

OpenUP

Duvivier Julien - Orban Pierre-Yves

30/04/2013

Plan

1. OpenUP, c'est quoi ?
2. Principes
3. Organisation
 - a. **Méthode**
(1) Rôles, (2) Disciplines, (3) Tâches , (4) Artifacts, (5) Directives
 - b. **Processus**
(1) Patterns, (2) Cycle de vie itération, (3) Delivery process
4. Micro incrément
5. Influences

OpenUP, c'est quoi ?

- 2005: Basic Unified Process par IBM
- 2006: OpenUP et EPF par Eclipse Foundation
- Process itératif basé sur RUP
 - Minimal
 - Complet
 - Extensible
- Approche agile
 - Collaboration et communication

Principes

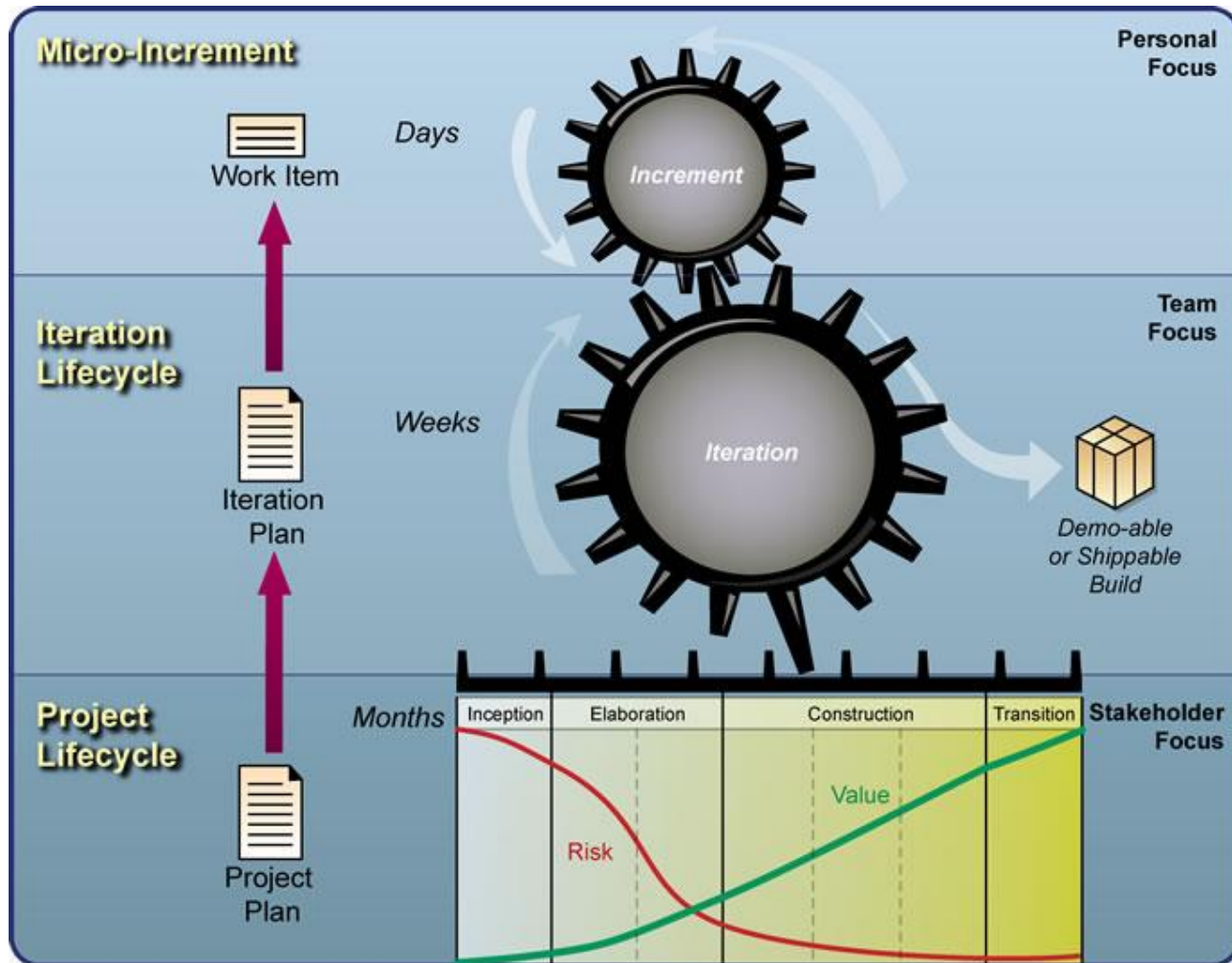
1. Collaborer pour aligner ses intérêts et partager les connaissances
2. Equilibrer les priorités afin de maximiser le bénéfices des parties prenantes
3. Se concentrer sur l'architecture très tôt pour minimiser les risques et organiser le développement
4. Evoluer pour continuellement recevoir des feedbacks et s'améliorer

