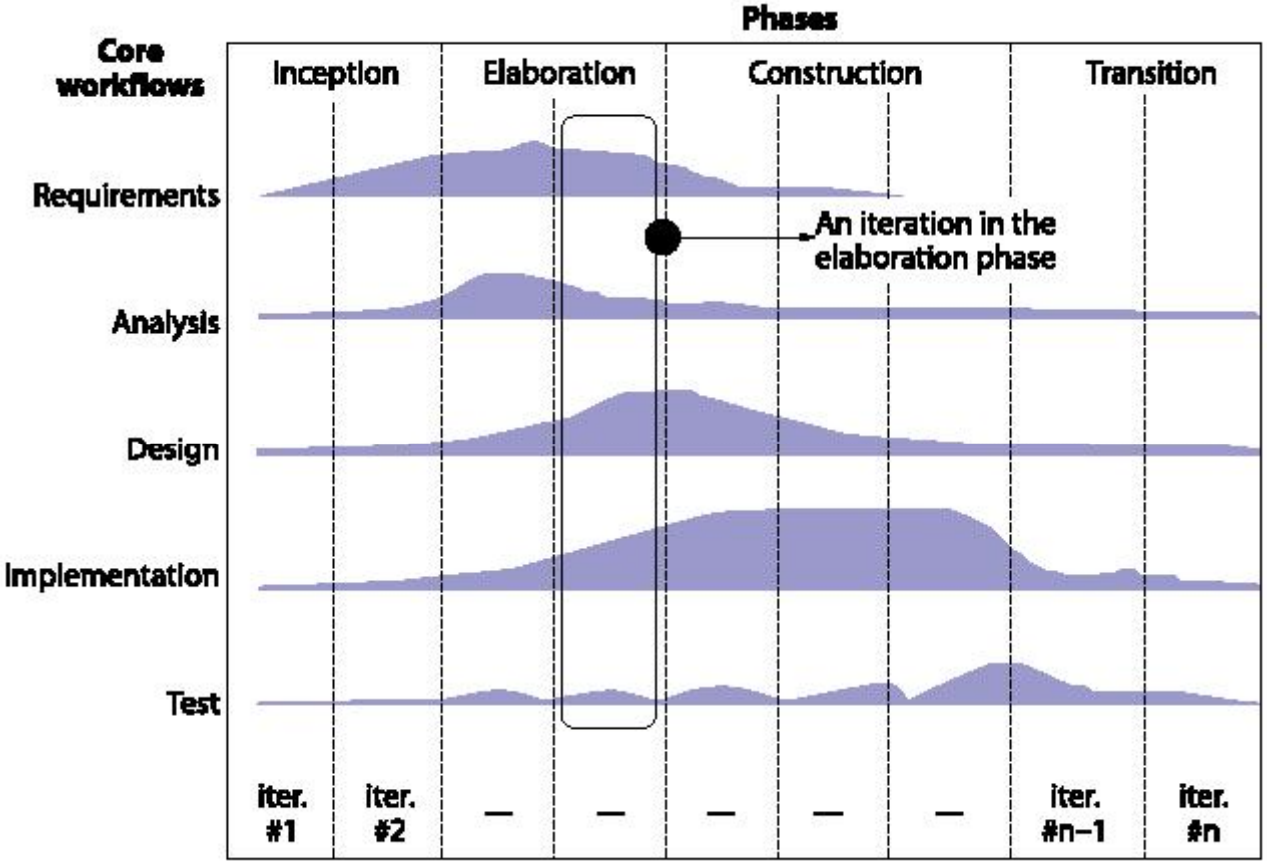
- permet de limiter les coûts
- un risque ou problème est confiné à une itération
- permet de limiter les risques de retard
- identification des risques et résolution des problèmes dès les premiers stades de développement se traduit par une accélération du rythme de développement
  - grâce à des objectifs clairs à court terme
  - facilite l'adaptation à l'évolution des besoins

La vie du Processus unifié



Le Processus unifié répète un certain nombre de fois une série de cycles constituant la vie d'un système...

Les phases d'un cycle



Chaque cycle se déroule sur une certaine durée découpées en 4 phases...

- A chaque itération, les développeurs identifient et spécifient les cas d'utilisations les plus pertinents, créent une conception (architecture), implémentent cette conception (en composants), et vérifie le tout.

Les phases d'un cycle

- **Création** (ou inception): Définir rôle et portée du produit : étude de rentabilité et objectifs du cycle de vie : répond à la question le projet vaut il la peine d'être entrepris.
- **Elaboration**: Créer l'architecture de référence, estimer les coûts et les risques, planifier la construction : Architecture du cycle de vie
- **Construction**: développer le système complet et s'assurer que le produit peut commencer à être utilisé.
- **Transition**: livraison du produit.

Les 4 "P" du Processus unifié

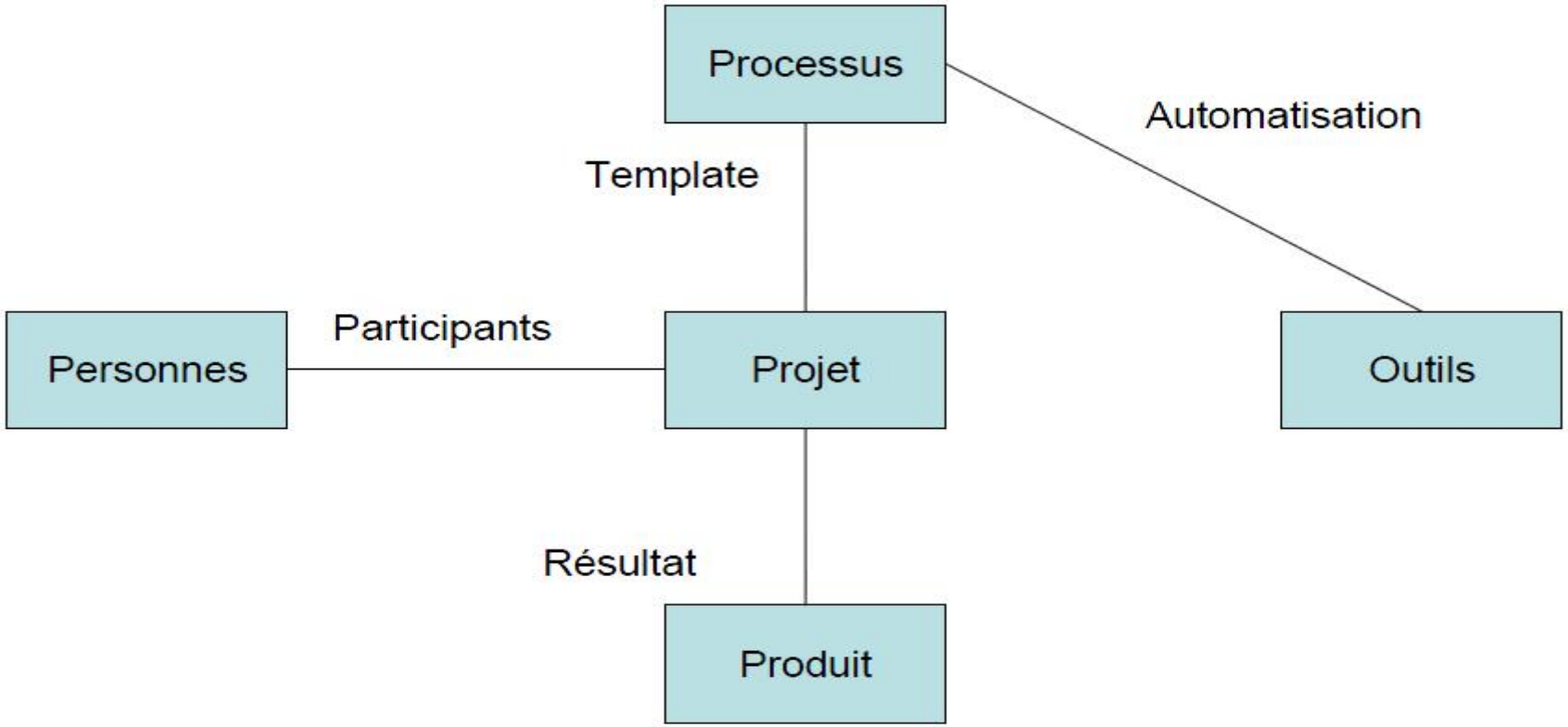
- Personnes : les architectes, développeurs, testeurs, utilisateurs, clients et autres intervenants sont les éléments moteurs de tout projet logiciel.

- Projet : élément d'organisation à travers lequel est géré le développement du logiciel. L'aboutissement d'un projet est le produit livré sous forme de version.

- Produit : ensemble d'artefacts (modèles) créés au cours du cycle de vie du projet, tels que les modèles, les codes sources, les exécutables et la documentation

- Processus : fournit une définition de l'ensemble des activités requises pour transformer en produit les besoins exprimés par les utilisateurs. Un processus offre un cadre générique à la création de projets.

Les 4 "P" du Processus unifié



Remarque : Les outils sont les logiciels permettant d'automatiser les activités définies par le processus

Variantes à Unified Process

- Agile Unified Process (AUP), a lightweight variation
- Rational Unified Process (RUP), the IBM / Rational Software development process
- Entrprise Unified Process (EUP), an extension of the Rationnal Unified Process
- Essential Uified Process (EssUP), a lightweight variation
- Rational Unified Process-System Engineering (RUP-SE), a version of RUP tailored by Rational Software for System Engineering
- Open Unified Process (OpenUP), the Eclipse Process Framework Software development process
- Oracle Unified Method (OUM), the Oracle development and implementation process
- 2TUP, une variante du Unied Process

RUP : Rational Unified Process
Processus Unifié de Rational

Processus Unifié de Rational

- **Rational Unified Process (RUP)** : est un processus de conception/développement de logiciel défini par Rational Software.

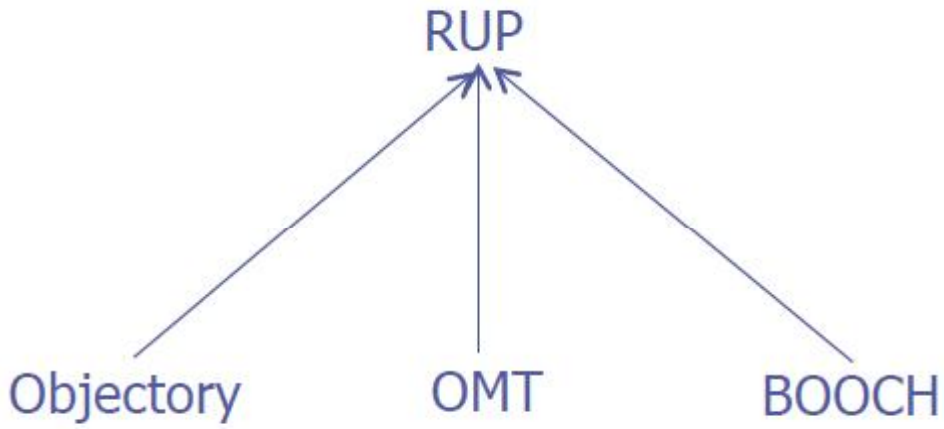
- Le Processus Unifié de Rational est un processus générique qui utilise UML comme langage de modélisation.

