

Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte
Campus Natal-Central

Diretoria Acadêmica de Gestão e Tecnologia da Informação
Curso de Tecnologia em Análise e Desenvolvimento de Sistemas



Eclipse Process Framework

Disciplina: Processo de Desenvolvimento de Software

Prof. Fellipe Aleixo (fellipe.aleixo@ifrn.edu.br)

Visão Geral do Projeto EPF

- É um projeto open source da Fundação Eclipse
- Iniciou em janeiro de 2006
- Tem como objetivo prover:
 - Um framework e ferramentas para edição, configuração e publicação de processos de software
 - Processos modelo
- EPF não é:
 - Aplicável apenas ao desenvolvimento Java no Eclipse
 - Voltado a criação do “processo perfeito”

Objetivos do EPF

- Prover um framework customizável para engenharia de processos de software
 - Prover um framework extensível e um conjunto de **ferramentas** “modelo” para engenharia de processos
 - Prover um **conteúdo de processo** par um conjunto de processos de desenvolvimento e gerenciamento de software
 - suportando iterativo, ágil e incremental
 - que possa ser extensível

Objetivos do EPF

Ferramenta Modelo:
EPF Composer

The screenshot shows the Eclipse Process Framework Composer (EPF Composer) interface. The main window displays the configuration for a task named "define_vision". The "General Information" section includes fields for Name, Presentation name, and Brief description. The "Detail Information" section includes fields for Purpose and Main description. Below these sections is a table listing the task's structure, including its presentation name, index, predecessors, model info, type, planned status, and repeat status.

Presentation Name	Index	Predecessors	Model Info	Type	Planned	Repeat	Multi
Inception Phase Iteration	0			Capability Pat...	true	false	false
Initiate Project	1		extends 'initiate_proje...	Activity	true	false	false
Define Vision	2			Task_Descip...	false	false	false
Plan the Project	3			Task_Descip...	false	false	false
Manage Iteration	4		extends 'manage_iter...	Activity	true	false	false
Plan Iteration	5			Task_Descip...	false	false	false
Manage Iteration	6			Task_Descip...	false	false	false
Assess Results	7			Task_Descip...	false	false	false
Manage Requirements	8	1	extends 'manage_req...	Activity	true	false	false
Find and Outline Requirements	9			Task_Descip...	false	false	false
Detail Requirements	10			Task_Descip...	false	false	false
Create Test Cases	11			Task_Descip...	false	false	false
Determine Architectural Feasibility	12	1	extends 'determine_ar...	Activity	true	false	false
Analyze Architectural Requirements	13			Task_Descip...	false	false	false
Demonstrate the Architecture	14			Task_Descip...	false	false	false

Exemplo de Processo Modelo:
OpenUP

The screenshot shows the OpenUP website in Mozilla Firefox. The page is titled "What is OpenUP?" and provides an overview of the OpenUP process. The diagram illustrates the OpenUP layers: micro-increments, iteration lifecycle, and project lifecycle. The diagram shows a flow from a Project Plan to a Work Item, then to an Iteration Plan, and finally to a Demo-able or Shippable Build. The project lifecycle is divided into four phases: Inception, Elaboration, Construction, and Transition. A red line represents Risk, which decreases over time, and a green line represents Value, which increases over time.

OpenUP layers: micro-increments, iteration lifecycle and project lifecycle

Personal effort on an OpenUP project is organized in **micro-increments**. These represent short units of work that produce a steady, measurable pace of project progress (typically measured in hours or a few days). The process applies intensive collaboration as the system is incrementally deployed by a committed, self-organized team. These micro-increments provide an automated feedback loop that drives adaptive decisions.

A Abordagem EPF

Standardize representation and manage libraries of reusable

Method Content

Content on agile development
Content on managing iterative development
Guidance on serialized java beans



JUnit user guidance
Content on J2EE
Configuration mgmt guidelines

Develop and manage Processes for performing projects

Process for Custom Application Development with J2EE
Process for Embedded System Development
Process for SOA Governance



Process assets patterns
Standard or reference processes
Enactable project plan templates
Corporate guidelines on compliance

Configure a cohesive process framework customized for my project needs

Create project plan templates for Enactment of process in the context of my project

Quem se Beneficia com o EPF?

1. Projetos individuais → usar os processos disponibilizados pelo EPF
2. Empresas de desenvolvimento → (i) adoção das “melhores práticas”, (ii) incorporar as lições aprendidas, (iii) padronizar a linguagem dentro da organização e (iv) endereçar necessidades específicas
3. Academia → atuando nos papéis de “consumidor” e de responsável por trazer o estado-da-arte no desenvolvimento de software para a “indústria”
4. Empresas de tecnologia → desenvolvimento de ferramentas adequadas aos processo EPF

Informações sobre o meta-modelo utilizado para armazenar às informações de um processo de software

REPRESENTAÇÃO DOS PROCESSOS NO EPF

Representação de Processos EPF

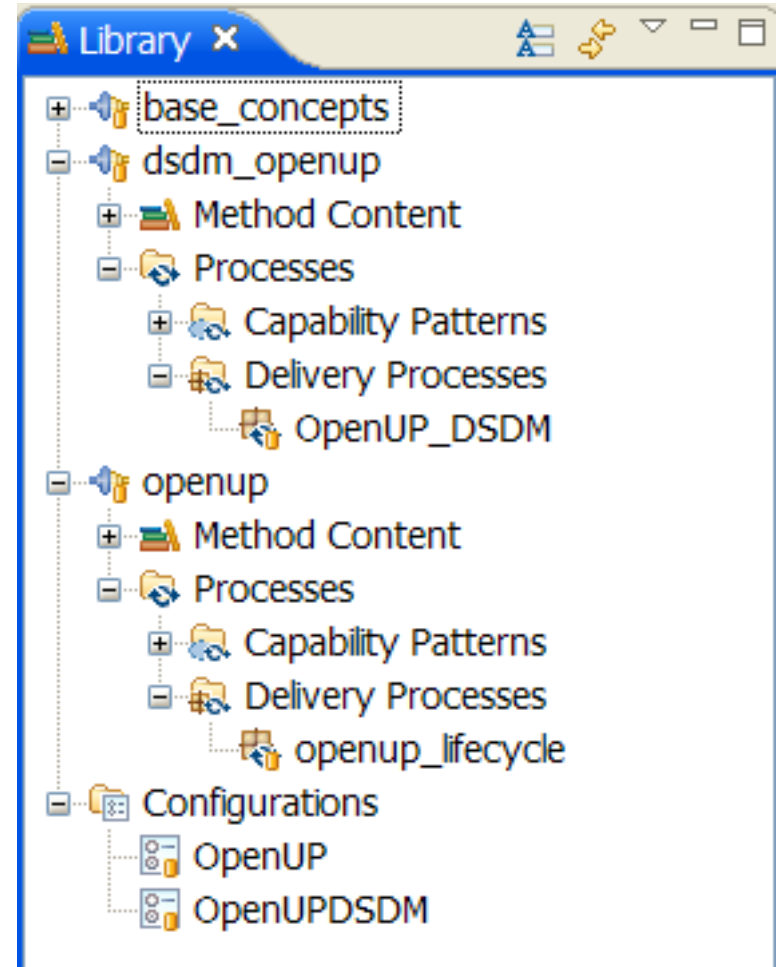
- O EPF armazena as informações de processo em um modelo segundo o meta-modelo UMA
 - UMA = *Unified Method Architecture*
 - O UMA foi desenvolvido com base no SPEM 1.0 e influenciou a criação do SPEM 2.0 (OMG)
 - SPEM = *Software Process Engineering Meta-model*
- Qualquer processo pode ser representado através destes meta-modelos

Conceitos Básicos: *Method Library*

- *Method Library*
 - Armazena um conjunto de elementos de método
- *Method Plug-in*
 - Representa um conjunto de “pacotes” de métodos e processos
- *Method Configuration*
 - Um subconjunto a lógico da *Method Library*
- *Delivery Process*
 - Uma abordagem completa e integrada para realização de tipo específico de projeto