Principes // Manifeste agile

1. Collaborer pour aligner ses intérêts et partager les connaissances
--> *"Individuals and interactions over process and tools"*
2. Equilibrer les priorités afin de maximiser le bénéfices des stakeholders
--> *"Customer collaboration over contract negotiation"*
3. Se concentrer sur l'architecture très tôt pour minimiser les risques et organiser le développement
--> *"Working software over comprehensive documentation"*
4. Evoluer pour continuellement recevoir des feedbacks et s'améliorer
--> *"Responding to change over following a plan"*

Organisation

Niveaux :



Méthode: les rôles (1)

- Rôles
 - Parties prenantes
 - Analyste
 - Architecte
 - Développeur
 - Testeur
 - Project Manager
 - Any role
- + 6 rôles spécifiques du déploiement
- + 2 rôles spécifiques de l'environnement

Méthode: les disciplines (2)

La méthode se concentre sur les disciplines suivantes :

1. Exigences
2. Architecture
3. Développement
4. Test
5. Project Management
6. Configuration & change management

Méthode: les tâches (3)

- *"Unit of work a role may be asked to perform"*
- Primary performer vs Additional performers
- Environ 35 tâches définies par défaut réparties dans 7 catégories (5 disciplines + deployment & environnement)
Ex:
 - Envision the architecture (Architecture)
 - Package the release (Deployment)
 - Create test cases (Test)
 - Deploy the process (Environment)
 - ...

Méthode: Artefacts (4)

- *"Something that is produced, modified or used by a task."*
- Pas de formalisme dans la représentation des artefacts
 - Ex: Photo d'un tableau blanc présentant l'architecture
- Environ 30 Artifacts définis par défaut réparties dans 7 catégories
Ex:
 - Glossary (Requirement)
 - Risk list (Project management)
 - Deployment plan (Deployment)
 - ...

Méthode : Directives (5)

- Ensemble de conseils et de checklist utiles pour atteindre les objectifs

The screenshot displays the OpenUP web application interface. On the left is a navigation tree with categories like 'Team', 'Guidance', 'Checklists', 'Concepts', and 'Examples'. The 'Examples' section is expanded, showing '4+1 Views of Software Architecture'. The main content area is titled 'Guidance > Examples > 4+1 Views of Software Architecture' and 'Example: 4+1 Views of Software Architecture'. It contains a paragraph explaining the example, followed by a 'Main Description' section. This section includes a diagram of the 4+1 Views of Software Architecture, which consists of a central 'Use-Case View' surrounded by four other views: 'Logical View', 'Implementation View', 'Process View', and 'Deployment View'. Below the diagram, there is a list of descriptions for each view: 'Use-case view', 'Logical view', 'Implementation view', and 'Process view'.

OpenUP

Where am I | Tree Sets |

View Discussion Edit New History

Home Manage

Guidance > Examples > 4+1 Views of Software Architecture

Example: 4+1 Views of Software Architecture

This example describes a possible set of views for describing a software architecture.

Expand All Sections Collapse All Sections

Relationships

Related Elements

- Architectural Views and Viewpoints

Back to top

Main Description

You may want to consider the following views (not all views are relevant to all systems or all the stakeholders). This set of views is known as the 4+1 Views of Software Architecture [KR95].