- RUP est l'une des plus célèbres implémentations de la méthode UP permettant de donner un cadre au développement logiciel.

Processus Unifié de Rational

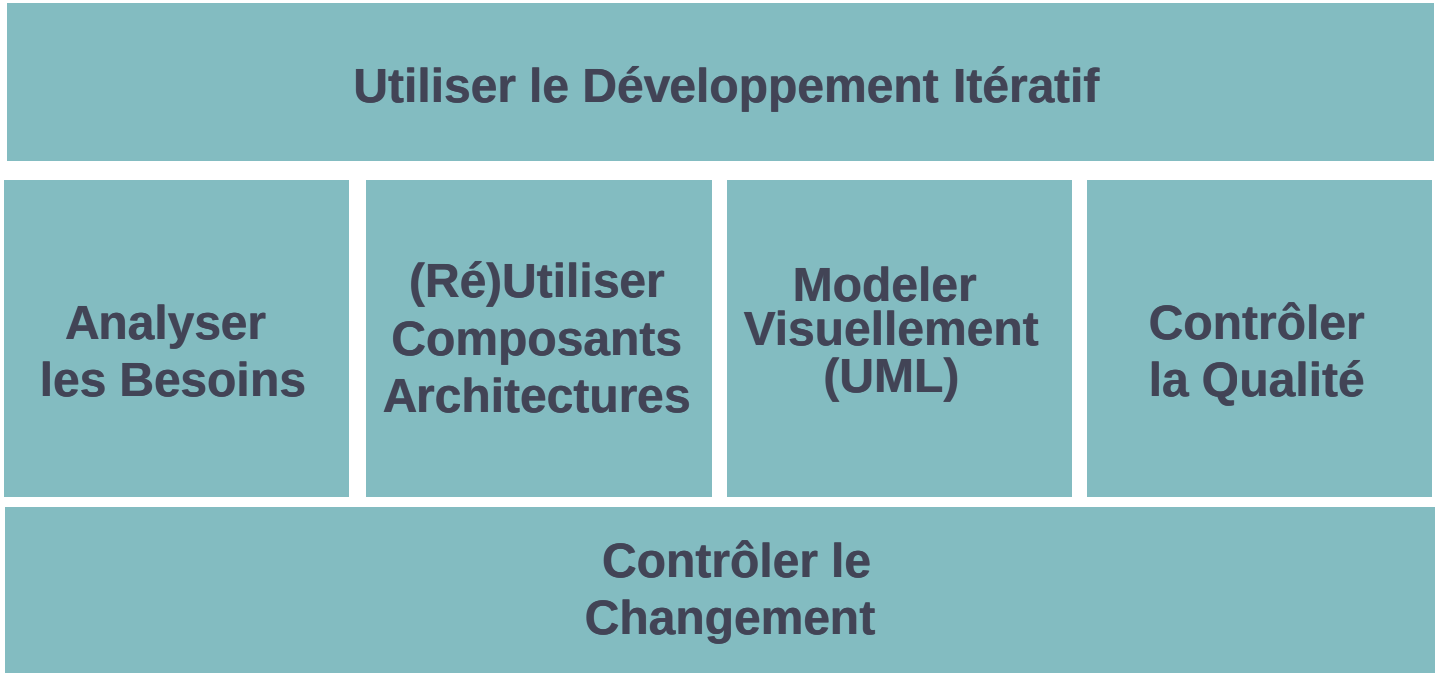
- Le développement est centré sur l'architecture
- Les activités de développement sont basés sur les scénarios d'utilisation.
- Le processus supporte les techniques orientés objets.
- Le processus est adaptable.
- Le processus favorise le contrôle de qualité et la gestion des risques

Qu'est ce que RUP ?



Le Processus Unifié de Rational permet les Meilleures Pratiques

Le processus Unifié Rational décrit comment appliquer les six directives de l'ingénierie logicielle



Le Processus Unifié de Rational permet les Meilleures Pratiques

- Les six meilleures pratiques fournissent les bases pour le Processus Unifié de Rational.
- Cependant, cette application nécessite des instructions étapes par étapes.
- Ces instructions sont fournies dans le Processus Unifié de Rational, qui comprend toutes les activités devant être appliquées pour construire un logiciel

UML et le processus unifié

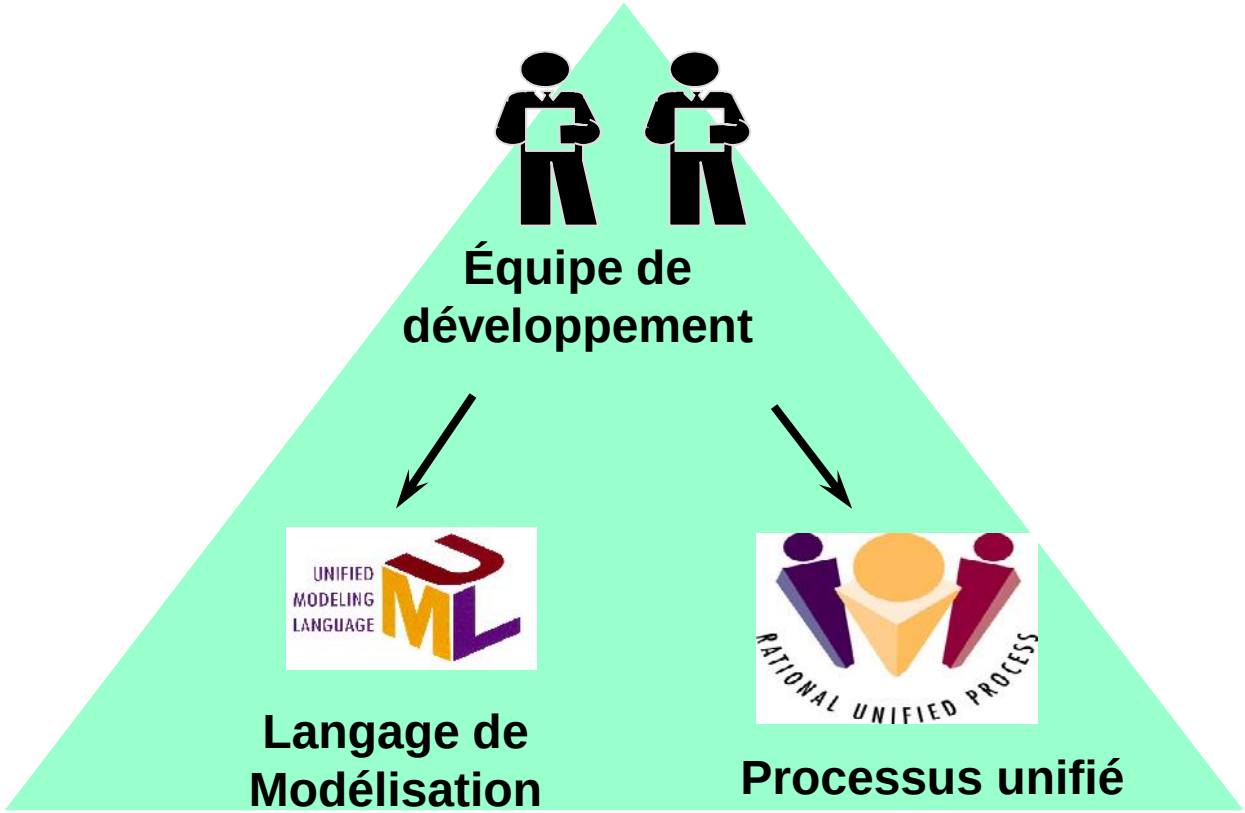
UML et processus unifié

- Si UML fournit un moyen standard de visualiser, spécifier, construire, documenter et communiquer les artefacts d'un système à logiciel prépondérant, nous admettons, qu'un tel langage doit être utilisé dans le cadre d'un processus logiciel complet.

- UML est un moyen, ce n'est pas une fin en soi. L'objectif est d'obtenir une application logicielle robuste, résistante et évolutive.

- Pour parvenir à ce but, il faut à la fois un processus et un langage…

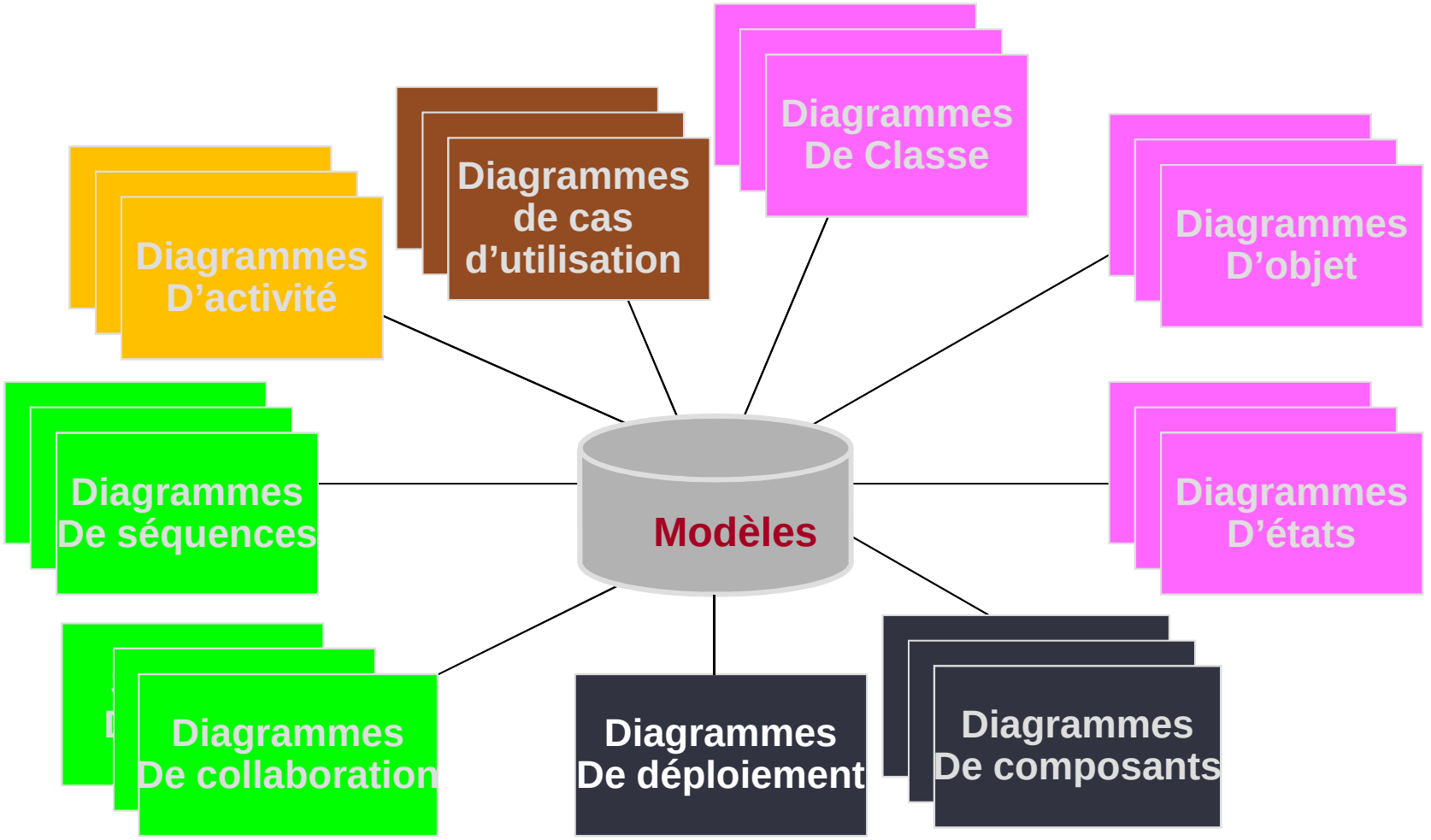
Dans la construction d'un système, un langage ne suffit pas.