Conceitos Básicos: *Method Library*

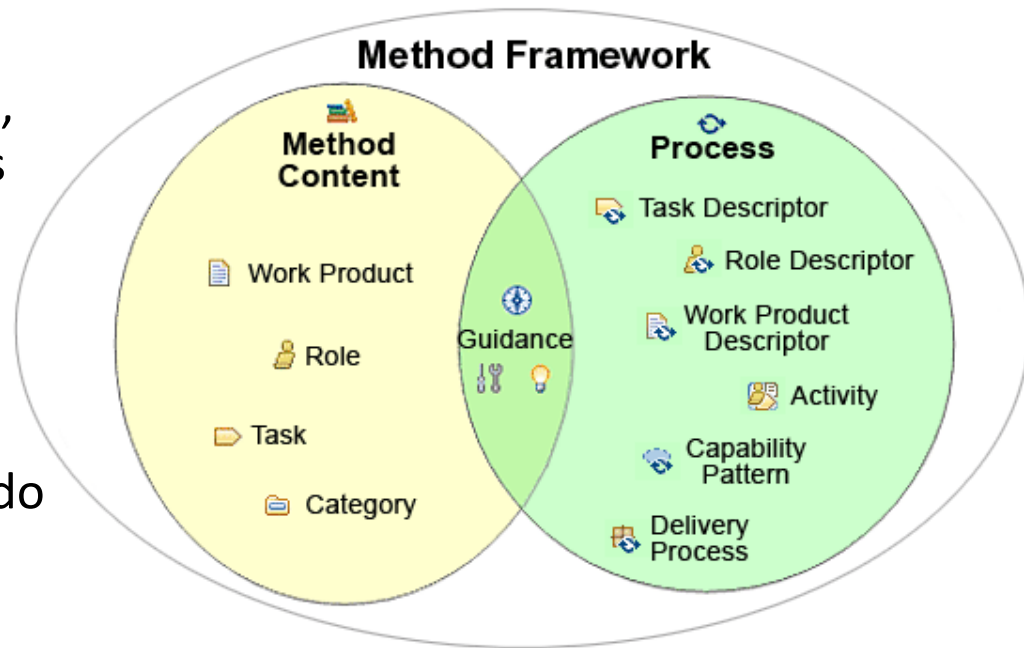
- Bibliotecas de métodos contém:
 - *Method plug-ins*
 - Configurações
- A biblioteca do OpenUP possui:
 - Três *method plug-ins*
 - base_concepts
 - dsdm_openup
 - openup
 - Dois *delivery processes*
 - Openup_DSDM
 - openup_lifecycle
 - Duas configurações
 - OpenUP
 - OpenUPDSDM



Conceitos Básicos:

Method Content e Process

- *Method Content* (quem, o que, por que, como)
 - Definição de papéis, tarefas, produtos de trabalho e seus relacionamentos
- *Process* (quando)
 - Sequência das Fases, Iterações, Atividades e *Milestones* → ciclo de vida do desenvolvimento
 - Define quando realizar as tarefas → diagrama de atividades e/ou *Work Breakdown Structures*

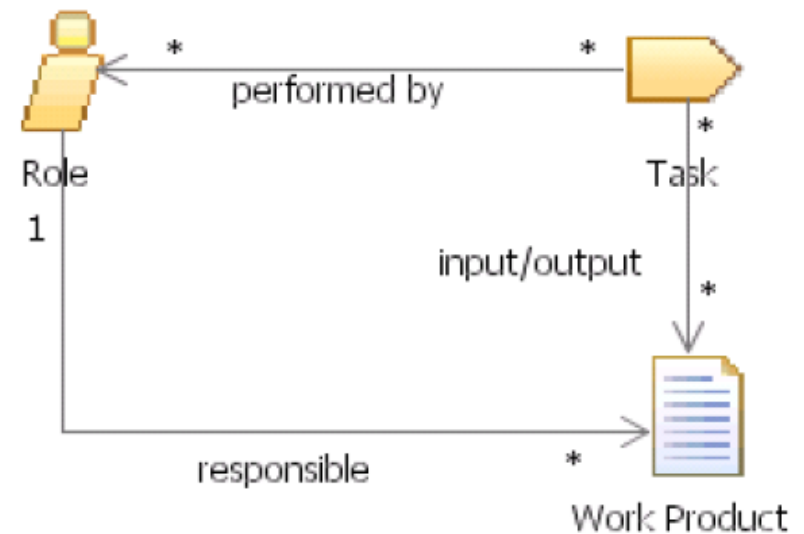


Method Content




CONTEUDO DE MÉTODO

(1) Papel

- Definem um conjunto de habilidades, competências e responsabilidades relacionadas
- Não representam indivíduos os quais podem desempenhar vários papéis
- Papéis realizam tarefas
- Papéis são responsáveis por produtos de trabalho



(2) Produto de Trabalho

- Na maioria dos casos representam elementos tangíveis usados, modificados e produzidos por uma tarefa
- Papéis usam produtos de trabalho para realizar tarefas e produzir/atualizar outros produtos de trabalho
- São responsáveis de um papel
- Podem ser de três tipos:
 -  Artefatos – item de configuração gerenciado
 -  Entregável – requisitado pelo *cliente/stakeholder*
 -  Resultado – efeito “intangível” de uma tarefa, como por exemplo um servidor ou ferramenta instalada
















(3) Tarefa

- Define uma unidade de trabalho a ser atribuída (geralmente com a duração de algumas horas)
- Desempenhada por papéis (primário e opcionais)
- Possuem um propósito claro, e provêem uma descrição de passo-a-passo do trabalho necessário para completar a tarefa (atingir meta)
- Modifica ou produz produtos de trabalho
- Não definem quando as mesmas deverão ser executadas no ciclo de vida

(4) Orientações

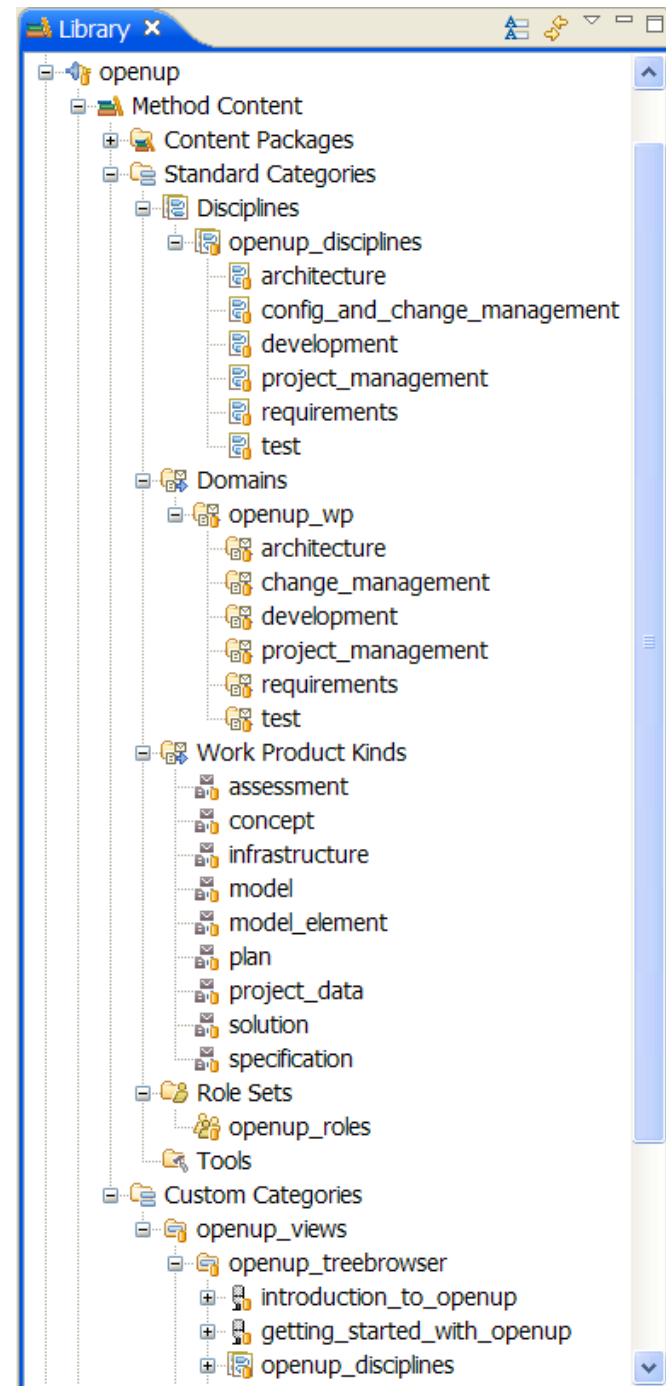
- Orientações podem ser associadas a um papel, tarefa e produtos de trabalhos
- Tipos diferentes para propósitos diferentes
- As orientações são usadas para detalhar a metodologia e oferecer informações de suporte
 - Tarefas dizem “o que” precisa ser feito, já as orientações detalham “como fazer”

Tipos de orientação:

- Checklist 
- Concept 
- Example 
- Guideline 
- Estimate 
- Considerations 
- Practice 
- Report 
- Reusable Asset 
- Roadmap 
- Supporting Material 
- Template 
- Term Definition 
- Tool Mentor 
- Whitepaper 

(5) Categorias

- Usadas para agrupar elementos de método relacionados
- São cinco as categorias padrão:
 - *Discipline*: agrupamento de tarefas
 - *Domain*: agrupamento de produtos de trabalho
 - *Work Product Kind*: similar ao *Domain*
 - *Role Set*: agrupamento de papéis
 - *Tool*: agrupamento de ferramentas
- Categorias pode ser aninhadas
- Podem ser definidas novas categorias
- Os elementos são categorizados através do editor de propriedades
- Usadas para criar visões no site web do processo



Conteúdo de processo

PROCESS CONTENT

(1) Padrão de Capacidade

- Definem a sequência de tarefas relacionadas, realizadas no intuito de alcançar um objetivo maior
- Tarefas pode ser especializadas para um dado contexto

Capability Pattern: Initiate Project

This capability pattern bundles tasks required to define the vision and create a project plan.

