

**15th Workshop on Software Engineering Education and
Reverse Engineering - Bohinj, Slovenia, 24-29.8.2015**

OpenUP: Comparison with RUP, and Evaluation of Tools Supporting the Process

Vangel V. Ajanovski

**vangel.ajanovski@finki.ukim.mk
<https://staff.finki.ukim.mk/ajanovski>**

Faculty of Computer Science and Engineering
Saints Cyril and Methodius University
Skopje, Macedonia

About This Presentation

- The Rational Unified Process (RUP) is covered to a large extent within several courses in the Information Systems (IS) and Software Engineering (SE) programs
 - A short introductory lecture on OpenUP would be beneficial for students
 - Describe the process and differences with RUP
 - Investigate and evaluate tools
 - Discussion about making the choice
- Relevant courses:
 - *IS Development Processes (1st cycle, 2nd cycle)*
 - *Analysis and Logical Design of IS (1st cycle)*
 - *Physical Design and Implementation of IS (1st cycle)*

Rational Unified Process

- RUP (Rational Unified Process)
 - It is an Iterative software development process created by Rational
 - Some people read the title as “**Rationally** unified process”, but ...
- History:
 - 1981 - Rational Machines was established
 - 1985 - Rational Environment (IDE for ADA) for R1000
 - 1994 - merged with Verdex => Rational Software
 - 1995 - Rumbaugh, Jacobson and Booch started unification of their work and methods => UML
 - Kruchten assembled a process framework, based on the Objectory process from Erickson's Objectory AB
=> Rational Objectory Process
 - Later renamed to RUP - in line with the Unification in UML
 - 2003 - IBM acquired Rational Software

OpenUP

- History continued
 - IBM created a subset of RUP for agile projects and released as an open-source method:
 - Basic Unified Process (BUP)
 - 2005 – BUP initially went under Eclipse Foundation
 - 2006 – BUP was renamed to OpenUP/Basic
- Some characteristics of the “trim”
 - OpenUP preserves the essentials from RUP
 - Less ceremony and more agility
 - A simpler process, targeting small and collocated teams
 - Small projects – teams 3-6 people, 3-6 months

OpenUP Principles

- Core principles
 - Balance competing priorities to maximize stakeholder value
 - Collaborate to align interests and share understanding
 - Focus on the architecture early to minimize risks and organize development
 - Evolve to continuously obtain feedback and improve

RUP Main Page

Rational. Method Composer

Search this Site:

Getting Started

Team

- Welcome
- Getting Started
- RUP Lifecycle
- Delivery Processes
- Roles
- Disciplines
- Work Products
- Guidance
- What's New
- About

Welcome

Welcome to the IBM(R) Rational Unified Process(R). RUP(R) provides best practices and guidance for successful software development.

Main Description

Learning

- Getting Started
- Key Principles

Navigation Links

- Roles
- Work Products (by domain)
- Processes

Resources

- Overview
- RUP on developerWorks
- Training
- IBM Rational Method Composer
- The Rational Edge

Disciplines

- Business Modeling
- Requirements
- Analysis & Design
- Implementation
- Test
- Deployment
- Configuration & Change Mgmt
- Project Management
- Environment

Phases

	Inception	Elaboration	Construction	Transition				
Business Modeling	[Area chart showing activity across phases]							
Requirements	[Area chart showing activity across phases]							
Analysis & Design	[Area chart showing activity across phases]							
Implementation	[Area chart showing activity across phases]							
Test	[Area chart showing activity across phases]							
Deployment	[Area chart showing activity across phases]							
Configuration & Change Mgmt	[Area chart showing activity across phases]							
Project Management	[Area chart showing activity across phases]							
Environment	[Area chart showing activity across phases]							
	Initial	E1	E2	C1	C2	CN	T1	T2

Iterations

Click on an area of the screen for more information.

The preceding figure illustrates the overall architecture of the RUP, which has two dimensions:

OpenUP Main Page

Eclipse Process Framework Composer Glossary | Index | Feedback | About

Where am I | Tree Sets | **OpenUP** Print

Getting Started | Core Principles | Roles | Work Products | Disciplines | Lifecycle

What is OpenUP?

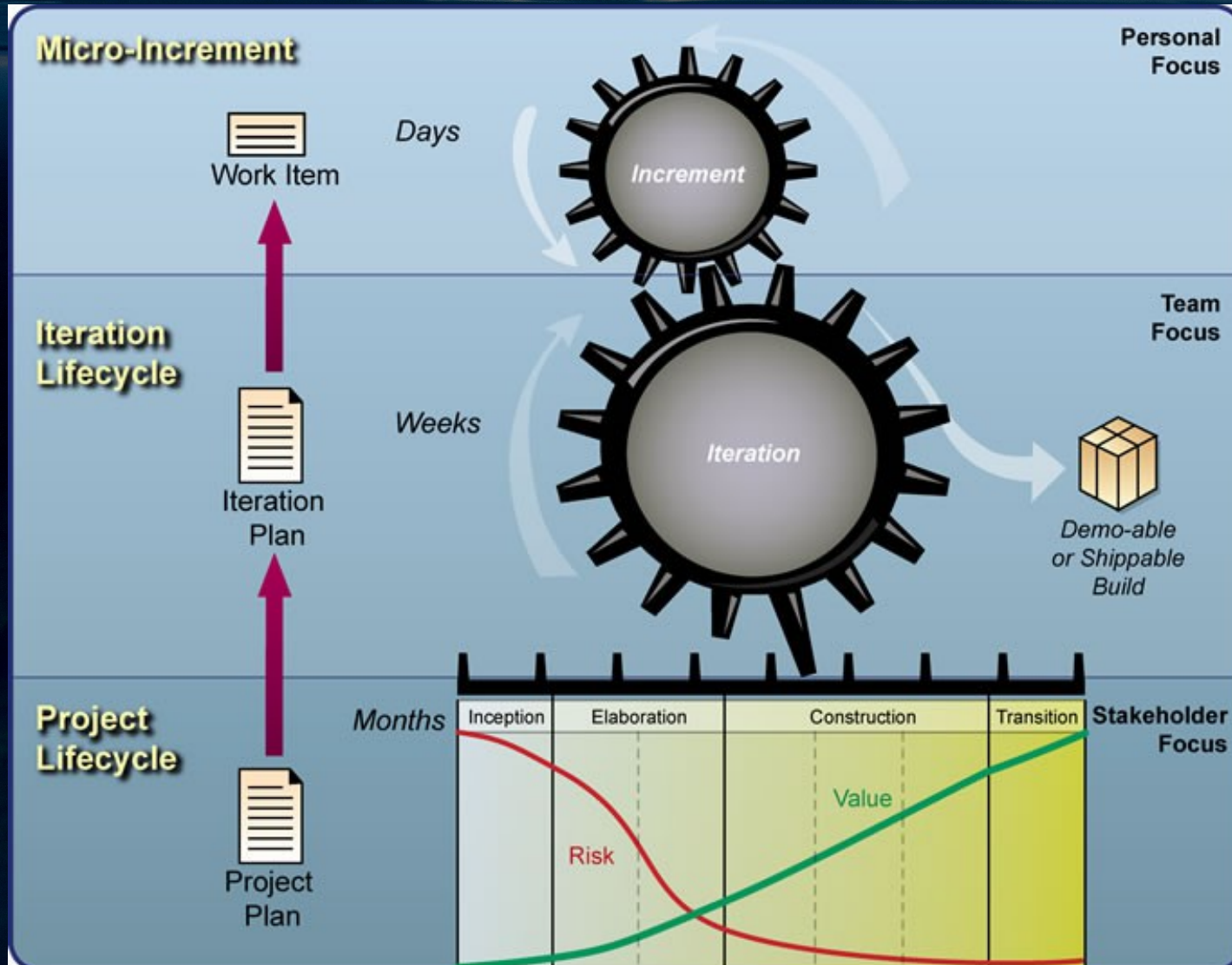
OpenUP is a lean Unified Process that applies iterative and incremental approaches within a structured lifecycle. OpenUP embraces a pragmatic, agile philosophy that focuses on the collaborative nature of software development. It is a tools-agnostic, low-ceremony process that can be extended to address a broad variety of project types.

