

Le Processus Unifié

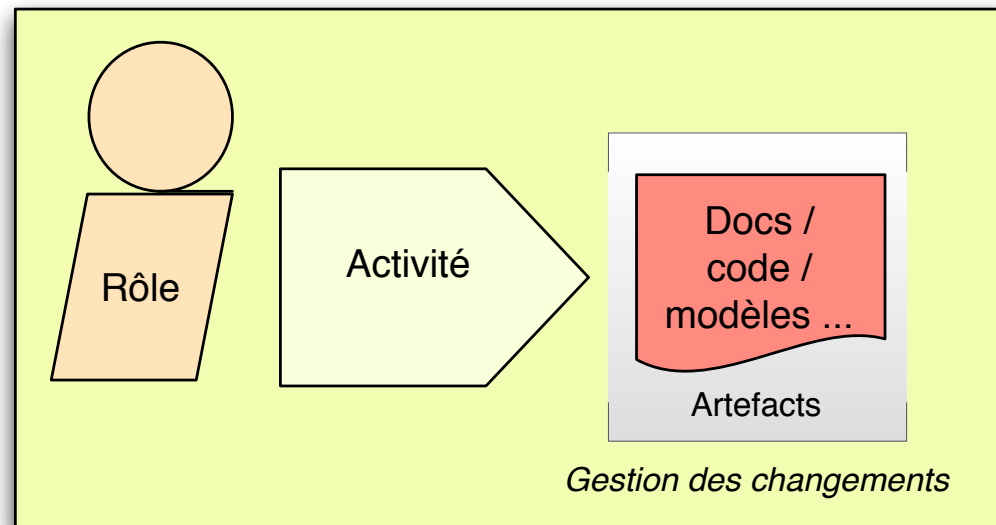
Une Démarche Orientée Modèle

jérémie.guiochet@laas.fr



Sommaire

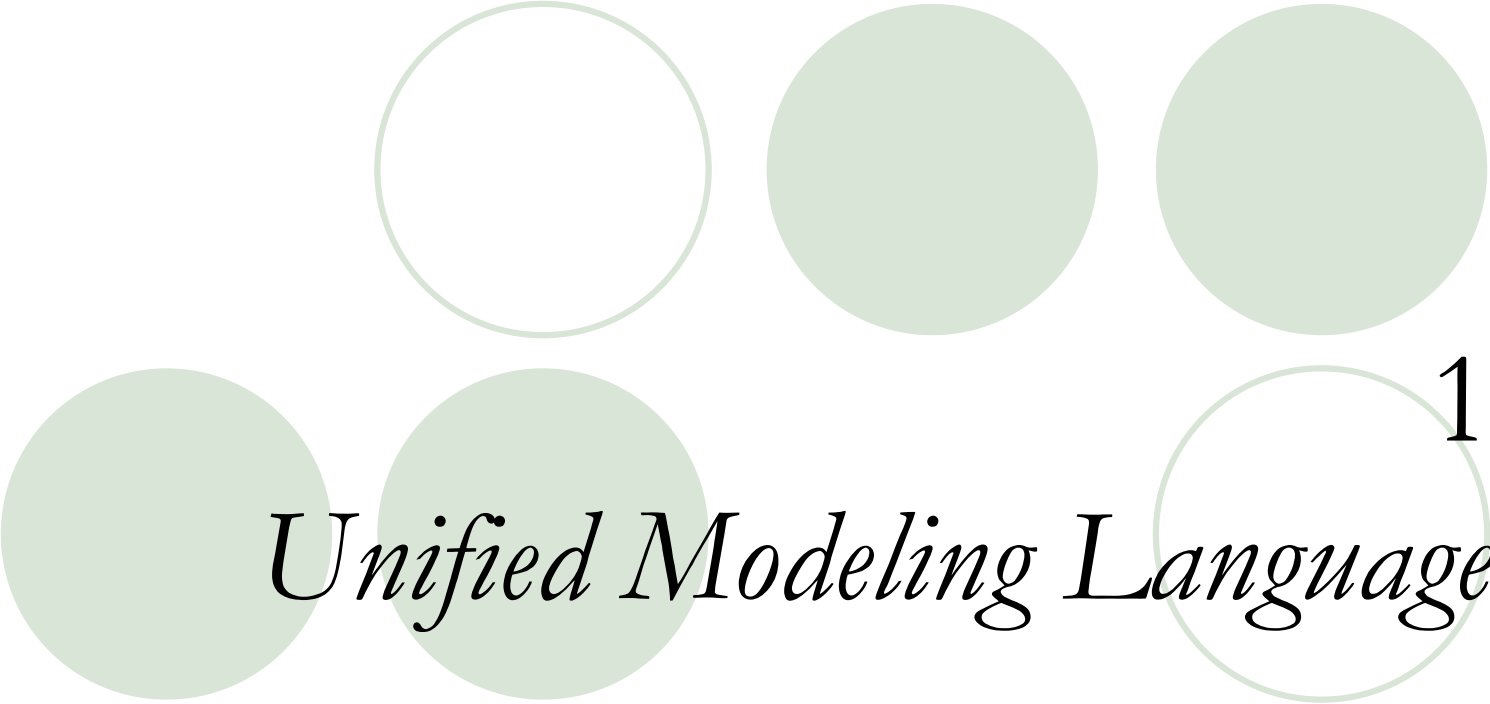
- Partie 1 : UML et processus unifié
- Partie 2 : Artefacts
- Partie 3 : Enchaînement d'itérations
- Partie 4 : Rôles
- Partie 5 : Gestion de la configuration et des changements
- Annexes
- Bibliographie





PARTIE 1

Rappels et compléments UML / Processus Unifié



*Unified Modeling Language*¹

Diagrammes d'UML2.0

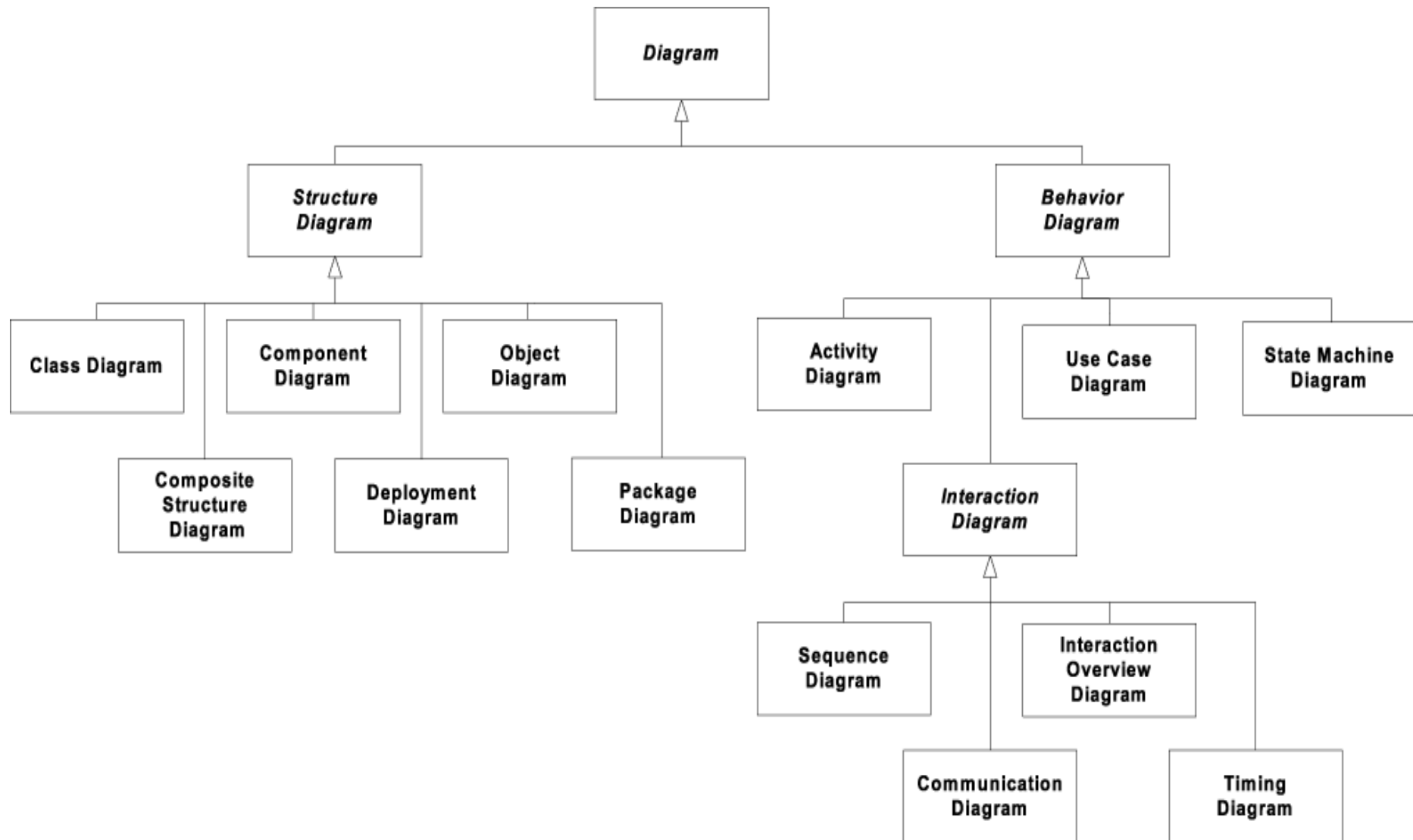


Diagramme d'objets

- représente *les objets et leurs relations* (correspond à un diagramme de collaboration simplifié, sans représentation des envois de messages).

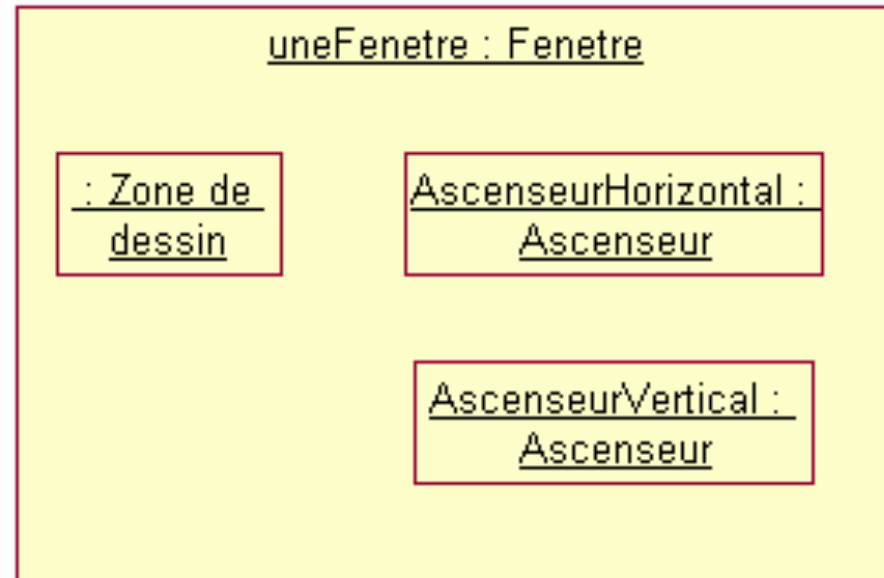


Diagramme de classes

- Représente *la structure statique* en terme de classes et de relations.

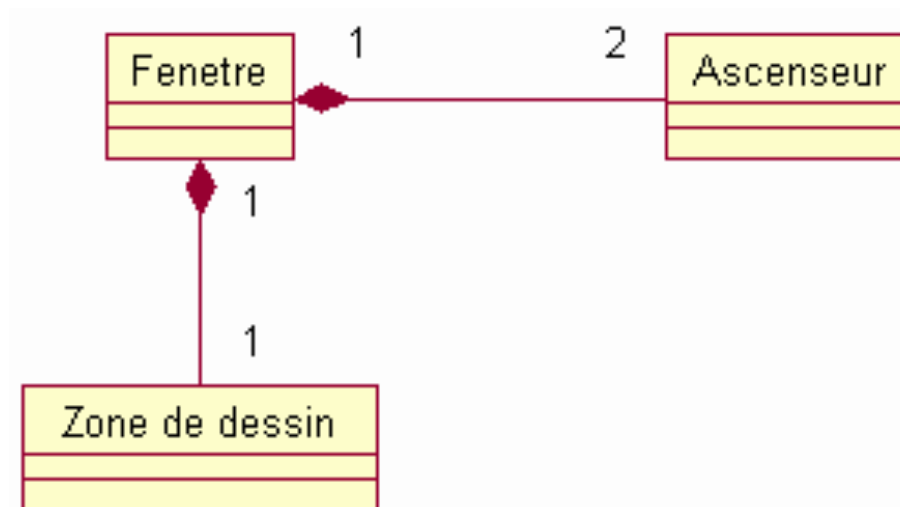


Diagramme de composants

- Représente les *composants physiques* d'une application.

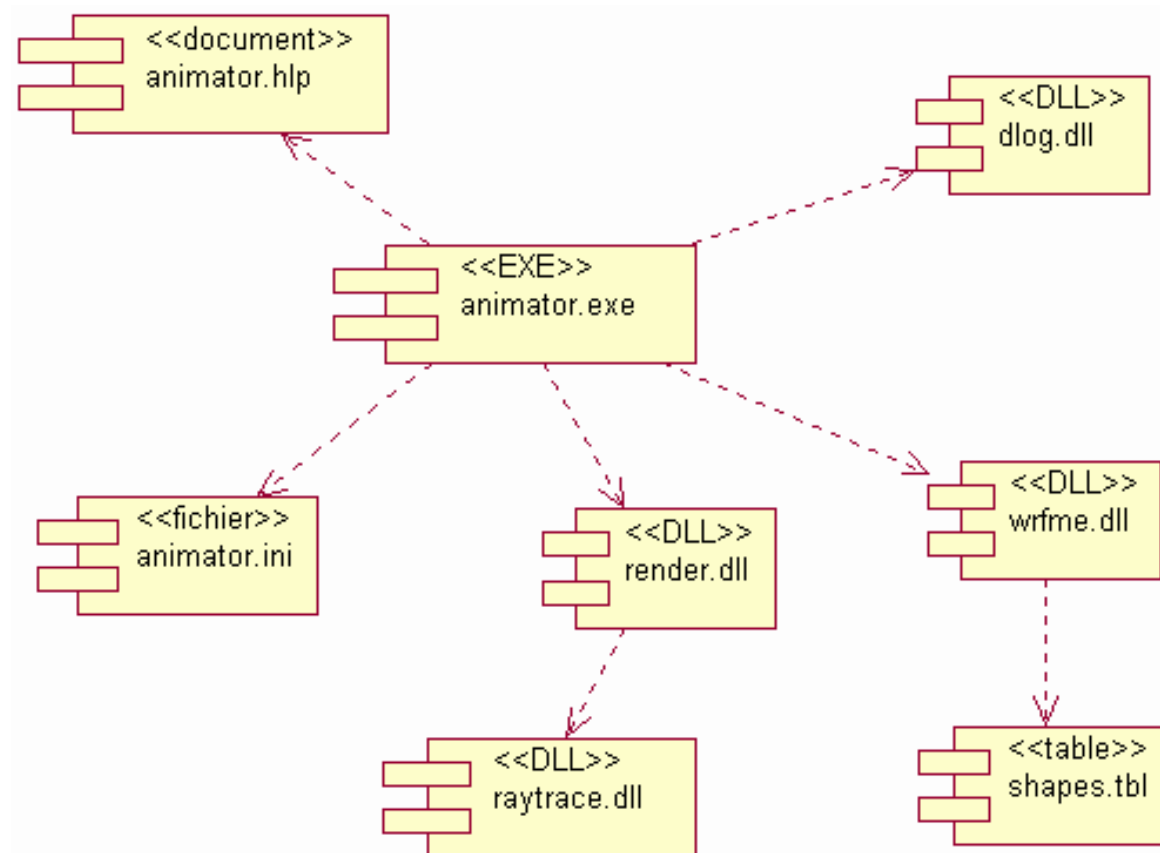


Diagramme de déploiement

- Représente le *déploiement des composants sur les dispositifs matériels.*

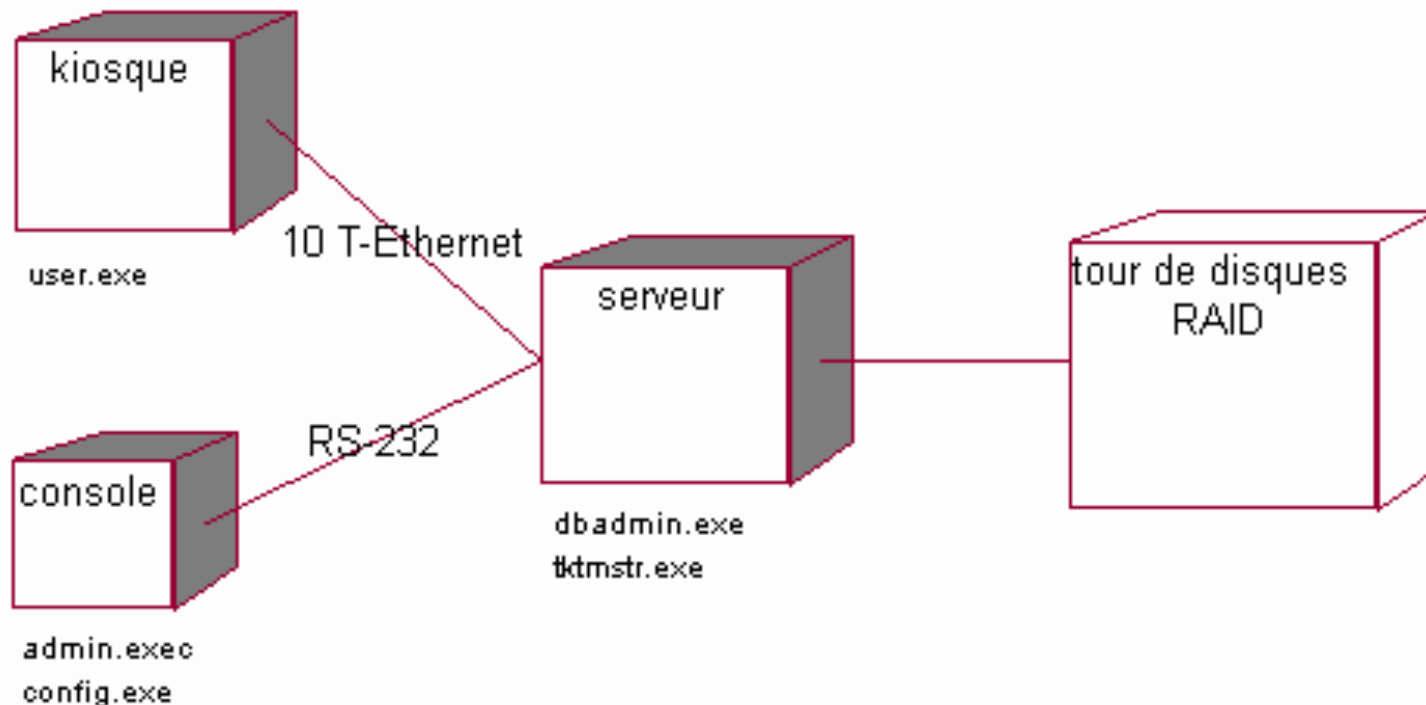


Diagramme de cas d'utilisation

- *Représente les objectifs en terme de fonctionnalités du système du **point de vue de l'utilisateur**.*

