GÉNIE LOGICIEL ORIENTÉ OBJET (GLO-2004)
ANALYSE ET CONCEPTION DES SYSTÈMES ORIENTÉS OBJETS (IFT-2007)

Automne 2016

Module 02 - Processus de développement
Présentation du livrable 1

Martin.Savoie@ift.ulaval.ca

B. ing, Chargé de cours, département d'informatique et de génie logiciel

UNIVERSITÉ LAVAL

Cette semaine!

- Retour commentaires d'étudiants
- Présentation du livrable 1
- Planifier vos rencontres d'équipe échéance premier livrable 2 octobre
  - Une rencontre toutes les semaines
- Processus de développement
- Analyse des besoins
  - Phase de conceptualisation / inception

Livrable #1 (TP1) : Analyse

- Produire les livrables (artefacts) associés aux deux activités (disciplines) suivantes du Processus Unifié : **Modélisation métier (Business Modeling)** et **Exigences (Requirements)**. Les livrables doivent se trouver dans l'état où ils seraient aux termes de la première itération de la phase d'élaboration.
- **Pour « découvrir » les livrables (artefacts) attendus, vous pouvez consulter le tableau reproduit dans l'énoncé du TP1.**
- Produires les maquettes/esquisses d'interface de votre application accompagné d'explication. Vous devez détailler les éléments qui ne sont pas trivial pour un néophyte.
- Vous devez en plus **produire un plan (Gantt)** illustrant le travail prévu pour les prochaines itérations (au sens défini par le Processus Unifié) jusqu'à la fin de la session.
- Votre rapport doit prendre la forme d'un seul document continu (format PDF) contenant textes et diagrammes.
- Vous devrez également fournir le fichier Visual Paradigm (un seul fichier) utilisé pour produire les schémas.
- **Évaluation de la contribution au sein de l'Équipe**

Livrable #2 (TP2) : Conception/interface

- Produire un modèle de conception (design model) comportant notamment les diagrammes UML pertinents.
- Fournir un Gantt mis à jour (en prenant soin d'indiquer l'état actuel des travaux).
- Programmer l'interface
- (Plus de détails à venir dans l'énoncé du TP2)

Livrable #3 (TP3) : Construction et démonstration

- Modèle de conception mis à jour
- Gantt mis à jour
- Démonstration d'une version préliminaire du logiciel
- (Plus de détails à venir dans l'énoncé du TP3)

Livrable #4 (TP4) : Construction et démonstration

- Modèle de conception mis à jour
- Gantt mis à jour
- Démonstration de la version finale du logiciel
- (Plus de détails à venir dans l'énoncé du TP4)

Bénéfices du projet

- Projet de grande taille
- Nombreux design alternatifs
- Les défis du travail d'équipe
- Réalisation concrète
- Un prix est en jeu!
- Le rôle du professeur: le client
- Le rôle du chargé de lab: consultant senior au sein de votre entreprise.

Livres de Java

- Disponible en version électronique (service « Safari » de la bibliothèque)

Wiki Java

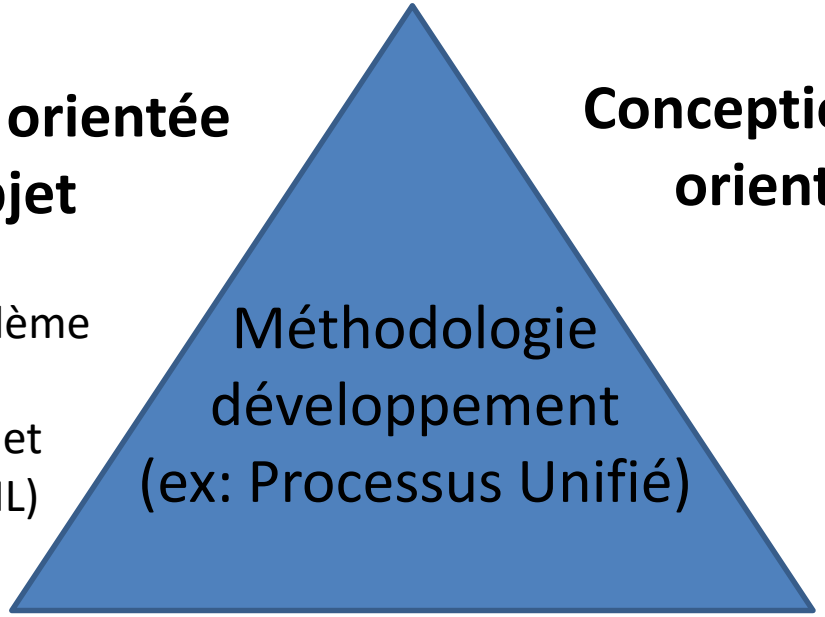
- Développé pour le cours
- Une série d'atelier « pas à pas » pour la construction d'une application en Java

Méthodologie/ processus de développement

Génie logiciel orienté objet

Analyse orientée objet

- Comprendre le problème
- Décrire la situation à l'aide de documents et diagrammes (ex: UML)



Conception (design) orienté objet

- Concevoir une solution informatique
- Tracer des plans (plus ou moins détaillés) sous la forme de documents et diagrammes (ex: UML)

Programmation orientée objet

Mettre en œuvre la solution à l'aide d'un langage (ex: Java)