The diagram illustrates the OpenUP lifecycle across three levels of focus:

- Project Lifecycle (Months):** Focuses on the overall project. It includes a **Project Plan** and is divided into four phases: **Inception**, **Elaboration**, **Construction**, and **Transition**. A red curve labeled **Risk** starts high in Inception and decreases through the other phases. A green curve labeled **Value** starts low in Inception and increases through the other phases.
- Iteration Lifecycle (Weeks):** Focuses on the team. It includes an **Iteration Plan** and results in a **Demo-able or Shippable Build**. It is represented by a large gear labeled **Iteration**.
- Micro-Increment (Days):** Focuses on the individual. It includes a **Work Item** and results in an **Increment**. It is represented by a smaller gear labeled **Increment**.





Arrows indicate the flow from Project Plan to Iteration Plan to Work Item, and from Iteration to Micro-Increment. The gears are interlocking, showing how the team's iteration feeds into the individual's micro-increment.

OpenUP Structured Life-cycle













Differences in Main Structure

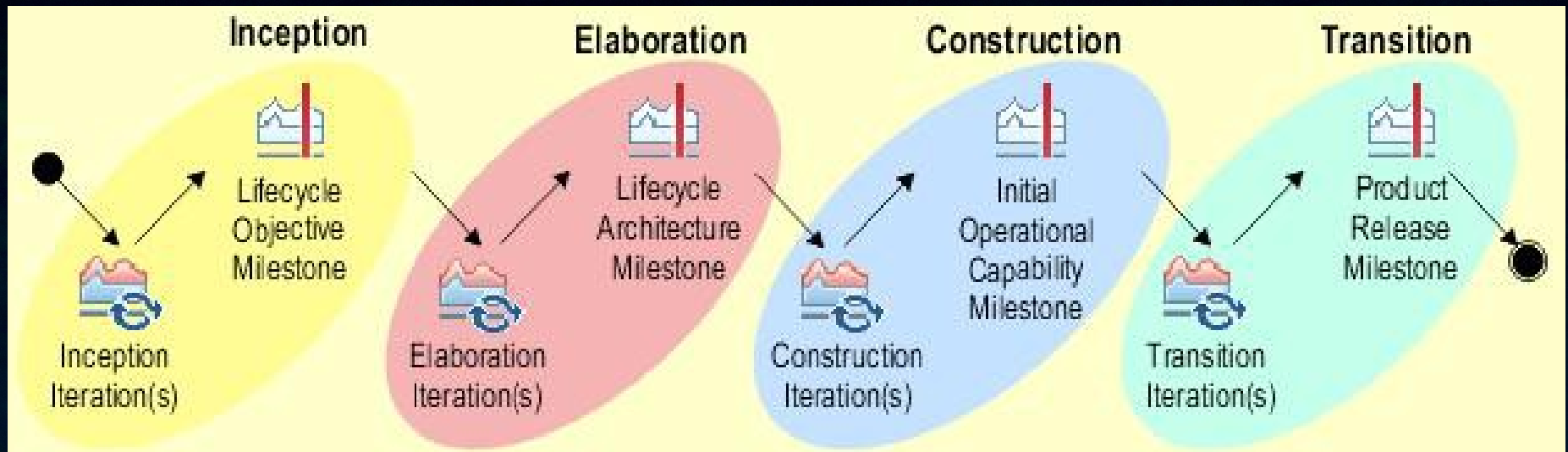
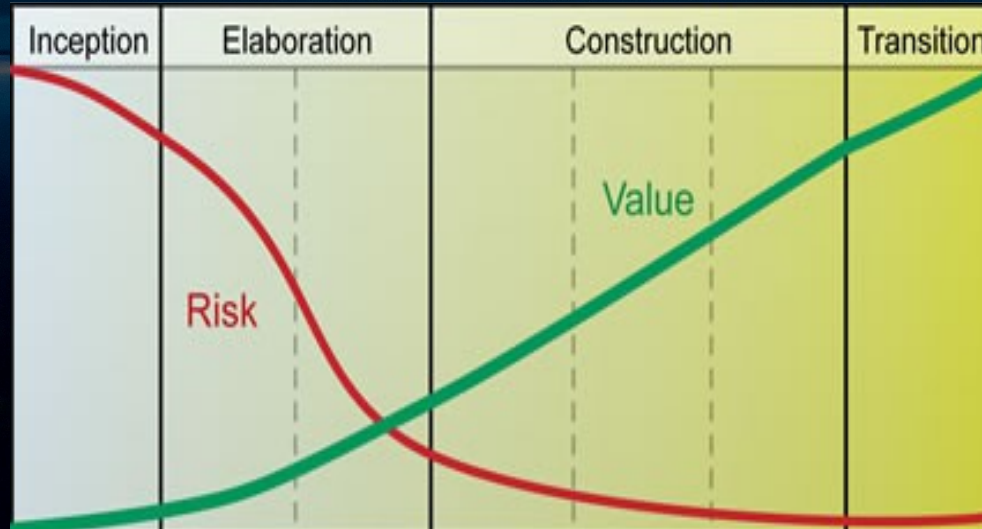
OpenUP

- + Introduction to OpenUP
- + Getting Started
- +  OpenUP Disciplines
- +  OpenUP Work Products
- +  OpenUP Roles
- +  OpenUP lifecycle
- + About
- OpenUP Copyright

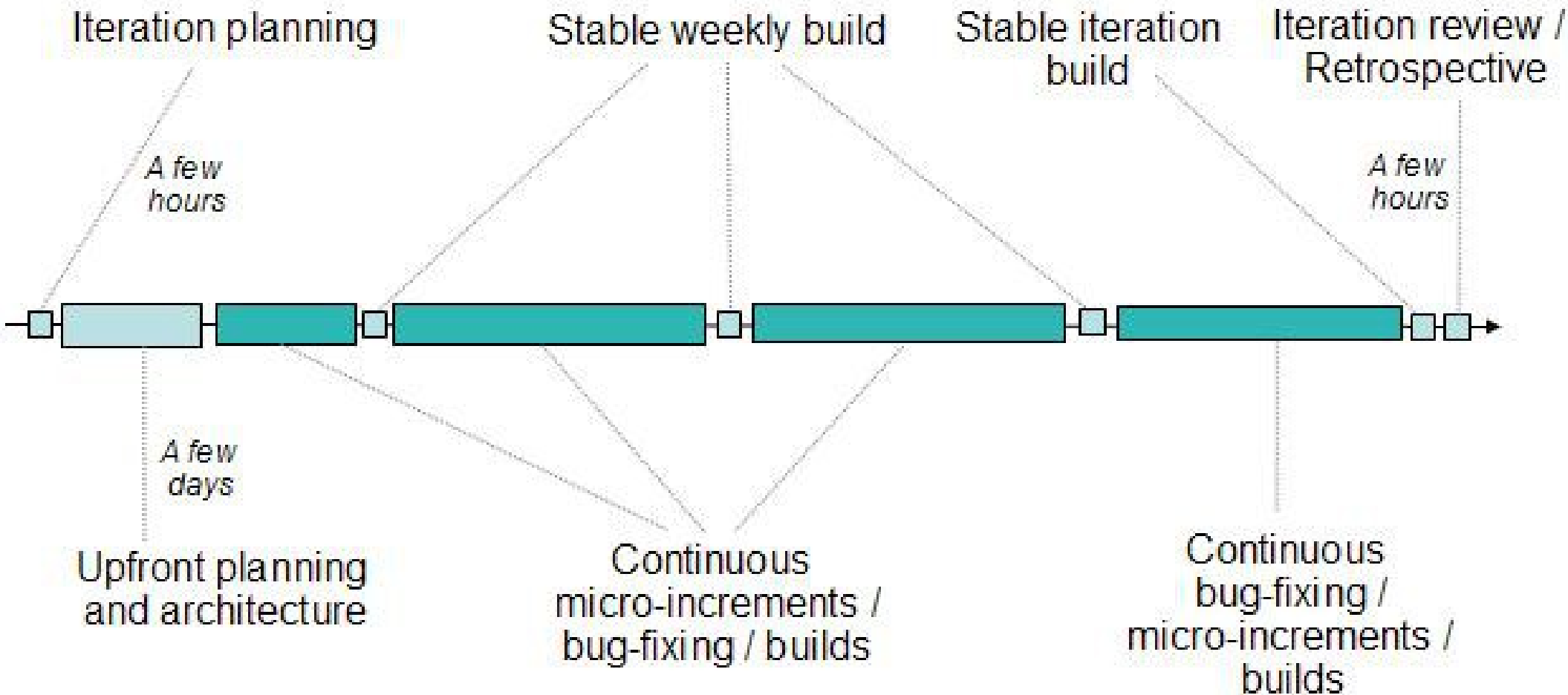
Team

-  Welcome
- +  Getting Started
- +  RUP Lifecycle
- +  Delivery Processes
- +  Roles
- +  Disciplines
- +  Work Products
- +  Guidance
- +  What's New
- +  About