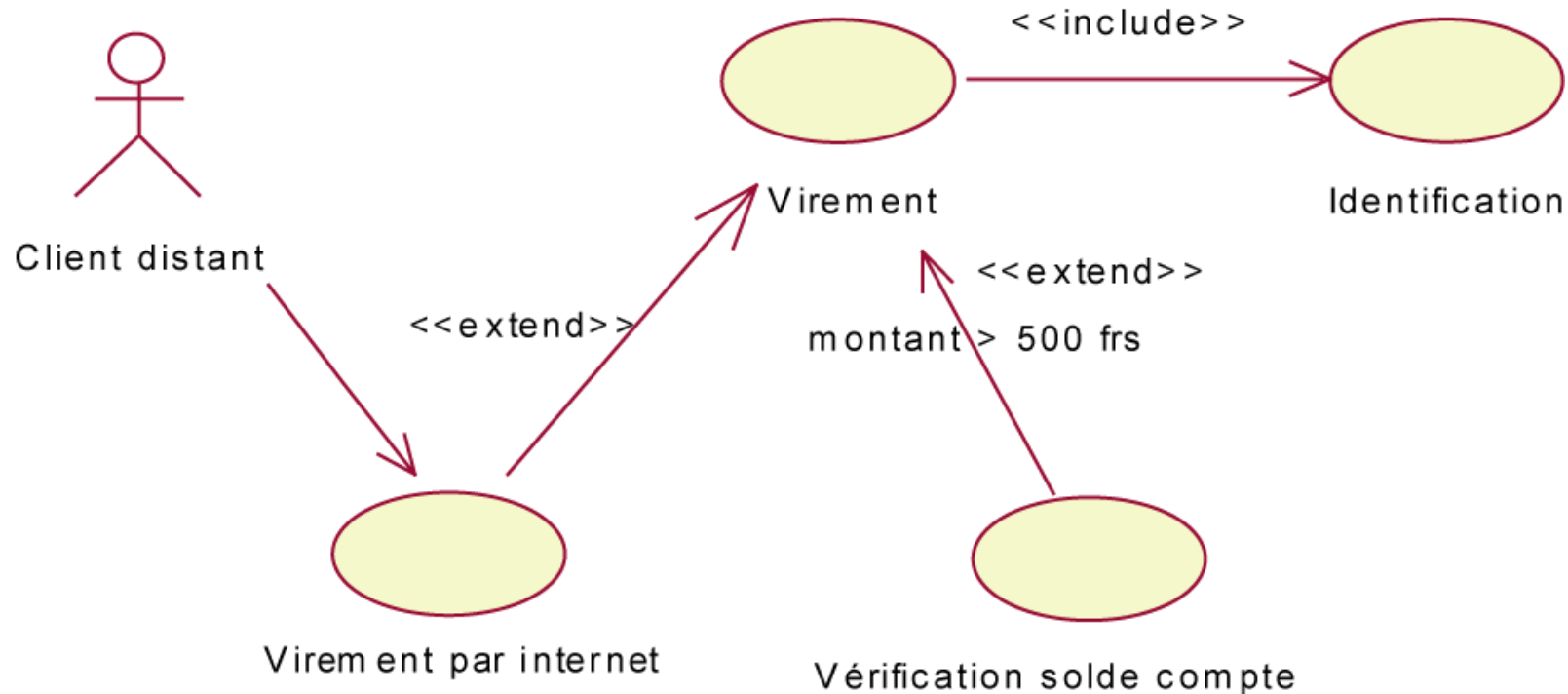


Diagramme de séquence

- Représentation *temporelle* des objets et de leurs interactions.

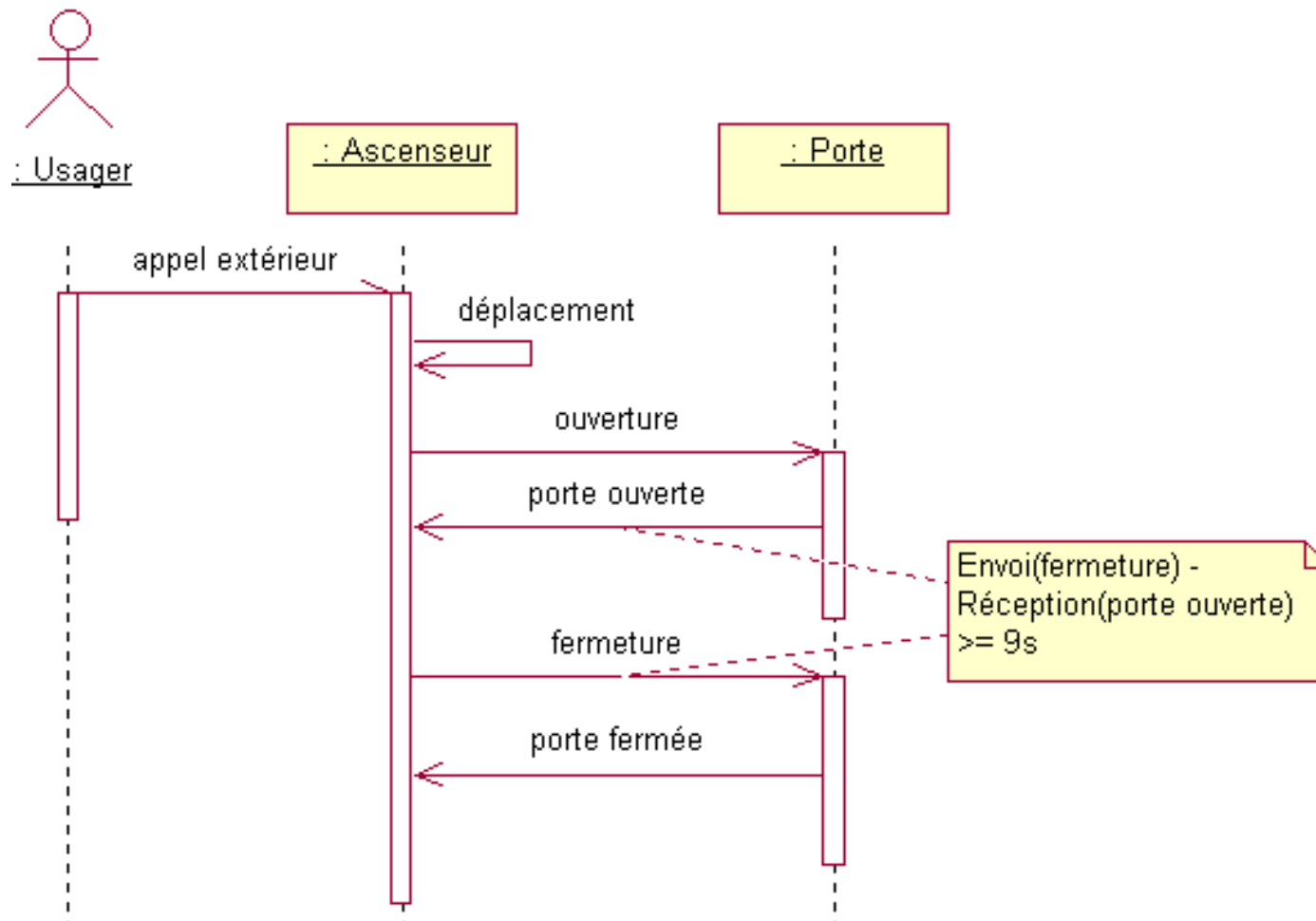


Diagramme de communication

- Représentation *spatiale* des objets, des liens et des interactions.

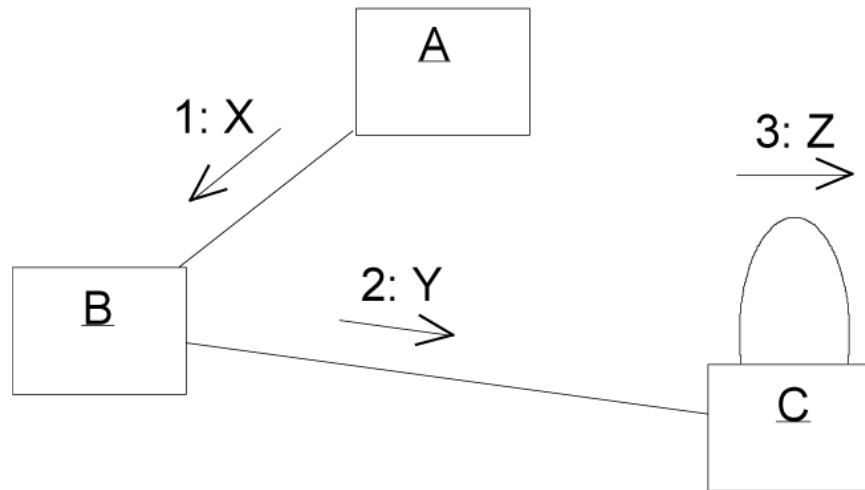


Diagramme d'états-transitions

- Représente *le cycle de vie* d'un objet

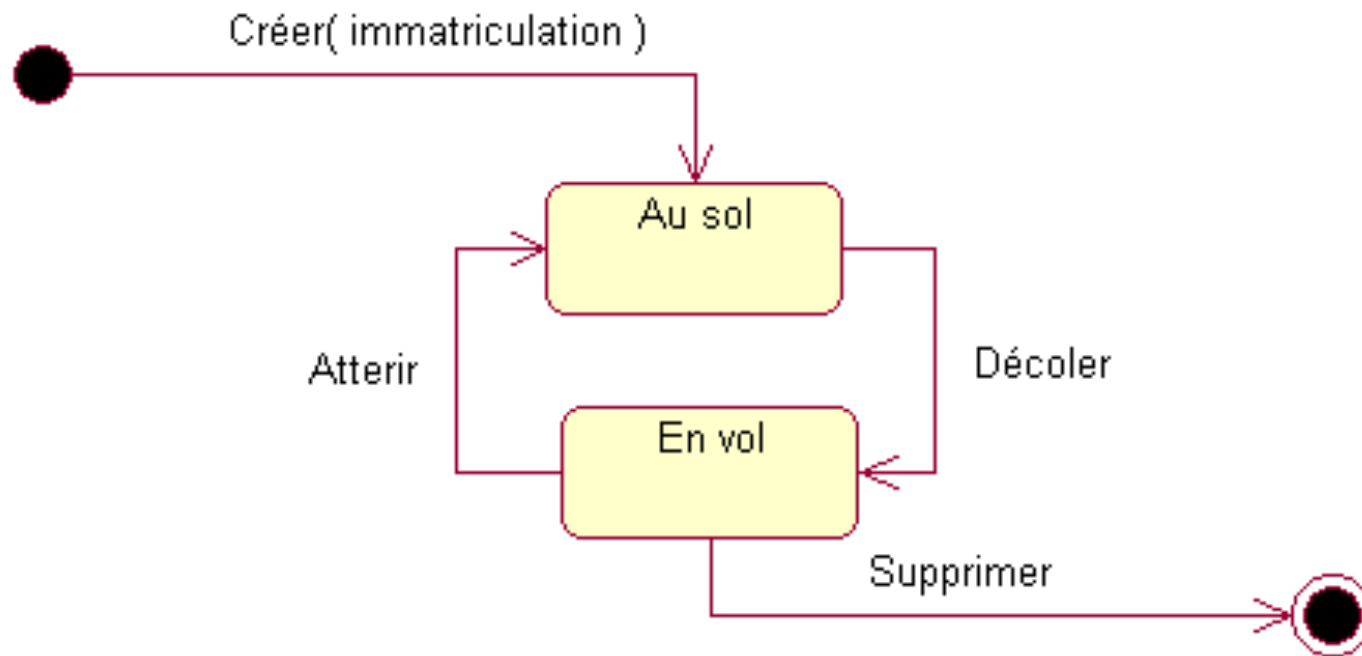
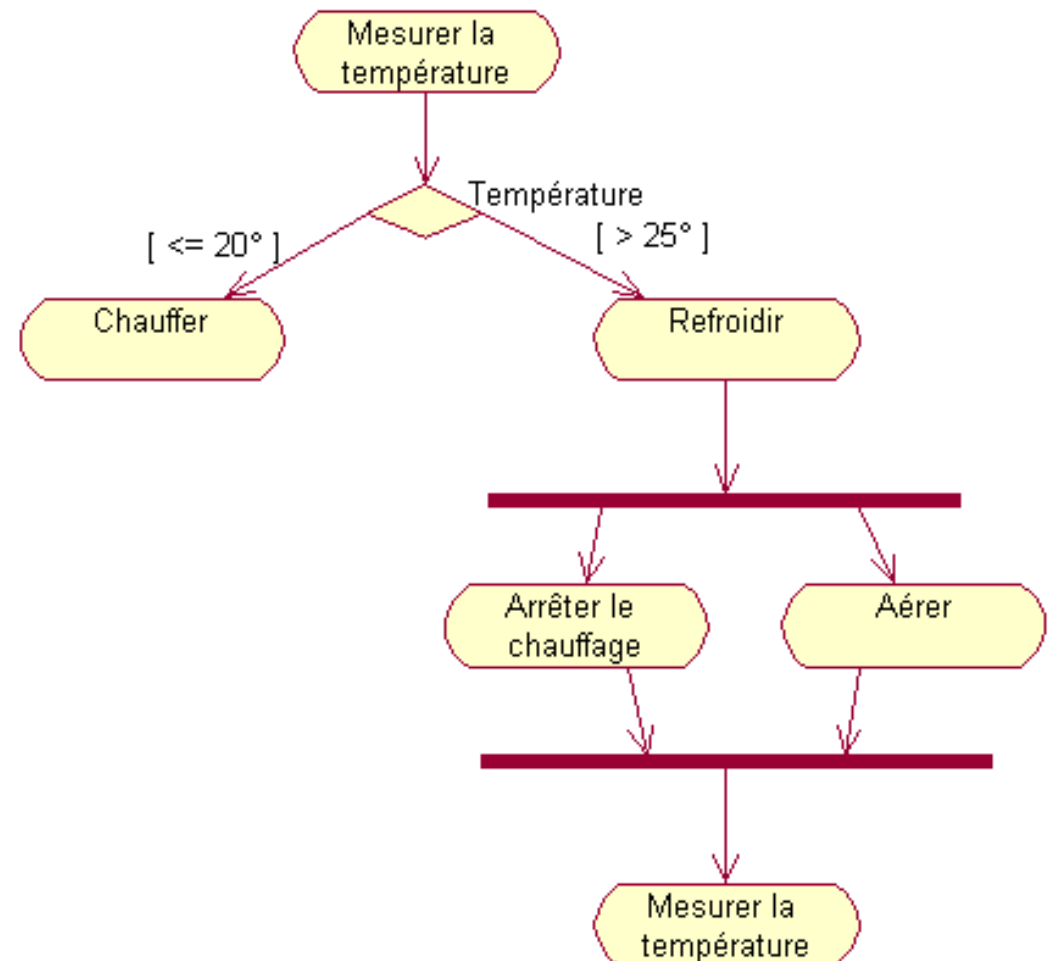


Diagramme d'activité

- Représente un enchaînement d'activités au sein d'une opération, d'un cas d'utilisation ou d'un processus métier.



