



- 1 Introduction**
- 2 User Perspective**
- 3 Developer Perspective**

## Context

- An end-user must be guided to apply domain activities
- The activities must be easily accessible to be executed and well organized

## Need

- Providing an interface with the main following features:
  - Ability to be declined by domain (e.g., technical, process)
  - Presentation of the activities by main topics and sub-topics
  - Ability to be customized

## Objective

- The Activity Explorer provides the main following features:
  - The entry point is an overview of pages; each page contains sections of activities
  - Navigation between pages
  - Extensibility by contribution of new pages, sections and activities

- 1 Introduction
- 2 User Perspective
- 3 Developer Perspective

# The Activity Explorer is exemplified on a system engineering process

## Overview

**Process Map of myProject**

- Operational Analysis**: Define Stakeholder Needs and Environment. Capture and consolidate operational needs from stakeholders. Define what the users of the system have to accomplish. Identify entities, actors, roles, activities, concepts.
- System Analysis**: Formalize System Requirements. Identify the boundary of the system, consolidate requirements. Define what the system has to accomplish for the users. Model functional dataflows and dynamic behaviour.
- Logical Architecture**: Develop System Logical Architecture. See the system as a white box. Define how the system will work so as fulfill expectations. Perform a first trade-off analysis.
- Physical Architecture**: Develop System Physical Architecture. How the system will be developped and built. Software vs. hardware allocation, specification and interfaces, deployment configurations, trade-off analysis.
- EPBS**: Formalize Component Requirements. Manage industrial criteria and integration strategy: what is expected from each designer/subcontractor. Specify requirements and interfaces of all configuration items.

Process Map | Operational Analysis | System Analysis | Logical Architecture | Physical Architecture | EPBS

## Page

**Operational Analysis**

- Define Operational Entites and Capabilities
- Define Operational Activities and describe Interaction
- Allocate Operational Activities to Operational Actors, Entities or Roles
  - [OAB] Create a new Operational Architecture Diagram
  - [ORB] Create a new Operational Role Diagram
  - [OES] Create a new Operational Entity Scenario
- Transverse Modeling

Viewer

type filter text

**Sections**

**Activities**

**Describe the state machine of the system**, specifying which are its modes and states. Among others, state machines can be created on the following kinds of elements, components, actors, classes (data), etc.