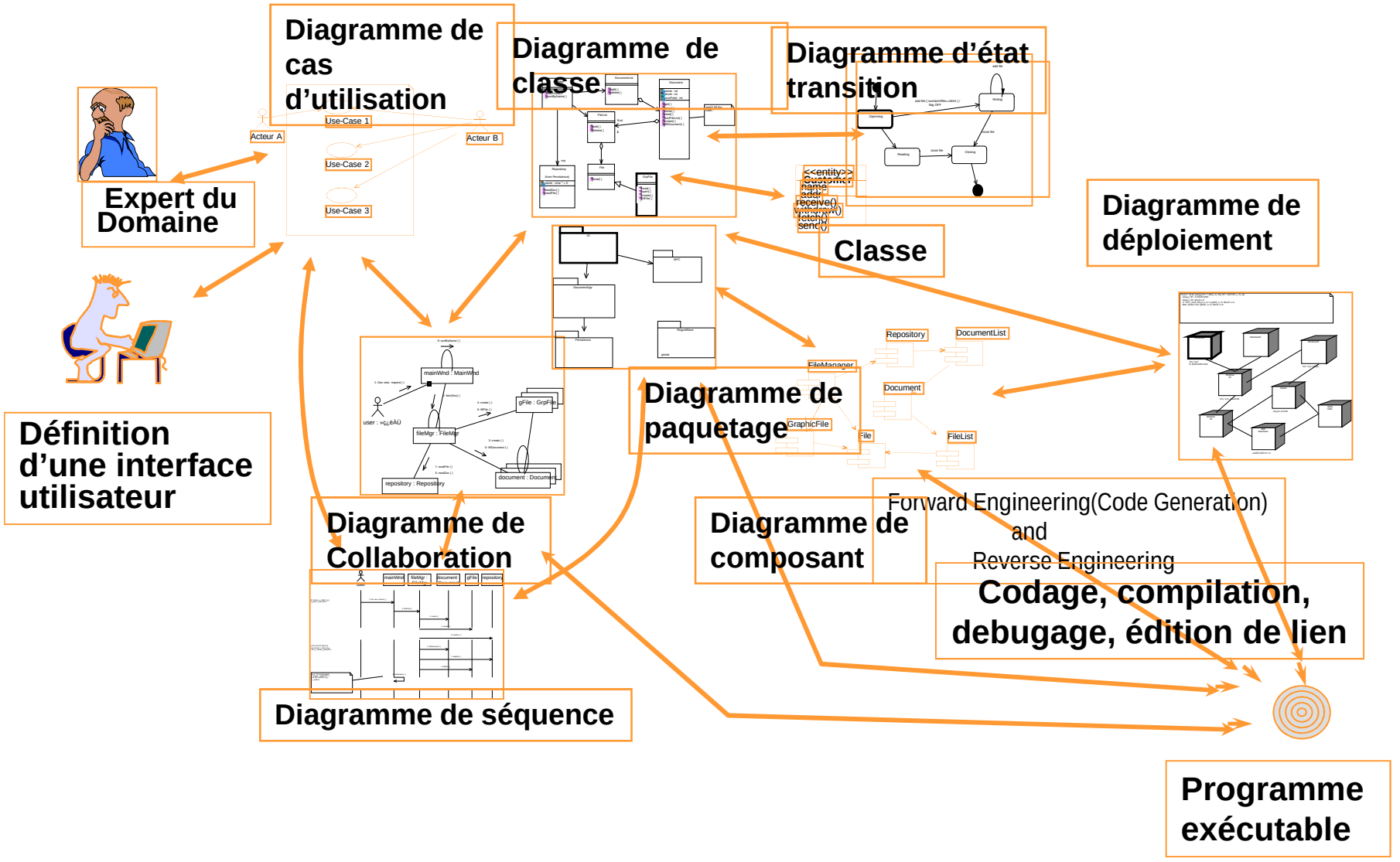
UML n'est pas un standard pour les processus de développement logiciel.

RUP est une démarche de développement qui est souvent utilisé conjointement au langage UML

UML fournit des diagrammes standardisés (UML 1.3)



Les diagrammes sont les artefacts clés



Les diagrammes sont les artefacts clés

- UML fournit un langage unique et commun de modélisation utilisable à travers plusieurs méthodes,
- Il définit le lien entre les coûts, les exigences et l'analyse, le design, l'implémentation, et les tests.
- UML facilite la communication entre tous les membres de l'équipe de développement.

RUP est piloté par les cas d’utilisation

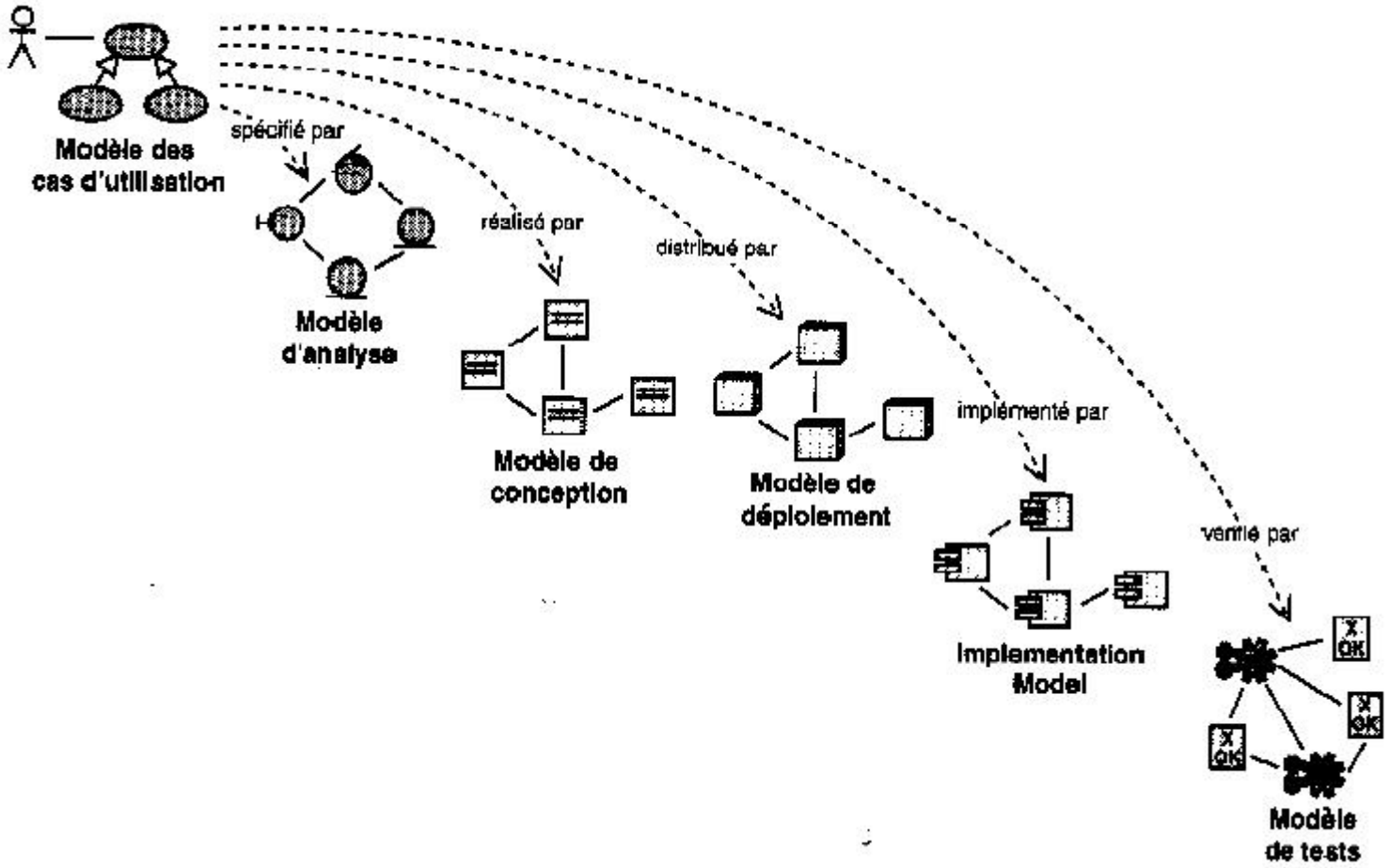
RUP est Piloté par les “cas d’utilisation”

Un processus guidé par les cas d'utilisation

- Capturer, organiser, et documenter les fonctionnalités et les contraintes requises
- Suivre et documenter les décisions et les compromis
- Les besoins du métier sont capturés et communiqués grâce aux cas d'utilisation
- Les cas d'utilisation sont des bons outils de planification

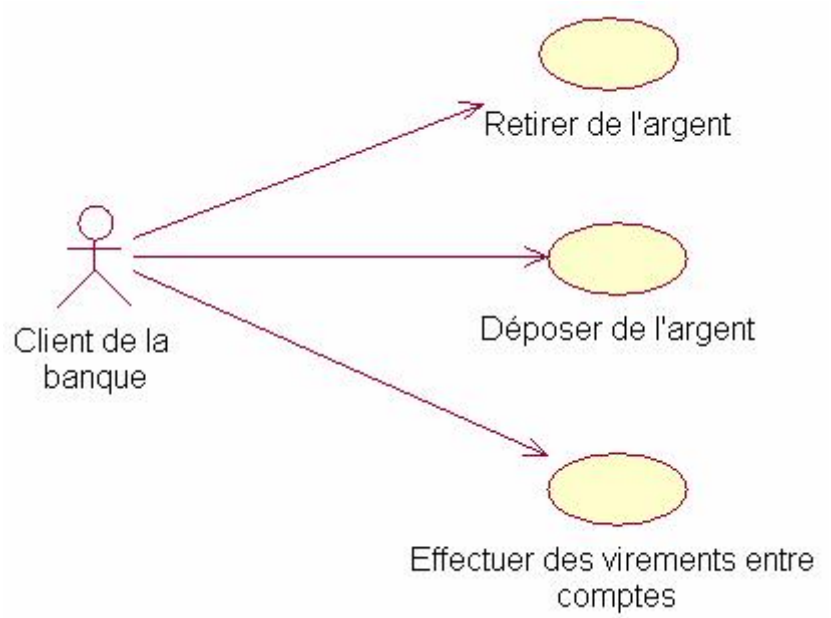
RUP est Piloté par les “cas d’utilisation”

à partir des cas d’utilisation , les développeurs créent une série de modèles UML.