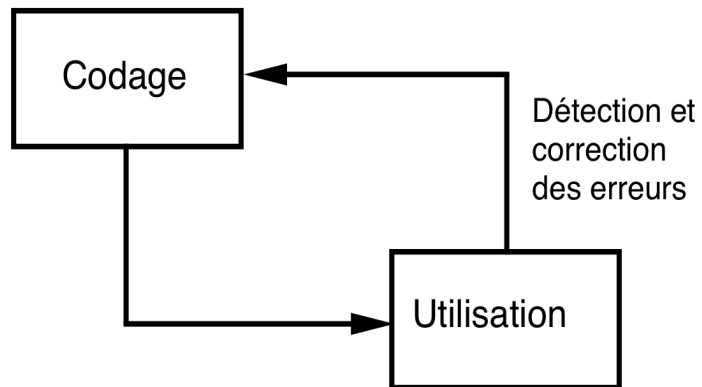


2

Les « meilleures » pratiques du développement logiciel

Projet Logiciel (1)

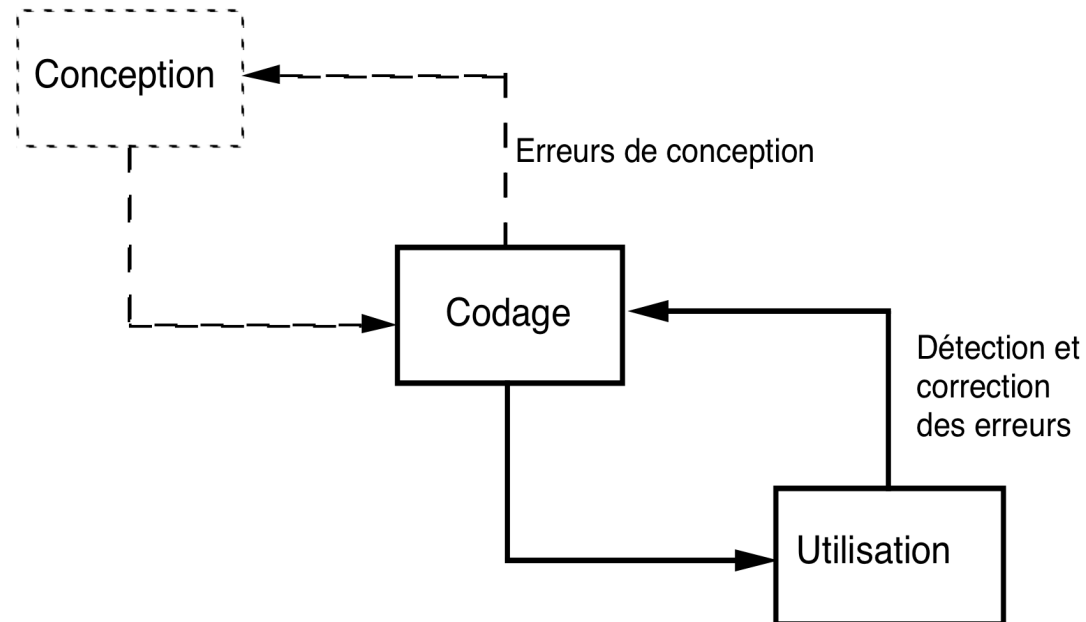
- Notion de Projet Logiciel



Petits Projets : Méthode descendante orientée programme

Projet Logiciel (2)

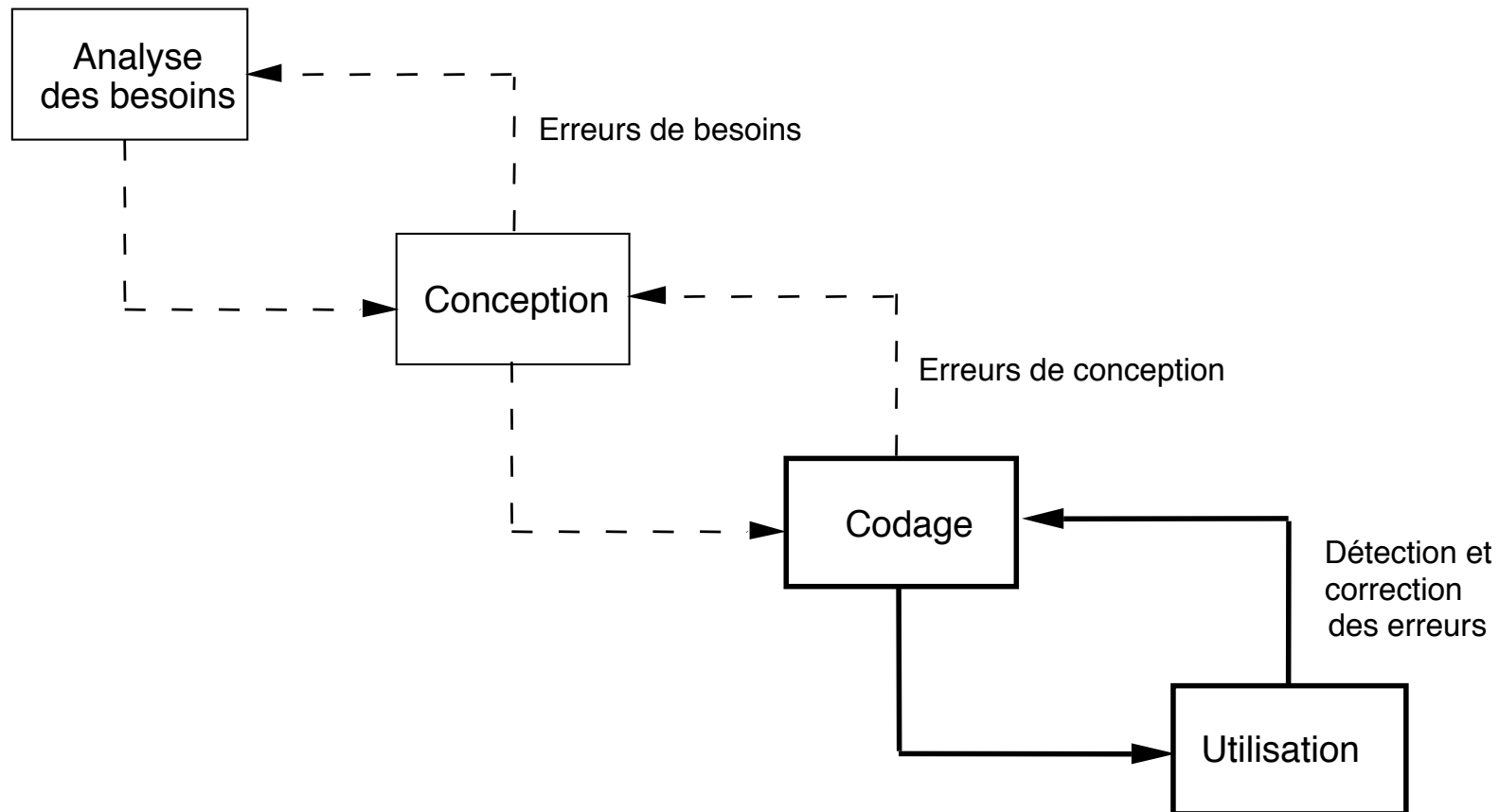
- Notion de Projet Logiciel



Petits Projets : Méthode descendante orientée programme

Projet Logiciel (3)

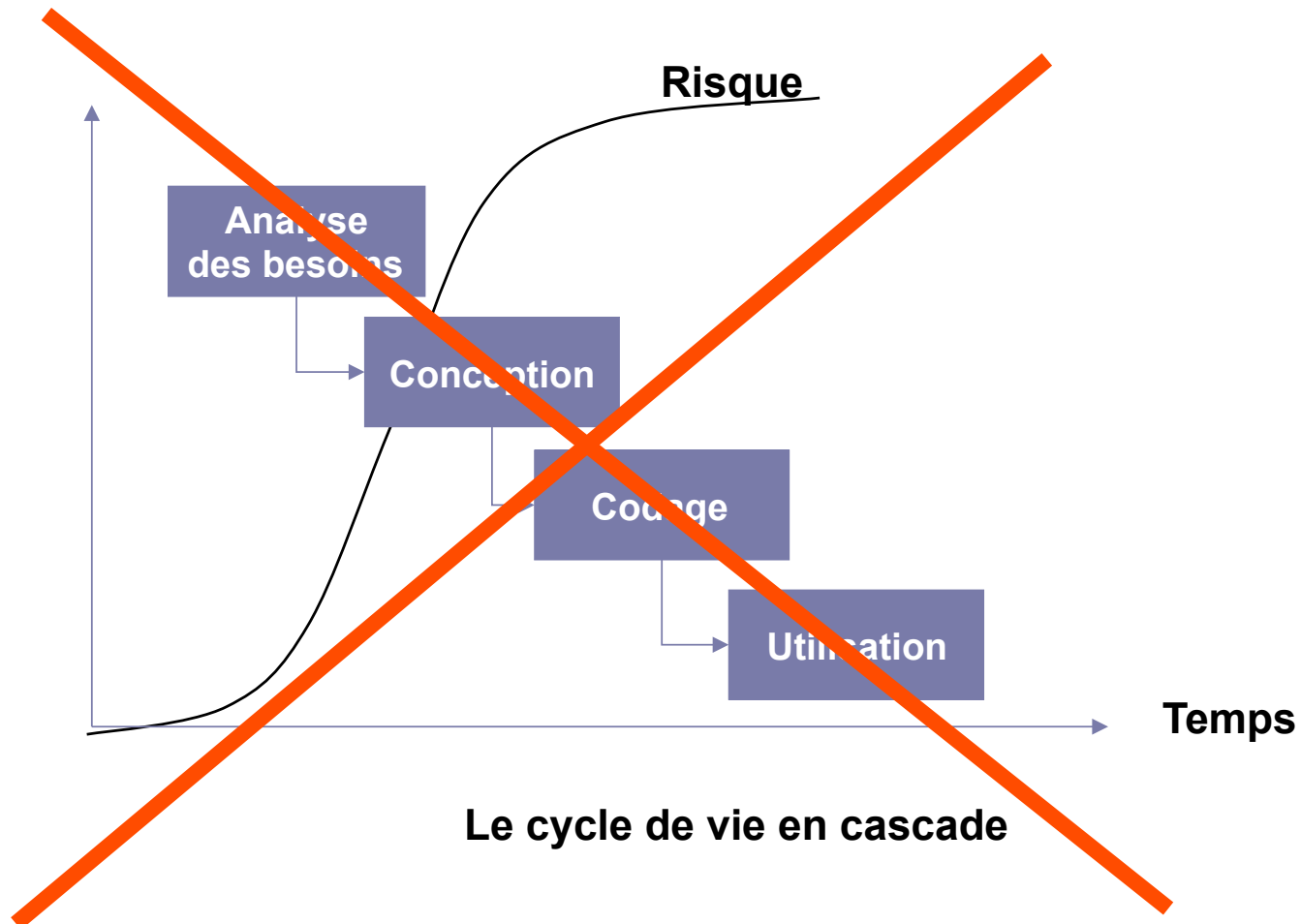
- **Notion de Projet Logiciel**



Petits Projets : Méthode descendante orientée programme

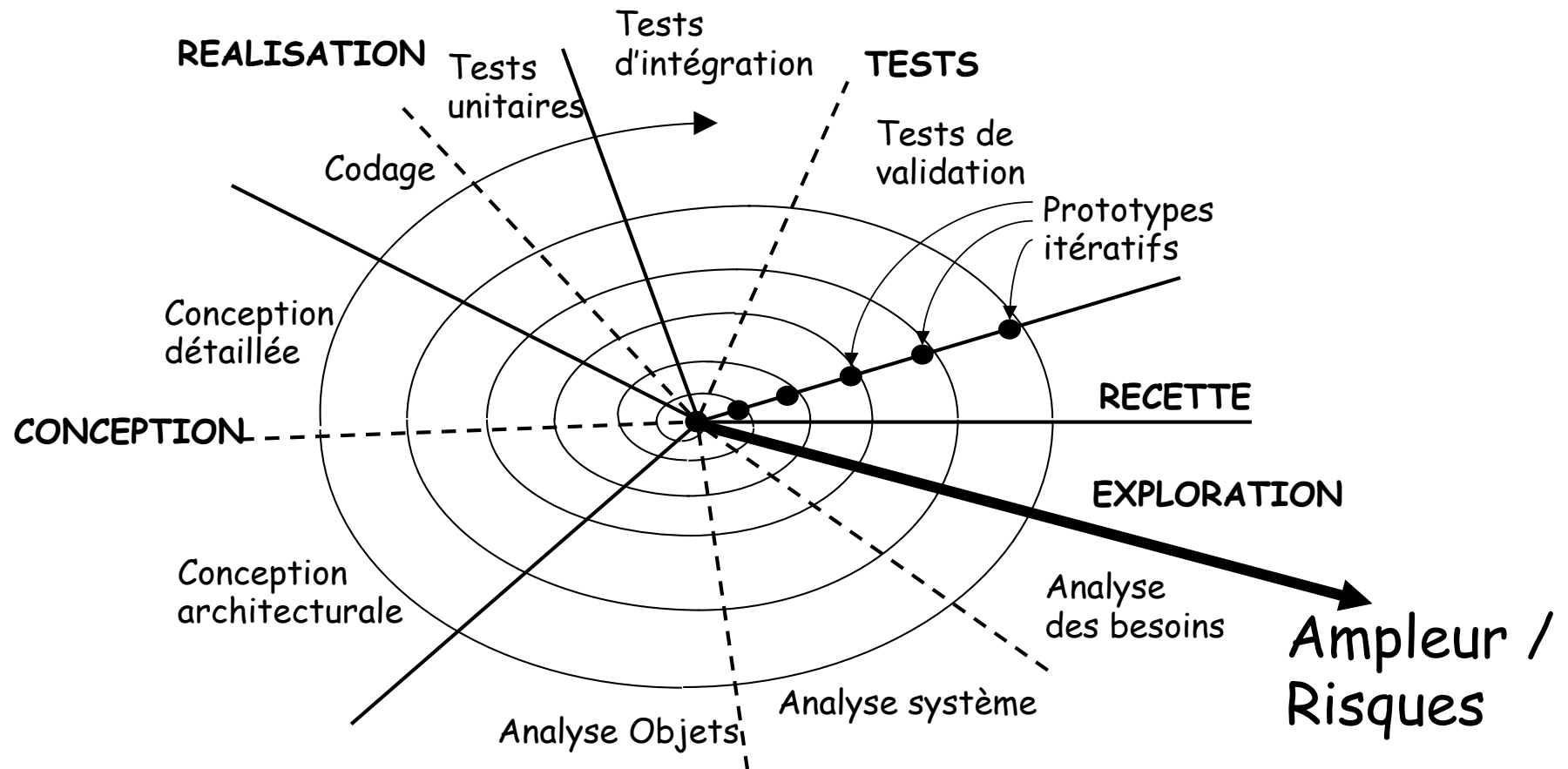
Développer le logiciel de façon itérative

*Bonne
pratique n°1*



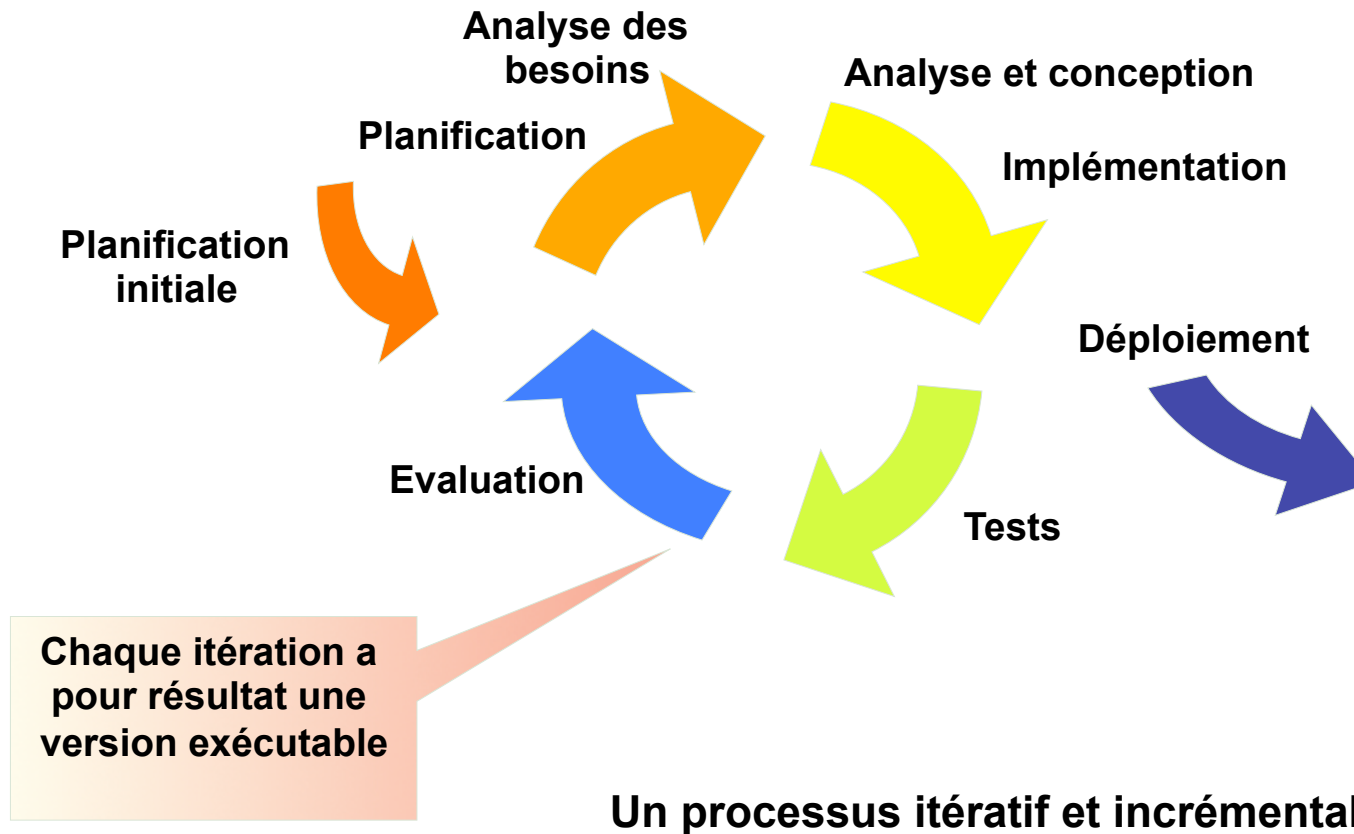
Développer le logiciel de façon itérative

- Un Processus Itératif: Cycle de Vie en Spirale



Développer le logiciel de façon itérative

*Bonne
pratique n°1*



Gérer les exigences

Bonne
pratique n°2

- Problématiques

- Modifications des exigences au cours du développement
- Identification et intégration de l'ensemble des besoins au cours du processus

- Propositions

- Organiser et décrire les fonctionnalités et les contraintes du système (*complétude et cohérence*)
- Evaluer les changements à apporter au cahier des charges et estimer leur impact (*modifiabilité*)
- Décrire les différentes décisions prises et en faire le suivi (*traçabilité*)

Utiliser et créer des architectures à base de composants

*Bonne
pratique n°3*

- L'architecture repose sur des décisions concernant :
 - L'organisation du système
 - Les choix des éléments structurels et de leurs interfaces
 - Leur comportement
 - Leur composition en sous systèmes

(Exemples : Composants Com, Corba, EJB, classes .Net, etc.)