Description Work Breakdown Structure Team Allocation Work Product Usage

Expand All Sections Collapse All Section

Work Breakdown		Steps	Index	Predecessors	Model Info	Type	Planned	Repeatable	Multiple	Occurrences	Ongoing	Event-Driven	Optional	Tea
Define Vision		4	1			Task Descriptor								
Analyst					Primary Performer	Role Descriptor	✓							
Stakeholder					Secondary Performer	Role Descriptor	✓							
Vision					Optional Input	Work Product Descriptor	✓							
Glossary					Output	Work Product Descriptor	✓							
Vision					Output	Work Product Descriptor	✓							
Plan the Project		4	2			Task Descriptor								
Project Manager					Primary Performer	Role Descriptor	✓							
Analyst					Secondary Performer	Role Descriptor	✓							
Architect					Secondary Performer	Role Descriptor	✓							
Stakeholder					Secondary Performer	Role Descriptor	✓							
Tester					Secondary Performer	Role Descriptor	✓							
Vision					Mandatory Input	Work Product Descriptor	✓							
Work Items List					Mandatory Input	Work Product Descriptor	✓							
Project Plan					Optional Input	Work Product Descriptor	✓							
Risk List					Optional Input	Work Product Descriptor	✓							
Project Plan					Output	Work Product Descriptor	✓							
Risk List					Output	Work Product Descriptor	✓							
Work Items List					Output	Work Product Descriptor	✓							

Descritores de Tarefa
(instância de Tarefa)

Descritores de Papel
(instância de Papel)

Descritores de Produto de Trabalho
(instância de PT)

(1) Padrão de Capacidade

- Podem ser “aninhados” e visualizados de forma gráfica
- Uma atividade é uma instância de um padrão de capacidade

Capability Pattern: Inception Phase Iteration



This iteration template defines the activities (and associated roles and work products) performed in a typical iteration in the Inception phase. Each activity and related goals are described.

Description

Work Breakdown Structure

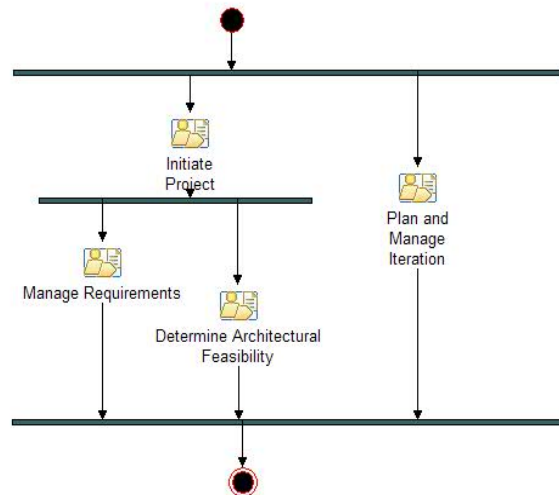
Team Allocation

Work Product Usage

Expand All Sections

Collapse All Sections

Workflow



Back to top

Atividade

instância de padrão de capacidade

Work Breakdown

Breakdown Element	Steps	Index	Predecessors	Model	Info	Type	Planned	Repeatable	Multiple Occurrences	Ongoing	Event-Driven	Optional	Team
Initiate Project	1					Activity	✓						
Plan and Manage Iteration	4					Activity				✓			
Manage Requirements	8	1				Activity	✓						
Determine Architectural Feasibility	12	1				Activity	✓					✓	

Back to top

(2) Processo de Entrega

- Definidos usando uma **Estrutura de Divisão de Trabalho** (Work Breakdown Structure) e/ou diagramas de atividade
- Define o ciclo de vida completo para um processo
- Pode incluir iterações, Fases, Milestones (tipos de Atividades)

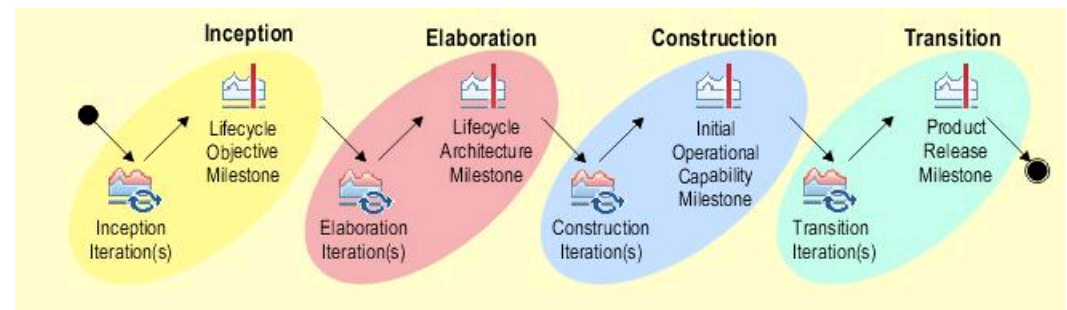
Delivery Process: OpenUP lifecycle



This delivery process defines an end-to-end software development lifecycle that supports the core principles of OpenUP. It is designed to support small, co-located teams in their daily activities.

Description Work Breakdown Structure Team Allocation Work Product Usage

Workflow



Work Breakdown

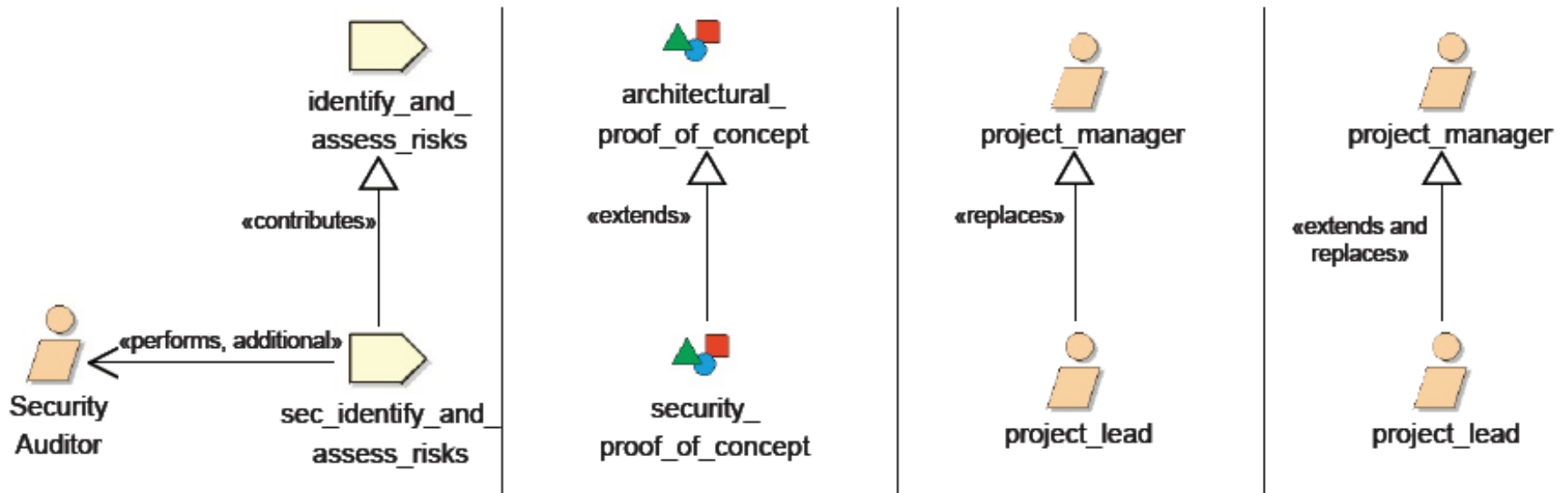
Breakdown Element	Steps	Index	Predecessors	Model Info	Type	Planned	Repeatable
<input type="checkbox"/> Inception Iteration [1..n]		1			Activity	✓	
<input type="checkbox"/> Initiate Project		2			Activity	✓	
<input type="checkbox"/> Plan and Manage Iteration		5			Activity		
<input type="checkbox"/> Identify and Refine Requirements		9	2		Activity	✓	
<input type="checkbox"/> Agree on the Technical Approach		13	2		Activity	✓	
Lifecycle Objectives Milestone		15	1		Milestone	✓	
<input type="checkbox"/> Elaboration Iteration [1..n]		16	15		Activity	✓	✓
Lifecycle Architecture Milestone		46	16		Milestone	✓	
<input type="checkbox"/> Construction Iteration [1..n]		47	46		Activity	✓	✓
Initial Operational Capability Milestone		68	47		Milestone	✓	
<input type="checkbox"/> Transition Iteration [1..n]		69	68		Activity	✓	✓
Product Release Milestone		86	69		Milestone	✓	