Besoins fonctionnels

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



Spécification 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.

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 l'environnement d'implémentation choisi; comme un ORB, un kit de construction d'IHM ou un système de gestion de base de données.

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.

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…

- Pour les **premiers use-cases** (les plus risqués), il convient ici d'établir un :

  1. **Modèle du domaine** : comprend les concepts importants du domaine représentés sous forme graphique (sous la forme de diagrammes UML)

  2. **Modèle de conception**: représentation logique du système incluant les concepts et les autres représentations essentielles de l'architecture (classes non associées aux concepts du domaine par exemple)

- Cela demande plusieurs **itérations** du **micro-process**

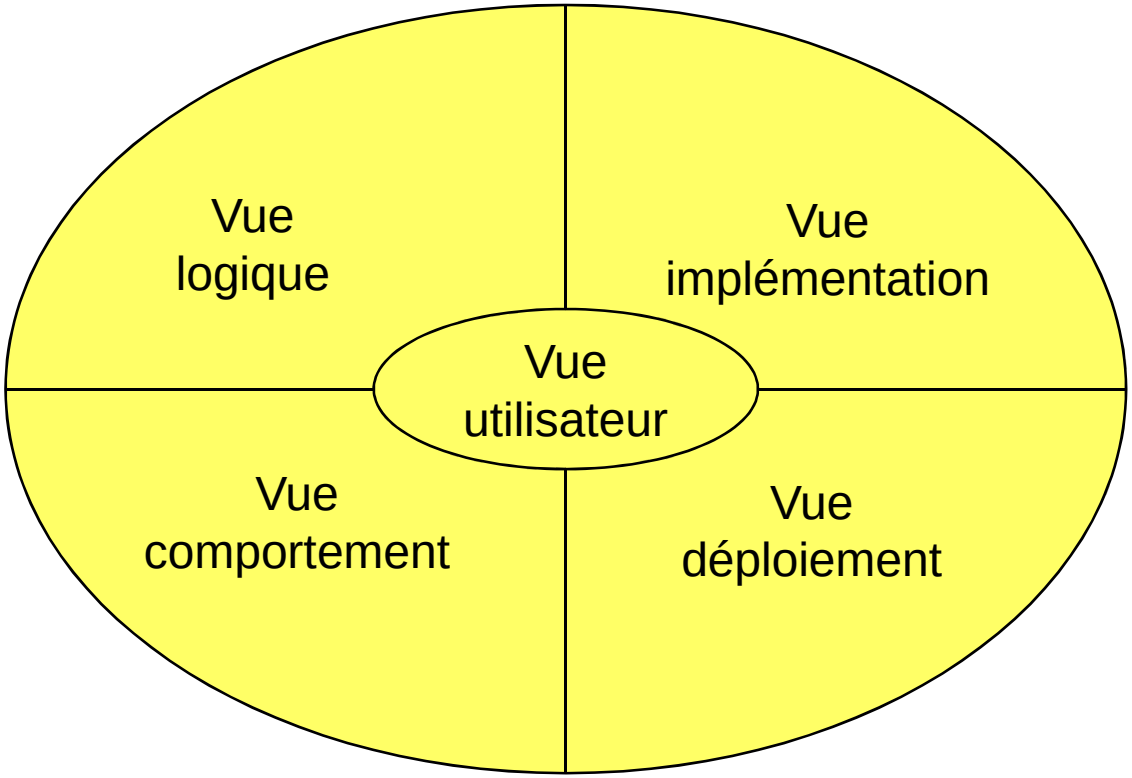
RUP est centré sur l'architecture

Architecture du système

Architecture et vues

Modélisation de différentes perspectives indépendantes et complémentaires

Architecture en couches et vues de Krutchen



Vue utilisateur

- La vue utilisateur présente le système du point de vue du propriétaire et de l'utilisateur final.
- La vue présente les buts et objectifs du système.
- La vue présente les requis et fonctionnalités du système.

Vue logique

- La vue logique présente les aspects statiques et structuraux du problème et de la solution.

Vue comportement

- La vue comportement présente les aspects dynamiques du problème et de la solution.
- La vue comportement présente les interactions et collaborations entre les éléments du système.

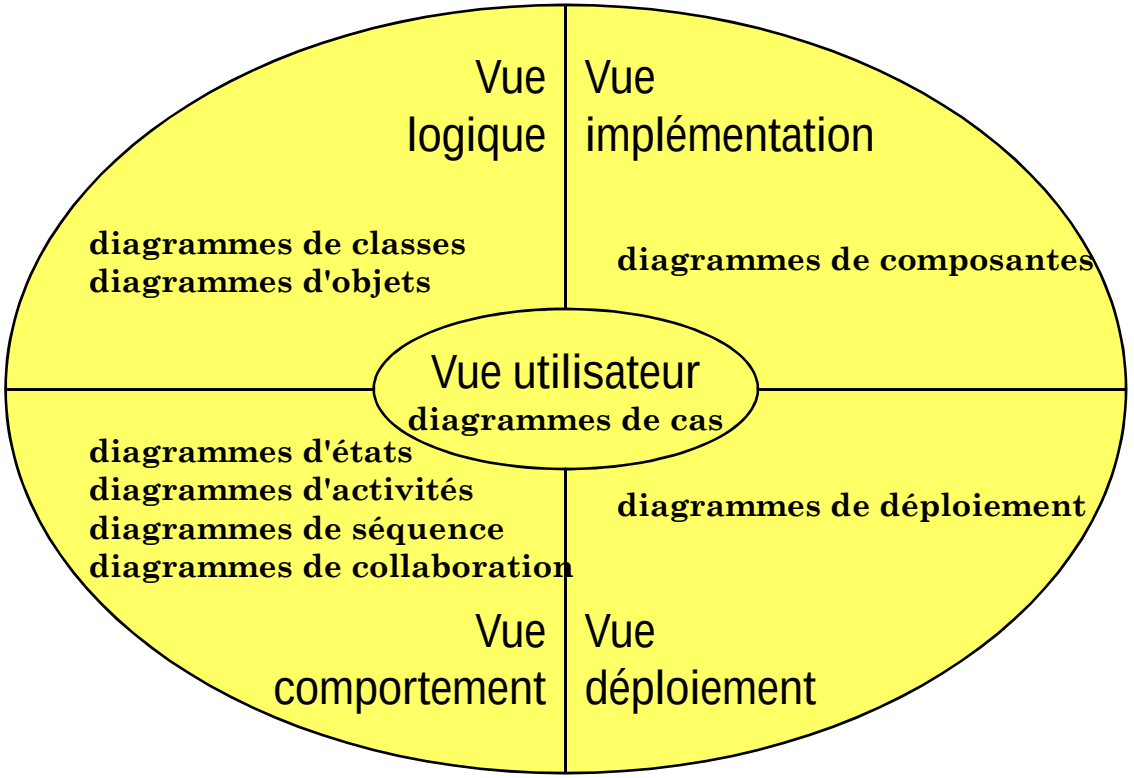
Vue implémentation

- La vue implémentation présente les aspects statiques, structuraux et dynamiques de la réalisation de la solution.
- La vue implémentation présente l'organisation du code de la solution.

Vue Déploiement

- La vue déploiement présente les aspects statiques et dynamiques de l'exécution de la solution.
- La vue déploiement présente les éléments physiques de la solution (processeurs, périphériques, ...)

Le processus de développement
Vues et diagrammes



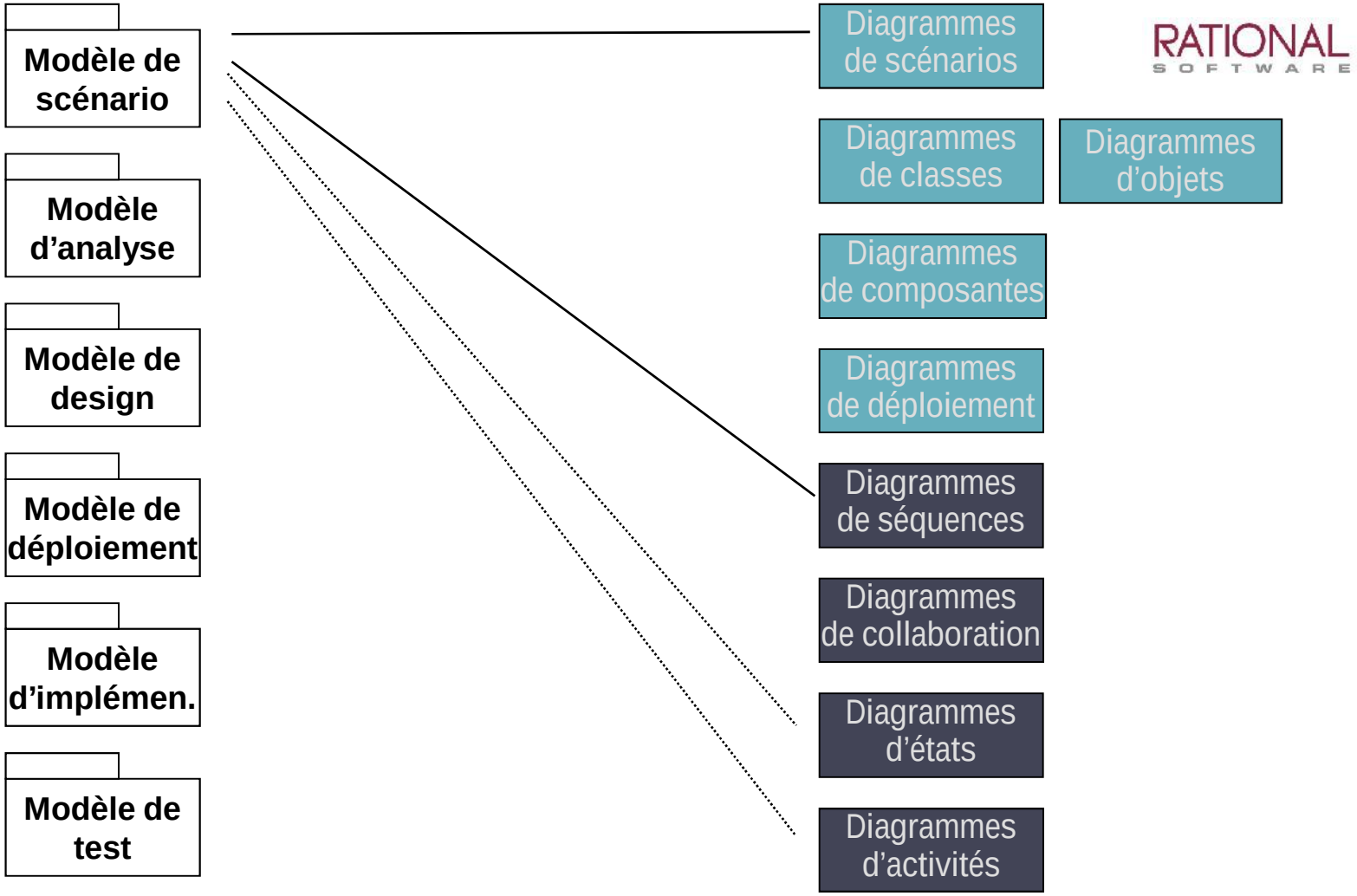
Les 9 modèles du Processus Unifié (1/2)