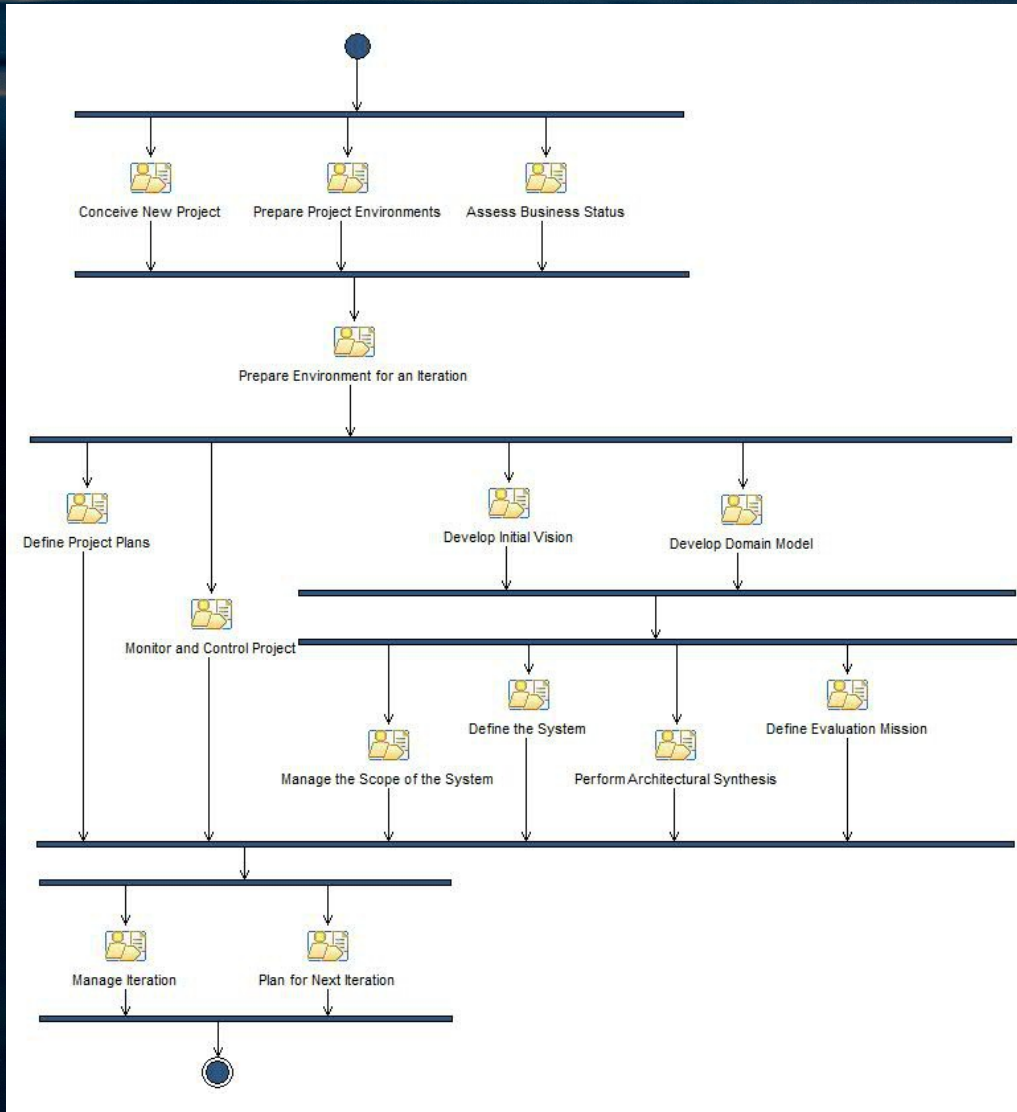
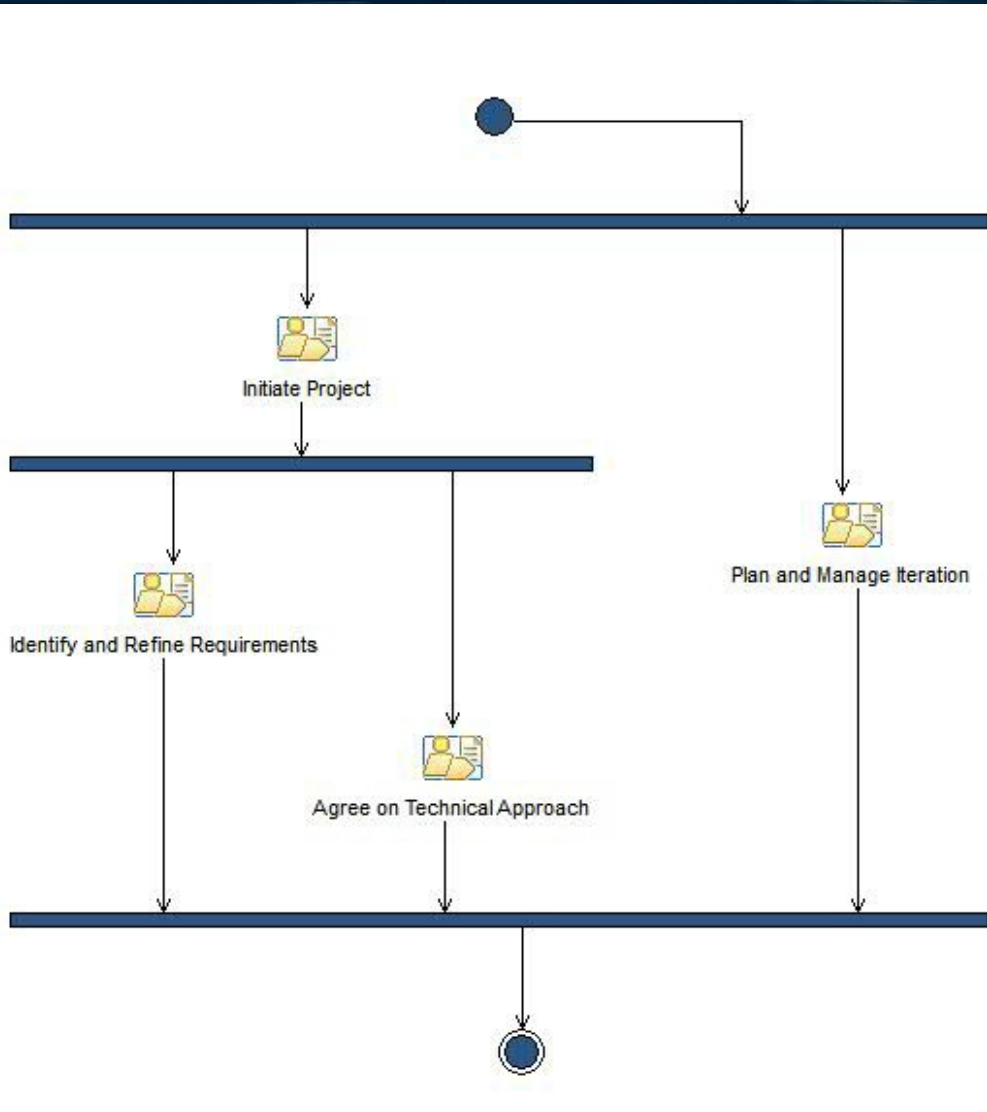
Project Life-cycle