(3) Variabilidade de Método

- Mecanismo que permite a customização do conteúdo do método sem modificar o seu conteúdo original
- Similar ao conceito de herança (OO)
 - Permite o reuso através da especialização
- Por exemplo, se um *plugin* B estende um *plugin* A:
 - Os elementos originais do *plugin* A permanecem intactos – todas as mudanças são definidas pelo *plugin* B
- Variabilidade de conteúdo é útil, por exemplo, para:
 - Alterar a descrição de um papel já existente
 - Adicionar passos para uma tarefa já existente
 - Adicionar orientações a uma tarefa já existente

(3) Variabilidade de Método

- Quatro tipos de variabilidade de método:
 - *Contribute* → adicionam conteúdo aos elementos base
 - *Extends* → herdam o conteúdo do elemento base e especializam alguns ou todos (ambos são publicados)
 - *Replace* → substituem o elemento base (só o novo elemento é publicado)
 - *Extends-Replace* → similar ao *extends*, mas o elemento base não é publicado



(4) Variabilidade de Processo

- Mecanismo de reuso similar à variabilidade de método
- Adicionalmente, Atividades também pode ser criadas a partir dos padrões de capacidade, das seguintes formas:
 - *Extends* → a atividade herda as propriedades de um padrão de capacidade. Atualizações no padrão de capacidade são automaticamente refletidas para a atividade
 - *Copy* → a atividade é criada com base no padrão de capacidade. Mas, não permanecem sincronizados
 - *Deep Copy* → similar ao copy, mas aplicado de forma recursiva para as atividades
 - *Local Variability* → quando um padrão de capacidade é definido (por Extends ou Copy) variabilidade local pode ser realizada

Ferramenta para autoria e publicação de processos de software

INTRODUÇÃO AO *EPF COMPOSER*

EPF Composer

- **EPF Composer** é desenvolvido sobre a plataforma Eclipse
- Suporta vários plug-ins do Eclipse
 - Ex.: Mylar
- Visões diferentes apresentam informações específicas
 - Ex.: a visão “Library” apresenta os plug-ins e seus conteúdos
- Perspectivas agrupam visões relacionadas
- Perspectivas padrão são:
 - *Authoring* → própria para a edição de conteúdos de método
 - *Browsing* → para visualizar uma prévia da publicação dos elementos de método e processo

Perspectiva *Authoring*

Authoring Perspective

Library View

Task Editor (form based)

Configuration View

The screenshot shows the Eclipse Process Framework Composer interface. The title bar indicates the path: C:\cmsynergy\uschi\ccm_wa\logan\OpenUP~\uschi\OpenUP. The menu bar includes File, Edit, Search, SYNERGY/CM, Configuration, Window, and Help. The toolbar contains icons for file operations and a toolbar with 'Authoring', 'Authoring Sy...', 'Telelogic SYN...', and 'Browsing' buttons.

The **Library View** on the left shows a tree structure of content packages and tasks. The **Task Editor (form based)** in the center displays the 'Task: define_vision' form. The **Configuration View** at the bottom shows a tree structure of configuration elements.

The **Task Editor (form based)** displays the following information:

- General Information:**
 - Name: define_vision
 - Presentation name: Define Vision
 - Brief description: Define the vision for the future system. Describe the problem and features based on Stakeholder requests.
- Detail Information:**
 - Purpose: The solution is proposed for a problem that everybody agrees on. Stakeholders collaborate with the development team to express and document their problems, needs, and potential features for
 - Main description:
 - Key considerations:
 - Alternatives:
- Version Information:**
 - Version: 1.0.0
 - Change date: Wednesday, February 28, 2007
 - Change description:
 - Authors:

The bottom of the window has a tabbed interface with 'Description', 'Steps', 'Roles', 'Work Products', 'Guidance', 'Categories', and 'Preview' tabs.

Perspectiva Authoring

Eclipse Process Framework Composer - C:\Home\Rational RUP team\TNG\Infrastructure\Samples\Beacon

File Edit Search Internal Configuration Window Help

Classic RUP (for large projects)

Library

rup_analysis_class

Work Product (Artifact): rup_analysis_class

General Information
Provide general information about this artifact.

Name: rup_analysis_class
Presentation name: Analysis Class
Unique ID:
Brief description: This work product specifies elements of an early conceptual model for 'things in the system which have responsibilities and behavior'.

Detail Information
Provide detailed information about this artifact.

Purpose: Analysis classes are used to capture the major system.

Main description: Analysis Classes specify elements of an early conceptual model of the system which have responsibilities and behaviors. They are prototypical classes of the system, and are a first abstraction that the system must handle. Analysis classes maintained in their own right, if a "high-level", can be desired. Analysis classes also give rise to the system.

Key considerations:

Notation
Provide notation information about this artifact.

Brief outline:

Representation options: Course Section

Tailoring
Provide tailoring information about this artifact.

Impact of not having:

Form based
plain text or...

...Rich Text editors

Work Product (Artifact): rup_analysis_class

Representation options:

Normal

CRC Card technique - see [WIR90] for details of this technique. On the front side of the card, capture the name and description of the class. An example for a Course in a course registration system is listed below.

Class Name	Course									
Description	The Course is responsible for maintaining information about a set of course sections having a common subject, requirements and syllabus.									
Responsibilities	To maintain information about the course.									
Attributes	<table border="1"><thead><tr><th>Name</th><th>Description</th><th>Type</th></tr></thead><tbody><tr><td>Course Title</td><td>The name of the course</td><td>string</td></tr><tr><td>Description</td><td>A short description of the course</td><td>string</td></tr></tbody></table>	Name	Description	Type	Course Title	The name of the course	string	Description	A short description of the course	string
Name	Description	Type								
Course Title	The name of the course	string								
Description	A short description of the course	string								

On the back of the card, draw a diagram of the class:

```
graph TD
    Course[Course] -- 0..* --> Section[Section]
    Section -- 1..* --> Professor[Professor]
    Section -- 1..* --> Student[Student]
    Section -- 1..* --> Room[Room]
    Section -- 0..* --> Textbook[Textbook]
    Section -- pre-requisite --> Section
```

Perspectiva *Browsing*

Browsing Perspective

Configuration View

Preview View

Eclipse Process Framework Composer - C:\cmsynergy\uschsi\ccm_wa\logan\OpenUP~uschsi\OpenUP

File Edit Search SYNERGY/CM Configuration Window Help

OpenUP

Configuration View

- Disciplines
 - OpenUP Disciplines
 - Architecture
 - Configuration and change man...
 - Development
 - Project Management
 - Requirements
 - Define Vision
 - Detail Requirements
 - Find and Outline Requireme...
 - Test
 - Uncategorized Tasks
 - Domains
 - Work Product Kinds
 - Role Sets
 - Tools
 - Processes
 - Custom Categories
 - Guidance

Content

Task: Define Vision

Define the vision for the future system. Describe the problem and features based on Stakeholder requests.

Disciplines: Requirements

Expand All Sections Collapse All Sections

Purpose

The solution is proposed for a problem that everybody agrees on. Stakeholders collaborate with the development team to express and document their problems, needs, and potential features for the system to be, so the project team can better understand what has to be done.

Back to top

Relationships

Roles	Primary Performer: <ul style="list-style-type: none">Analyst	Additional Performers: <ul style="list-style-type: none">ArchitectProject ManagerStakeholder
Inputs	Mandatory: <ul style="list-style-type: none">None	Optional: <ul style="list-style-type: none">VisionWork Items List
Outputs	<ul style="list-style-type: none">GlossaryVisionWork Items List	
Process Usage	<ul style="list-style-type: none">Initiate Project > Define Vision	

Back to top

Steps

Identify Stakeholders

Expand All Steps Collapse All Steps

Configuração

- Seleção de um subconjunto da biblioteca de método para ser publicado em HTML ou exportado para MS Project ou XML

The screenshot displays the Eclipse Process Framework Composer interface for configuring the OpenUP project. The main window is titled "Eclipse Process Framework Composer - C:\cmsgynergy\uschi\ccm_wa\logan\OpenUP~uschi\OpenUP". The "Library" view on the left shows a tree structure of project components, with a blue callout bubble labeled "Configurations" pointing to the "Configurations" folder. The "Configuration: OpenUP" view in the center shows the "Configuration Content" section, which includes a "Content" list and two "Add these Categories" and "Subtract these Categories" sections. A blue callout bubble labeled "Select Content" points to the "Content" list. The "Content" list includes various components such as "base_concepts", "dsdm_openup", "harmony_core", "harmony_itsw_cntxt", "harmony_itsw_dashboard", "harmony_itsw_docexpress", "harmony_itsw_doors", "harmony_itsw_epf", "harmony_itsw_focalpoint", "harmony_itsw_logiscope", "harmony_itsw_proc", "harmony_itsw_rhapsody", "harmony_itsw_synergy", "harmony_itsw_systemarchitect", "harmony_itsw_tau", "openup", "tlog_support_for_openup", "Method Content", "Content Packages", "Processes", and "tlog_support_for_openup". The "Add these Categories" and "Subtract these Categories" sections include "openup", "Disciplines", "Domains", "Work Product Kinds", "Role Sets", and "Custom Categories". The "Configuration" view at the bottom shows a list of categories including "Disciplines", "Domains", "Work Product Kinds", "Role Sets", "Tools", "Processes", "Custom Categories", and "Guidance".