Logical View

Implementation View

Use-Case View

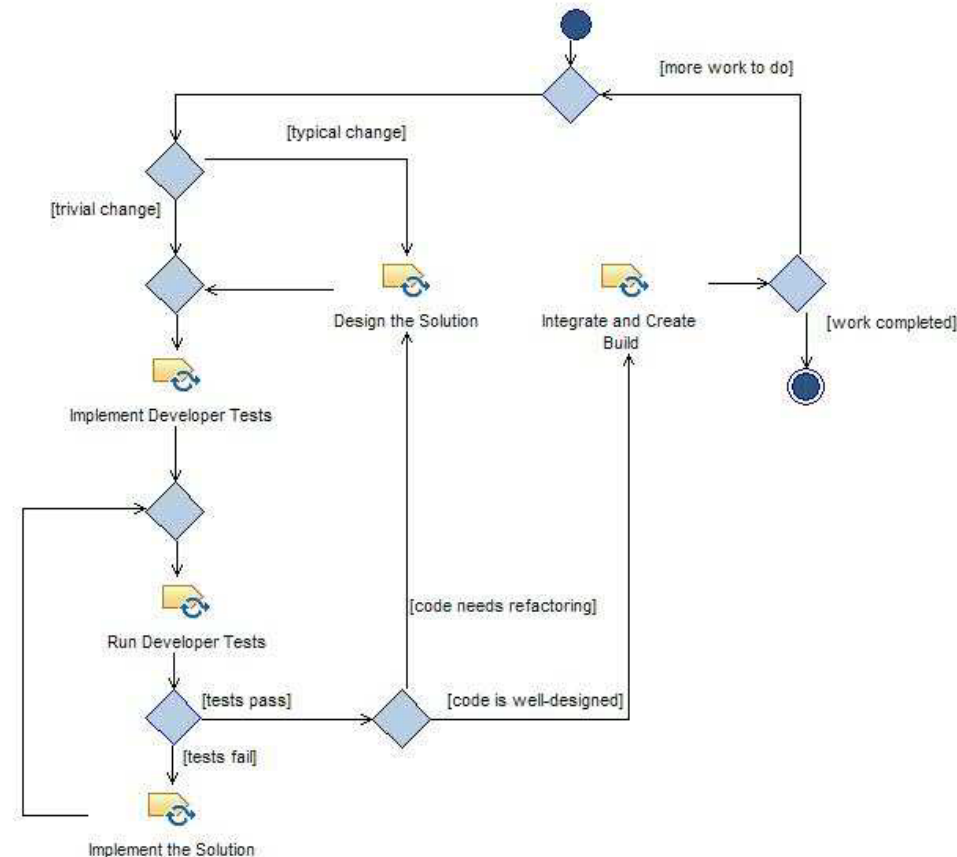
Process View

Deployment View

- Use-case view:** Describes functionality of the system, its external interfaces, and its principal users. This view is mandatory when using the 4+1 Views, because all elements of the architecture should be derived from requirements.
- Logical view:** Describes how the system is structured in terms of units of implementation. The elements are packages, classes, and interfaces. The relationship between elements shows dependencies, interface realizations, part-whole relationships, and so forth. Note: This view is mandatory when using the 4+1 Views of Software Architecture.
- Implementation view:** Describes how development artifacts are organized in the file system. The elements are files and directories (any configuration items). This includes development artifacts and deployment artifacts. This view is optional when using the 4+1 Views.
- Process view:** Describes how the run-time system is structured as a set of elements that have run-time behavior and interactions. Run-time structure often bears little resemblance to the code structure. It consists of readily changing networks of communication objects. The elements are components that have run-time processes (processes).

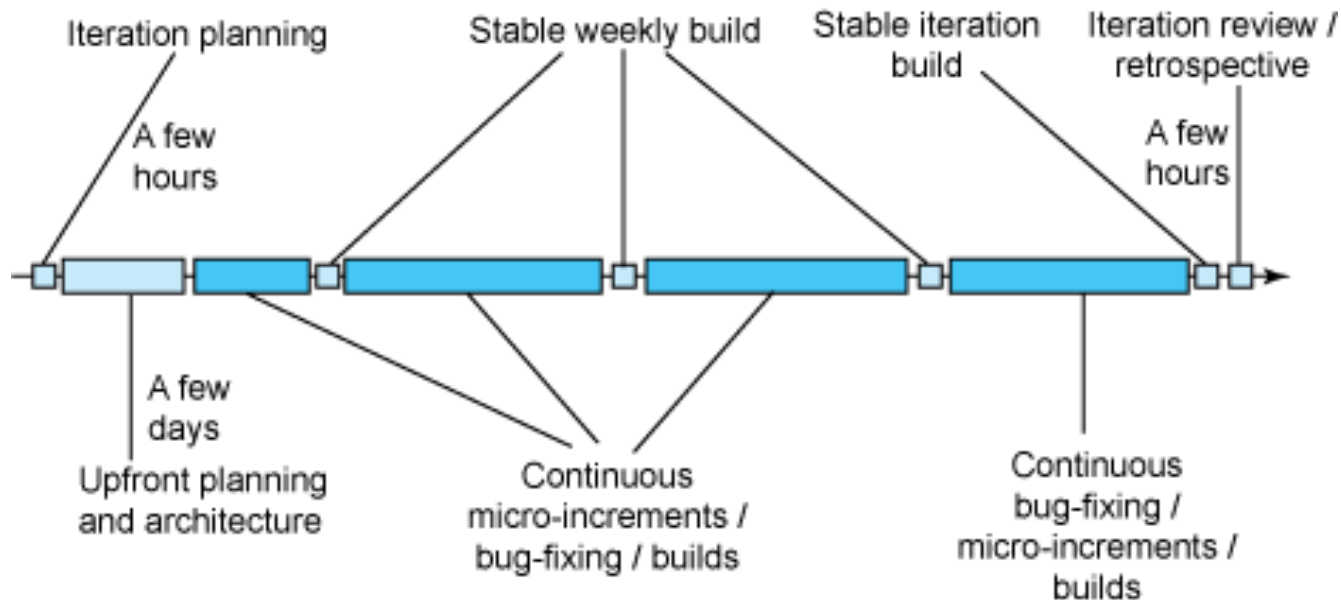
Processus : Patterns (1)