- 1. Modèle d'affaires
  - présente une abstraction de l'organisation
- 2. Modèle de domaine
  - présente le contexte du système
- 3. Modèle de scénarios
  - présente les fonctionnalités du systéme
- 4. Modèle d'analyse (optionnel)
  - présente un aperçu de la solution
- 5. Modèle de design
  - présente l'architecture de la solution

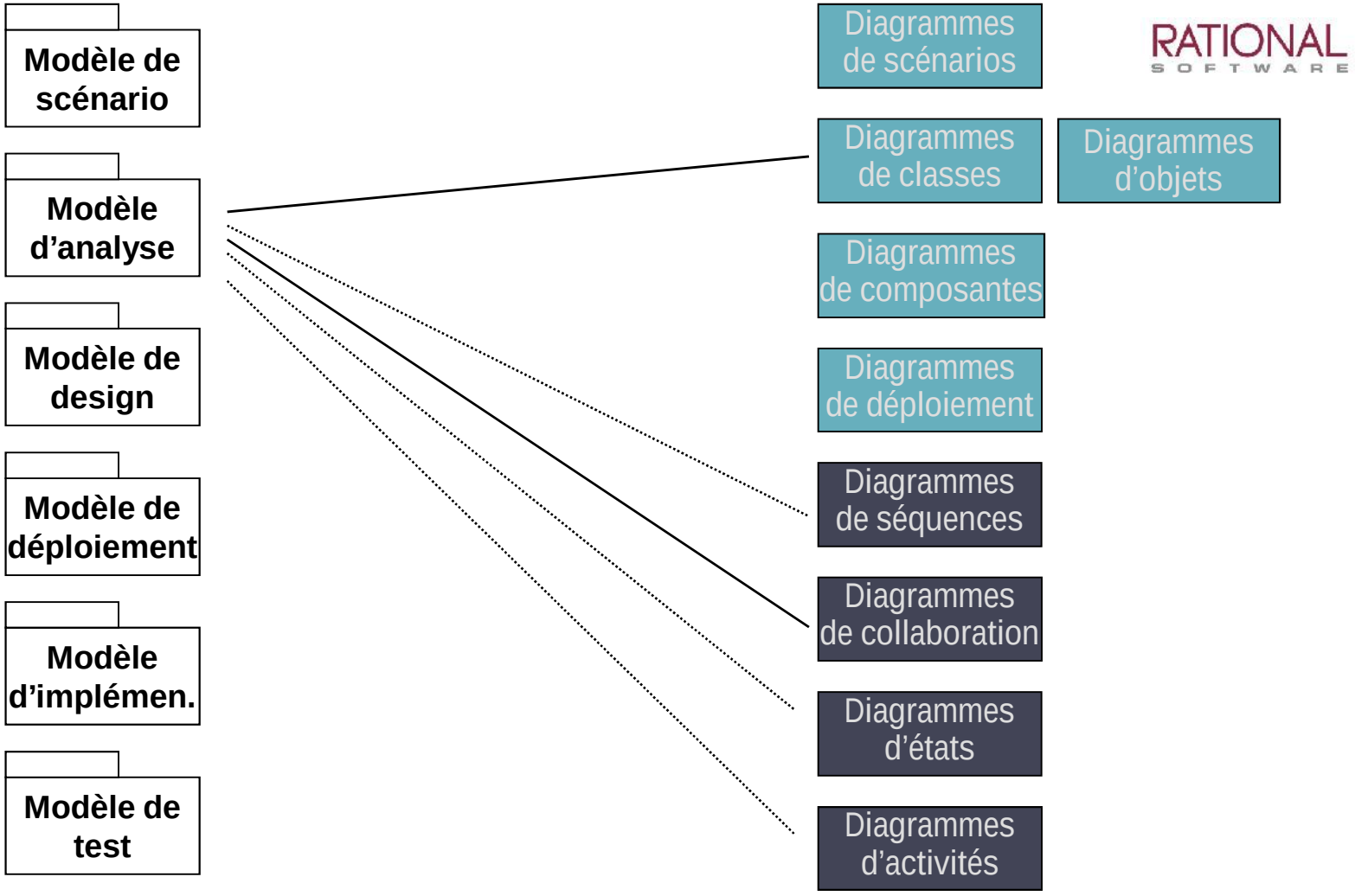
Les 9 modèles du Processus Unifié (2/2)

- 6. Modèle de processus (optionnel)
  - présente les mécanismes de concurrence et de synchronisation
- 7. Modèle de déploiement
  - présente la topologie des composantes physiques
- 8. Modèle d'implémentation
  - présente l'organisation des éléments à assembler et livrer
- 9. Modèle de test
  - présente les scénarios de test et validation