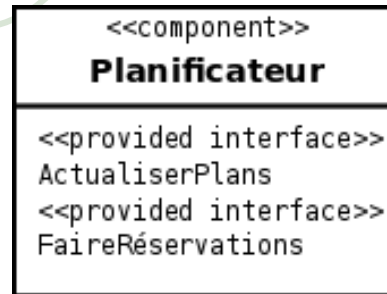
Rappel : Diagramme de composants

- Description du système modélisé sous forme de composants réutilisables et mettre en évidence leurs relations de

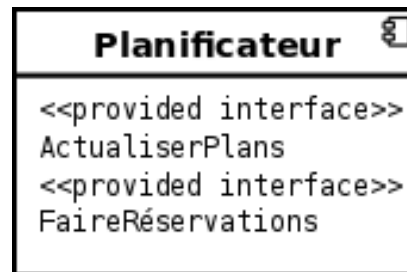


Stéréotypes composant

(A)



(B)



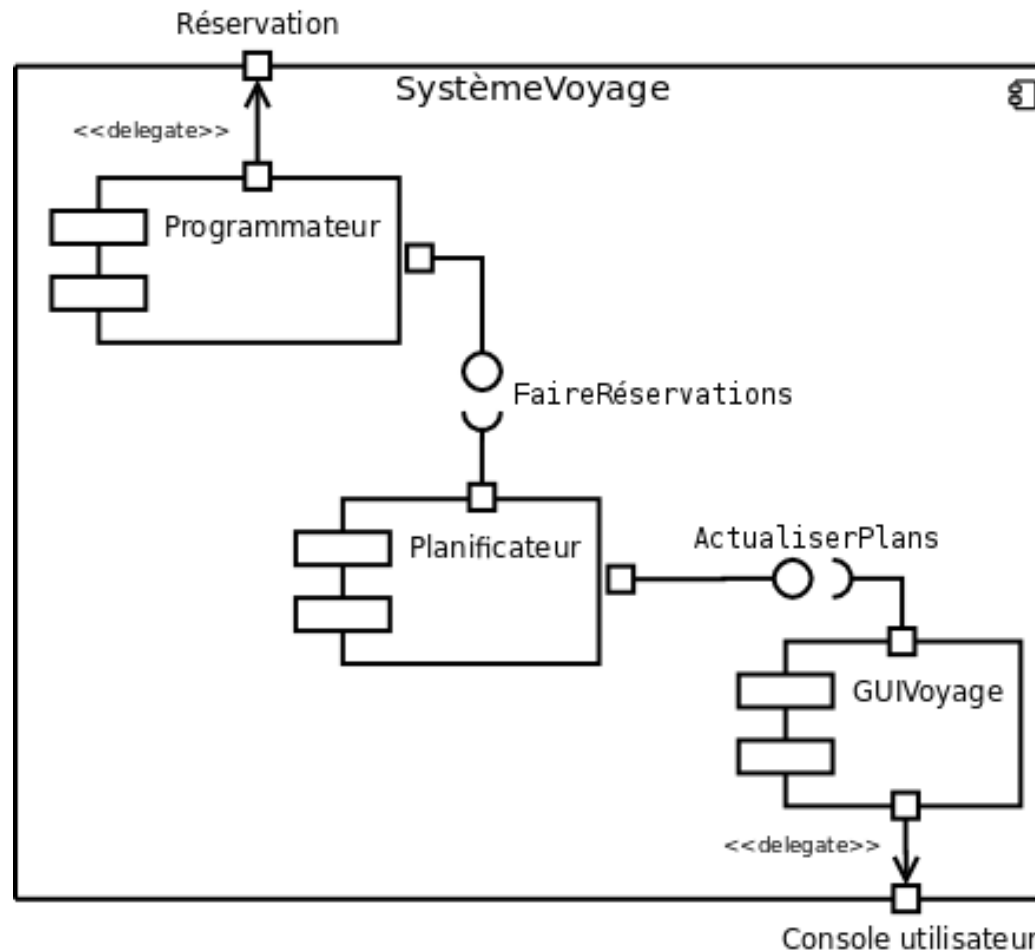
(C)



(D)



Représentation de l'implémentation d'un composant complexe



Modéliser graphiquement le logiciel

Bonne
pratique n°4

Utilisation des diagrammes UML, et plus globalement de la notion de *modèle*.

- ➡ « Tout » n'est pas modèle
- ➡ « Tout modèle » n'est pas UML

Vérifier la qualité du logiciel

*Bonne
pratique n°5*

- Fonctionnalités, fiabilité et performances
 - Tests (de « benchmark », de configuration, de fonctionnement, d'installation, d'intégrité, de charge, de performance, de stress, ...)
 - Rapports de conception (ou revues)

Exercice



Décrivez en quelques lignes l'organisation du processus de développement de votre entreprise (précisez la taille des équipes de développement).

Déterminez si ce processus intègre à votre connaissance les cinq pratiques vues précédemment.



3

L'essentiel du Processus Unifié

Qu'est ce que le Processus Unifié ? (1/3)

- **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)
 - Il existe d'autres approches (qui ne sont en général pas complètement antinomique), comme par ex. les méthodes « agile », beaucoup moins orientées modèle, comme XP (eXtreme Programming), scrum, etc.

Qu'est ce que le Processus Unifié ? (2/3)

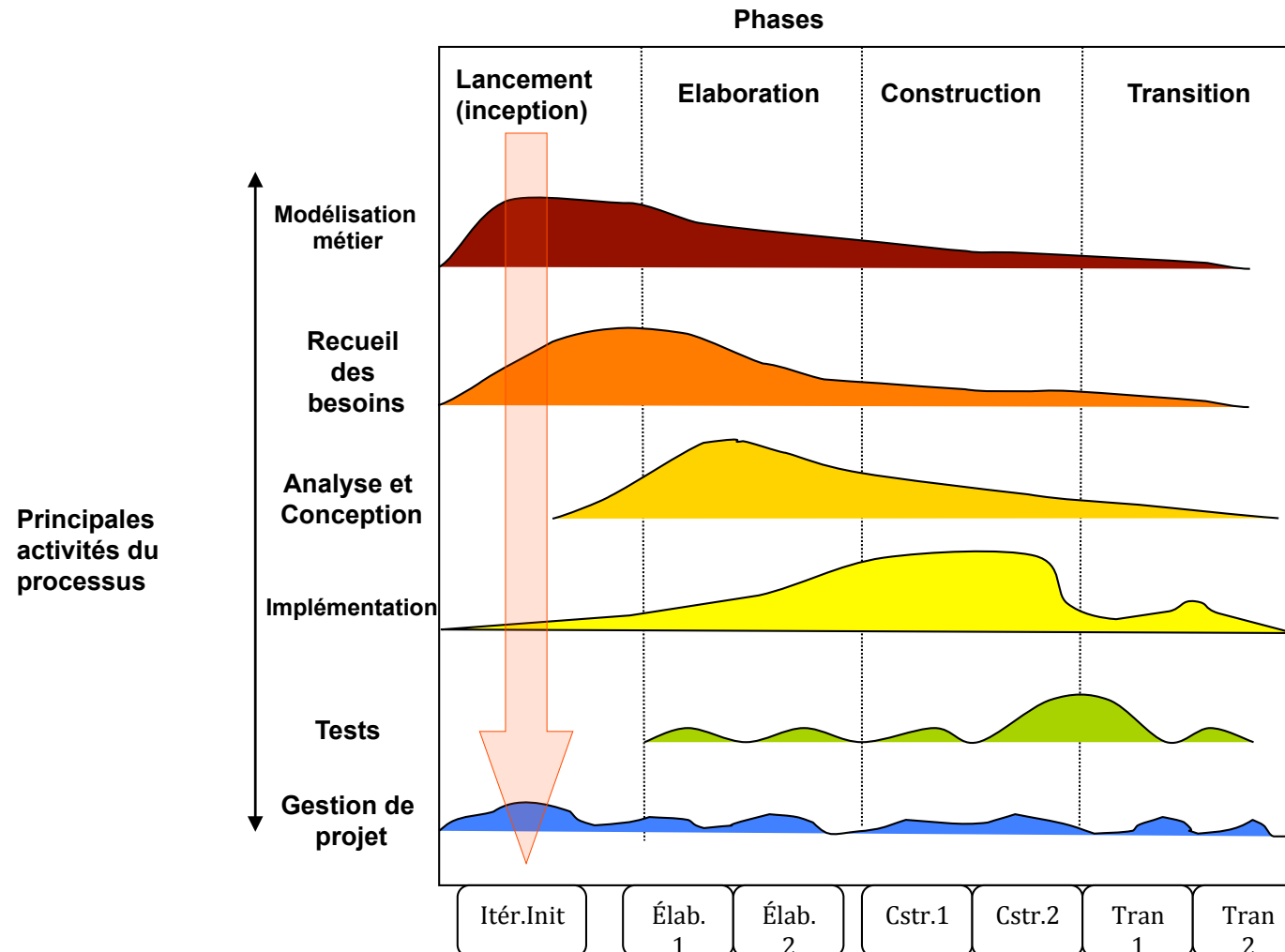
- Le processus unifié permet d'affecter des tâches et des responsabilités au sein d'une organisation de développement logiciel
 - Approche disciplinée pour des gros projets (chef de projet, analystes, intégrateur, intervenants, etc.)
 - Approche à adapter pour des petits projets
 - Pas particulièrement conçu pour le développement de systèmes embarqués

Qu'est ce que le Processus Unifié ? (3/3)

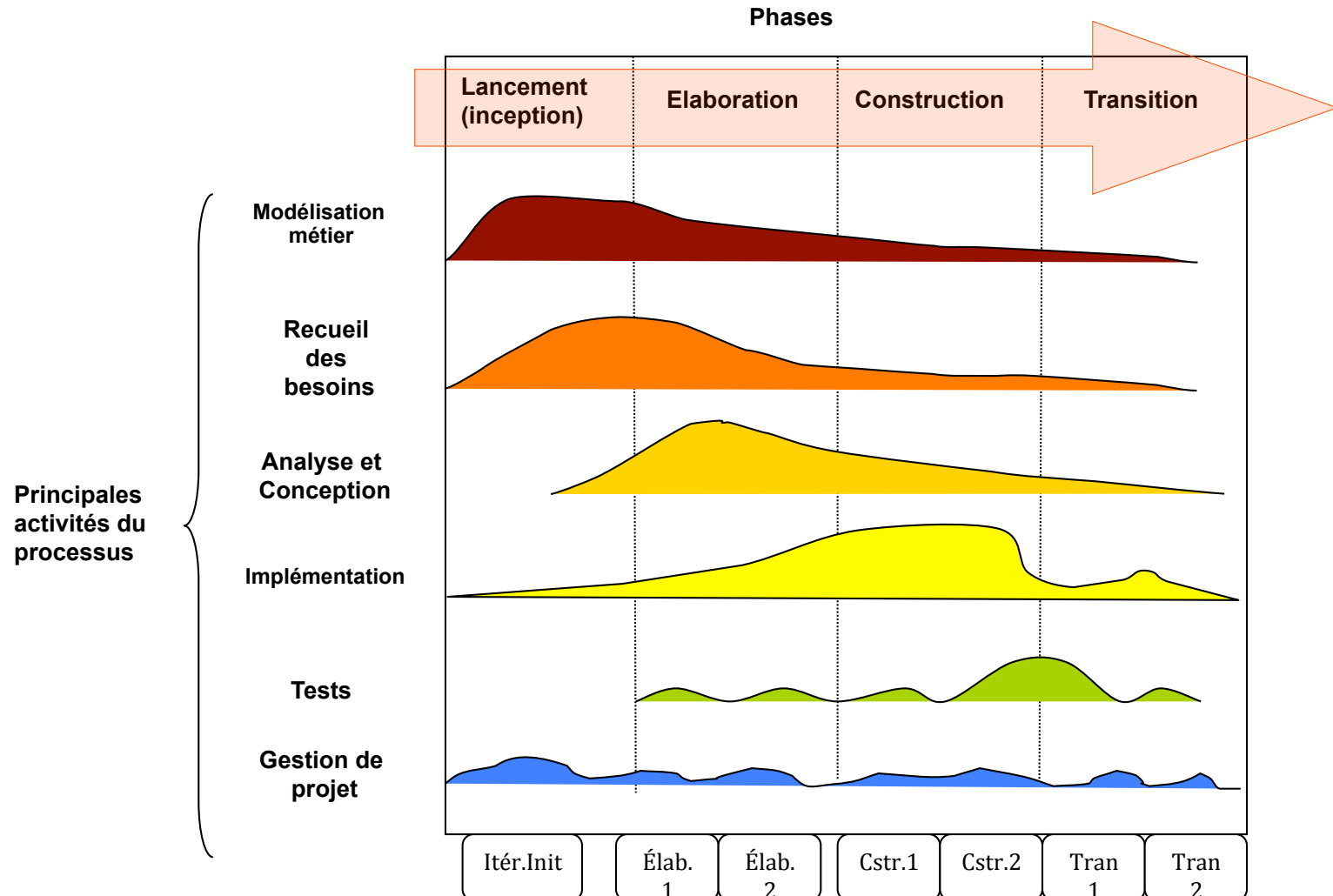
- Le processus unifié utilise le langage UML
- Le processus unifié est piloté par les cas d'utilisation
- Centré sur l'architecture
- Itératif et incrémental

La structure logique du Processus

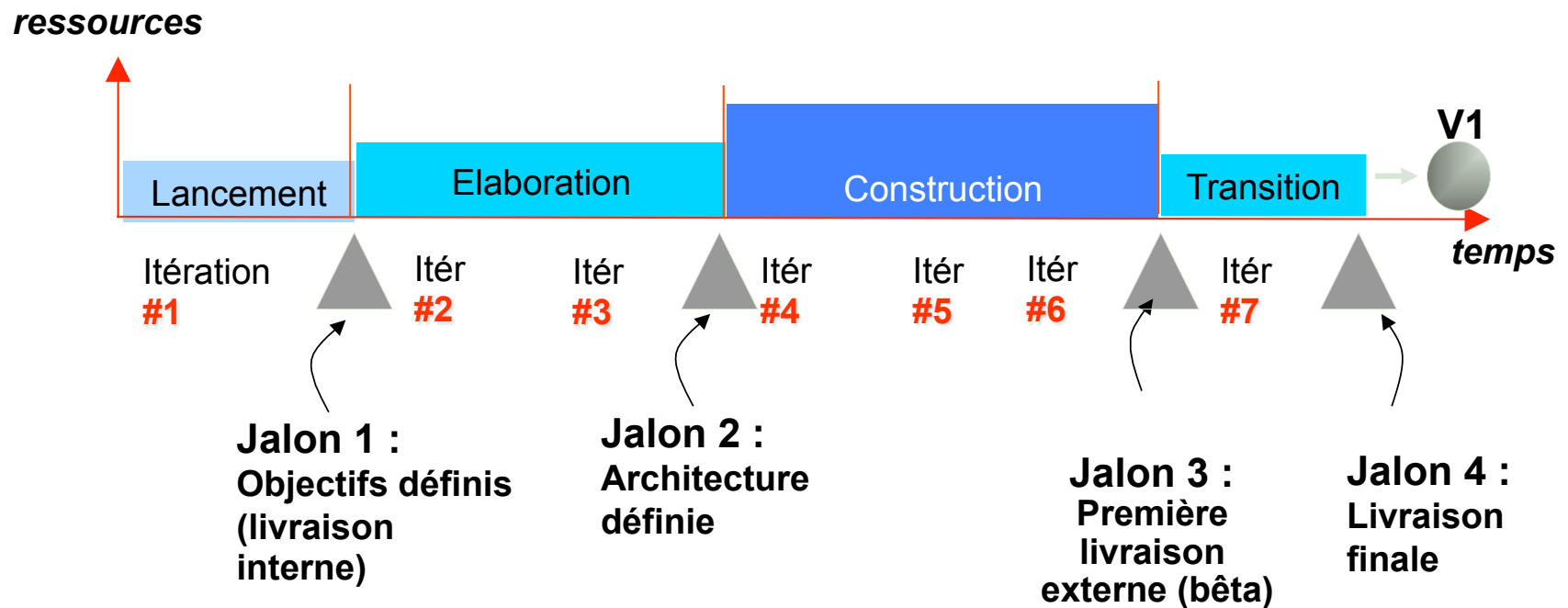
[Rational UP]