UNIVERSITÉ LAVAL

Méthodologie / processus de développement logiciel

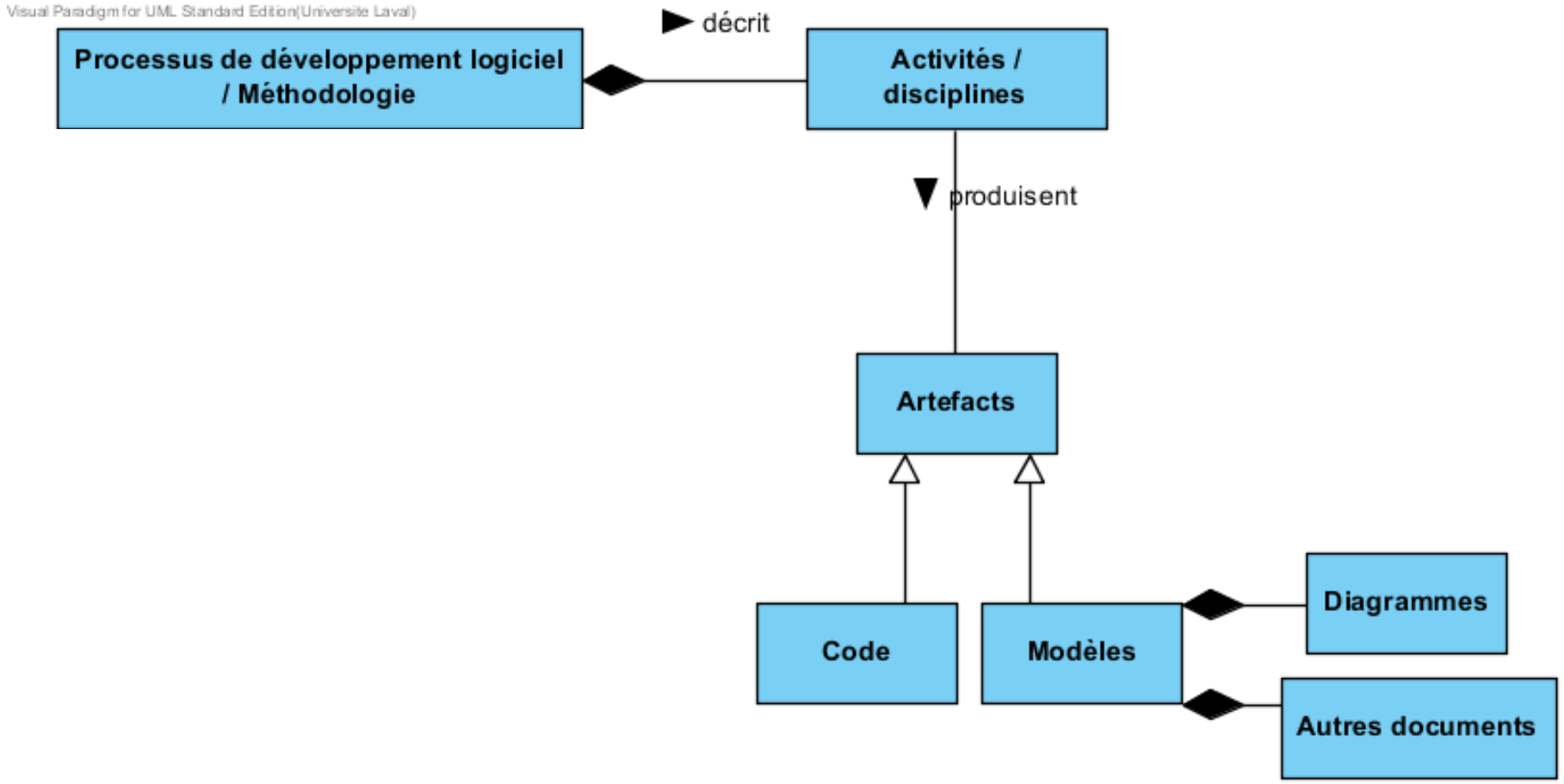
- Précise une méthodologie à suivre pour développer un logiciel
- Spécifie les étapes / activités à réaliser et les livrables (« artefacts ») à produire
- Décrit notamment quand et comment procéder à l'analyse, à la conception, à la programmation, etc..

Processus de développement de logiciel

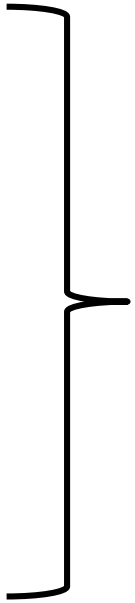
- Une **méthode** de design **orientée objet** repose sur **trois bases**:
  - une ***notation*** permettant de décrire les ***modèles (ex: UML)***;
  - un ***processus*** de construction des ***modèles***;
  - des ***outils*** facilitant la construction et la description des modèles.