Configuração: Definição das Visões

- Categorias agrupam elementos relacionados
- Visões definidas pela seleção de categorias

The screenshot displays the Eclipse Process Framework Composer interface for configuring the OpenUP project. The main window is titled "Configuration: OpenUP".

Standard Categories: A callout points to the "Library" view on the left, which shows a tree structure of categories. The "Standard Categories" folder is expanded, showing sub-categories like "Method Content", "Content Packages", "Disciplines", "Domains", "Work Product Kinds", "Role Sets", and "Tools".

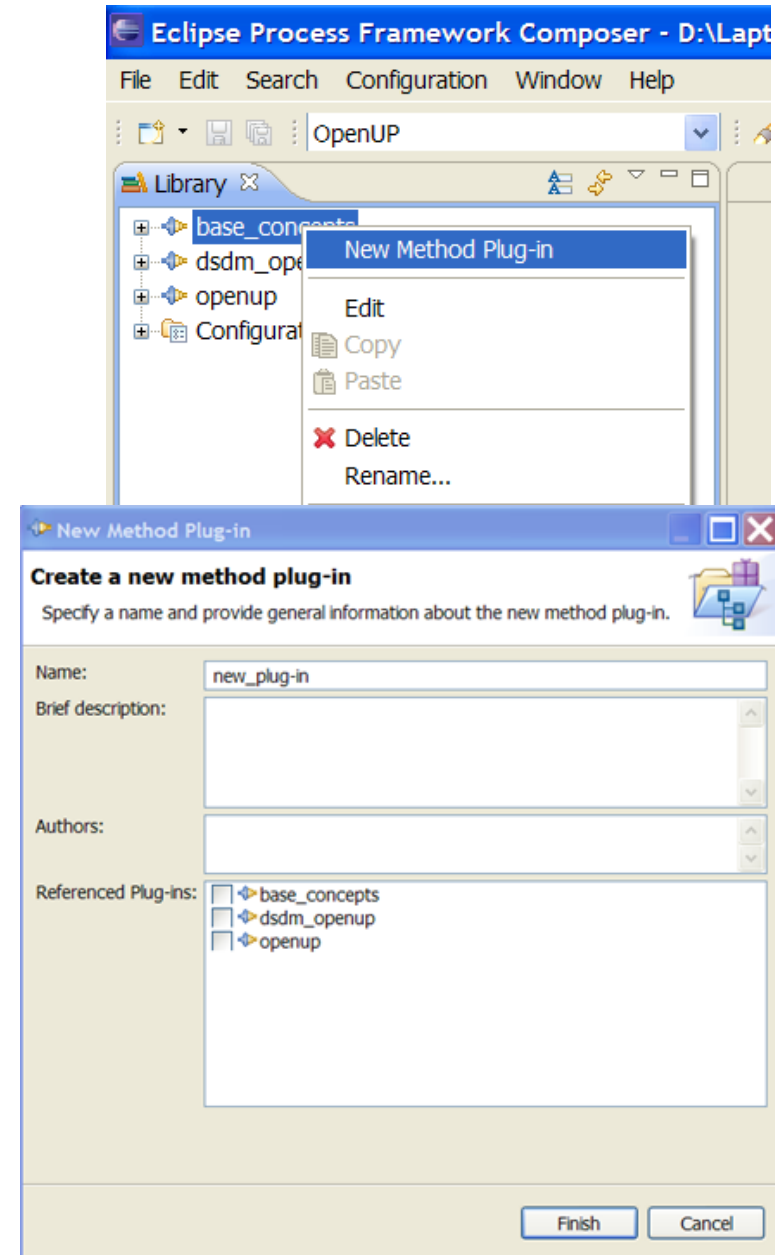
Custom Categories: A callout points to the "Custom Categories" folder in the Library view, which contains sub-categories like "openup_views", "openup_treebrowser", "introduction_to_openup", "getting_started_with_openup", "openup_disciplines", "openup_wip", "openup_roles", "openup_lifecycle", "about", and "openup_copyright".

Define Views: A callout points to the "Published Navigation Views" section in the Configuration: OpenUP window. This section includes instructions on how to create a view by selecting a category. Below the instructions are buttons for "Add View...", "Remove View", "Make Default", and "Order". The "openup_treebrowser" view is currently selected, showing a list of content elements: "Introduction to OpenUP", "Getting Started", "OpenUP Disciplines", "OpenUP Work Products", "OpenUP Roles", "OpenUP lifecycle", "About", and "OpenUP Copyright".

At the bottom of the interface, there is a "Configuration" view showing a summary of the configuration, including "Disciplines", "Domains", "Work Product Kinds", "Role Sets", "Tools", "Processes", "Custom Categories", and "Guidance".

Criar um *Plugin*

- Clicando com o botão direito sobre um componente existente na biblioteca
 - Selecione “New Method Plug-in”
- Defina:
 - Nome (minúsculas, sem espaços)
 - Descrição (opcional)
 - Autor (opcional)
 - Plug-ins referenciados
- “Referenced Plug-ins” define a visibilidade a outros plug-ins



Criar um *Plugin*

New Plugin

The screenshot displays the Eclipse Process Framework Composer interface. The main window is titled "Method Plug-in: my_new_plugin". The left sidebar shows a tree view of the project structure, with "my_new_plugin" selected. The right pane contains the configuration form for the plug-in, which is divided into several sections:

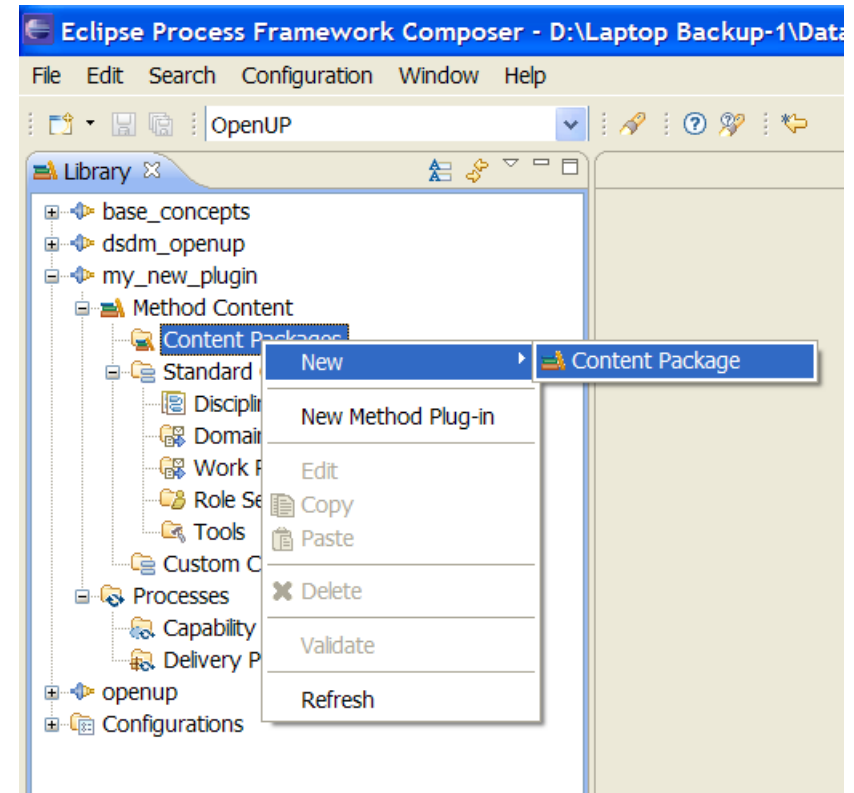
- General Information:** Name: my_new_plugin; Brief description: This plugin is created for training purposes.
- Version Information:** Version: (empty); Change date: (empty); Change description: (empty); Authors: Chris Sibbald; Copyright: (empty).
- Referenced Plug-ins:** A list of plug-ins referenced by this method plug-in, including "openup", "base_concepts", and "dsdm_openup".

The bottom of the window shows a table with columns for "Property" and "Value".

Property	Value

Criar um *Method Content Package*

- Clicar com o botão direito sobre “Content Packages”
 - Selecione “New Content Package”
- O novo pacote é criado
- Este é aberto no editor



Criar um *Method Content Package*

New Content Package

The screenshot displays the Eclipse Process Framework Composer interface. The 'Library' view on the left shows a tree structure with 'Method Content' expanded to 'Content Packages', where 'my_content_package' is selected. The 'Configuration' view at the bottom left shows a list of categories including Disciplines, Domains, Work Product Kinds, Role Sets, Tools, Processes, Custom Categories, and Guidance. The main editor area is titled 'Content Package: my_content_package' and contains the following sections:

- General Information**: Provide general information about this content package.
 - Name: my_content_package
 - Brief description: This content package was created for training purposes.
- Dependencies**: This section displays dependencies of this content package to other content packages.