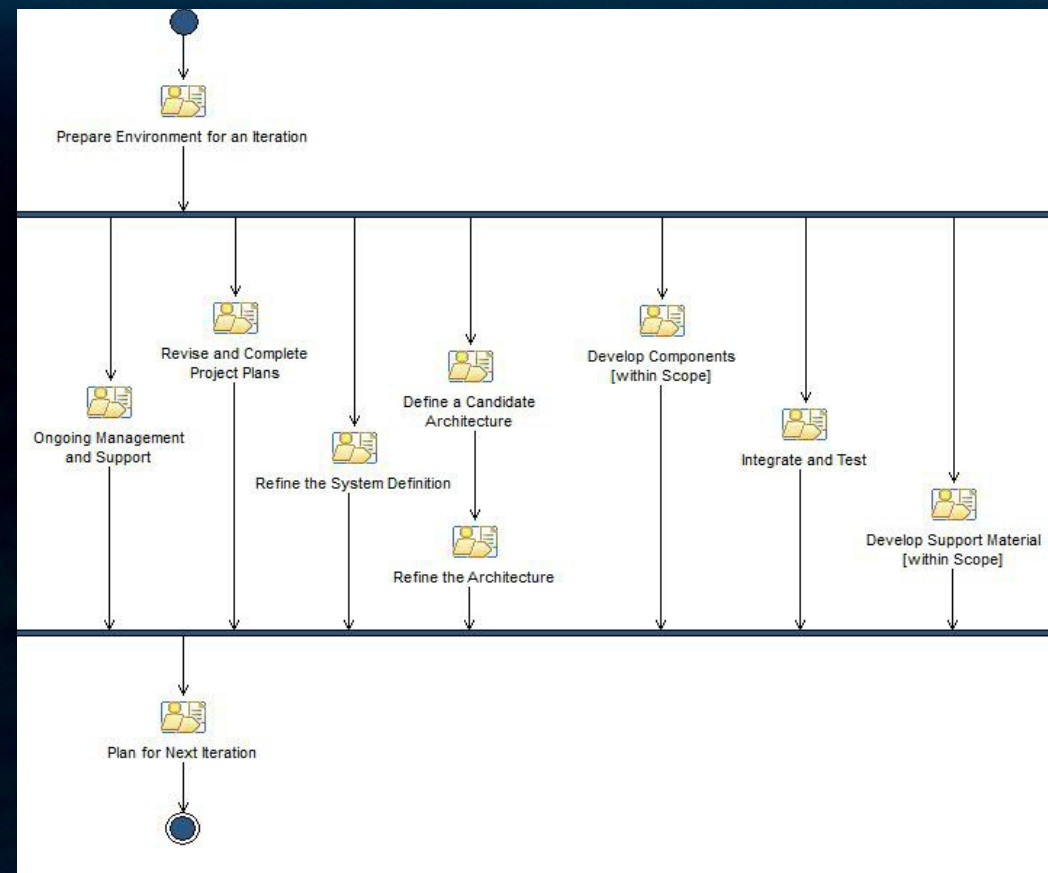
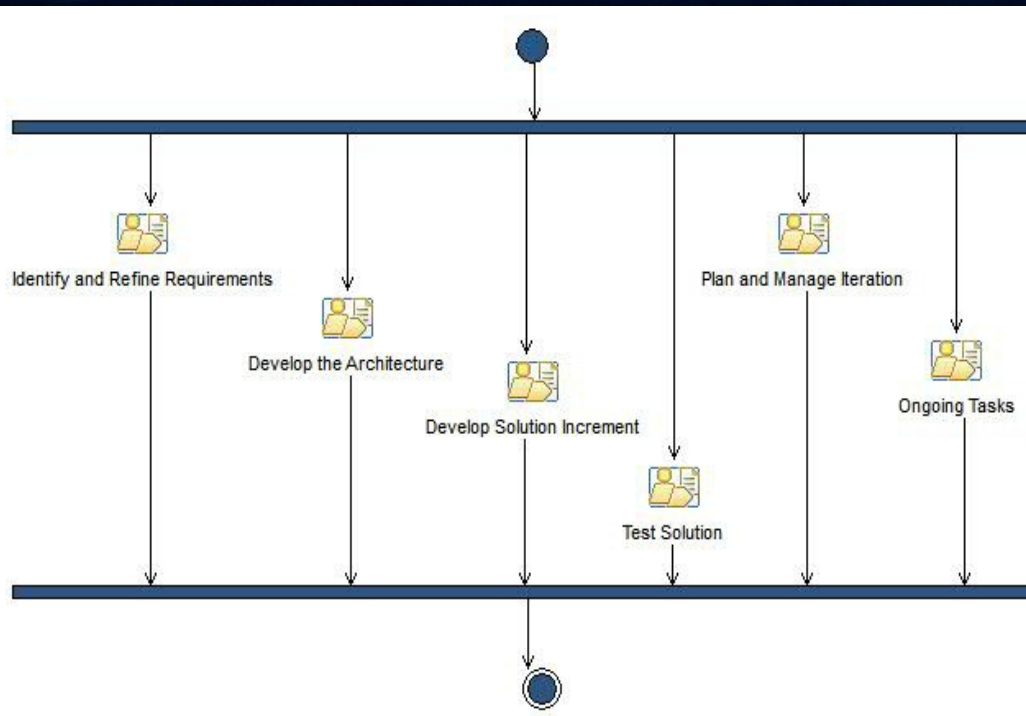
Iteration Life-cycle