Le processus **doit permettre de gérer**

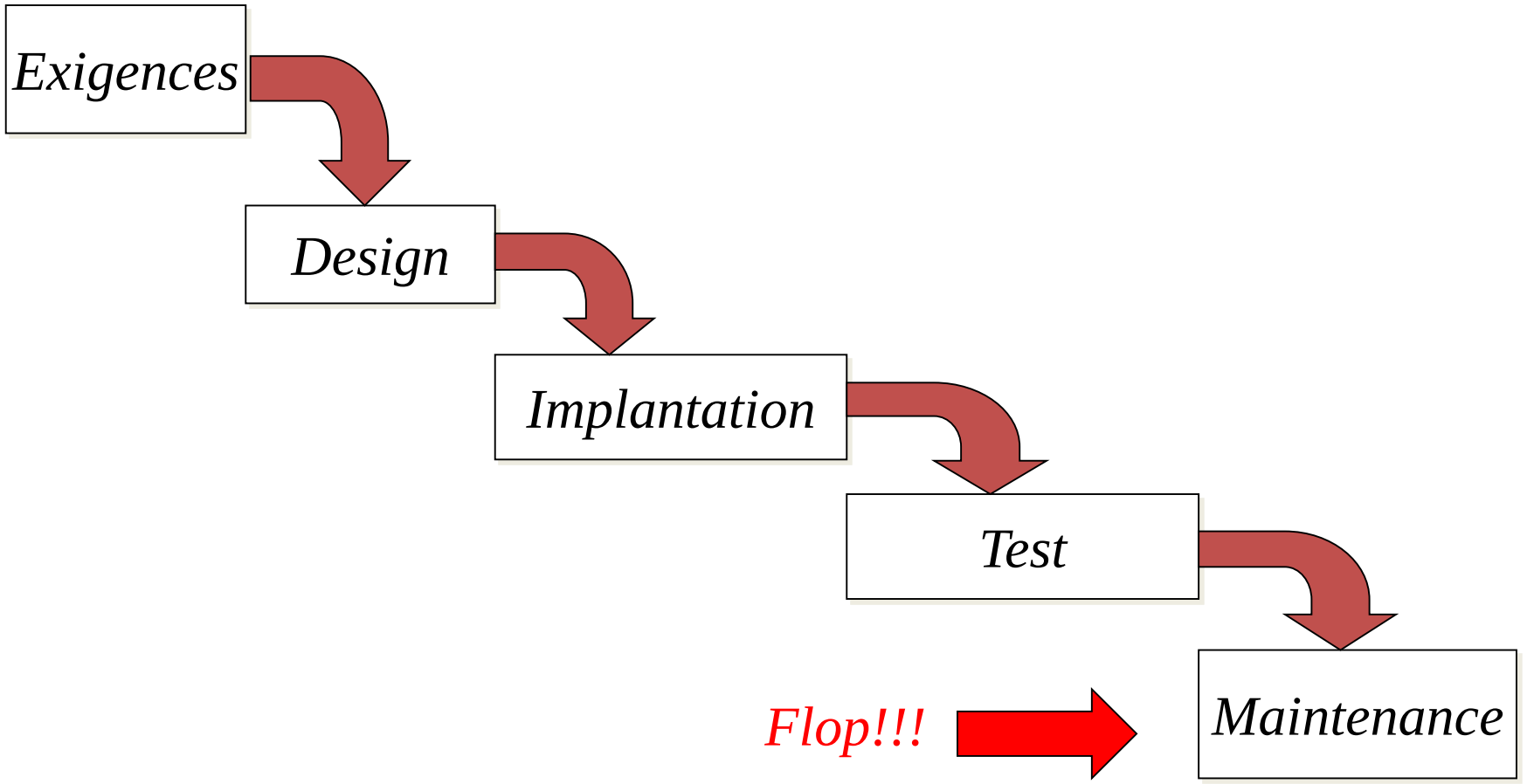
- la ***complexité*** du ***problème***;
- la ***complexité*** de gestion du ***processus*** de conception du logiciel par une équipe (plusieurs solutions sont possibles et il faut choisir la meilleure);
- la ***complexité*** associée à la ***flexibilité*** offerte par le logiciel;



- Plusieurs approches de développement de logiciel existent:
  - Waterfall (“en cascade”) – Méthode d’ingénierie classique
  - Spirale
  - Unified Process (UP)
  - Méthodes agiles (Agile Software Development Methods)
  - Méthodes “extrêmes” (Extreme Programming Methods)