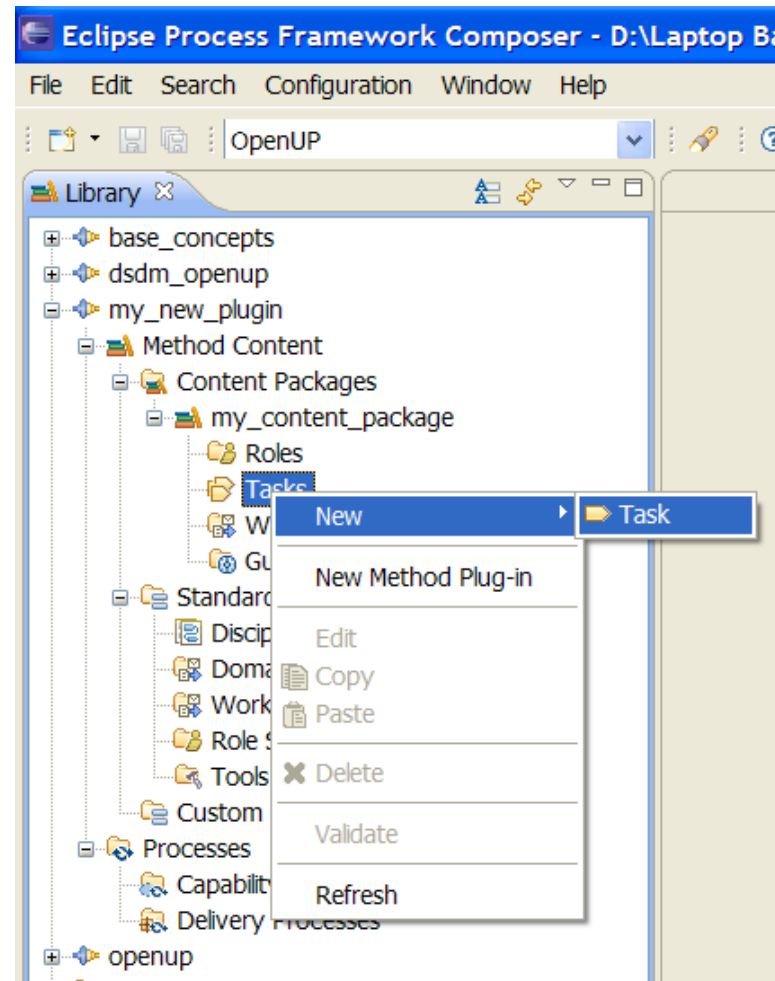
At the bottom, there is a 'Description' field and a 'Properties' table with columns for Property and Value.

Property	Value

The status bar at the bottom shows the file path: D:\Laptop Backup-1\Data\Telelogic\EPF2\EPF Composer\epf-c...-1.2.0-win32\epf-composer\OpenUP\my_new_plugin\plugin.xml

Criar uma Tarefa

- Clicar com o botão direito sobre “Tasks” no pacote de conteúdo desejado
 - Selecione “New Task”
- A nova tarefa é criada
- Este é aberto no editor



Criar uma Tarefa

New Task

The screenshot displays the Eclipse Process Framework Composer interface. The 'Library' view on the left shows a tree structure with 'my_task' selected under 'Tasks'. A blue callout bubble labeled 'New Task' points to this entry. The main editor area shows the configuration for 'Task: my_task', divided into 'General Information' and 'Detail Information' sections. The 'General Information' section includes fields for Name (my_task), Presentation name (My Task), and Brief description (This task was created for training purposes). The 'Detail Information' section includes fields for Purpose, Main description, and Key considerations. The bottom of the interface shows a 'Properties' table with columns for Property and Value.

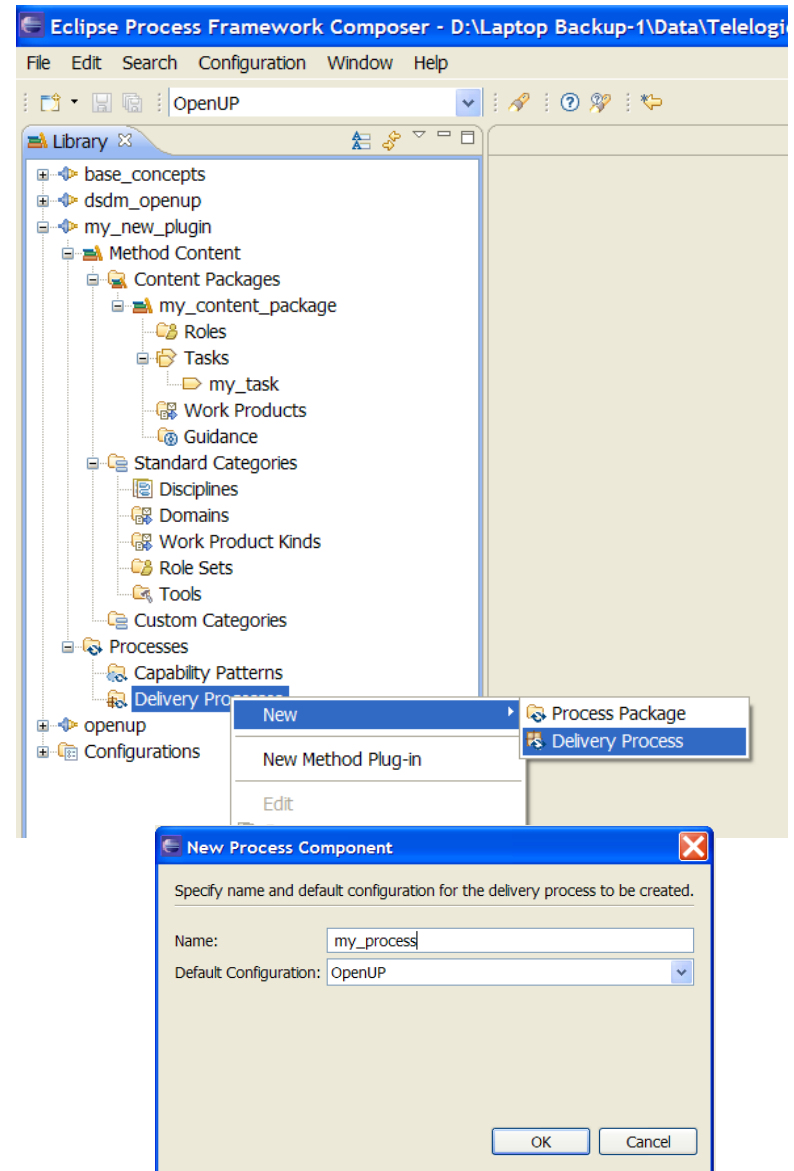
Property	Value

Editar uma Tarefa

- O editor de tarefas possui uma série de abas:
 - *Description* → definir uma visão geral da tarefa
 - *Steps* → definir os passos para a tarefa
 - *Roles* → definir os responsáveis pela tarefa
 - *Work Products* → definir artefatos de “entradas” e “saídas”
 - *Guidance* → associar elementos de orientação à tarefa
 - *Categories* → categorizar a tarefa
 - *Preview* → ver o resultado final
- Alguns campos possuem a capacidade de uma edição com mais recursos (*Ritch Text Editing*)