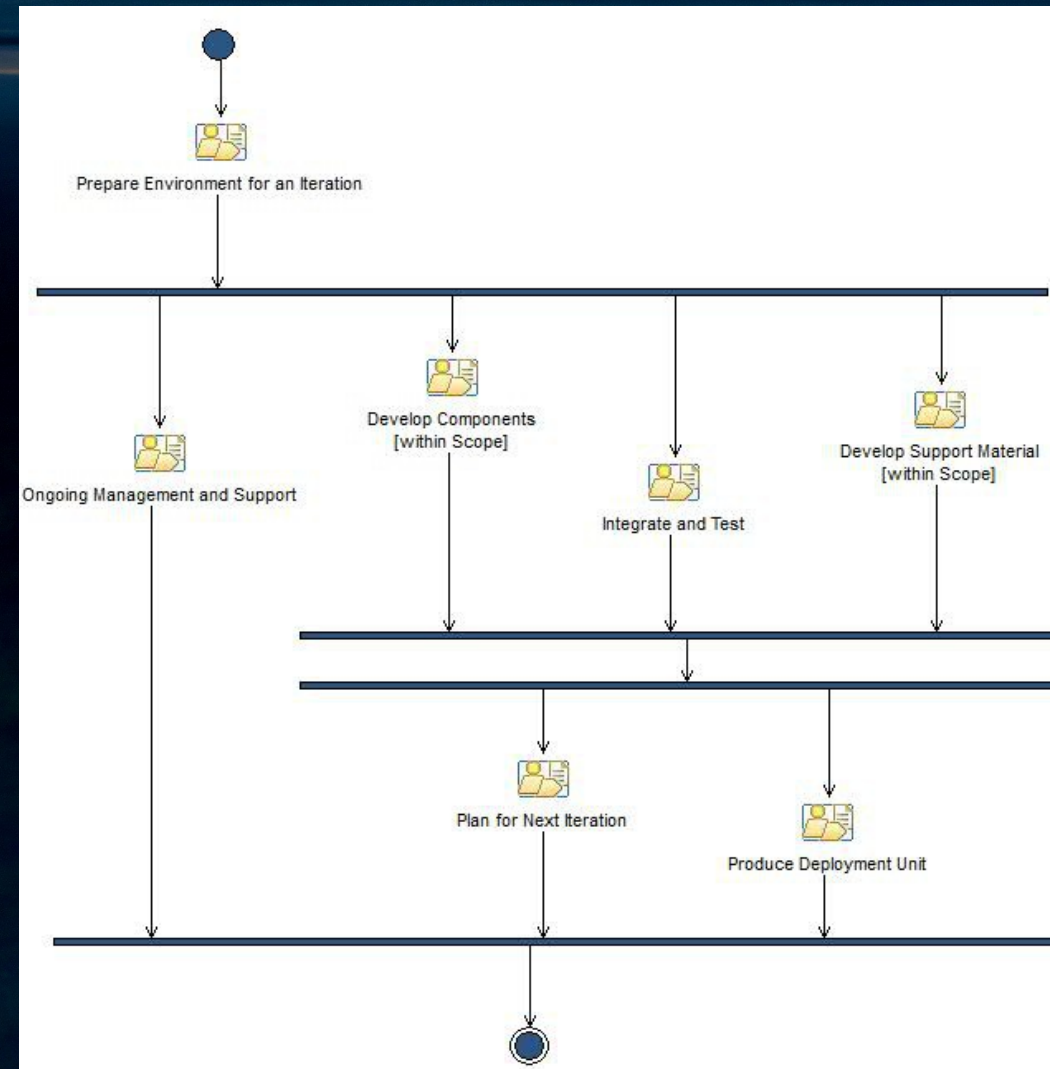
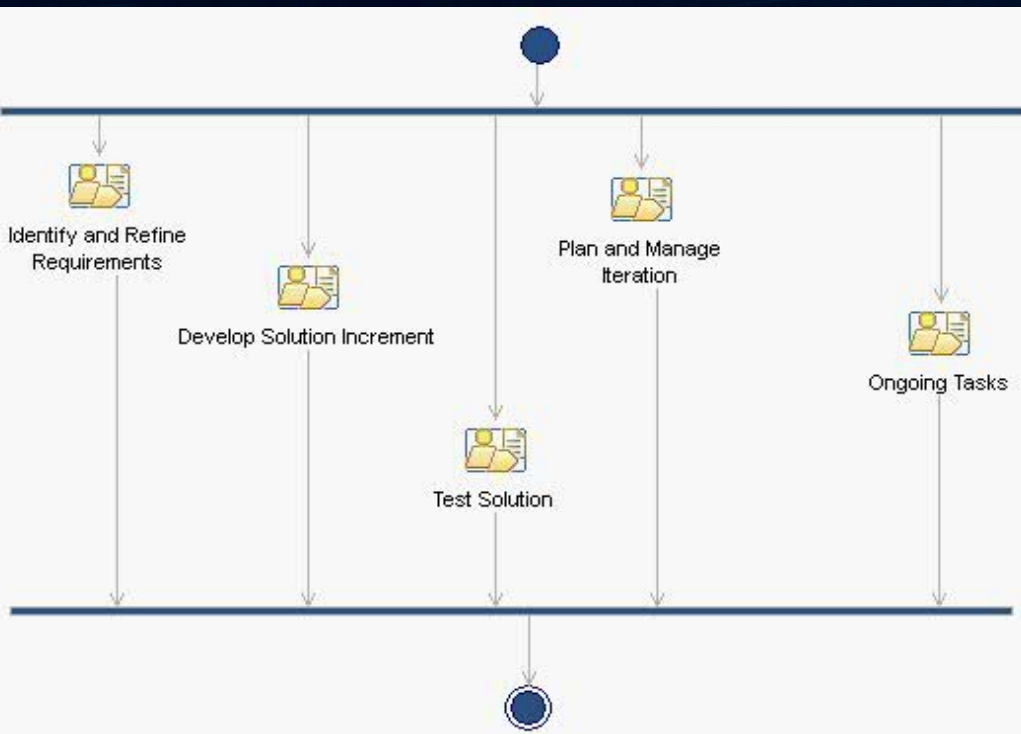
Inception Phase



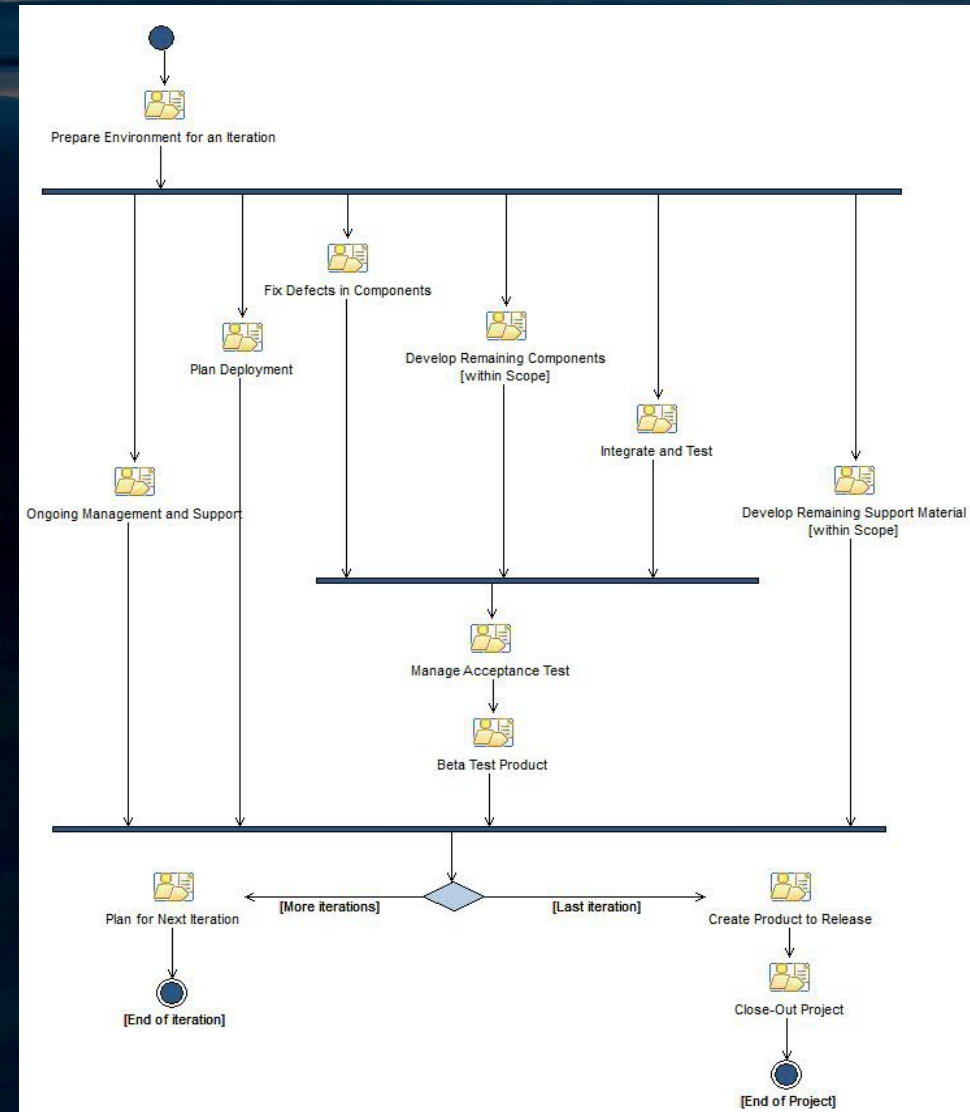
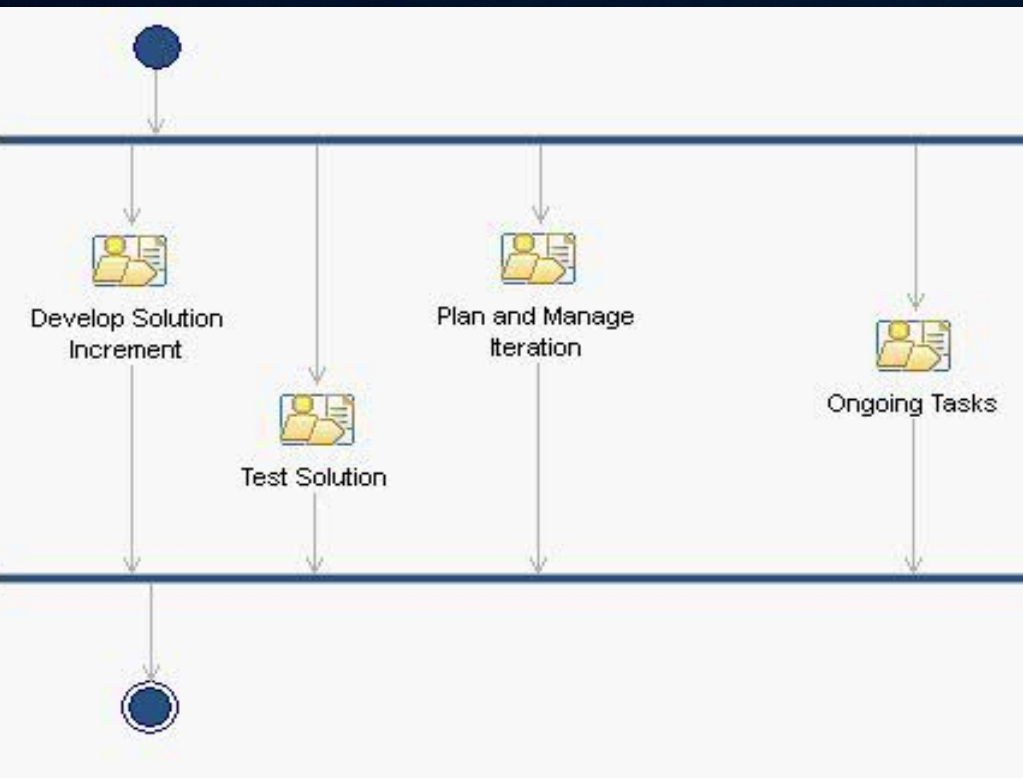
Elaboration Phase