Les quatre phases



Les phases —, les itérations (#n) et les jalons ▲



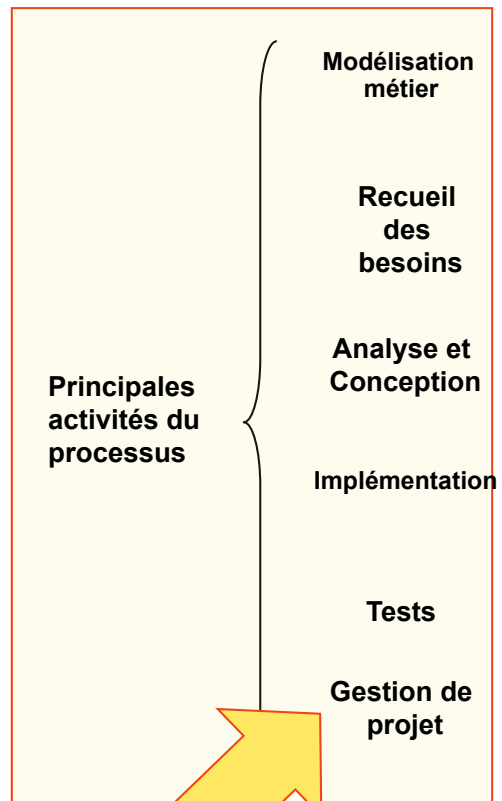
➡ Les Jalons correspondent à des étapes d'évaluation de la phase terminée, et de lancement de la phase suivante



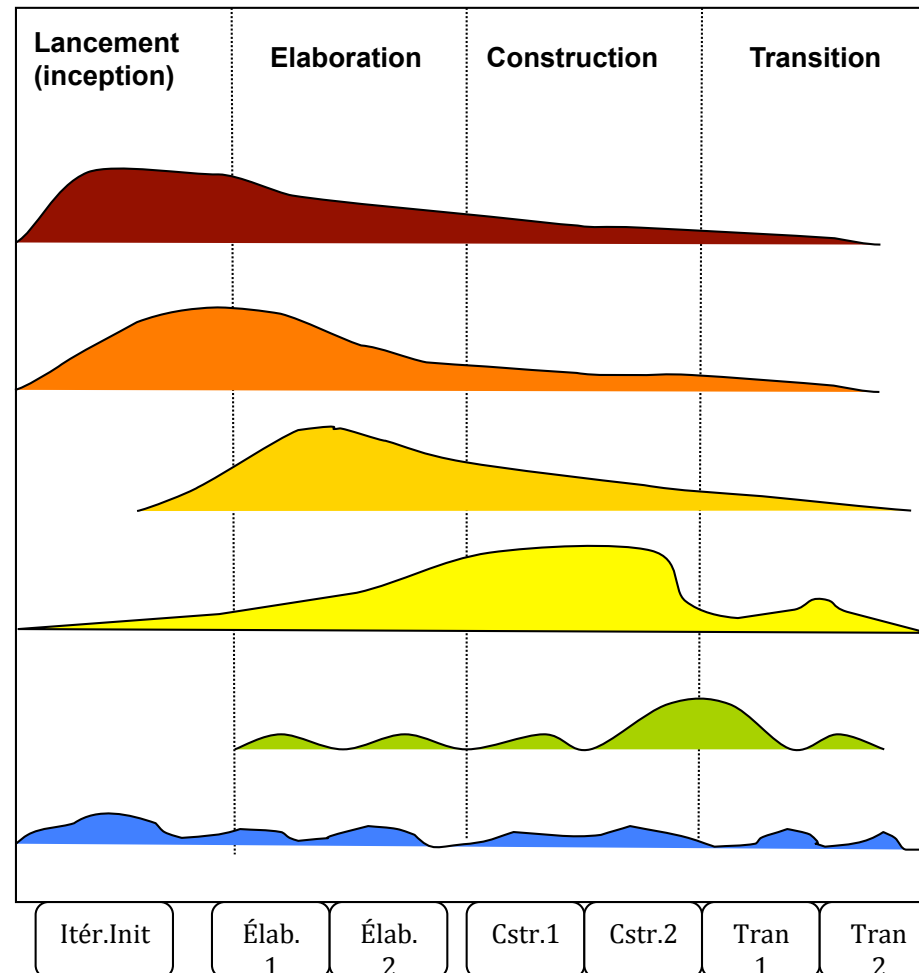
5

Les enchaînements d'activités

Vue générale



Phases



Gestion de projet

- « La **gestion de projet** est la mise en œuvre de connaissances, de ressources, de compétences, d'outils et de techniques qui permettent le lancement, la planification, la réalisation, le pilotage et la clôture d'un projet dans un cadre temporel et budgétaire, pour atteindre des objectifs précis. »
- De nombreux aspects ne sont pas couverts dans cette présentation (gestion personnel, budget, contrats, etc.)
[Voir cours gestion de projet qui couvre également les aspects « Phases d'un projet », « Artefacts », et « Rôles »]
- Objectifs
 - Planifier/évaluer un projet itératif
 - Gérer les risques
 - Contrôler les progrès (délais, coûts, qualité, efforts, satisfaction client, productivité, etc.)

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

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)
- Planification et prévision de coût
- Ebauche de l'IHM (prototypage et validation par l'utilisateur)

Analyse et conception

- Transformation des besoins utilisateurs en modèles UML
- Unified Modeling Language (pas une méthode mais un langage)
- Modèle d'analyse et modèle de conception, (éventuellement modèle de données)
- Étape importante :
 - de l'analyse à la conception : assigner des responsabilités aux classes ➡ Les patrons GRASP (General Responsibility Assignment Software Patterns) [Présentation ultérieure]

Implémentation



- Développement incrémental
- Découpage en couches
- Composants d'architecture
- Retouche des modèles et des besoins
- Unités indépendantes, équipes différentes

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)

Exercice



- Décrivez en quelques lignes les phases que vous suivez lorsque vous êtes seul à développer un logiciel.
- Est ce que vous retrouvez certains éléments du Processus Unifié ?
- Quels tests effectuez vous lorsque vous développez seul ?



PARTIE 2

Les artefacts*
(ou livrables, ou délivrables)

*artificiel actis

Les artefacts

- Éléments produits lors du développement (documents, modèles, fichiers compilés, composants, etc.)
- ➔ Nous nous intéresserons ici aux artefacts de documentation et à leur gestion (création, modification, livraison)

Préambule

- Les parties des documents sont présentées ici à titre d'exemple.
- Suivant la taille du projet, il faut étendre ou réduire ces documents (voire les supprimer)
- Dans tous les cas, les documents doivent être mis régulièrement à jour, avec une gestion des changements (tableau de mise à jour dans l'en-tête de chaque doc)

Artefact ou pas ? Règles de base

- Objectif des artefacts : produire un meilleur logiciel
- Trop d'artefacts produit perte de temps, dispersion de l'équipe, augmentation des coûts
- ➔ Restez concentré sur le logiciel exécutable
- ➔ En cas de doute sur l'opportunité d'un artefact, abstenez vous

Template des documents

<nom du projet>

Nom du document

Version <1.0>

Historique des révisions

Date	Version	Description	Auteur
<jj/mm/aa>	<x.x>	<details>	<name>
...

Sommaire

1. Introduction
 1. Objectifs
 2. Définitions, acronymes, et abréviations
 3. Références
2. <Sections spécifiques au document>

Ensemble d'artefacts



- L'ensemble de gestion
- L'ensemble des exigences
- L'ensemble de conception
- L'ensemble d'implémentation
- L'ensemble de déploiement



L'ensemble de gestion de projet

- Document de vision
- Plan de développement logiciel (planning des phases, rôles, responsabilités, jalons, activités)
- Liste des risques
- Planning d'itération
- Bilan d'itération
- Étude de rentabilité
- Points en suspens
- Glossaire
- Données sur des anomalies

Document de vision

[En-tête]

Sommaire



1. Problème

[motivations pour la création du logiciel]

2. Énoncé de la vision

[description du concept du système et intérêts]


3. Principaux acteurs concernés

[acteurs du développement et utilisateurs]

4. Caractéristiques du logiciel

[liste des principaux CU, performances, etc.]

Exemple de sommaire du document de vision étendu

- 
1. Introduction
 - 1.1 Purpose
 - 1.2 Scope
 - 1.3 Definitions, Acronyms, and Abbreviations
 - 1.4 References
 - 1.5 Overview
 2. Positioning
 - 2.1 Business Opportunity
 - 2.2 Problem Statement
 - 2.3 Product Position Statement
 3. Stakeholder and User Description
 - 3.1 Market Demographics
 - 3.2 Stakeholder Summary
 - 3.3 User Summary
 - 3.4 User Environment
 - 3.5 Stakeholder Profiles
 - 3.5.1 <Stakeholder Name>
 - 3.6 User Profiles
 - 3.6.1 <User Name>
 - 3.7 Key Stakeholder or User Needs
 - 3.8 Alternatives and Competition
 - 3.8.1 <aCompetitor>
 - 3.8.2 <anotherCompetitor>

4. Product Overview
 - 4.1 Product Perspective
 - 4.2 Summary of Capabilities
 - 4.3 Assumptions and Dependencies
 - 4.4 Cost and Pricing
 - 4.5 Licensing and Installation
5. Product Features
 - 5.1 <aFeature>
 - 5.2 <anotherFeature>
6. Constraints
7. Quality Ranges
8. Precedence and Priority
9. Other Product Requirements
 - 9.1 Applicable Standards
 - 9.2 System Requirements
 - 9.3 Performance Requirements
 - 9.4 Environmental Requirements
10. Documentation Requirements
 - 10.1 User Manual
 - 10.2 Online Help
 - 10.3 Installation Guides, Configuration
 - 10.4 Labeling and Packaging

Plan de développement

Distribution dans le temps des ressources humaines (rôles et responsabilités) et matérielles



1. Structure organisationnelle
2. Rôles et responsabilités
3. Planning du projet
 1. Planning des phases
 2. Objectifs des itérations
 3. Délivrables
 4. Découpage en phases, identification des jalons et objectifs des itérations

Nb : Utilisation de tableaux, diagrammes (Gantt par ex. pour des projets importants)

Redmine pour préparer plan de développement

- Voir Redmine et les tickets “evolution” pour préparer le plan de développement
- Préparer la phase de lancement

Liste des risques



1 <Identificateur de risque> *[nom descriptif ou numéro]*

1.1 Niveau de risque

1.2 Description

1.3 Conséquences

1.4 Indicateurs

1.5 Stratégie de traitement du risque

2 <Risque suivant>

[...]

Types de risques

Risk type	Possible risks
Technology	<p>The database used in the system cannot process as many transactions per second as expected.</p> <p>Software components that should be reused contain defects that limit their functionality.</p>
People	<p>It is impossible to recruit staff with the skills required.</p> <p>Key staff are ill and unavailable at critical times.</p> <p>Required training for staff is not available.</p>
Organisational	<p>The organisation is restructured so that different management are responsible for the project.</p> <p>Organisational financial problems force reductions in the project budget.</p>
Tools	<p>The code generated by CASE tools is inefficient.</p> <p>CASE tools cannot be integrated.</p>
Requirements	<p>Changes to requirements that require major design rework are proposed.</p> <p>Customers fail to understand the impact of requirements changes.</p>
Estimation	<p>The time required to develop the software is underestimated.</p> <p>The rate of defect repair is underestimated.</p> <p>The size of the software is underestimated.</p>

Exemples de risques projet

(source Ian Sommerville 2004)

Risk	Affects	Description
Staff turnover	Project	Experienced staff will leave the project before it is finished.
Management change	Project	There will be a change of organisational management with different priorities.
Hardware unavailability	Project	Hardware that is essential for the project will not be delivered on schedule.
Requirements change	Project and product	There will be a larger number of changes to the requirements than anticipated.
Specification delays	Project and product	Specifications of essential interfaces are not available on schedule
Size underestimate	Project and product	The size of the system has been underestimated.
CASE tool under-performance	Product	CASE tools which support the project do not perform as anticipated
Technology change	Business	The underlying technology on which the system is built is superseded by new technology.
Product competition	Business	A competitive product is marketed before the system is completed.

Planning d'itération



[en-tête]

Sommaire

1. Planning

[Description temporelle des jalons intermédiaires, versions beta, demos, etc.]

2. Ressources

[Matérielles, humaines, et financières]

3. Cas d'utilisation

[C.U. et scénarios qui seront développés dans cette itération]

4. Critères dévaluation

[Fonctionnalités, performances, mesures de qualité, etc.]

Bilan d'itération



// ce document est le pendant du plan d'itération => reprendre point par point du plan et valider / invalider les éléments

// rajouter une section pour les difficultés rencontrées et les solutions appliquées

Élément du bilan d'itération

- Faire une table avec le degré d'avancement des cas d'utilisation -> mettre en valeur les CU traité pendant l'itération réalisée
- Nom du CU / % avancement
- Explication pour % non complet

L'ensemble des exigences

- Demandes des intervenants
- Modèle des CU
- Spécifications supplémentaires
- Modèle métier
- Prototypes jetables interface

Demandes des intervenants

- Document permettant de noter et référencer les demandes des intervenants (développeurs, sous-traitants, clients, etc.), concernant des modifications d'exigences (fonctionnelles ou non)
- Date / Nom / Description / Prise en compte ou non

Modèle des cas d'utilisation

(Ensemble des fiches de C.U. et du diagramme des C.U.)



1. Nom
2. Résumé
3. Acteurs
4. Description des scénarios
 1. Scénario nominal
 2. Scénarios alternatifs
5. Pré conditions
6. Post conditions
7. Enchaînements alternatifs (description textuelle ou tableau)
8. Autres spécifications :
 1. Besoins d'IHM, ergonomie
 2. Robustesse, confidentialité, performances (temps de réponse, charge), disponibilité, etc.

Action de l'acteur principal	Action du système	Action de l'acteur secondaire
1. action1	2. Réponse du système et traitement	3. Réception d'une information

Spécifications supplémentaires



1. Introduction
2. Fonctionnalités
 - 2.1 <Exigence fonctionnelle 1>
3. « Utilisabilité » (anglicisme de usability)
 - 3.1 <Exigence d'utilisabilité1>
4. Sûreté de fonctionnement
 - 4.1 <Exigence SdF 1>
5. Performance
 - 5.1 <Exigence de performance 1>
6. Contraintes de conception
 - 6.1 <Contrainte 1>
7. Aide en ligne et système d'aide
8. Composants achetés
9. Interfaces
 - 9.1 Interfaces utilisateur
 - 9.2 Interfaces matérielles
 - 9.3 Interfaces logicielles
 - 9.4 Interfaces de communication
11. Exigences de licence
12. Copyright
13. Normes applicables

Modèle métier



- Diagramme de classes métier
- Diagramme de C.U. métier
- Diagramme d'activités métier

L'ensemble de conception

- Description de l'architecture
- Modèle de conception
 - Diagrammes de classes complémentaires, de séquence détaillés, états, composants, déploiement, etc...
- Plan de test
- Cas de test

Description de l'architecture

- diagrammes de package avec choix architecture logique
 - Layers (stricte ou lâche cad avec des objets partagés):
Présentation, Application, Domaine,... (Multitiers : 3-tiers, 2-tiers, etc.)
 - Model-View-Controller (actif/passif)
 - SOA (service oriented architecture)
 - centralisé ou distribuée(par ex. Peer-to-peer)
 - Blackboard, middleware, etc.
- de classes, de séquence pour spécifier les classes collaborantes inter packages, patterns et bibliothèques utilisées

Plan de test



1. Éléments ciblés par les tests
2. Vue d'ensemble des tests planifiés
 - 2.1 Descriptif des test inclus dans le plan de dev.
 - 2.2 Descriptif des autres tests exclus du plan de dev.
 - 2.3 Descriptif des tests candidats à une inclusion dans le plan de dev.
3. Types et techniques de test
 - 3.1 Test d'intégrité
 - 3.2 Test fonctionnel
 - 3.3 Test de "benchmark"
 - 3.4 Test de l'interface graphique
 - 3.5 Test de performance
 - 3.6 Test de charge
 - 3.7 Robustesse
 - 3.8 Test de stress
 - 3.9 Test de sécurité et de contrôle d'accès
 - 3.11 Test de configuration
 - 3.12 Test d'installation

➡ Il faut documenter les tests envisagés pour l'application

Plan de test

Objectifs de la technique :	<i>[Description des objectifs]</i>
Technique:	<i>[Description de la technique utilisée, par ex. « Exécuter chaque scénario et vérifier que... »]</i>
Oracles:	<i>[Stratégies utilisées par la technique pour observer précisément le succès ou la défaillance du test]</i>
Outils requis :	<i>[Besoins en terme de scripts, fichiers de log, etc...]</i>
Critère de succès :	<i>[Condition pour que l'ensemble du test soit positif]</i>
Considérations particulières :	<i>[Contraintes liées au type de test, recommandations, remarques, etc...]</i>

Cas de test



Cas de test : *<Nom du cas de test>*

Numéro d'identification : *<Identifiant>*

Catégorie : *<Catégorie de cas de test>*

[Les valeurs possibles de catégorie sont Fonctionnel et Performance]

1. Introduction

[Cette section décrit brièvement le rôle et l'objectif du Cas de Test]

2. Enchaînement des événements

[Décrire les étapes que le testeur doit suivre. Les réponses (R) sont les résultats attendus par le test.]