Criar um Delivery Process

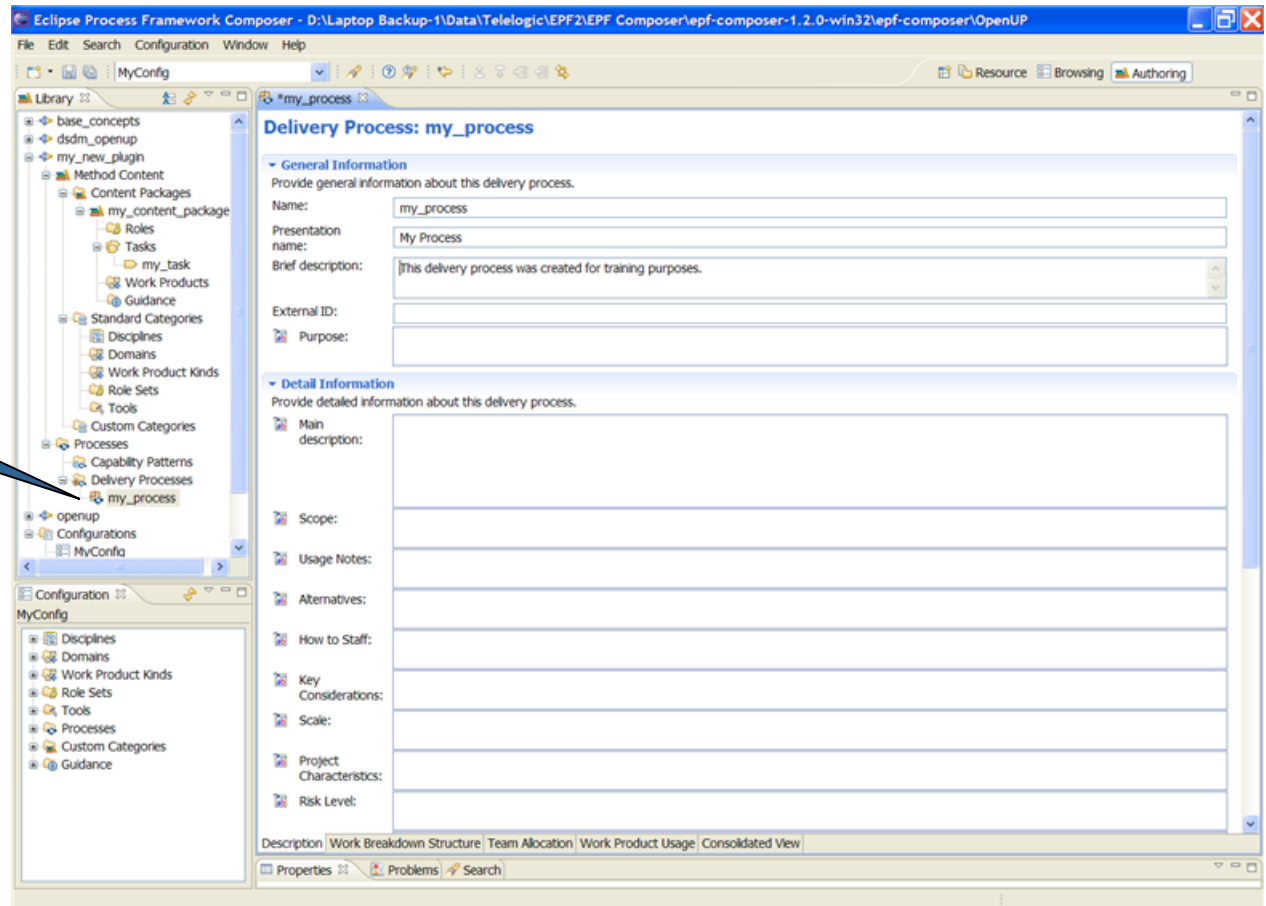
- Clicar com o botão direito sobre “Delivery Process”
 - Selecione “New Delivery Process”
- Define-se:
 - Nome (minúsculas, sem espaço)
 - Configuração padrão



Criar um *Delivery Process*

- O editor permite a alteração/atualização do conteúdo

New Delivery Process

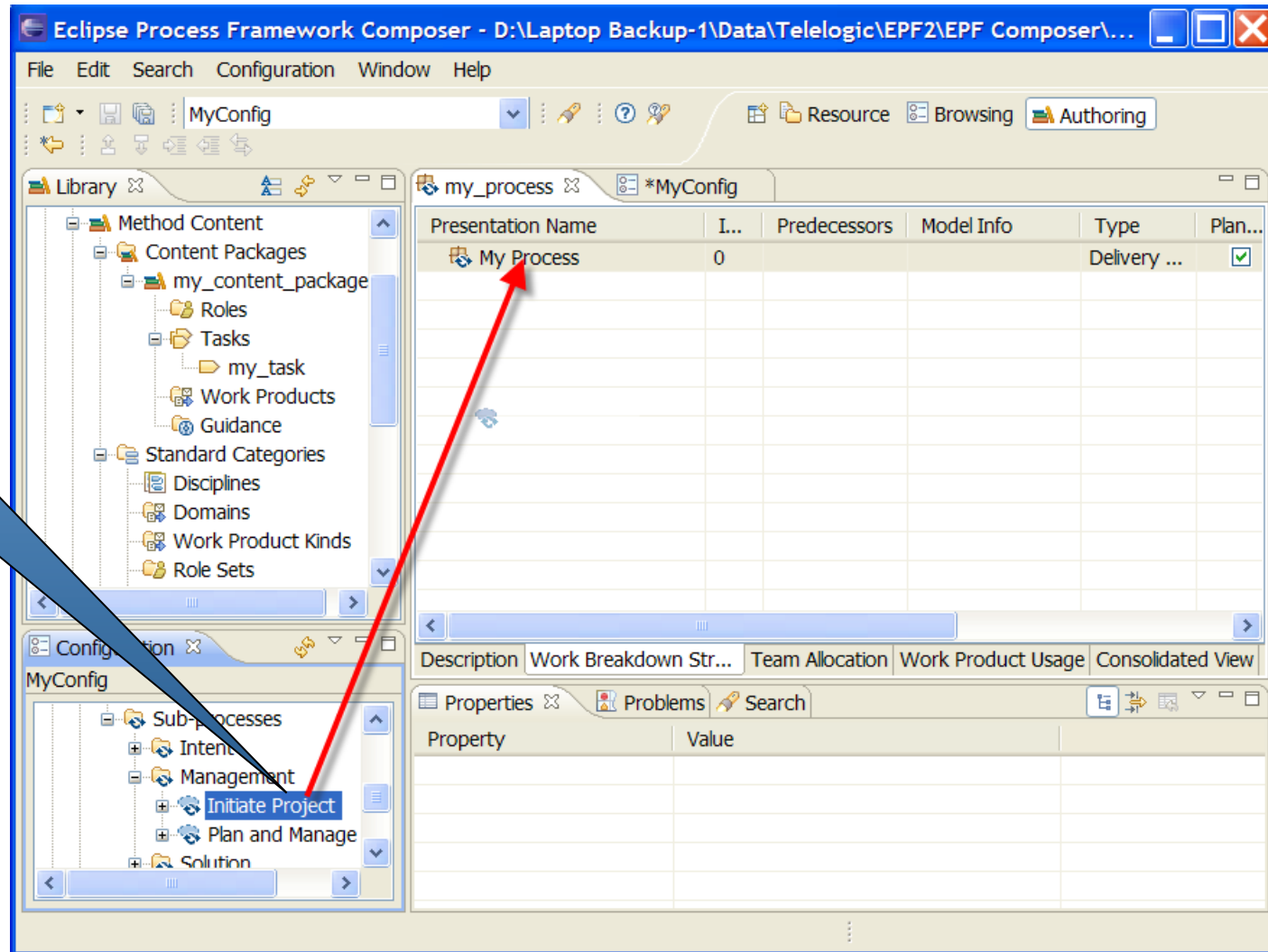


Criar um *Delivery Process*

- O editor do Delivery Process possui uma série de abas:
 - *Description* → definir atributos gerais do processo
 - *WBS* → definir atividades e seus relacionamentos
 - *Team Allocation* → visualizar e editar papéis
 - *Work Products Usage* → visualizar e editar produtos de trabalho
 - *Consolidated View* → visão consolidada (preenchida automaticamente)