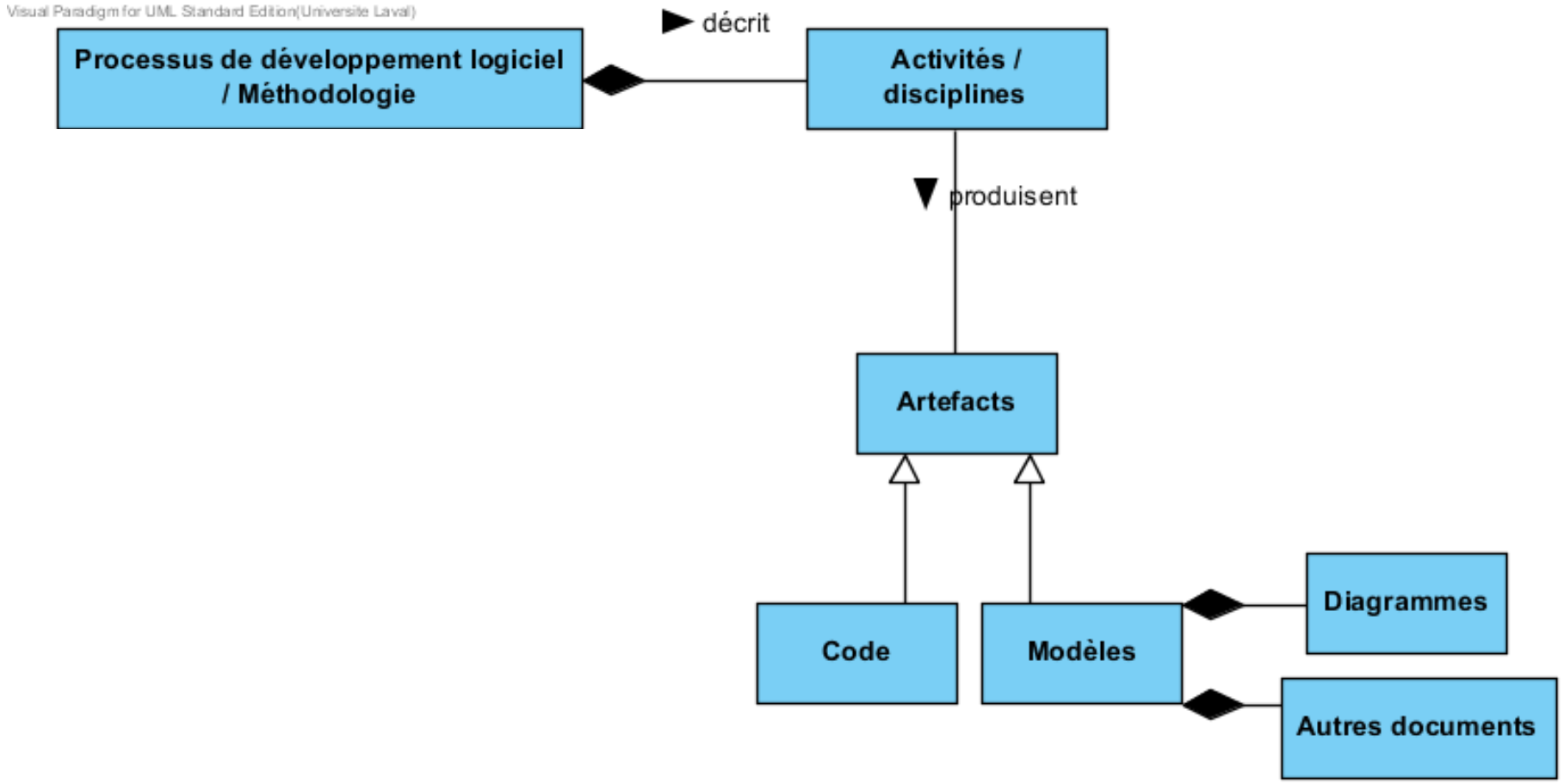


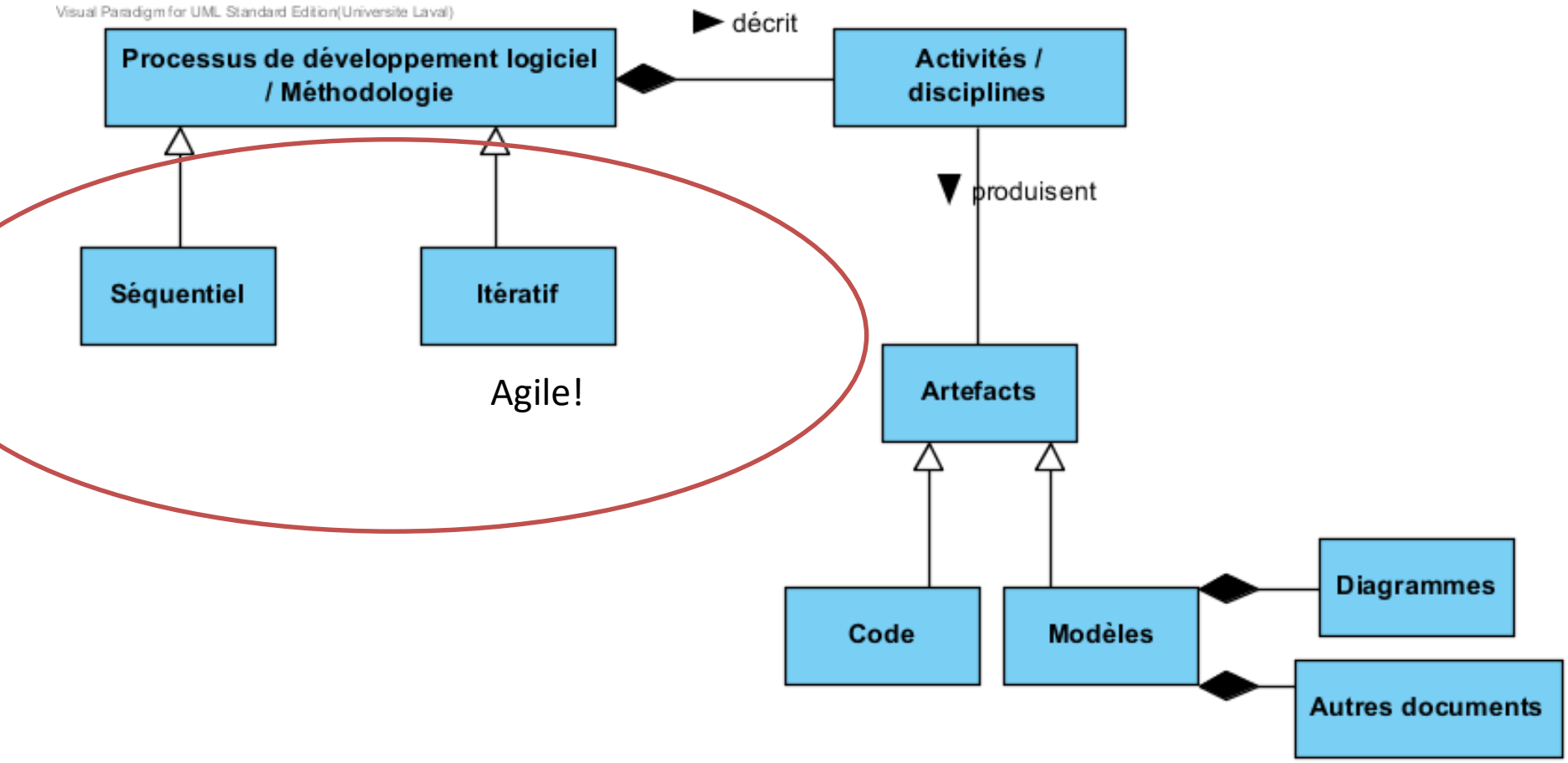
Modèle *WATERFALL (en cascade)* *de développement de logiciels*