1. Étape 1

R. Réponse 1.

2. Étape 2

R. Réponse 2

3. Étape 3

R. Réponse 3

3. Besoins techniques

[Besoins techniques liés à ce cas de test, par. Ex. un simulateur d'environnement, une machine sous windowsNT, etc.]

3.1 <Premier besoin >

[...]

L'ensemble d'implémentation

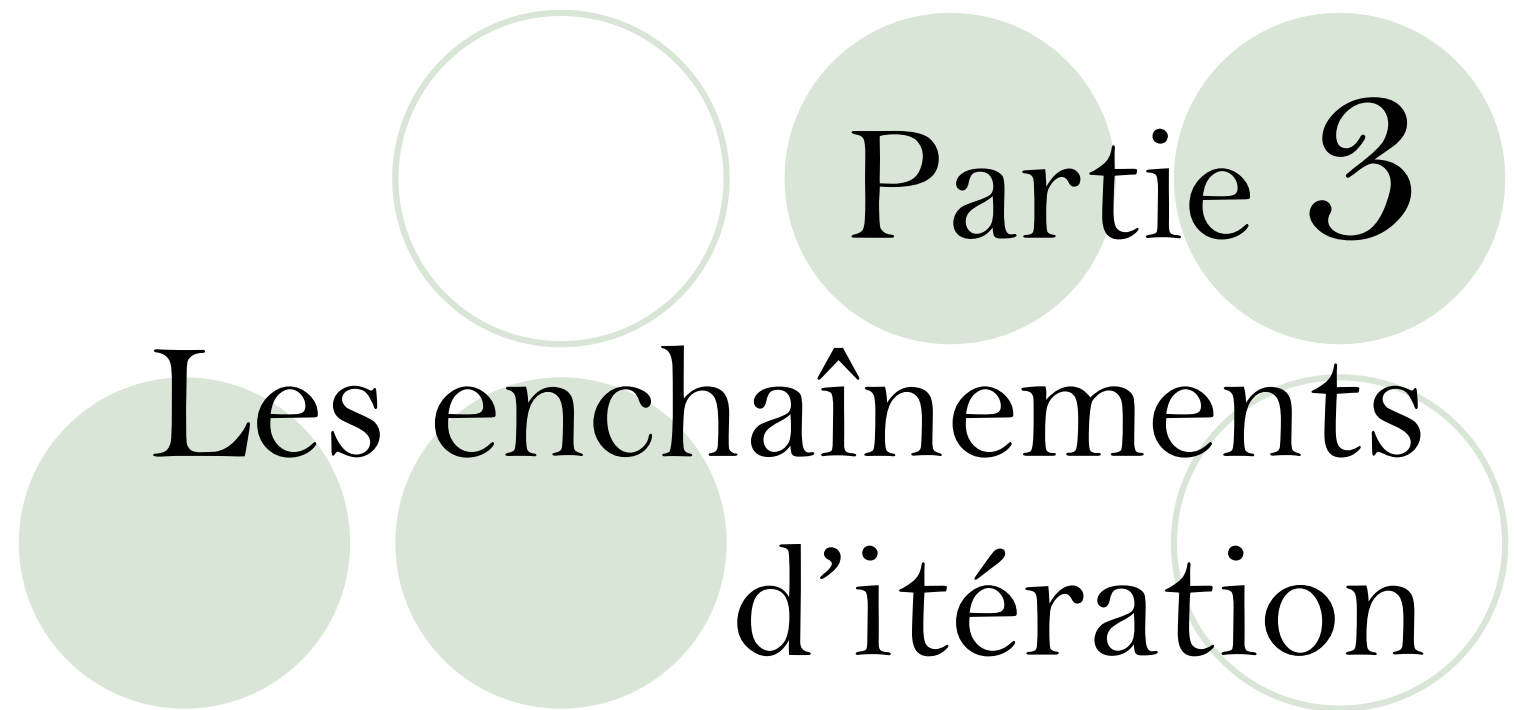
- Le code source et les exécutables
- Les fichiers de données associés (par ex. Javadoc) ou les fichiers permettant de les produire

L'ensemble de déploiement

- Scripts et guide d'installation
- Documentation utilisateur
- Matériel de formation



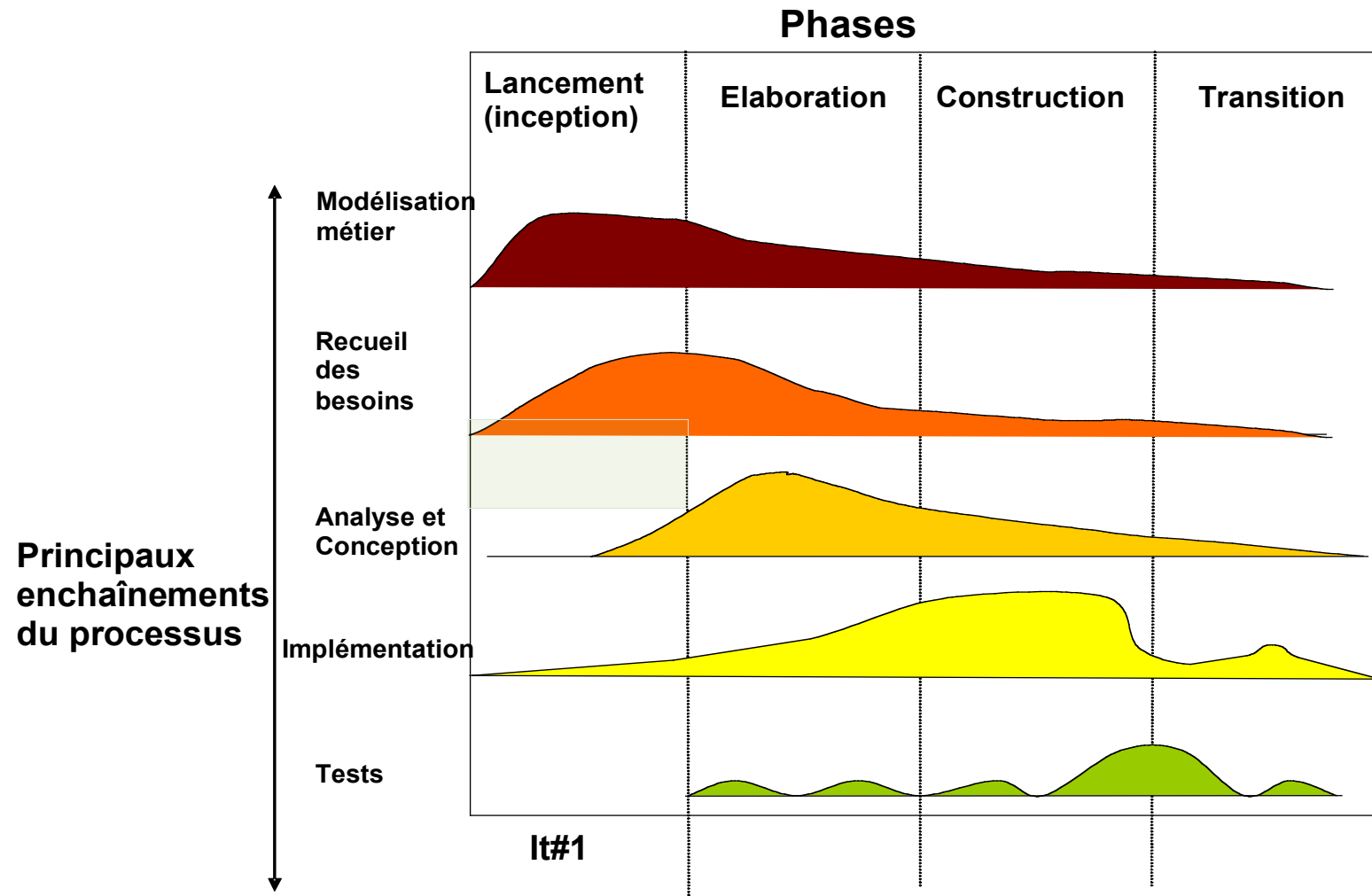
- Aller sur <http://jdbv.sourceforge.net/> et répondre aux questions suivantes:
 - Que fait le logiciel ?
 - Combien de personnes et combien de temps pour le développement ?
 - Quelle architecture ?
 - Quelles fonctionnalités ?
 - Qu'est ce qu'il reste à faire ?



Partie 3

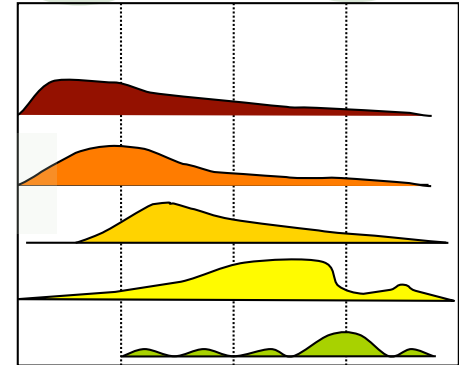
Les enchaînements d'itération

Phase de lancement



Phase de lancement (inception) (1/4)

- Comprendre le système à construire (vision générale et limites du projet)
- Recueillir les besoins utilisateurs
- Spécifier les principaux cas d'utilisation et scénarios
- Définir une architecture candidate
- Evaluer les coûts et planning
- Les principaux risques



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. Comprendre les coûts, les délais et les risques
 5. 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 et/ou un artefact:modèle métier, développer des artefact: proto d'interface jetables)

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.) (artefact:document d'architecture)
- Technologies utilisées (éventuellement faire des tests d'implémentation pour estimer les risques liés à une technologie)

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

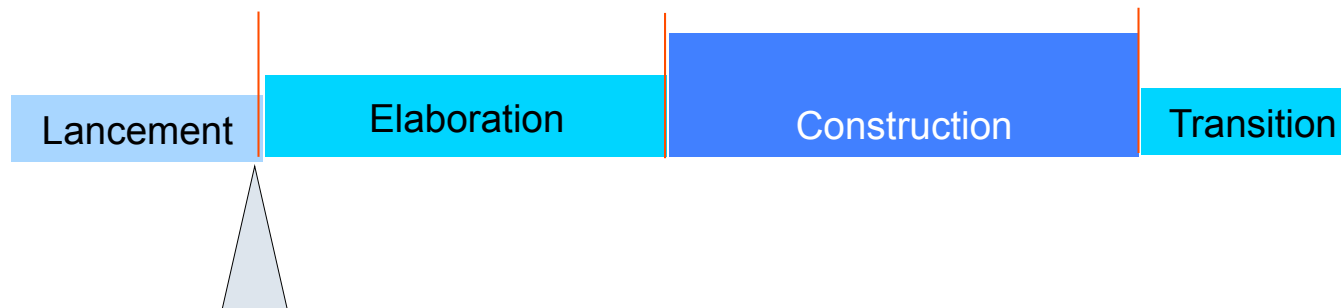
- Examiner la faisabilité du projet (étude de rentabilité, artefact: liste des risques)
- Établir le plan du projet (artefact : plan de développement du logiciel)

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, mettre à jour liste de risques, coûts et délais

Revue de projet : Jalon fin de lancement

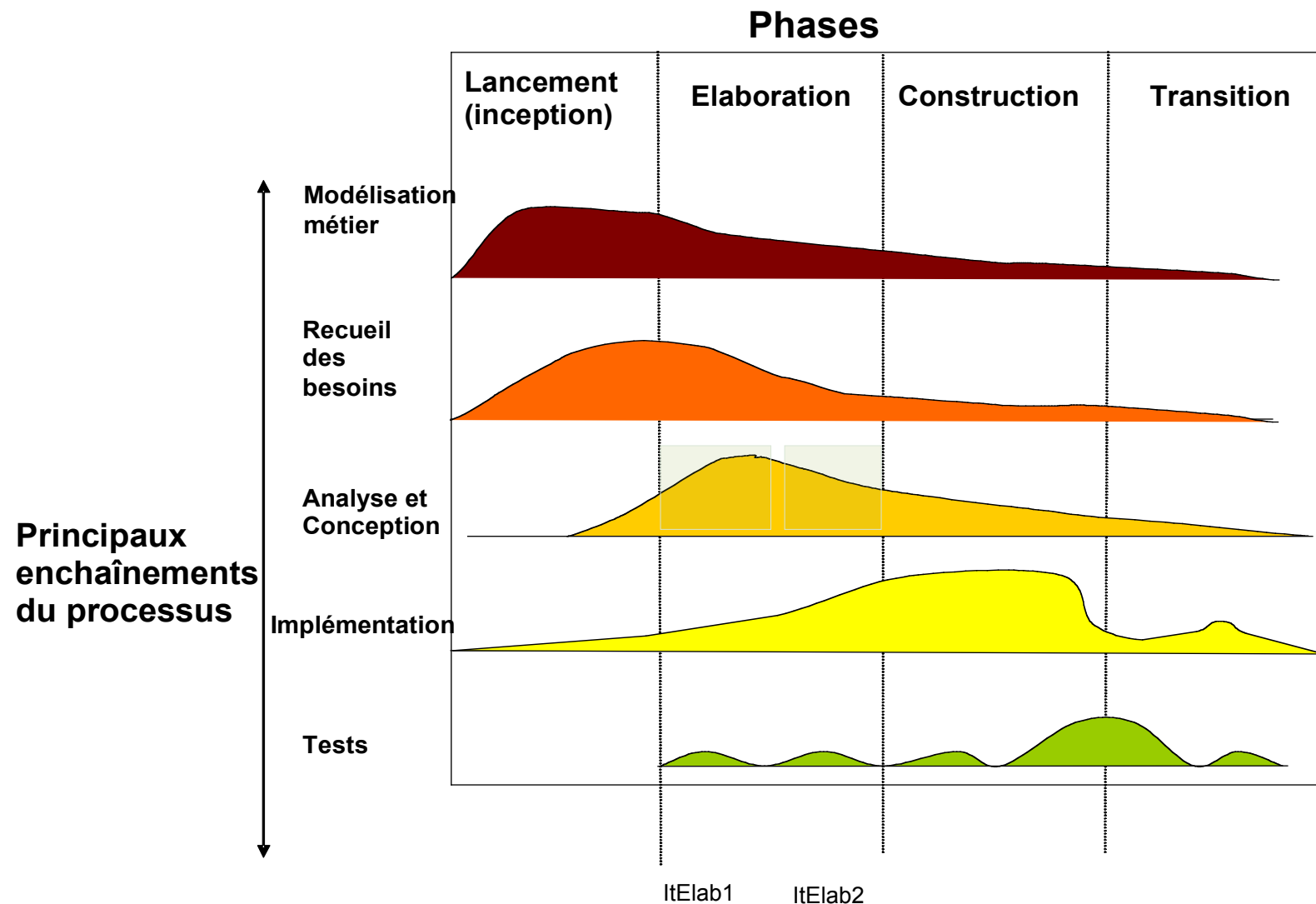
- Les différents intervenants valident:
 - le périmètre du projet, coûts et délais
 - la liste des exigences
- Les risques initiaux sont identifiés et il existe une stratégie de réduction pour chacun d'eux
- L'ensemble des artefacts produit est complet, à jour et cohérent
- NB : à chaque fin d'itération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





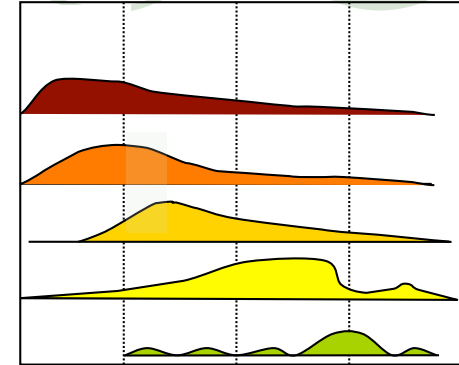
Exercice : Lister l'ensemble des artefacts produits pendant la phase de lancement

Phase d'élaboration



Phase d'élaboration

- Modèle des cas d'utilisation complet
- Exigences supplémentaires
(non fonctionnelles et celles qui ne sont pas associées à des c.u.)
- Raffiner l'architecture logicielle
- Un prototype architectural exécutable
- Un manuel utilisateur préliminaire



La phase d'élaboration

- Construction d'un squelette d'architecture en intégrant les risques majeurs et affiner les plans de projet
- Objectifs :
 1. Comprendre en détail les exigences
 2. Concevoir, implémenter, valider l'architecture
 3. Réduire les risques essentiels et estimer plus exactement le budget
 4. Affiner le plan de développement

NB: en général une à trois itérations (artefact : plan d'itération)

Objectif 1 : comprendre les exigences en détail

- Mettre à jour tout au long de cette phase :
 - Le modèle des C.U. (certains C.U. très simples et ne présentant aucun risque ne doivent pas être formalisés)
 - Le glossaire
- Hiérarchiser et faire des packages de C.U.