Construction Phase

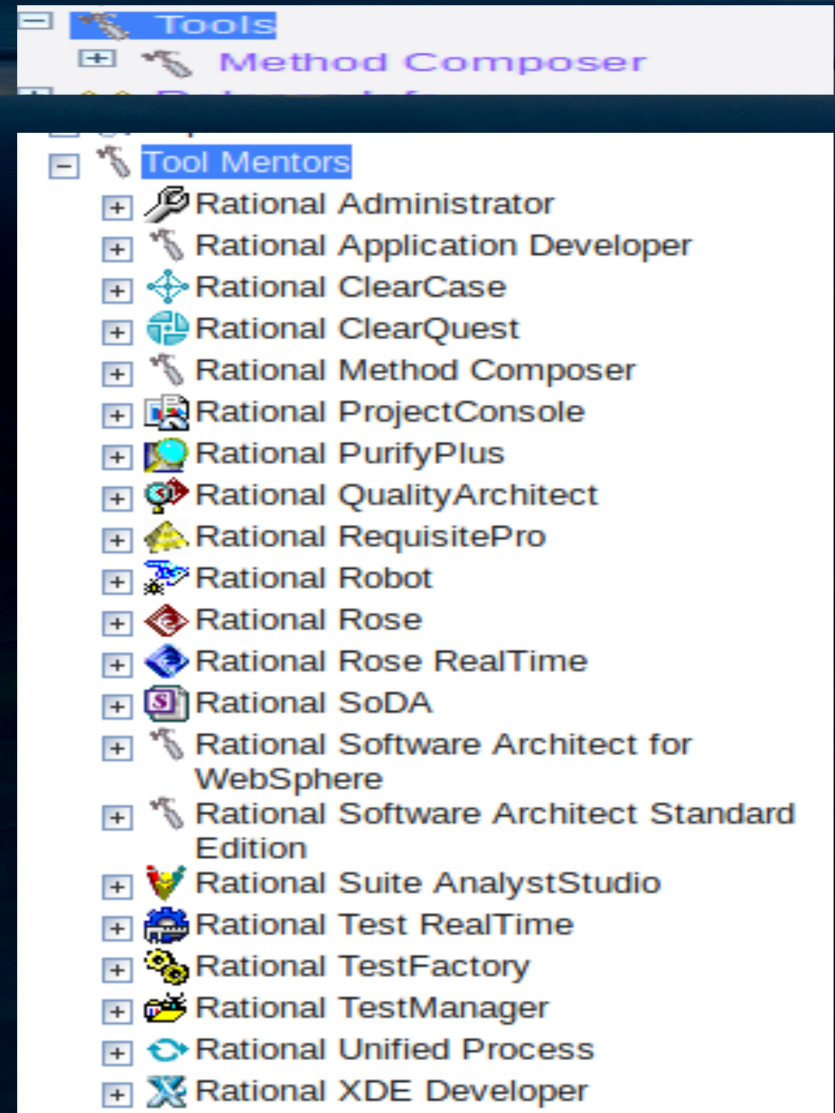


Transition Phase



Tools

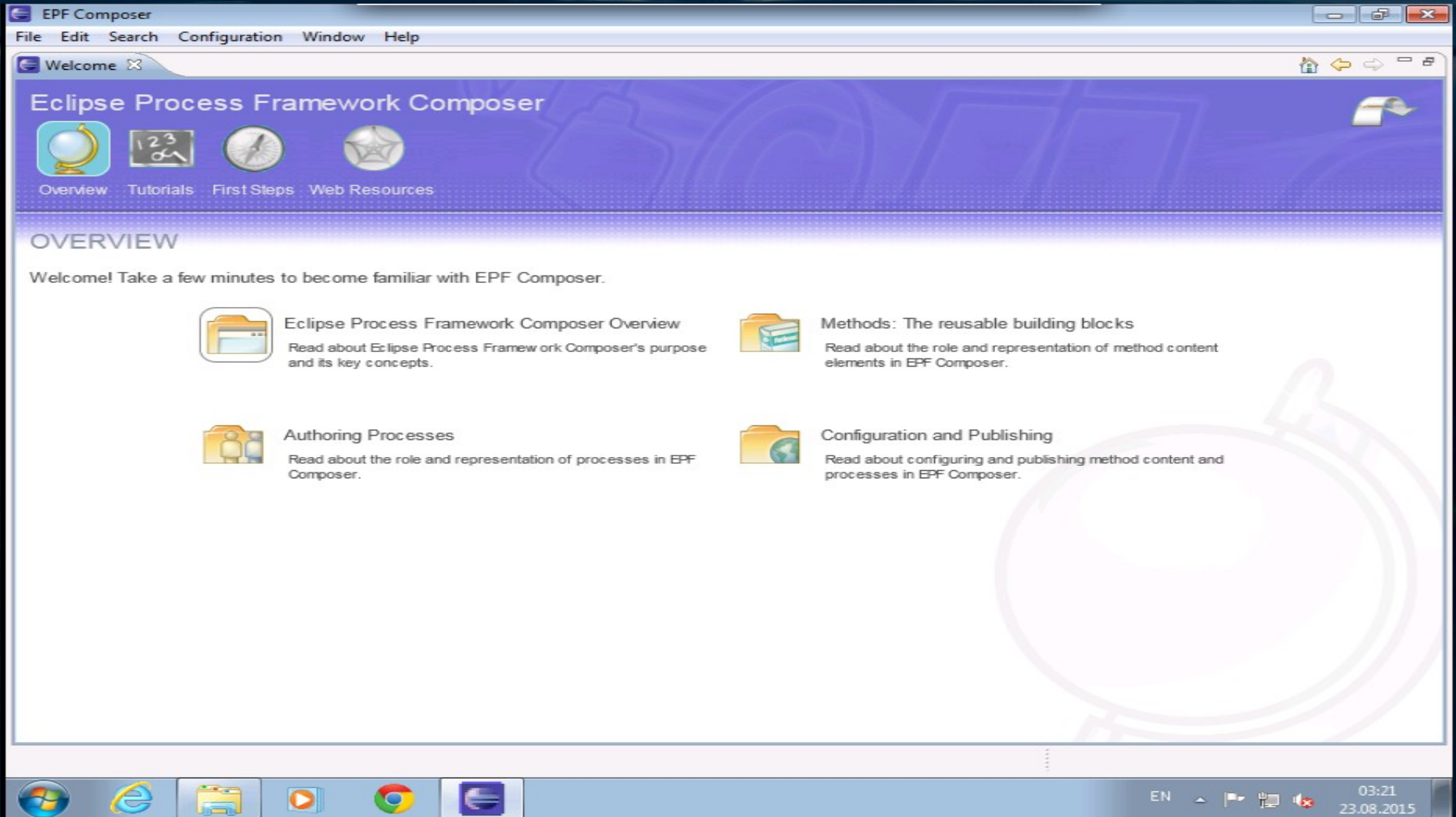
- Rational Method Composer
- Eclipse EPF Composer
 - Windows and Linux
 - Latest rel.: 1.5.1.7 from July 22, 2015
 - <http://www.eclipse.org/epf>
- Installation problems
 - Only in 32-bit format
 - Options:
 - Install a 32-bit sub-system (real or vm)
 - Compile from source