Waterfall

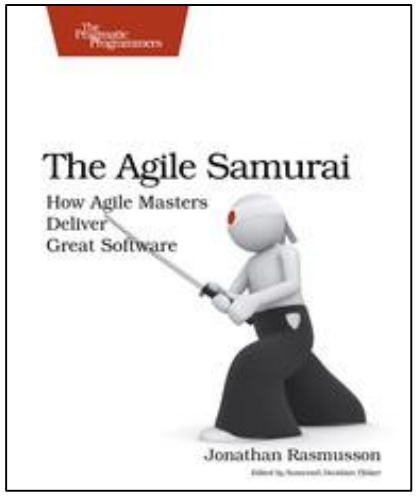
- Approche classique d'ingénierie formée d'étapes successives
  - Analyse, Conception, Implantation, Integration, Test, Transition
- Cette méthode est très rigide et exige que toutes les spécifications soient complètes avant que la conception et l'implantation commencent
- Elle est applicable avec succès pour de petits projets, mais mène à des échecs cuisants sur des projets d'envergure s'étalant sur de longues périodes de temps





Méthodes itératives

- Approche impliquant de courtes itérations (ex: deux semaines) incluant toutes les étapes de développement logiciel:
  - Requirements, design, implantation, intégration, test et acceptation par le client
- Favorise le travail en équipe en mettant l'accent sur la collaboration entre les membres de l'équipe plutôt que sur la documentation
- Favorise la participation étroite du client au développement du produit
- Exemple: Méthode agile



http://books.openlibra.com/pdf/AgileSamurai.pdf

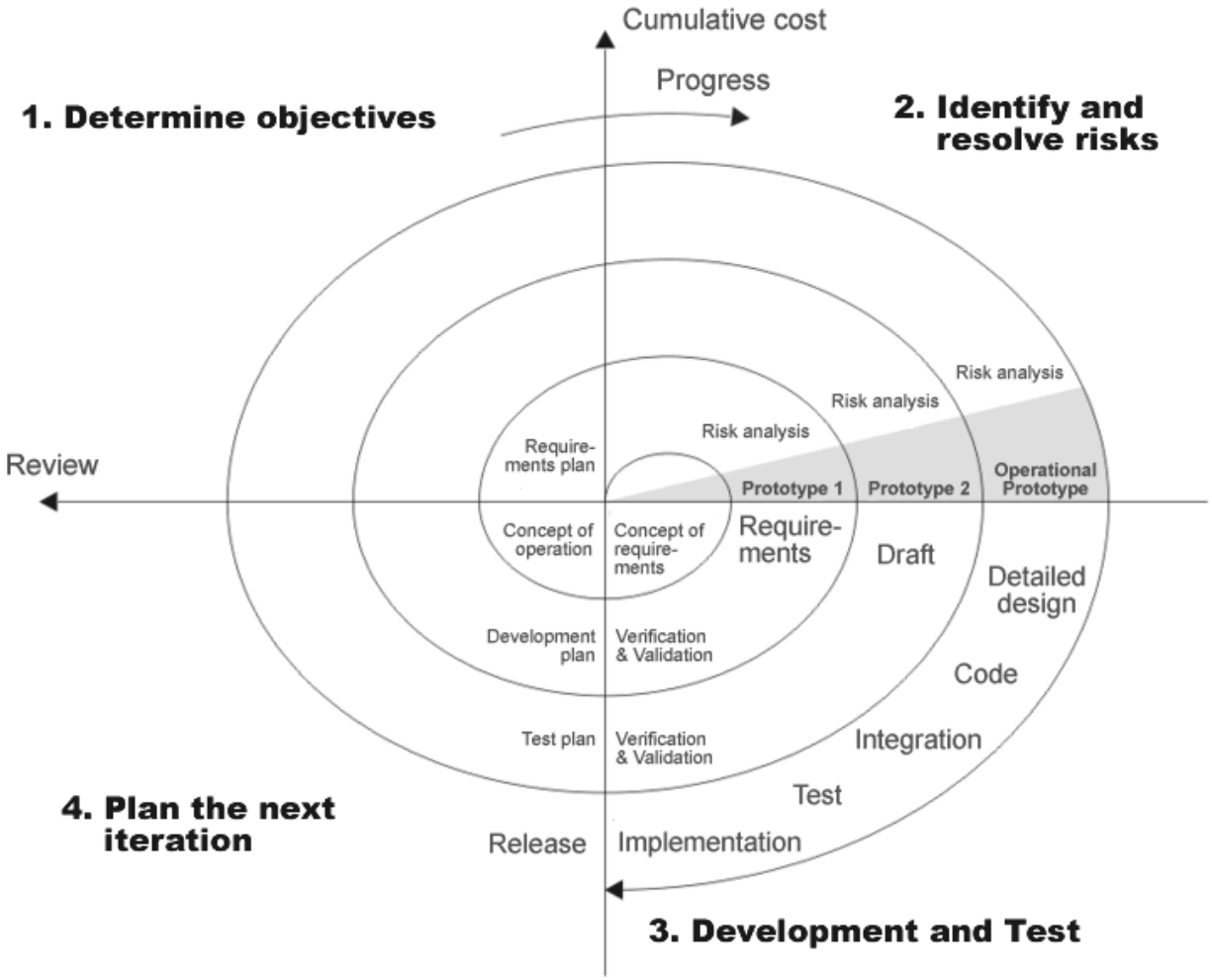
Avantages d'une méthode itérative

- Réduction des risques
  - Identifier plus tôt les problèmes potentiels
  - Évaluer rapidement la vitesse de progression
- Développer une architecture robuste
  - Évaluation rapide de l'architecture: amélioration, correction possible
- Gérer des besoins évolutifs (et de façon évolutive)
  - Les utilisateurs (le client) donne une rétroaction constante
  - Répondre à ces rétroactions nécessite un changement incrémental (plutôt qu'une refonte totale du système).
- Permettre le (les) changement(s)
  - Adapter le système aux problèmes, aux besoins, aux requis
- Apprendre tôt dans le processus de développement
  - Tout le monde obtient une compréhension du domaine (des processus relatif au domaine) tôt.