Objectif 2 : Concevoir, implémenter, et valider l'architecture

- Architecture: définir les sous-systèmes, les composants, et leurs interfaces (utiliser au maximum des *patrons* d'architecture)(artefact : description de l'architecture)
- Déterminer les C.U. significatifs du point de vue architectural et fonctionnel -> identifier les C.U. critiques
- Concevoir les C.U. critiques (description architecture + artefact : modèle de conception)
- Concevoir la base de données
- Décrire la concurrence, les processus, les *threads*, et la distribution physique
- Identifier les patrons de conception
- Implémenter et tester les scénarios critiques (artefact : plan de test, cas de test)

Objectifs 3 & 4

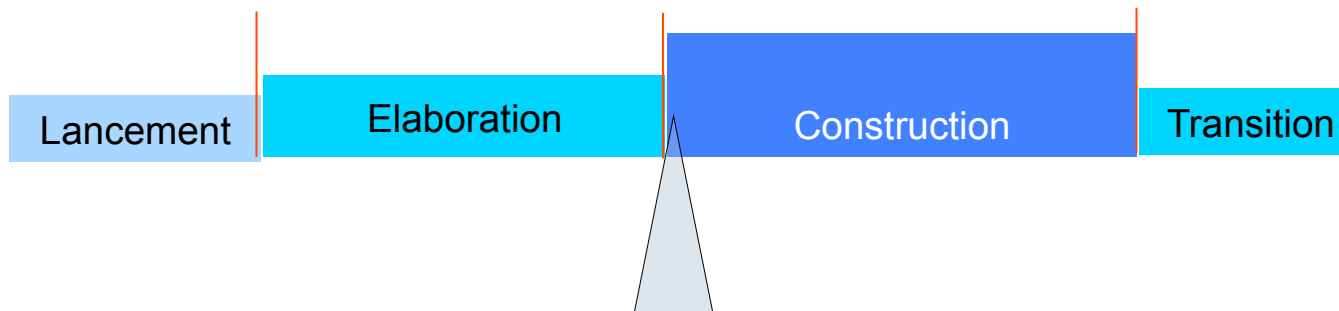
- Réduire les risques essentiels et estimer au mieux les délais et coûts et raffiner le plan de développement
 - Utiliser toutes les informations provenant des activités de conception pour mettre à jour les risques, délais et coûts.

Les itérations pendant l'ELABORATION

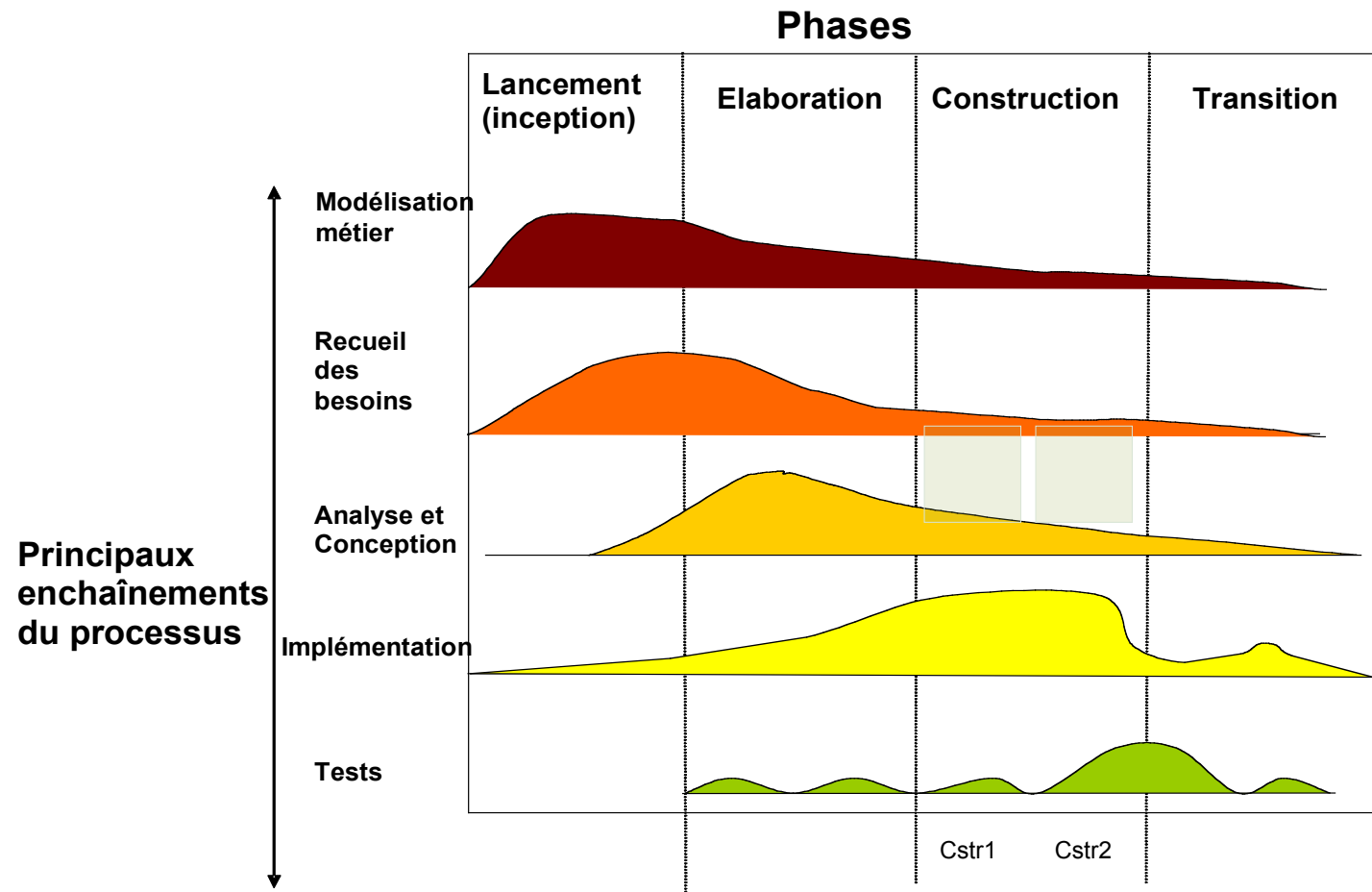
- Basées sur des proto qui implémentent:
 - Les C.U. critiques (une it# par C.U.)
 - Les scénarios d'un C.U. critique (1 it / seq)
 - Les parties (avec des stubs) d'un scénario
 - Les parties (avec des stubs plus ou moins simples) d'un scénario
- Guident la distribution du travail (notamment pour faire des packages)

Revue de projet : jalon fin d'élaboration

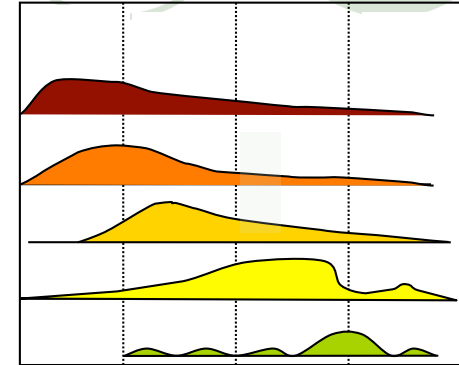
- Évaluer :
 - Stabilité vision et exigences
 - Stabilité architecture
 - Crédibilité du plan d'itération
- En cas de doute : refaire une itération



Phase de construction



Phase de construction (3/4)



- Gestion des ressources matérielles (installation sur plateformes choisies)
- Optimisation du processus pour réduire les coûts de développement
- Améliorer la qualité
- Compléter les modèles suivant les besoins d'implémentation
- Produit : le logiciel installé sur les plate-formes choisies, les manuels d'utilisation, description de la version mise en place

La phase de construction

- Phase concentrée sur la conception, l'implémentation et les tests
- Objectifs :
 1. Minimiser les coûts de développement et obtenir un certain degré de parallélisme
 2. Développer de façon itérative un logiciel prêt à la transition vers les utilisateurs

NB : même pour de petits projets, il faut plusieurs itérations (entre 2 et 4)

Objectif 1 : Minimiser les coûts de développement et obtenir un certain degré de parallélisme

- S'organiser autour de l'architecture
- Gestion de la configuration
- Gestion des demandes de changement
- Appliquer l'architecture
- Assurer une progression continue

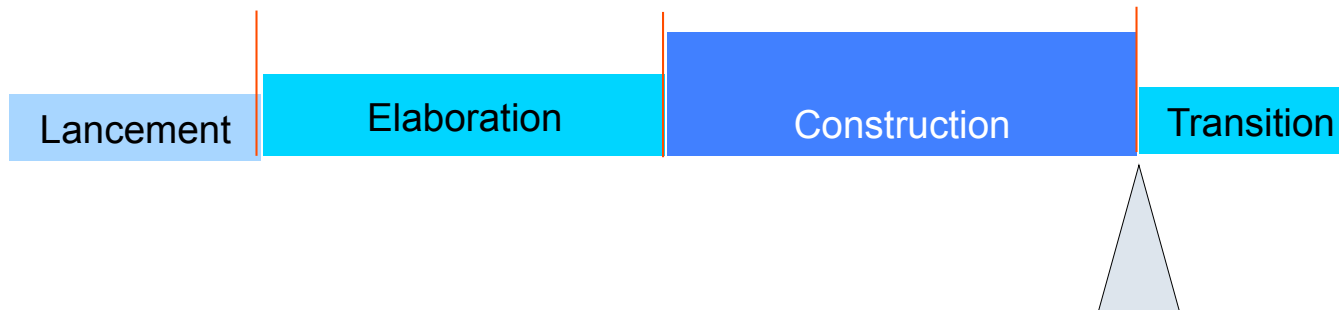
Objectif 2 : Développer de façon itérative un logiciel prêt à la transition vers les utilisateurs

- Décrire les C.U. restants et les spécifications supplémentaires (modèle des C.U, spec suppl)
- Terminer la conception (modèle de conception)
- Concevoir la base de données
- Coder et exécuter les tests unitaires (compléter les cas de test)
- Effectuer les tests d'intégration et système
- Premiers déploiements et boucle de rétroaction

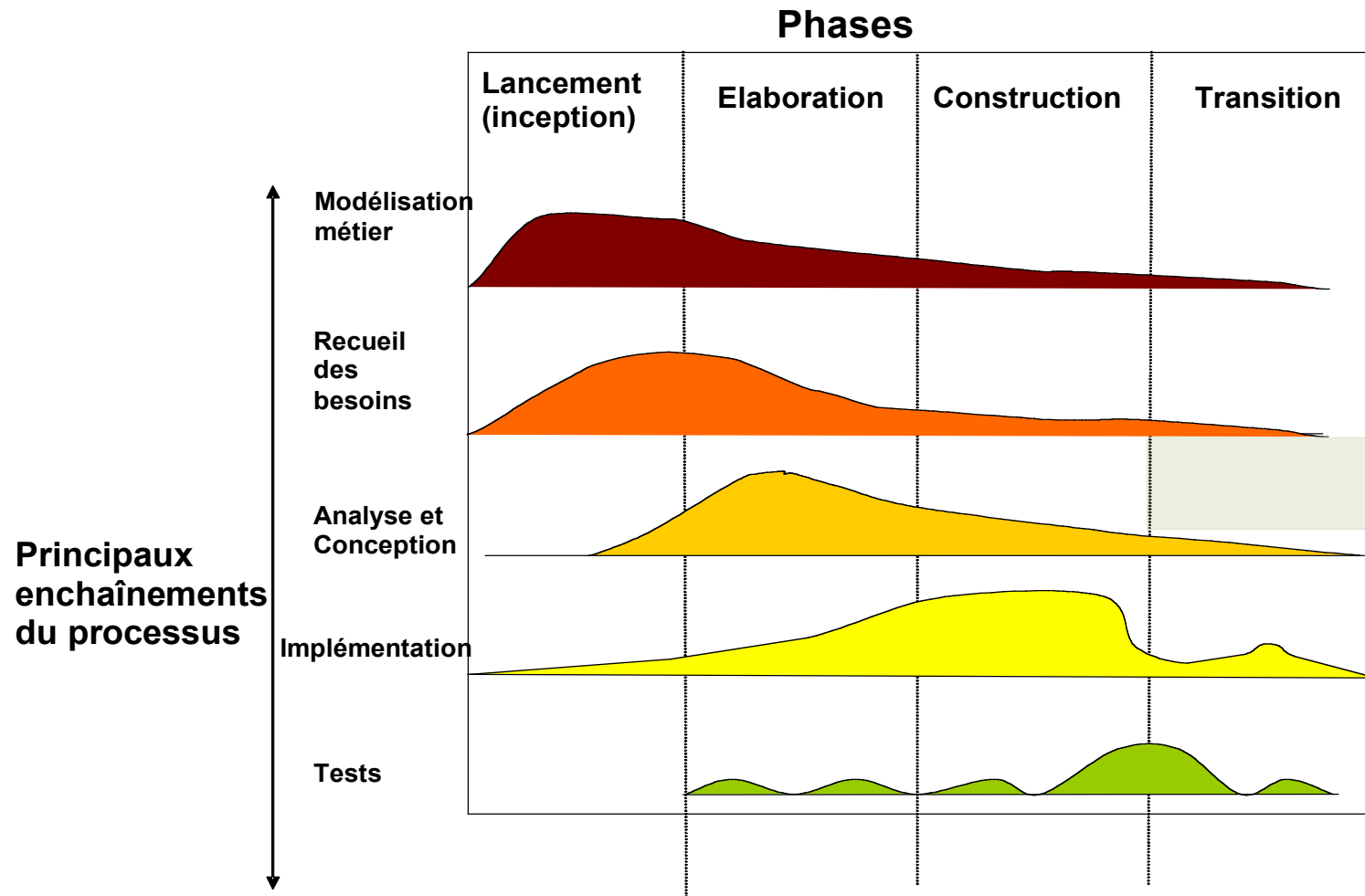
Revue de projet : le jalon fin de Construction

- Évaluation :

- Version du logiciel est elle stable ?
- Tous les intervenants sont prêts ?
- Dépenses réelles/prévisionnelles acceptables ?

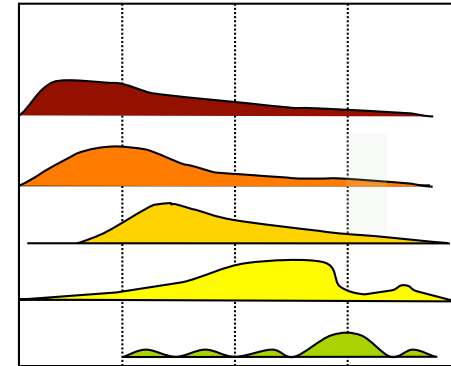


Phase de transition



Phase de transition (4/4)

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



Phase de transition

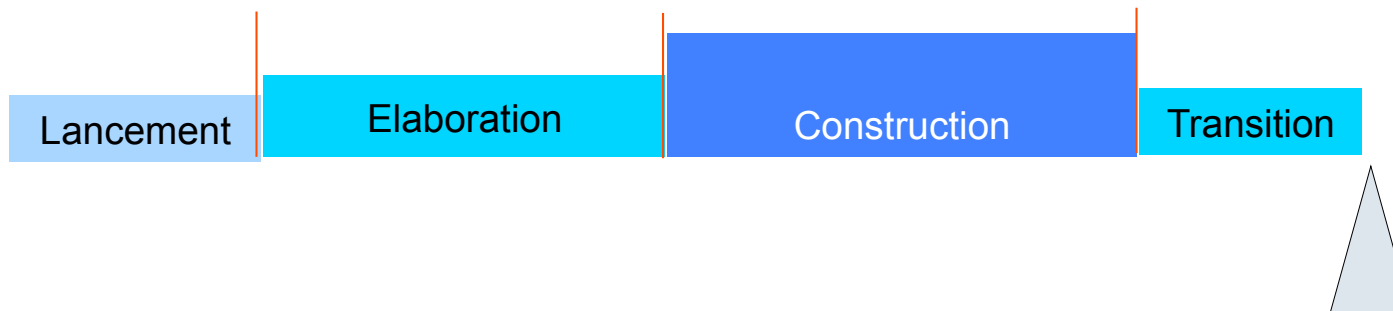


- Objectifs :

- Exécuter les tests bêta
- Former les utilisateurs
- Préparer le site de déploiement
- Préparer le lancement
- Obtenir l'accord des intervenants
- Améliorer les performances futures

Revue de projet : Jalon Fin de la Transition

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



Exemple applicatif

- Réaliser les documents de la phase de lancement pour le projet suivant:
 - Le mirail souhaite créer une plateforme pédagogique et administrative commune pour la gestion de l'enseignement à distance (possibilité de réutiliser moodle)
 - SED mirail :

http://sed.univ-tlse2.fr/servlet/com.univ.collaboratif.utils.LectureFichiergw?ID_FICHER=1317125365447&ID_FICHE=1474&INLINE=FALSE



PARTIE 4

Les rôles

Les rôles



The diagram consists of five circles arranged in a pentagonal pattern. The top circle is an empty outline. The bottom-left and bottom-right circles are solid light green. The top-right and middle-right circles are light green and contain the text for project roles. The bottom-right circle is a light green outline containing the text for testing roles.