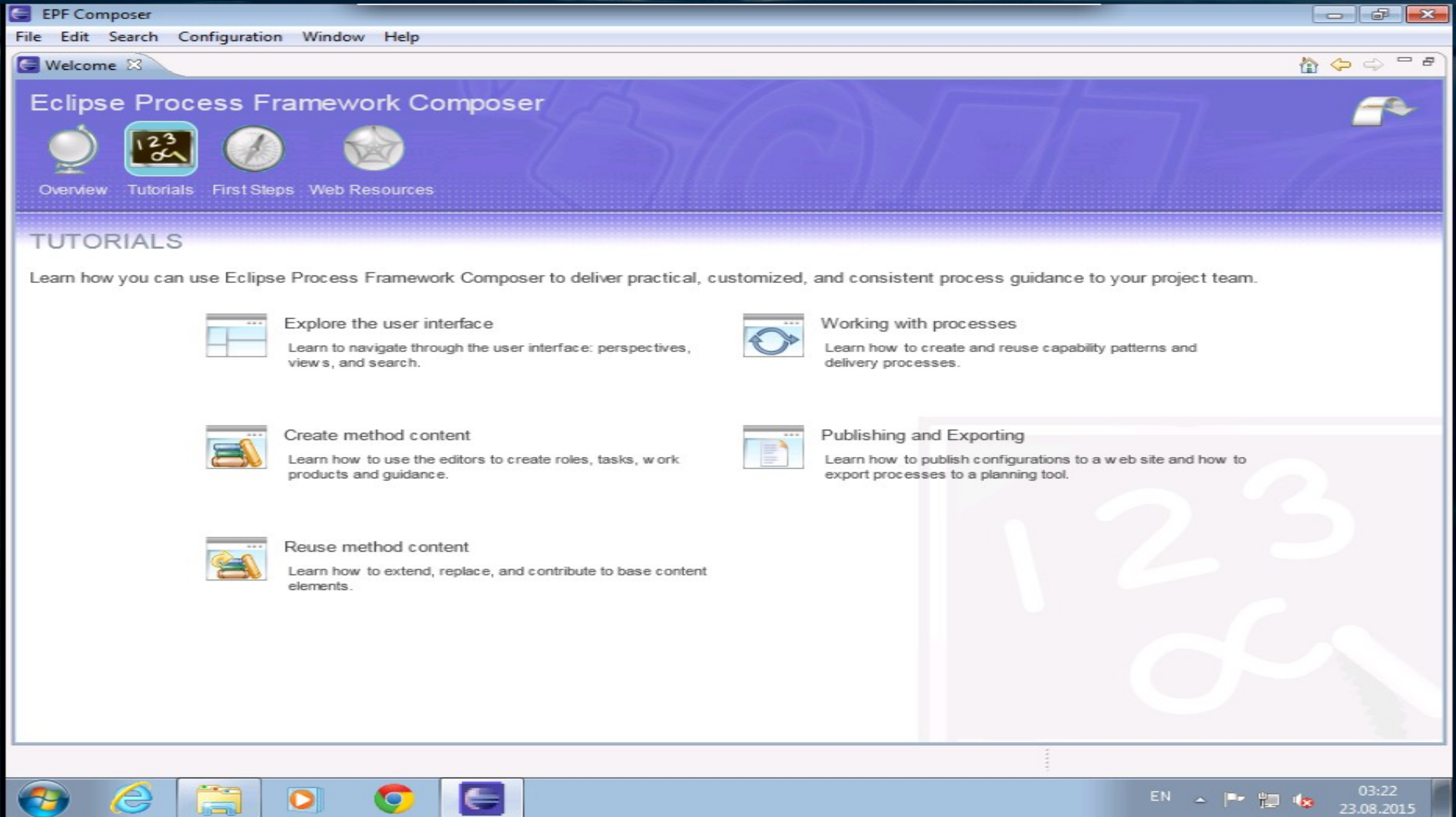
Eclipse Process Framework Composer



EPF Composer Overview



EPF Composer Tutorials



The screenshot shows the EPF Composer application window. The title bar reads "EPF Composer" and the menu bar includes "File", "Edit", "Search", "Configuration", "Window", and "Help". The main content area is titled "Eclipse Process Framework Composer" and features a "TUTORIALS" section. Below this section, there is a list of five tutorial topics, each with an icon and a brief description:

- Explore the user interface**: Learn to navigate through the user interface: perspectives, views, and search.
- Working with processes**: Learn how to create and reuse capability patterns and delivery processes.
- Create method content**: Learn how to use the editors to create roles, tasks, work products and guidance.
- Publishing and Exporting**: Learn how to publish configurations to a web site and how to export processes to a planning tool.
- Reuse method content**: Learn how to extend, replace, and contribute to base content elements.

The Windows taskbar at the bottom shows the Start button, Internet Explorer, File Explorer, a media player, Google Chrome, and the EPF Composer icon. The system tray on the right displays the language "EN", system icons, and the date and time "03:22 23.08.2015".

EPF Composer - A Multi-process tool

Libraries Download

EPF Practices

This library contains all the practices included in EPF. It is intended to be used by a process engineer to assemble, customize, and publish practices.

Download library: **1.5.1.5**

Build of EPF Composer used: **1.5.1.5**

Release Notes

OpenUP

The following libraries contain specific extensions and versions of OpenUP.

Download library: **OpenUP/DSDM 1.0, OpenUP Portuguese 1.0**

Build of EPF Composer used: **1.2.0**

Scrum

The following libraries contain specific extensions and versions of Scrum.

Download library: **Scrum 1.5, Scrum Portuguese 1.5**

Build of EPF Composer used: **1.5**

XP

The following libraries contain specific extensions and versions of XP.

Download **XP 0.1** library

Build of EPF Composer used: **1.2.0M4-N.20070724-2201**

Download **XP Portuguese 0.1** library

Build of EPF Composer used: **1.5**

EPF Composer - Using a Method Library

The screenshot displays the Eclipse Process Framework Composer interface. The title bar indicates the application is running on a Windows system, with the file path: C:\Users\ajan.LABSTUDENT1.004\Downloads\OpenUP_library_1.0_20070801\OpenUP. The main window is titled 'architecture' and shows the configuration for the 'Architecture' discipline.

Library View: A tree view on the left shows the project structure. The 'openup' folder is expanded to show 'Method Content', which includes 'Content Packages', 'Standard Categories', and 'Disciplines'. Under 'Disciplines', the 'openup_disciplines' folder is expanded to show 'architecture', 'config_and_change_mana', 'development', 'project_management', 'requirements', and 'test'.

Architecture Discipline Configuration:

- Discipline: Architecture**
 - This discipline explains how to create an architecture from architecturally significant requirements. The architecture is built in the Development discipline.
 - Buttons: Expand All Sections, Collapse All Sections
- Relationships**

Reference Workflows	<ul style="list-style-type: none">Agree on the Technical ApproachDevelop the Architecture
Tasks	<ul style="list-style-type: none">Outline the ArchitectureRefine the Architecture

Back to top
- Main Description**

The purpose of this discipline is to evolve a robust architecture for the system.

The Architecture discipline is related to other disciplines, as follows:

 - The **Requirements** discipline provides the architecturally significant requirements.
 - The **Development** discipline designs and implements the architecture.
 - The **Test** discipline verifies the stability and correctness of the architecture.
 - The **Project Management** discipline plans the project, and each iteration.