Criar um *Delivery Process*

Drag/Drop CP

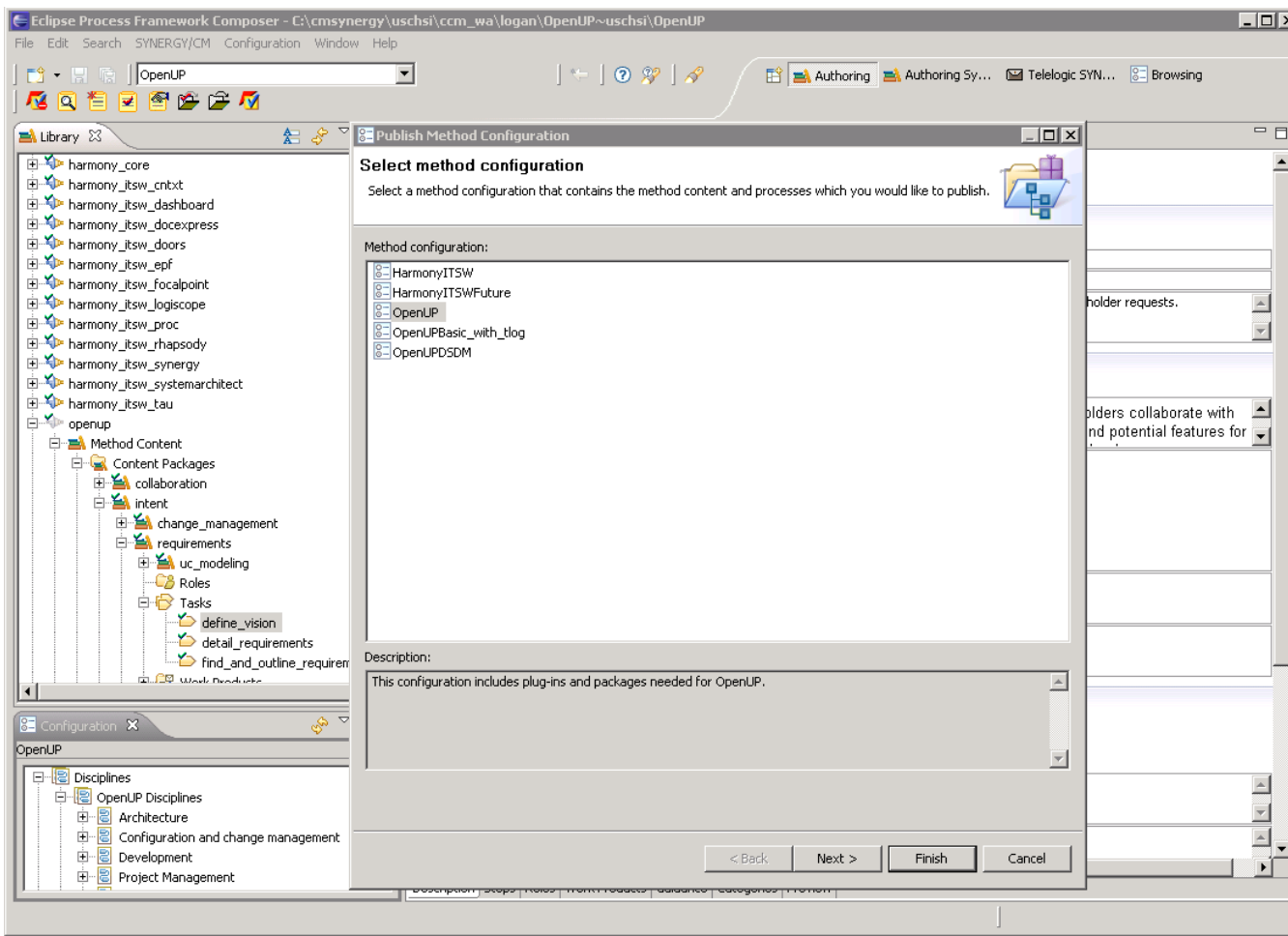


Criar um Delivery Process

- Ao adicionar um padrão de capacidade ao delivery process é possível:
 - *Extend* → irá manter um link ao CP, que caso seja atualizado, a mudança irá se refletir no *delivery process*
 - *Copy* → irá criar uma cópia local do CP, a qual não estará ligada ao CP original
 - *Deep Copy* → similar ao *copy*, mas é aplicado recursivamente aos sub-CPs
- A mais comum é a *Extend*

Publicação

- “Configuration | Publish” para iniciar o assistente de publicação



Website Resultado

http://usma-doorsdb/epf/openup/ - Microsoft Internet Explorer

Eclipse Process Framework Composer

Where am I | Tree Sets

OpenUP

- Introduction to OpenUP
 - Core Principles
 - Minimal, Complete, and Extensible
 - Iteration Lifecycle
 - Micro-Increments
 - Project Lifecycle
 - The OpenUP Family
 - Who Should Use OpenUP
 - Getting Started
 - OpenUP Roadmap
 - Resources for Modifying Models
 - Resources for Contributing Models
 - OpenUP Disciplines
 - Architecture
 - Configuration and Change Management
 - Development
 - Project Management
 - Requirements
 - Test
 - OpenUP Work Products
 - Architecture
 - Development
 - Project Management
 - Requirements
 - Test
 - OpenUP Roles
 - Analyst
 - Any Role
 - Architect
 - Developer
 - Project Manager
 - Stakeholder
 - Tester
 - OpenUP lifecycle
 - About
 - OpenUP Copyright

Main Description

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 layers across three focus areas: Personal Focus, Team Focus, and Stakeholder Focus. The vertical axis represents time: Days, Weeks, and Months. The horizontal axis represents project phases: Inception, Elaboration, Construction, and Transition. A red curve labeled 'Risk' starts high in the Inception phase and decreases through the other phases. A green curve labeled 'Value' starts low in the Inception phase and increases through the other phases. The layers are: 1. Micro-Increment (Personal Focus, Days): Shows a 'Work Item' leading to an 'Increment' (represented by a gear). 2. Iteration Lifecycle (Team Focus, Weeks): Shows an 'Iteration Plan' leading to an 'Iteration' (represented by a larger gear) which results in a 'Demo-able or Shippable Build' (represented by a cube). 3. Project Lifecycle (Stakeholder Focus, Months): Shows a 'Project Plan' leading to the overall project progression through the phases.

OpenUP layers: micro-increments, iteration lifecycle and project lifecycle

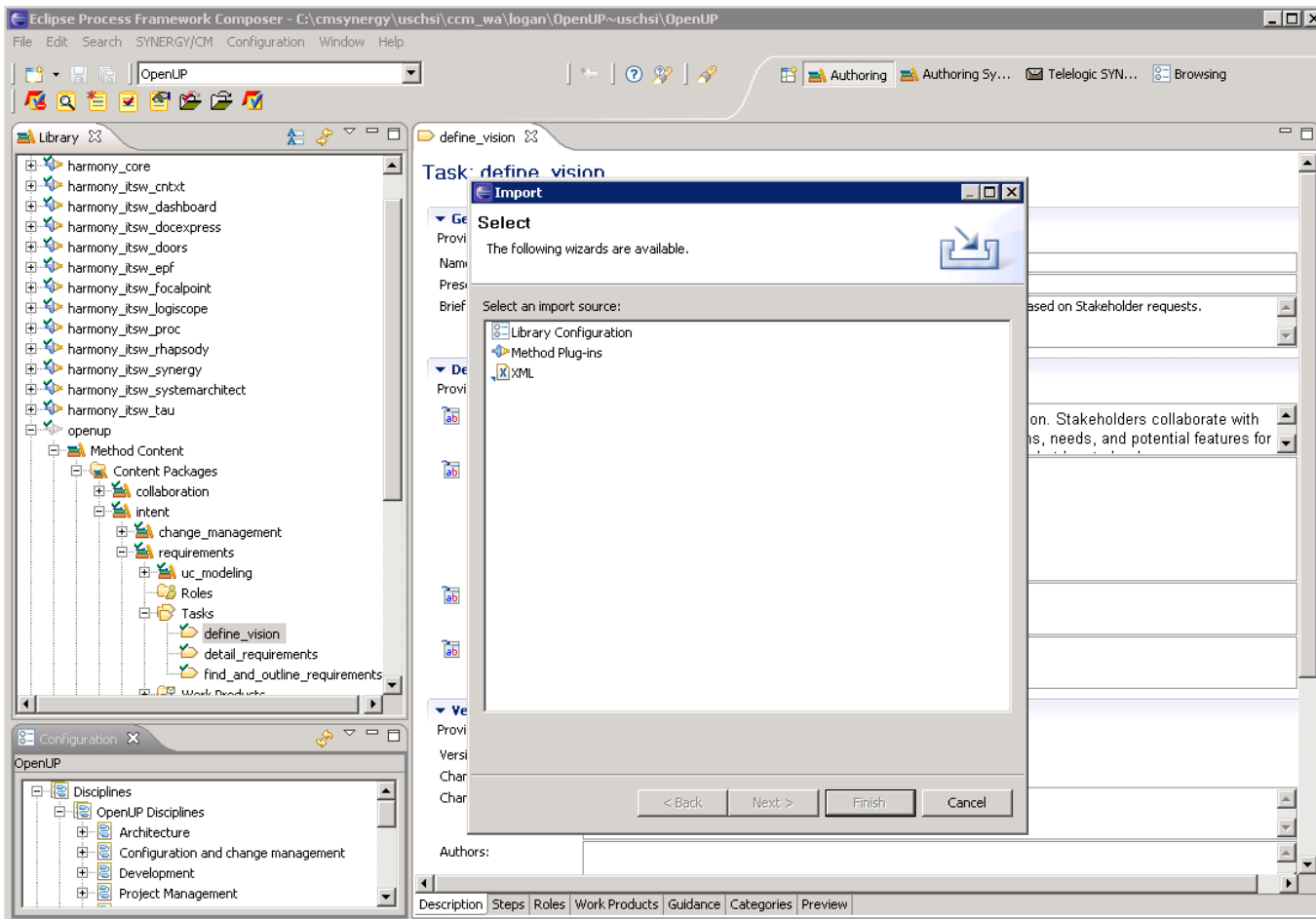
Personal effort on an OpenUP project is organized in **micro-increments**. These represent short units of work that produce a steady, measurable pace of project progress (typically measured in hours or a few days). The process applies intensive collaboration as the system is incrementally developed by a committed, self-organized team. These micro-increments provide an extremely

OpenUP lifecycle

Local intranet

Importação

- “File | Import” para iniciar o assistente de importação
- Pode ser importado uma Configuração, um plug-in, ou XML



Exportação

- “File | Export” para iniciar o assistente de exportação
- Pode exportar uma Configuração, plug-in, XML ou template MS Project