Spirale

- Approche classique visant à réduire les impacts négatifs de la méthode Waterfall en lui ajoutant des étapes de prototypage. Méthode proposée par Boehm à la fin des années 1980.
- Combine une approche Top-Down (analyse à la Waterfall) à une approche Bottom-Up (développement par prototypage)
- Propose un développement en itérations assez longues (6 mois à 2 ans) faisant évoluer des prototypes vers le produit final avec la participation du client

Spirale (suite)



http://en.wikipedia.org/wiki/File:Spiral_model_(Boehm,_1988).png

- Mis à part l'approche Waterfall, les méthodes en spirale, PU, et les méthodes agiles en général, favorisent l'aspect itératif et incrémental.
- Au lieu de voir toutes ces méthodes, nous allons nous concentrer sur l'apprentissage de l'une d'entre elles, l'apprentissage des autres se faisant avec un effort plus réduit
- Nous allons nous concentrer sur l'approche PU

Processus Unifié (PU)

- Processus de développement itératif reposant sur UML et proposé à l'origine par Booch, Rumbaugh et Jacobson (livre publié en 1999)
- De nouvelles versions (Rational Unified Process) ont été publiées depuis (propriété de IBM)
- Le processus original (UP) peut être vu comme un tronc commun à plusieurs méthodes itératives

	Activités (appelées *disciplines* dans le Processus Unifié)	**Modèles** et artefacts générés
Analyse	Modélisation domaine d'affaires / Business modeling / Modélisation métier *synonymes!*	**Modèle du domaine**: (1) diagramme de classe « conceptuel », (2) parfois un diagramme d'activités
	Analyse des besoins / Exigences / Requirements	(3) Énoncé de vision
		Modèle de cas d'utilisation / Use-case model : (4) diagramme des cas d'utilisation, (5) texte des cas d'utilisation, (6) diagramme de séquence système
		(7) Spécifications supplémentaires
		(8) Glossaire
	Design / Conception	**Modèle de conception / Design model** : (9) diagrammes de classes, (10) diagrammes d'interaction, (11) tout autre diagramme UML pertinent selon le contexte
	Implémentation	(12) Code
	…	

Comme dans le processus en cascade?

- NON!

UP: Plusieurs itérations, à chaque itération on fait du travail relié à plusieurs disciplines