Chef de projet

Analyste

Architecte

Développeur

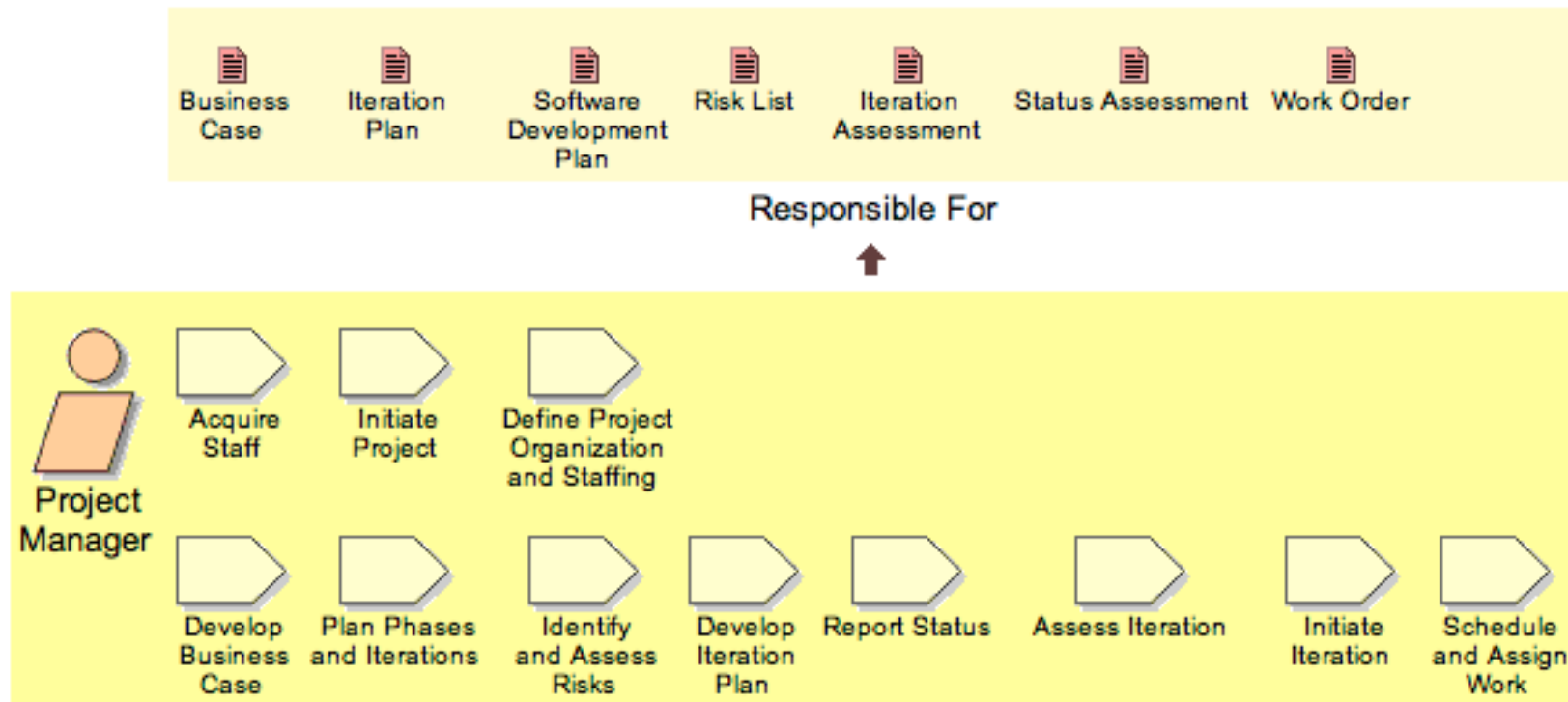
Testeur

[...]

Le chef de projet

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

Activités du chef de projet dans le RUP

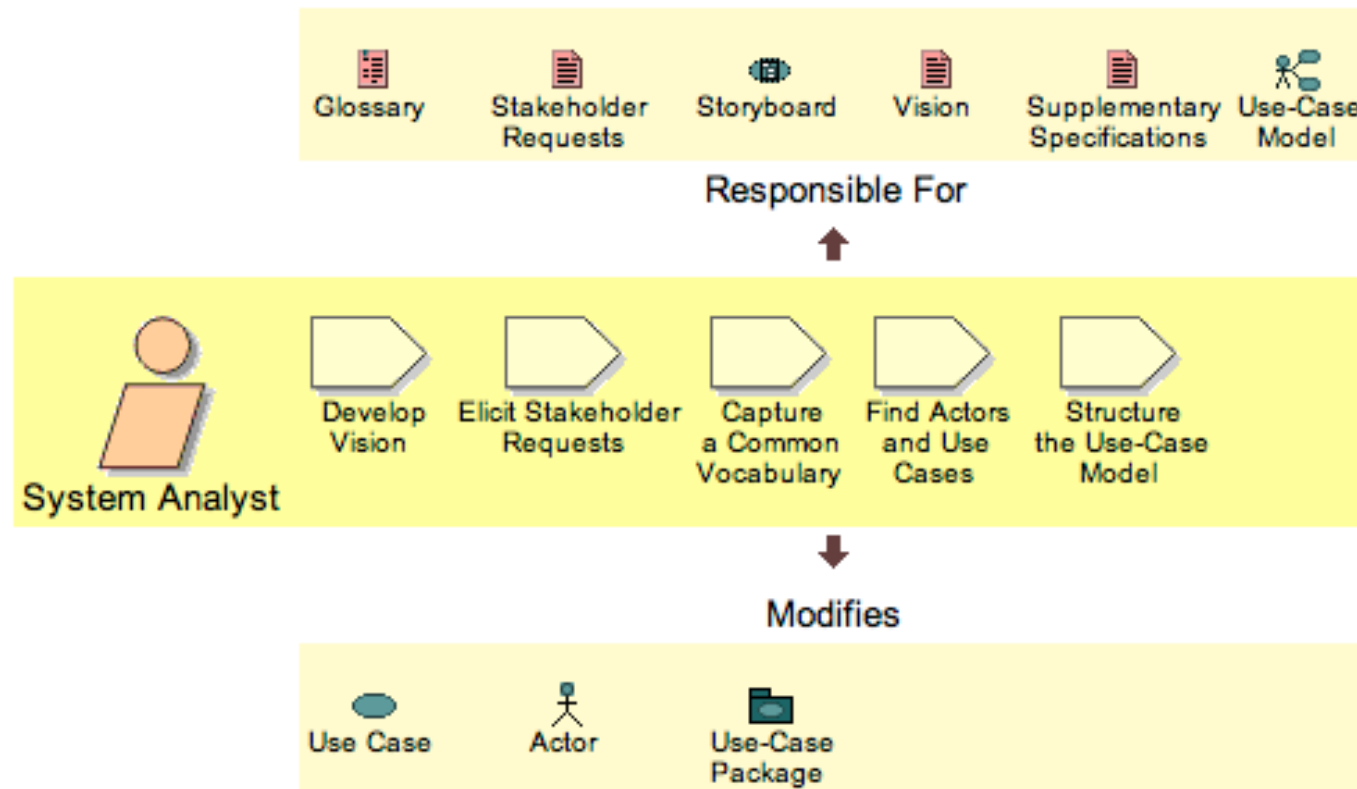


L'analyste



- Activités : Modélisation métier, Exigences, Analyse et conception
- 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

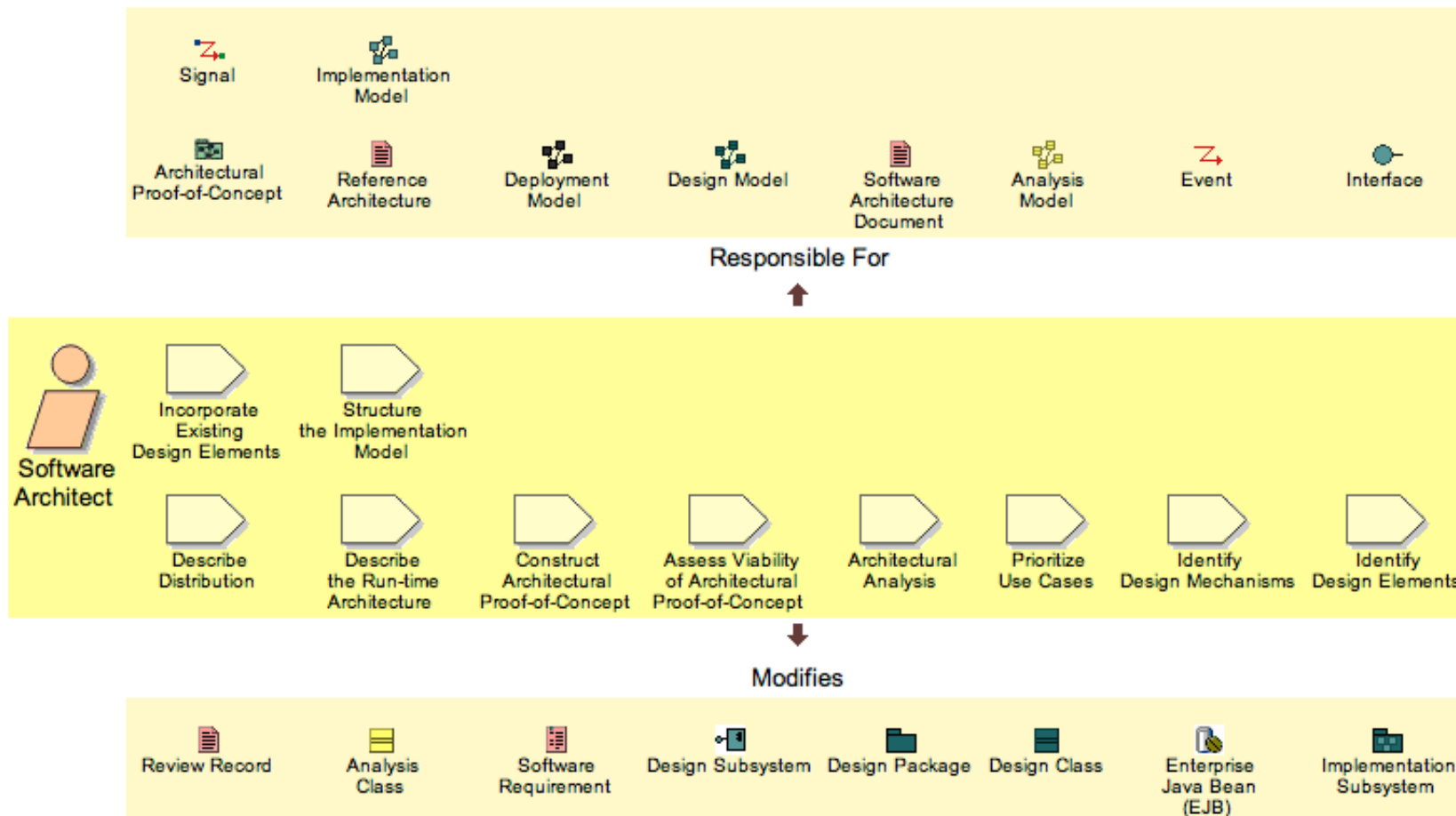


L'architecte



- Coordonne les artefacts techniques
- Communication entre les équipes de développement
- Choix d'une architecture en fonction :
performance, robustesse, réutilisation, compréhensibilité,
modularité, contraintes techniques, sûreté de fonctionnement
- Document d'architecture
- Validation par le prototype architectural

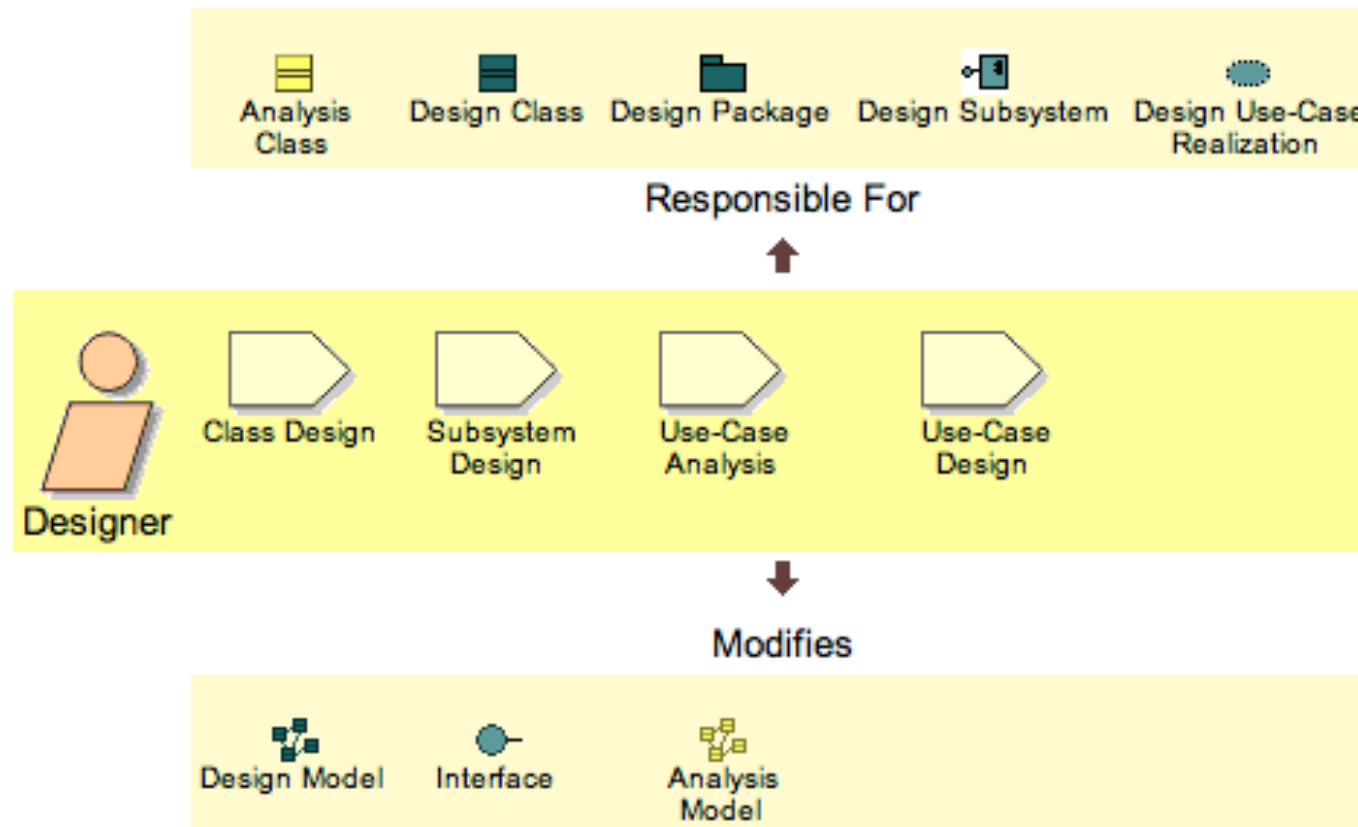
Activités de l'architecte dans le RUP



Le développeur

- Activités :
 - Analyse et conception
 - Implémentation
- Conception des réalisations des C.U.
- Conception des tests

Activités du développeur dans le RUP



Le testeur



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



PARTIE 5

La gestion de la configuration et des changements pour un travail collaboratif

Objectifs de la gestion de la configuration et des changements

- Garder la trace de tous les éléments tangibles qui participent au développement (artefacts)
- Suivre leur évolution
- Participer de manière collaborative à leur création/modification

Gestion de la configuration

- Gestion des versions :
 - gestion des sources et des espaces de travail (numéroter les versions d'artefacts, garder un historique, possibilité de retour)
 - CVS, SVN, Git, etc.
- Gestion de configuration
 - Gestion des dépendances : entre artefacts (par exemple des fichiers java)
 - PVCS, Perforce, etc.
- Liste de logiciels : http://fr.wikipedia.org/wiki/Logiciel_de_gestion_de_versions

Gestion des changements

- Traiter les modifications du système sollicitées par les intervenants
 - Analyser les conséquences
 - Suivre les demandes jusqu'à leur accomplissement

Enchaînement d'activités



1. Le chef de projet établit les procédures de gestion de la configuration
2. Le responsable de la configuration (à partir du plan d'architecture logicielle) crée et alloue les espaces de travail aux développeurs
3. Les développeurs modifient les artefacts de leur espace de travail, ils soumettent des demandes de changement
4. L'équipe établit les impacts/côûts/etc. et approuve ou non les demandes
5. Les modifications sont intégrées et testées

Définition des espaces de travail des développeurs

- Répartition structurelle:
 - Par package / sous-système
- Répartition dynamique:
 - Par C.U. / scénarios
- Deux approches :
 - ➔ Écriture exclusive
 - ➔ Écriture partagée
- ➔ Le choix des approches de répartition des espaces de travail et d'écriture dépendent des projets (et notamment de la taille du logiciel et du nombre de développeurs)

Problèmes inhérents à la gestion de configuration

- Systèmes centralisés (type cvs) :
 - Pas de gestion locale des versions
 - Attendre d'avoir une version compilable pour un commit est parfois long
 - Technique du « merge » peu fiable
- Pas de vérification de dépendance
- Solutions en cours de développement (git)



Annexes pour le projet

Redmine



- Planifiez votre plan de développement directement en utilisant des demandes sous redmine
- Générez le Gantt associé (en pdf)

Utilisation de IBM RSA

- Créer une structure de projet suivante :
 - Modèle des cas d'utilisation
 - Acteurs
 - Acteur1
 - Cas d'utilisation
 - UC01 ...
 - Diagramme des UC
 - Modèle d'analyse et de conception
 - Réalisation des UC
 - UC01
 - Diag. Seq. UC01 scenario nominal
 - UC02 ...

Utilisation de IBM RSA suite

- Modèle d'analyse et de conception (suite)
 - Package “mon logiciel”
 - Classe1
 - Diag états Classe1
 - Classe2
 - Diagramme de classe

Utilisation de IBM RSA suite

- Utilisation IBM RSA
 - Génération de doc html
 - Génération de code java
 - Vérification statique
 - Utilisation du SVN

Bibliographie

P. Kroll et P. Kruchten, *Guide pratique du RUP*,
CampussPress, 2003

P. Kruchten, *Introduction au Rational Unified Process*,
Eyrolles, 2000

Craig Larman, *Applying UML and patterns - An
introduction to object oriented analysis and design
and the Unified Process*, Prentice Hall, 2004

(<http://www.cs.bgu.ac.il/~oosd051/uploads/stuff/ApplyingUMLandPatterns.pdf>)

Exemple de projet entièrement RUP :

<http://jdbv.sourceforge.net/>