- *Définit l'ensemble des tâches à effectuer pour répondre à un besoin du projet.*
- Itéré tant que l'objectif n'est pas atteint.
- Unité de base du Delivery process



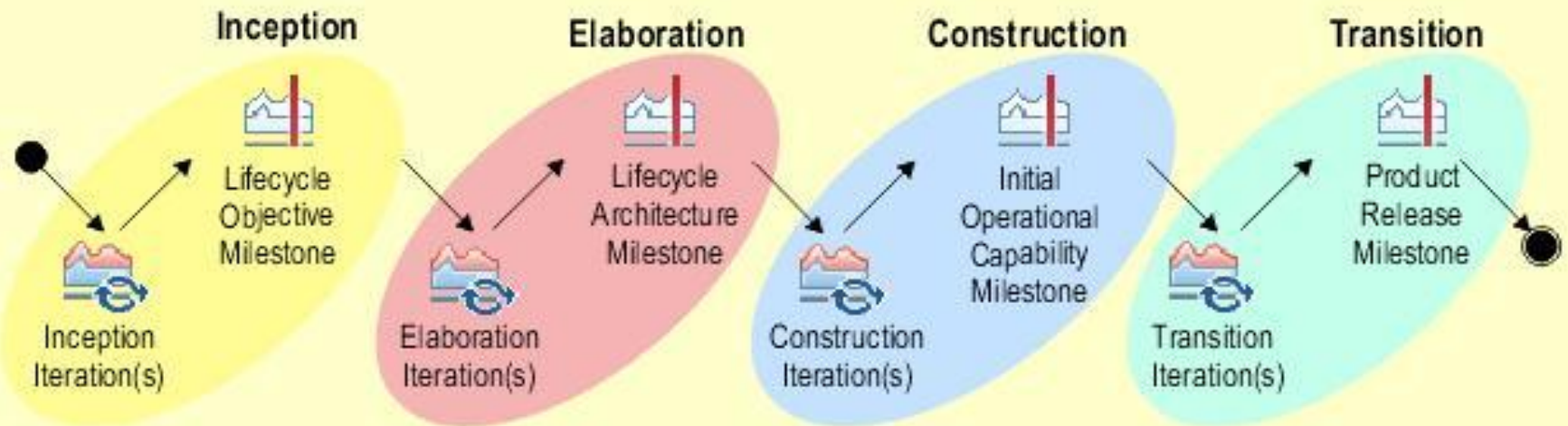
Processus : Cycle de vie d'une itération (2)

1. Meeting pour planifier l'itération
2. Exécuter et tester les micro incréments
3. Résolution des erreurs et possible ajout de nouvelles caractéristiques
4. Rétrospective et évaluation