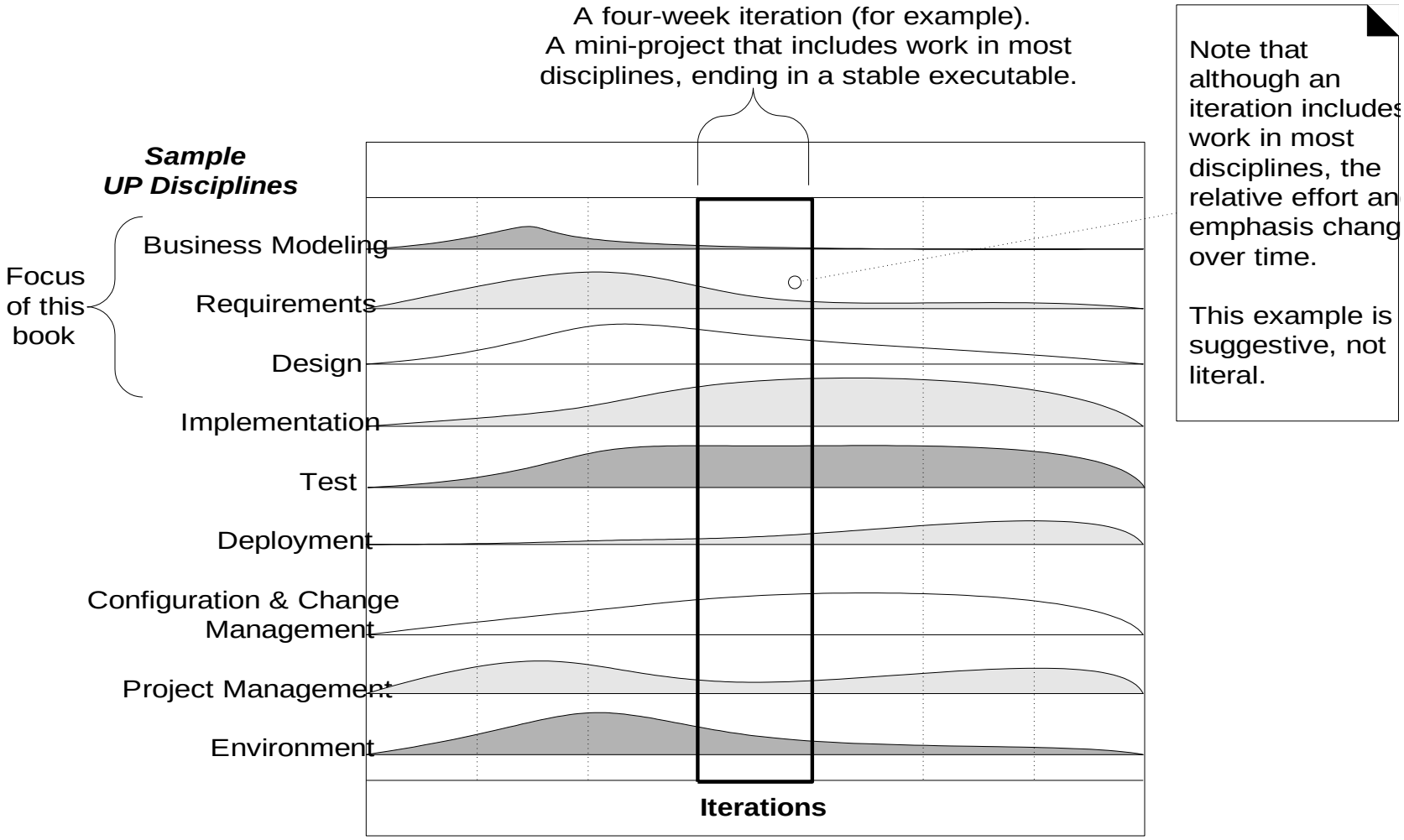
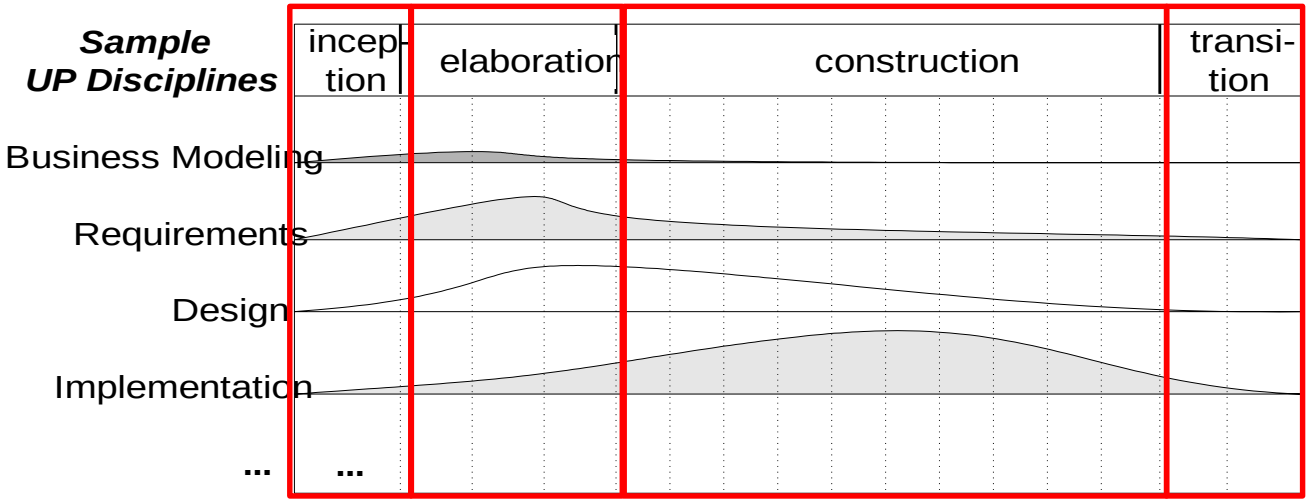
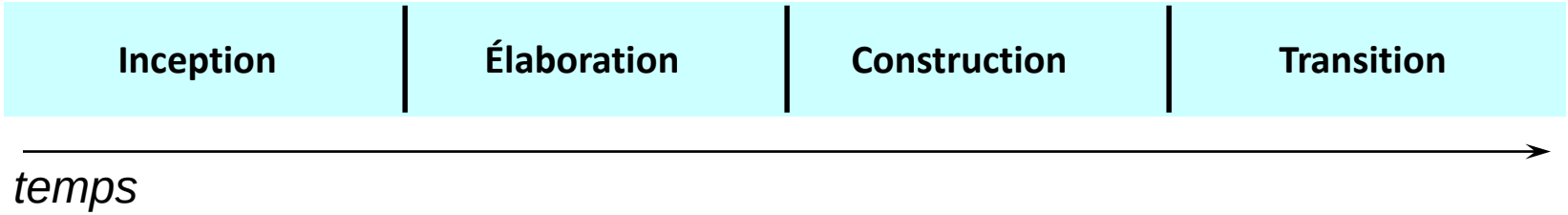