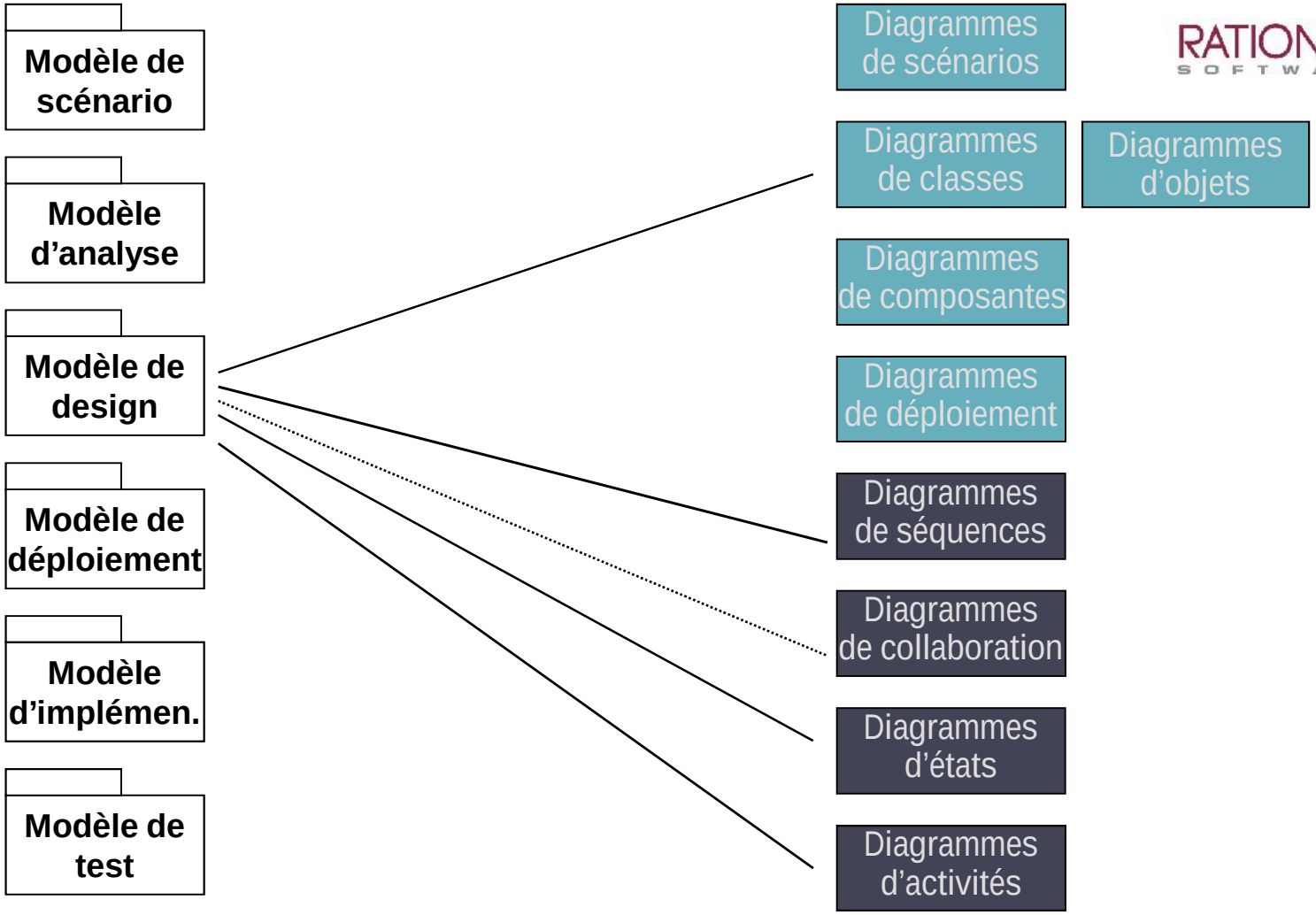
Modèle de scénario



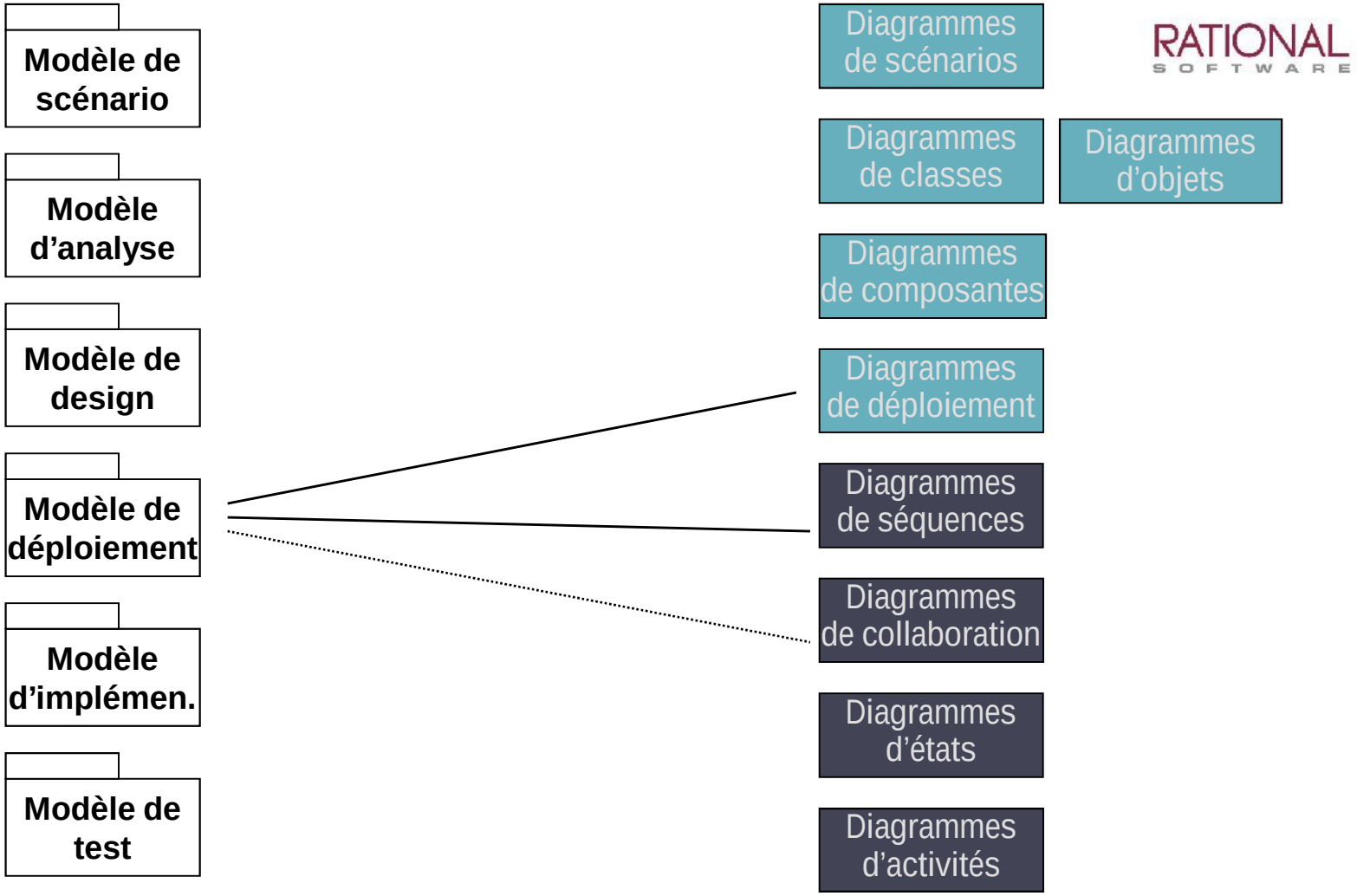
Modèle d'analyse



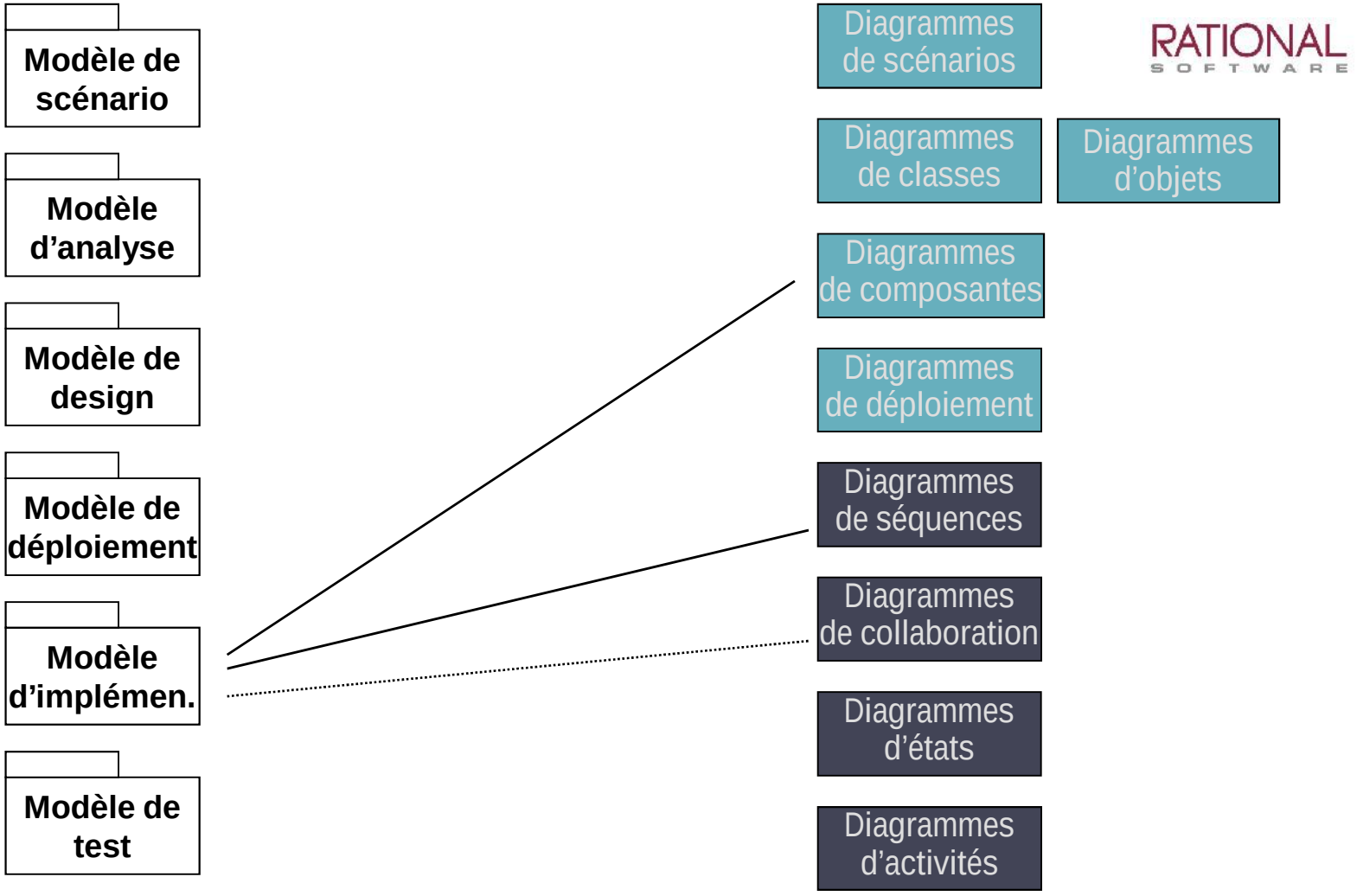
Modèle de design



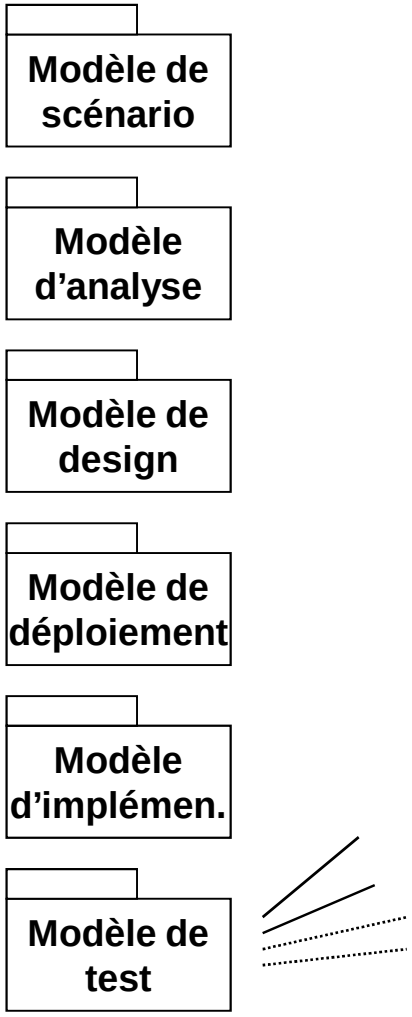
Modèle de déploiement



Modèle d'implémentation



Modèle de test



Diagrammes de scénarios

RATIONAL SOFTWARE

Diagrammes de classes

Diagrammes d'objets

Diagrammes de composantes

Diagrammes de déploiement

Diagrammes de séquences

Diagrammes de collaboration

Diagrammes d'états

Diagrammes d'activités

Les étapes du processus unifié Rational

- Capture des besoins
- Analyse
- Design
- Implémentation
- Test

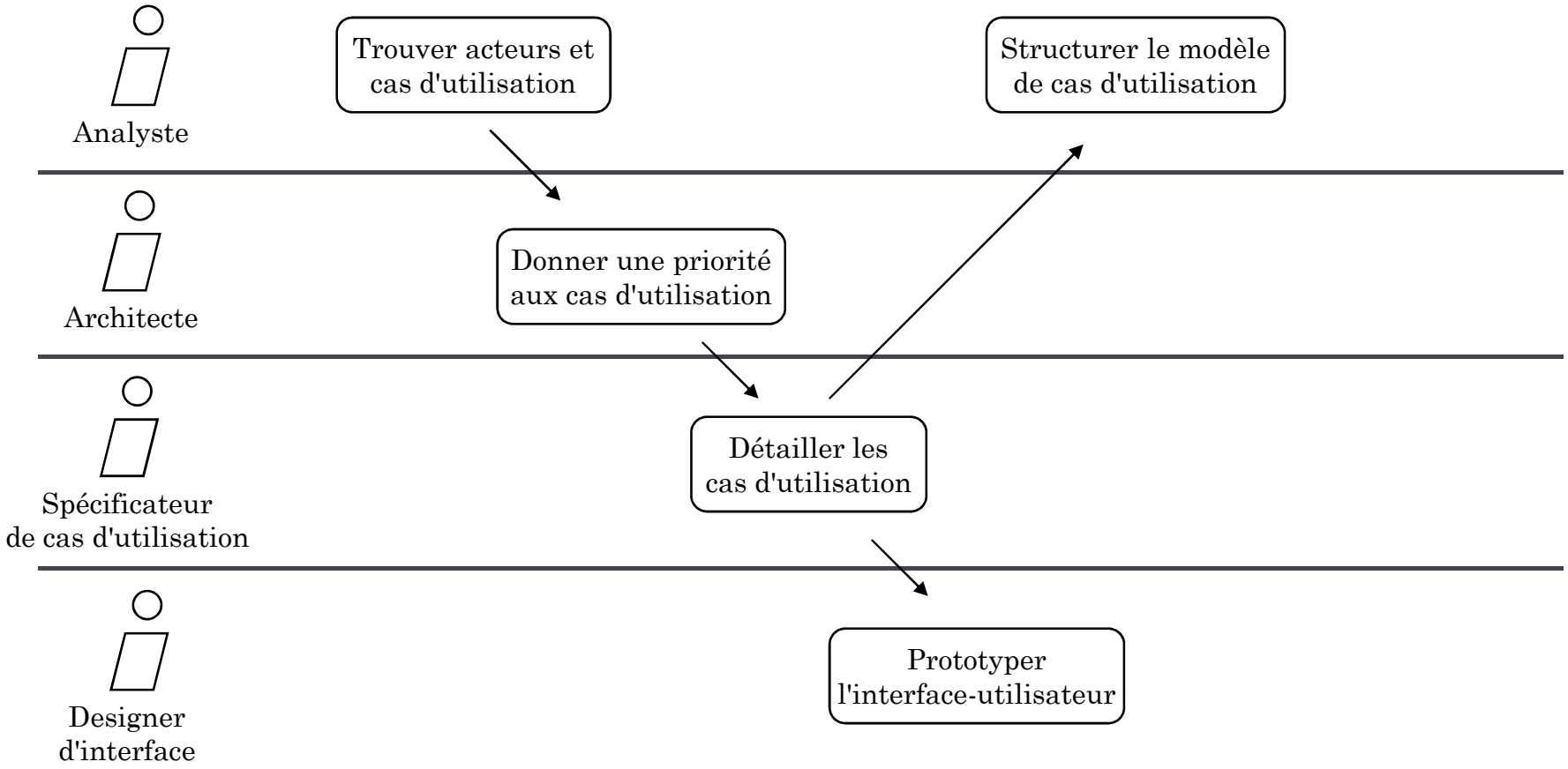
Capture des besoins
Les produits

- Modèle de domaine
- Modèle des cas d'utilisation
  - Acteur
  - Cas d'utilisation
- Flux d'événements
  - Description textuelle des actions du système quand le cas d'utilisation est exécuté
- Description de l'architecture
- Glossaire
- Prototype d'interface