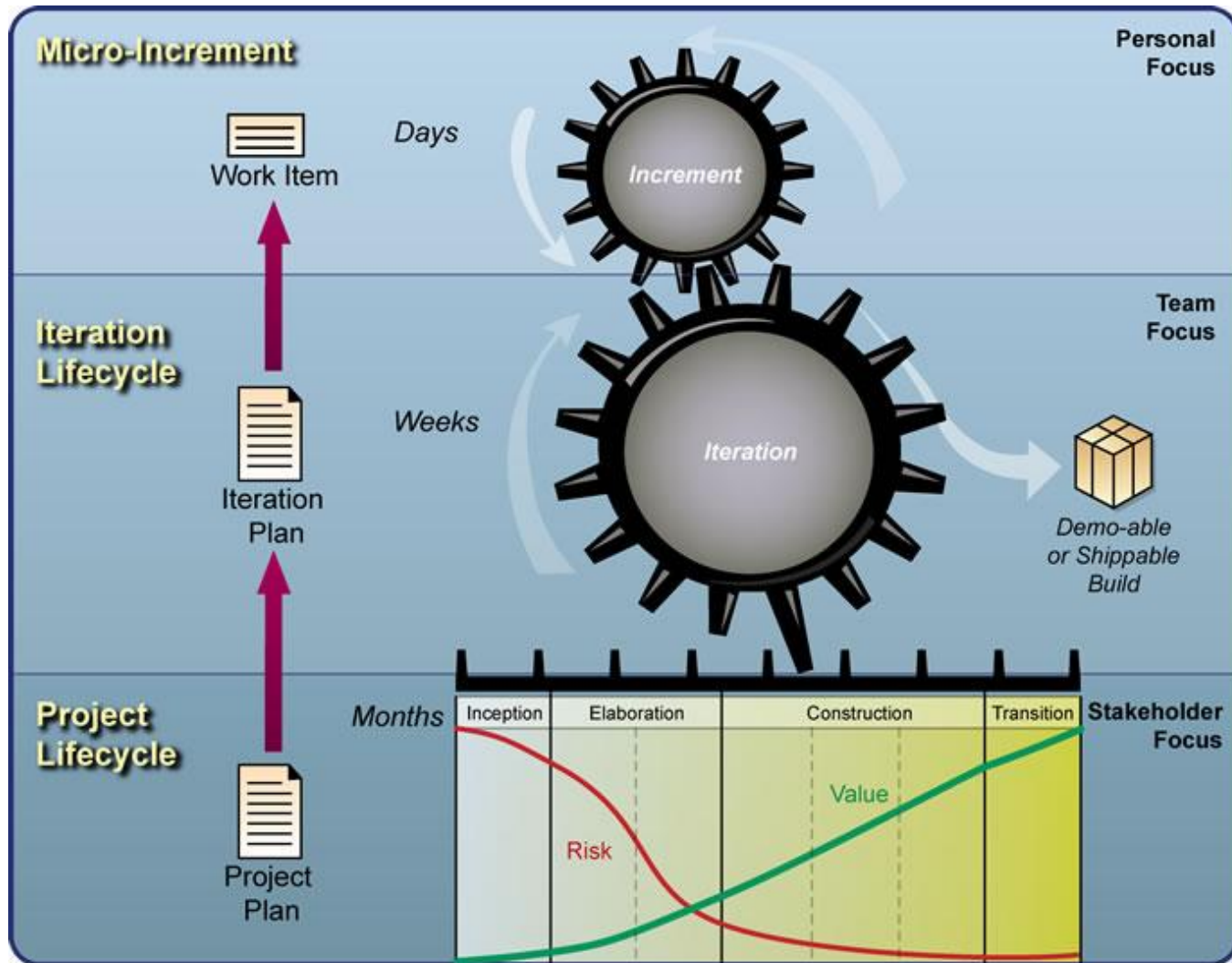


Processus : Delivery process (3)

- Décrit la vie complète du projet, mais ne remplace pas la réalité
- Composé de Patterns à itérer autant de fois que nécessaire pour chacune des 4 phases
- Le nombre d'itération dépend fort du type de projet



Micro-increment



Micro-increment

- *Résultat d'un travail pouvant aller de quelques heures à quelques jours d'une personne ou de plusieurs.*
- **BUT :**
 - Séparer le travail en plus petites unités de sorte que chacune contribue à la valeur ajoutée du projet.
 - Boucle très courte de feedback
- OpenUP ne fournit pas une liste complète des micro incréments possibles.

Influences

- **Scrum et XP** : modèle itératif court avec livrable en fin de chaque itération
- **RUP** : reprend le concept de phase en y ajoutant la "Self-organization" et l'itération
- **Eclipse Way** : Agile et itératif

Sources

OpenUP

<http://epf.eclipse.org/wikis/openup/>

<http://www.eclipse.org/epf/general/OpenUP.pdf>

<http://www.methodsandtools.com/PDF/mt200801.pdf>

<http://www.ibm.com/developerworks/rational/library/sep07/kroll/>

EPF

http://en.wikipedia.org/wiki/Eclipse_Process_Framework

<http://www.eclipse.org/epf/>