Fig. 2.7

Dans UP, les itérations sont regroupées en grandes phases

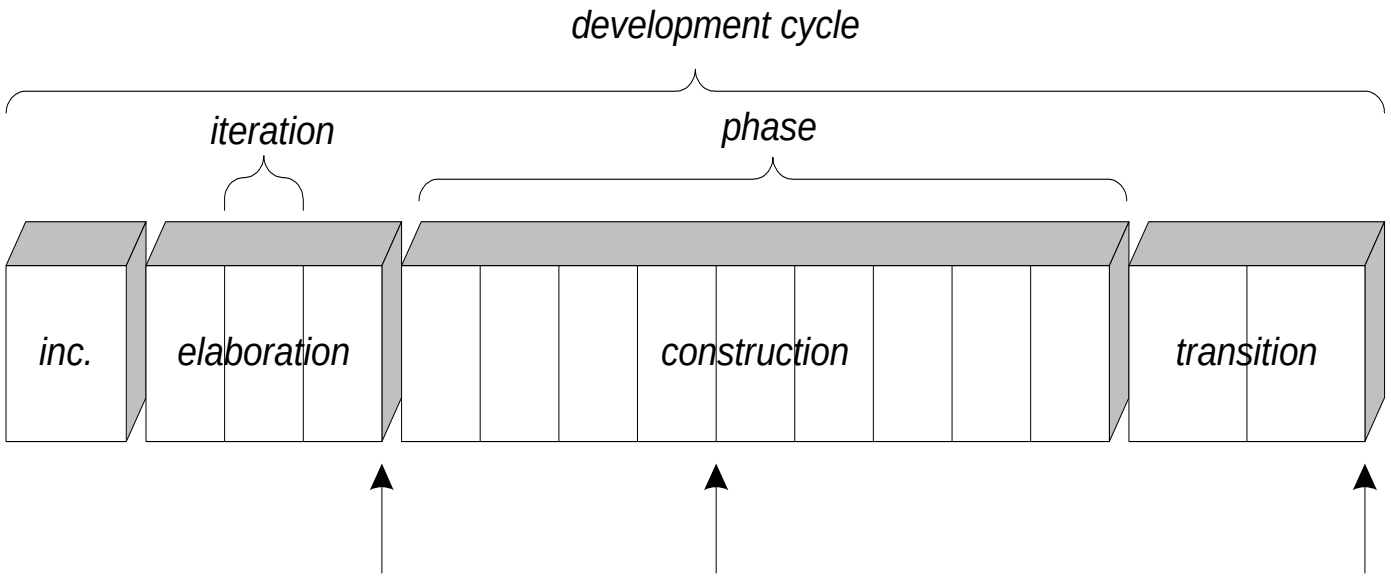


Phases du processus unifié (PU) - Unified Process (UP)



- Le PU organise le travail et les iterations en 4 phases principales:
  - Conceptualisation/Inception: Définir la portée du projet
  - Élaboration: Planifier le projet, développer les fonctionnalités et l'architecture de base
  - Construction: Terminer le développent du produit
  - Transition: Transférer le produit à sa communauté d'utilisateurs

Fig. 2.6



milestone

An iteration end-point when some significant decision or evaluation occurs.

release

A stable executable subset of the final product. The end of each iteration is a minor release.

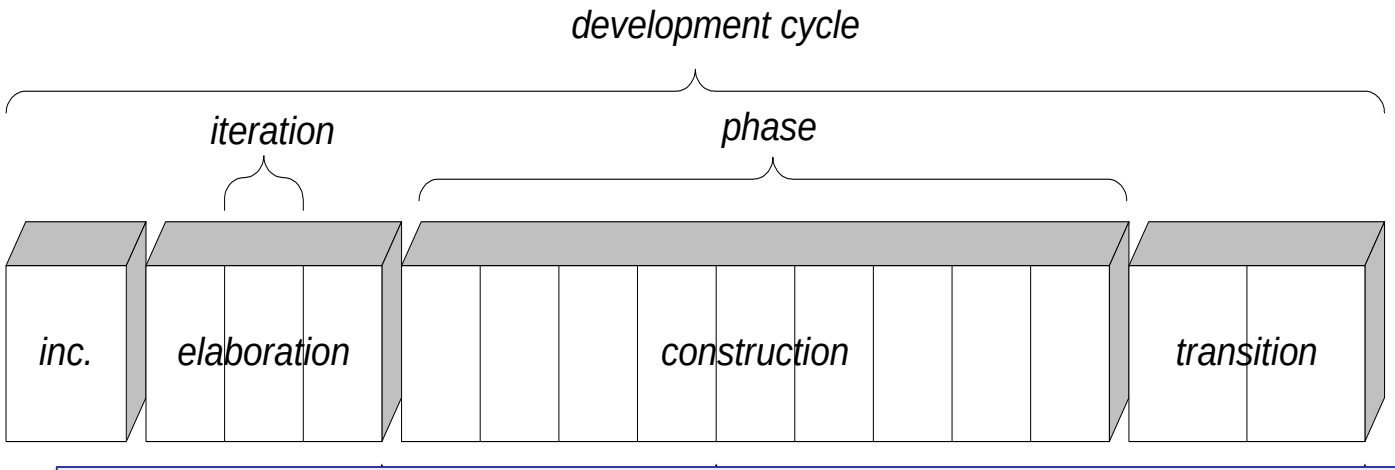
increment

The difference (delta) between the releases of 2 subsequent iterations.

final production release

At this point, the system is released for production use.

Fig. 2.6



4 grandes *phases*

Chaque phase est composée d'*itération**S.***

Dans chaque itération, on réalise du travail associé à plusieurs *activités* / disciplines.

Les *artefacts* sont les livrables produits.

Discipline	Artifact Iteration	Incep. Il	Elab. EL.En	Const. CL.Cn	Trans. T1..T2
Business Modeling	Domain Model		s		
Requirements	Use-Case Model	*s*	r		
	Vision	*s*	r		
	Supplementary Specification	*s*	r		
	Glossary	*s*	r		
Design	Design Model SW Architecture Document Data Model		s s s	r r	
Implementation	Implementation Model (code)		s	r	r

S: start
R: refine

	Activités (appelées *disciplines* dans le Processus Unifié)	**Modèles** et artefacts générés
Analyse	Modélisation domaine d'affaires / Business modeling / Modélisation métier *synonymes!*	**Modèle du domaine**: (1) diagramme de classe « conceptuel », (2) parfois un diagramme d'activités
	Analyse des besoins / Exigences / Requirements	(3) Énoncé de vision
		Modèle de cas d'utilisation / Use-case model : (4) diagramme des cas d'utilisation, (5) texte des cas d'utilisation, (6) diagramme de séquence système
		(7) Spécifications supplémentaires
		(8) Glossaire
	Design / Conception	**Modèle de conception / Design model** : (9) diagrammes de classes, (10) diagrammes d'interaction, (11) tout autre diagramme UML pertinent selon le contexte
	Implémentation	(12) Code
	…	

À faire cette semaine

- Lecture des chapitres 1, 2 et 3
- S'assurer de pouvoir comprendre et définir les termes et concepts suivants
    - Processus / méthodologie, analyse, conception, artefact, modèle, UML, itération, phase

- Formation des équipes (déjà fait?)
- Lecture (et compréhension) de l'énoncé pour le projet de session/livrable 1
- Installation Visual Paradigm
- Installation NetBeans
- S'intéresser à Java