Capture des besoins
Les acteurs

- Analyste
- Spécificateur de cas d'utilisation
- Designer d'interface
- Architecte

Capture des besoins
Le processus



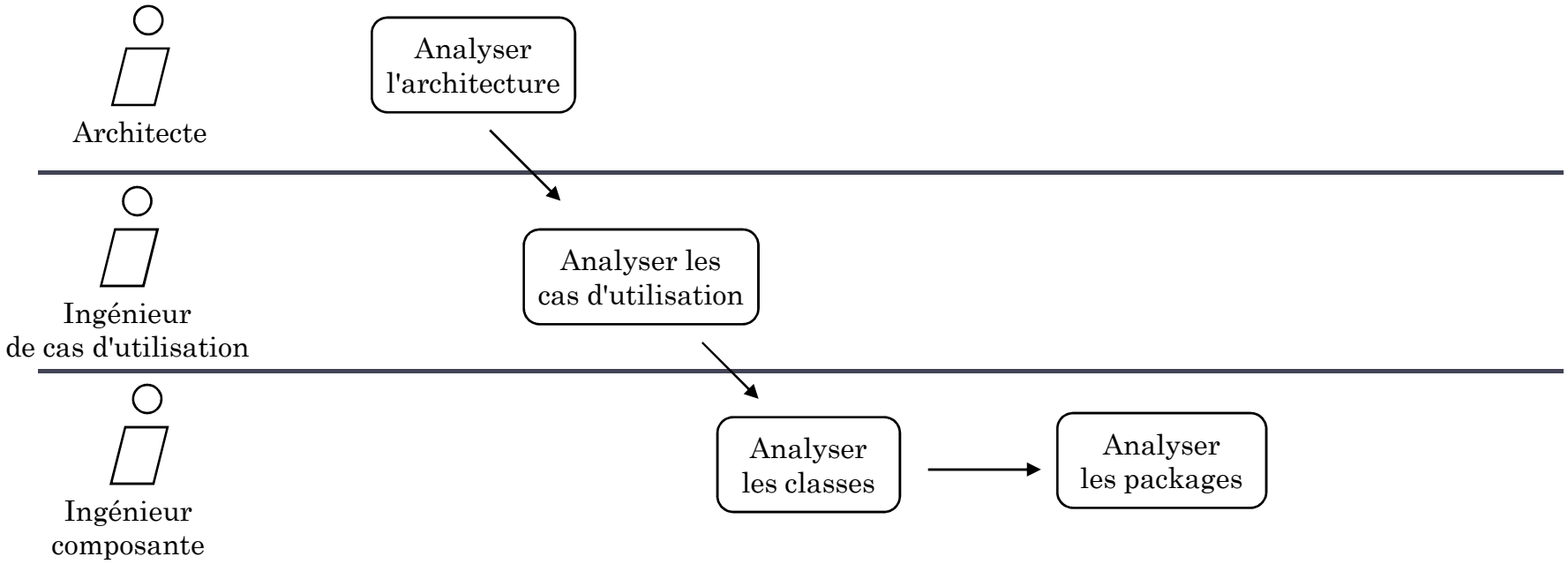
Analyse
Les produits

- Architecture d'analyse
- Diagrammes de classe
  - Classe de frontière (interface)
  - Classe d'entités
  - Classe de contrôle
- Diagrammes d'interaction

Analyse
Les acteurs

- Architecte
- Ingénieur de cas d'utilisation
- Ingénieur de composante

Analyse
Le processus



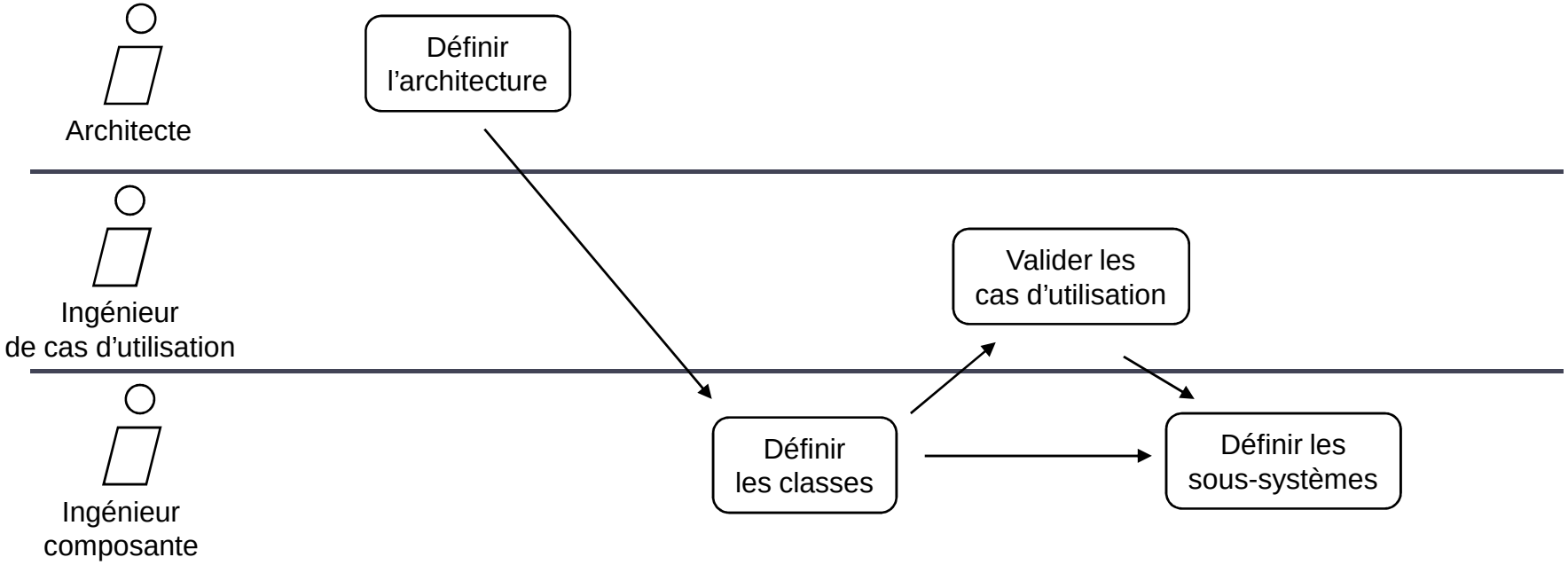
Design
Les produits

- Architecture de design
- Diagrammes de classe d'implémentation organisés en sous-système
- Diagrammes d'interaction
- Modèle de déploiement

Design
Les acteurs

- Architecte
- Ingénieur de cas d'utilisation
- Ingénieur de composante

Design
Le processus



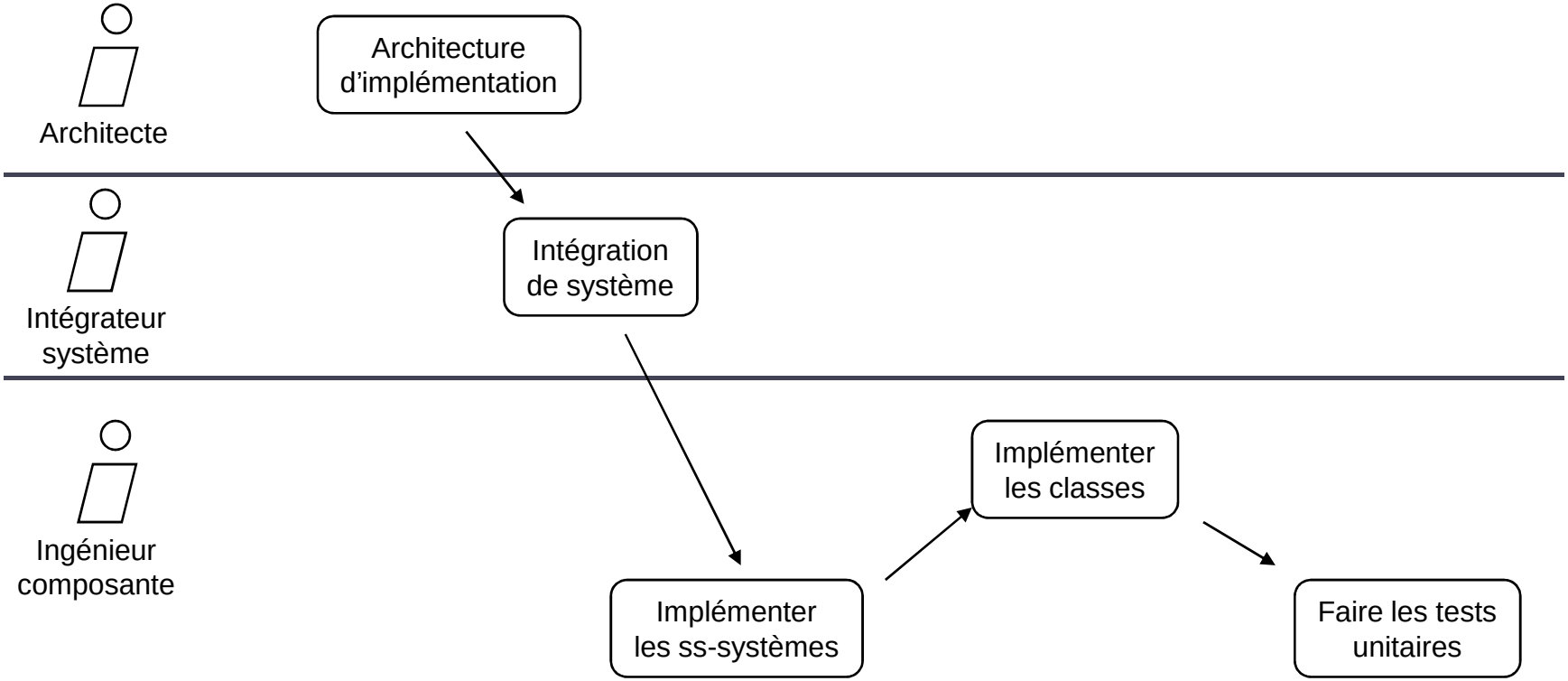
Implémentation
Les produits

- Architecture d'implémentation
- Composantes organisées en sous-systèmes
- Plan d'intégration