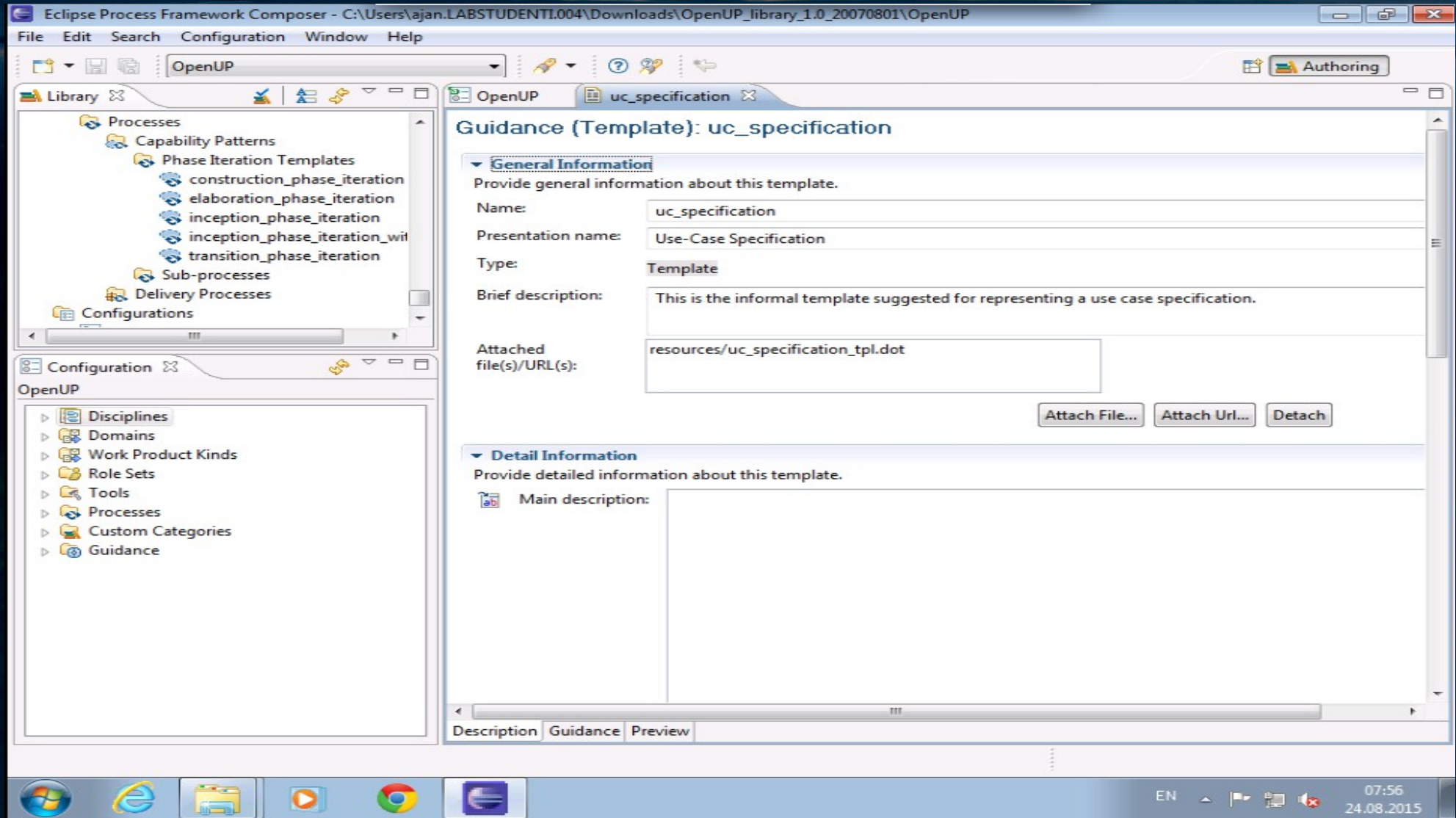
Back to top
- Key Considerations**

Creating and applying architectural mechanisms is essential for creating a robust architecture. See more

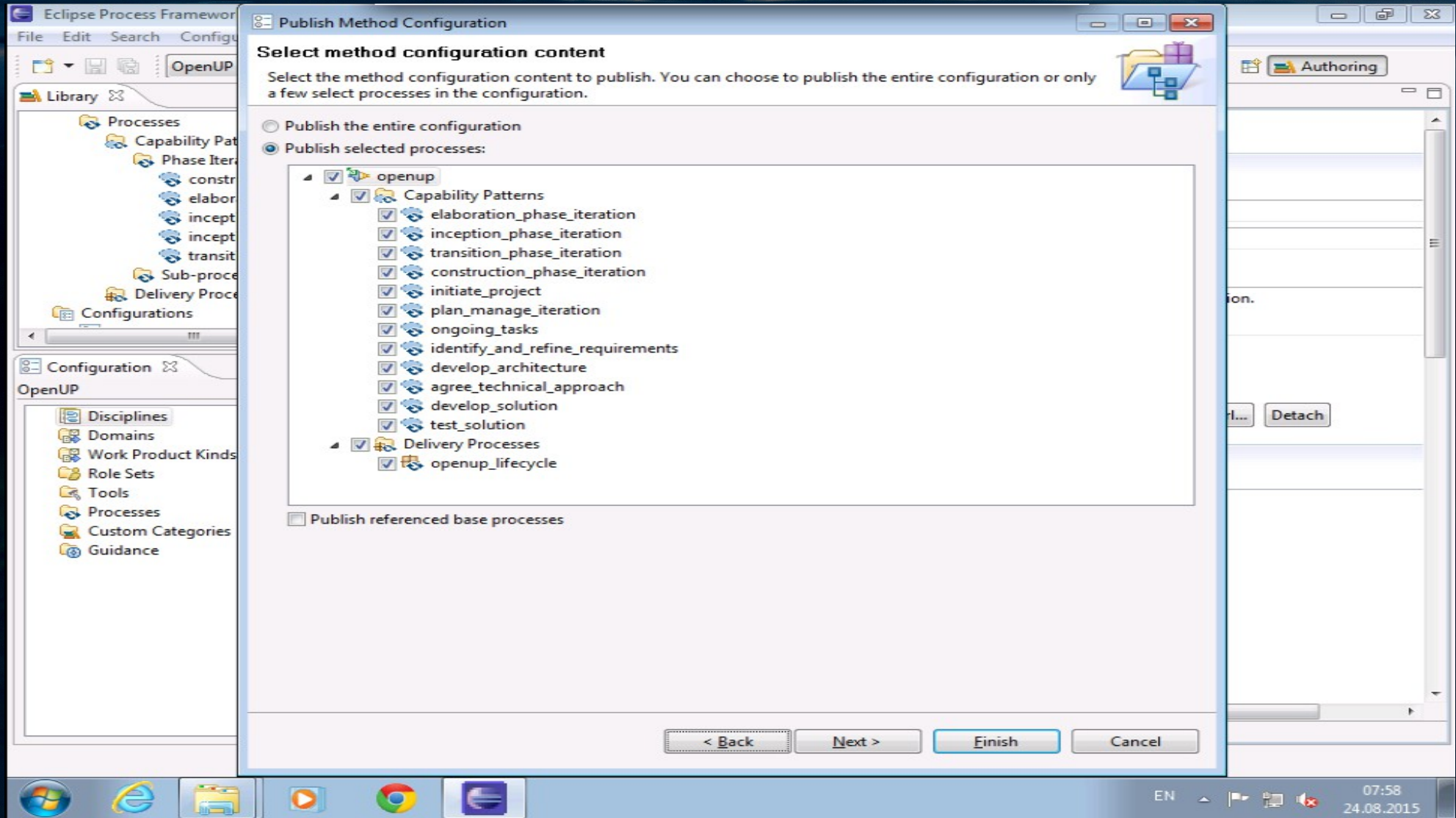
At the bottom of the configuration window, there are tabs for 'Description', 'Tasks', 'Reference Workflow', 'Guidance', and 'Preview'. The 'Description' tab is currently selected.

The Windows taskbar at the bottom shows the system clock as 03:30 on 23.08.2015.

EPF Composer - Editing Process Elements



EPF Composer - Publishing the Process



Questions and Comments