Implémentation
Les acteurs

- Architecte
- Intégrateur système
- Ingénieur de composante

Implémentation
Le processus



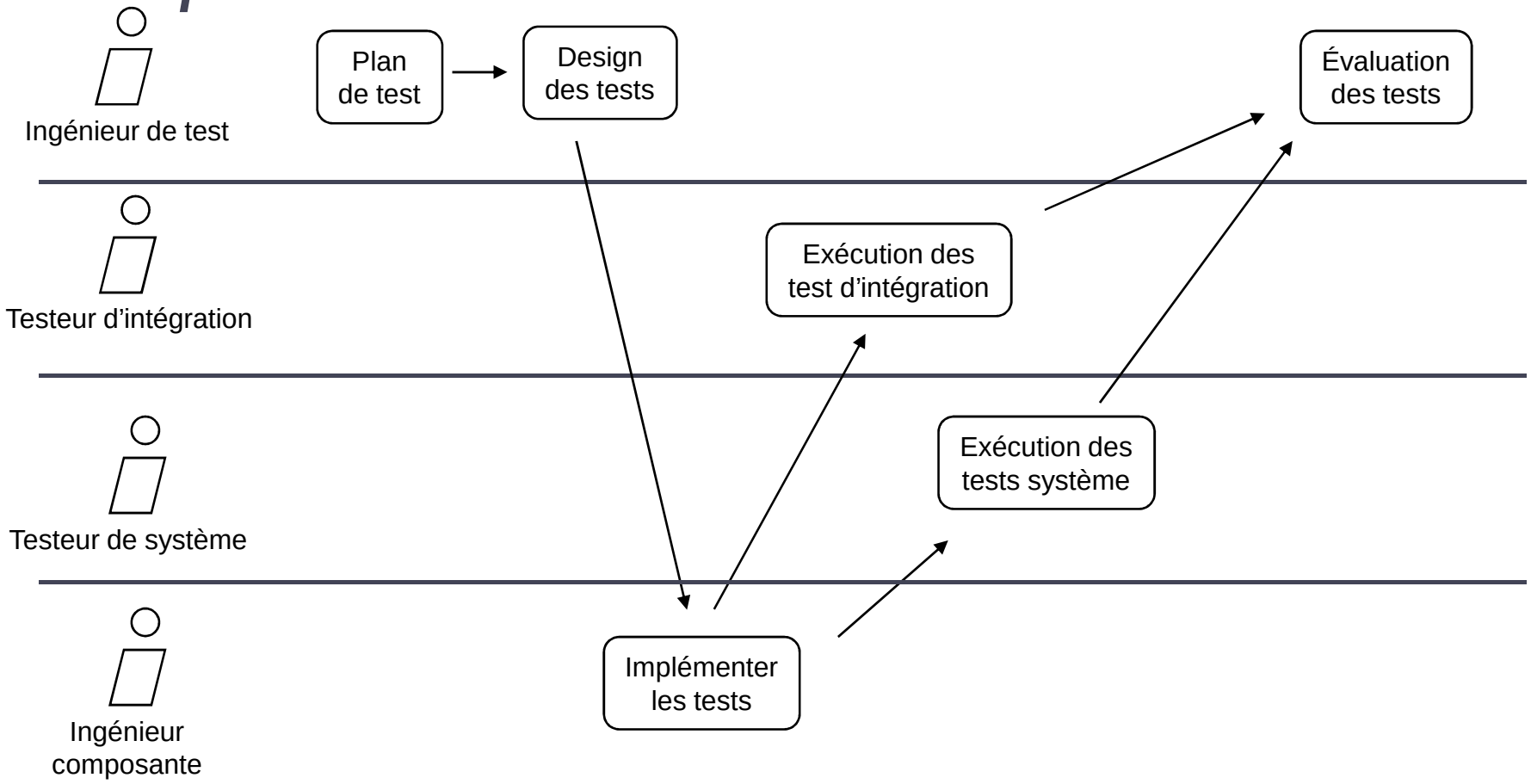
Test
Les produits

- Plan de tests
- Cas de test
- Procédures de test
- Composantes de test

Test
Les acteurs

- Ingénieur de test
- Testeur d'intégration
- Testeur de système
- Ingénieur de composante

Test
Le processus



Sommaire
Le processus unifié Rational

Sommaire
Le processus unifié Rational

Nous avons vu le Processus Unifié Rational :

- Le processus est itératif.
- Il est centré sur l'architecture, basé sur les scénarios d'utilisation et adaptable.

Sommaire
Le processus unifié Rational

- 6 étapes:
  - Besoins,
  - Analyse,
  - Design,
  - Implémentation,
  - Test,
  - Déploiement

Sommaire
Le processus unifié Rational

- 5 vues:
  - Vue utilisateur
  - Vue logique
  - Vue processus
  - Vue Implémentation
  - Vue déploiement

Sommaire
Le processus unifié Rational

- 9 modèles:
  - Modèle d'affaires
  - Modèle de domaine
  - Modèle de scénarios
  - Modèle d'analyse (optionnel)
  - Modèle de design
  - Modèle de processus (optionnel)
  - Modèle de déploiement
  - Modèle d'implémentation
  - Modèle de test

Sommaire

Le processus unifié Rational

- 9 diagrammes:
  - Diagramme de classe
  - Diagramme d'objets
  - Diagramme de cas
  - Diagramme de séquence
  - Diagramme de collaboration
  - Diagramme d'états
  - Diagramme d'activités
  - Diagramme de composantes
  - Diagramme de déploiement

Sommaire
Le processus unifié Rational

- Le tableau suivant donne un exemple de tableau d'analyse multidimensionnelle, permettant de visualiser sous forme de tableau de bord l'état des différents indicateurs, synthétisés à des niveaux plus ou moins élevés du modèle RUP

Méthodes agiles :
XP : eXtreme Programming

eXtreme Programming

- eXtreme Programming, une méthode basée sur des pratiques qui sont autant de boutons poussés au maximum
- Méthode qui peut sembler naturelle mais concrètement difficile à appliquer et à maîtriser :
  - Réclame beaucoup de discipline et de communication (contrairement à la première impression qui peut faire penser à une ébullition de cerveaux individuels).
  - Aller vite mais sans perdre de vue la rigueur du codage et les fonctions finales de l'application.
- Force de XP : sa simplicité et le fait qu'on va droit à l'essentiel, selon un rythme qui doit rester constant.

Valeurs d'XP

- Communication
  - XP favorise la communication directe, plutôt que le cloisonnement des activités et les échanges de documents formels.
  - Les développeurs travaillent directement avec la maîtrise d'ouvrage

- Feedback
  - Les pratiques XP sont conçues pour donner un maximum de feedback sur le déroulement du projet an de corriger la trajectoire au plus tôt.

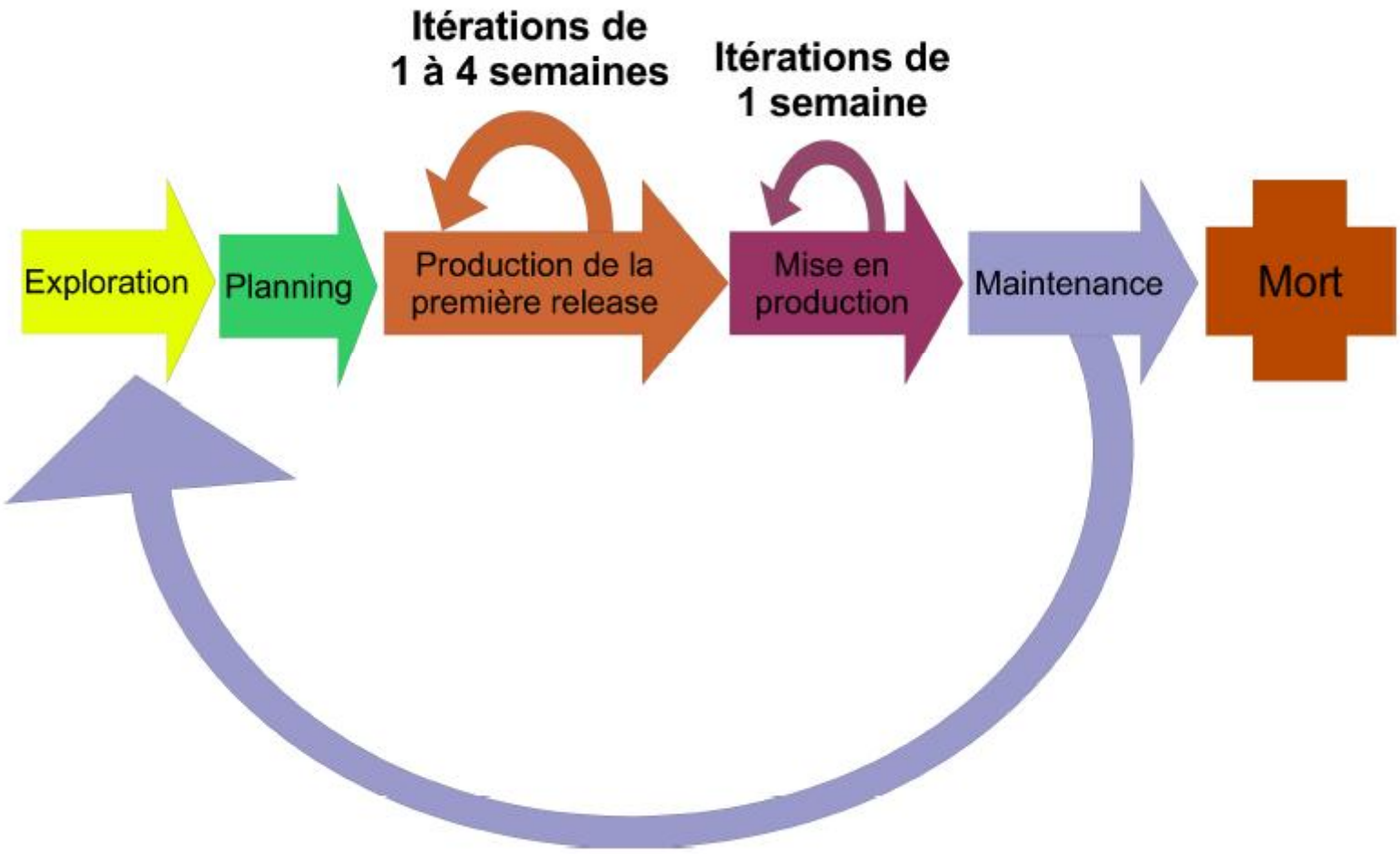
- Simplicité :
  - Du processus
  - Du code

- Courage :
  - d'honorer les autres valeurs
  - de maintenir une communication franche et ouverte
  - d'accepter et de traiter de front les mauvaises nouvelles.

Pratiques d'XP

- XP est fondé sur des valeurs, mais surtout sur 13 pratiques réparties en 3 catégories
  - Gestion de projets
  - Programmation
  - Collaboration

Cycle de vie XP



XP vs RUP

- **Inconvénients de XP**
  - Focalisation sur l'aspect individuel du développement, au détriment d'une vue globale et des pratiques de management ou de formalisation
  - Manquer de contrôle et de structuration, risques de dérive
- **Inconvénients de RUP**
  - Fait tout, mais lourd, usine à gaz
  - Parfois difficile à mettre en œuvre de façon spécifique.

- XP pour les petits projets en équipe de 12 max
- RUP pour les gros projets qui génèrent beaucoup de documentation

Bibliographie

- « The RUP, an Introduction »

P. Kruchten, Addison-Wesley 2000

- « The unified Software Developement Process »

I. Jacobson, G. Booch, J. Rumbaugh, Addison-Wesley 1999

- « Modélisation Objet avec UML »

P.-A. Muller, N. Gaertner, Eyrolles, 2002

http://www-306.ibm.com/software/rational/

- Rational Unified Process :

http://www-306.ibm.com/software/awdtools/rmc/

- Software Architect

http://www-306.ibm.com/software/awdtools/architect /swarchitect /

"Agile and Iterative Development: A Manager's Guide", Craig Larman, Addison Wesley/Pearson Education, 2003