**Describe the domain element and actually exchanged data.**

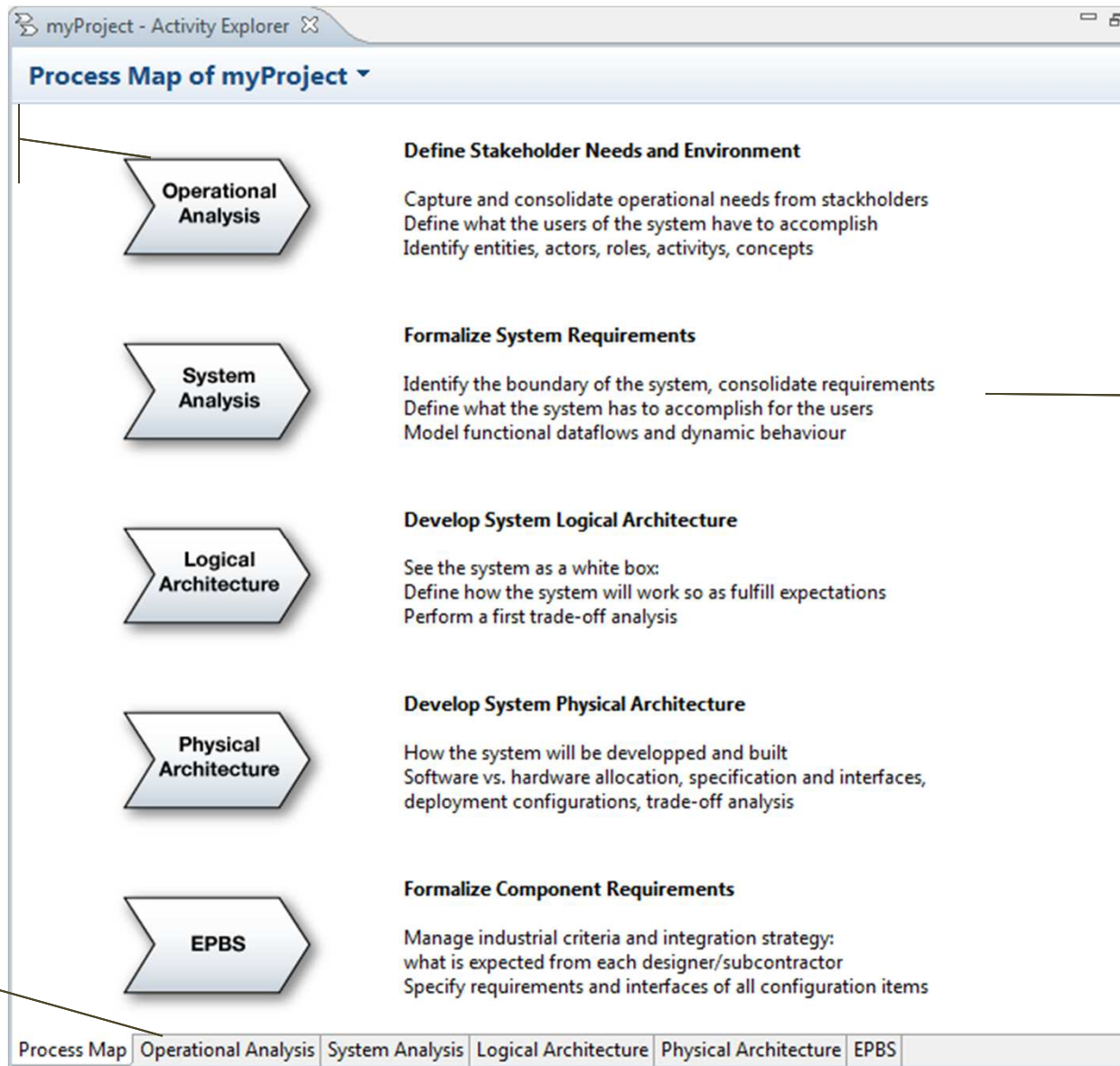
- Domain Elements: modeling elements of the domain should not "polluted" by technical consideration (e.g: internal representation of data, database storage, access methods, etc.). In the beginning, concentrate only on elements that provide high-level semantic: related to the domain.
- Data actually exchanged between components: used for example to type parameters in interface operations. These data have to be unambiguous and consistent.

Both the domain element and actual data are described in a Class diagram.

Process Map | Operational Analysis | System Analysis | Logical Architecture | Physical Architecture | EPBS

This document is not to be reproduced, modified, adapted, published, translated in any material form in whole or in part nor disclosed to any third party without the prior written permission of Thales© THALES 2013 – All rights reserved.

Link to Activity Explorer page



Description

Page

OPEN

## Objective

- Providing the entry point of the activities
- Providing an overview of the activities organized by pages

## User Actions

- Displaying the overview page
- Navigating between activity pages
- Selecting an activity page

**Page title** points to the window title 'myProject - Activity Explorer'.

**Current page** points to the 'Operational Analysis' activity in the process map.

**Section** points to the 'Allocate Operational Activities to Operational Actors, Entities or Roles' section.

**Activities** points to the list of activities under the section, including '[OAB] Create a new Operational Architecture Diagram', '[ORB] Create a new Operational Role Diagram', and '[OES] Create a new Operational Entity Scenario'.

**Activity's icon** points to the icon of the '[OES] Create a new Operational Entity Scenario' activity.

**Section description button popup** points to a popup window containing the following text:

**Describe the state machine of the system**, specifying which are its modes and states. Among others, state machines can be created on the following kinds of elements, components, actors, classes (data), etc.

**Describe the domain element and actually exchanged data.**

- Domain Elements: modeling elements of the domain should not "polluted" by technical consideration (e.g.: internal representation of data, database storage, access methods, etc.). In the beginning, concentrate only on elements that provide high-level semantic related to the domain.
- Data actually exchanged between components: used for example to type parameters in interface operations. These data have to be unambiguous and consistent.

Both the domain element and actual data are described in a Class diagram.

This document is not to be reproduced, modified, adapted, published, translated in any material form in whole or in part nor disclosed to any third party without the prior written permission of Thales. © THALES 2013 – All rights reserved.



The screenshot shows the 'myProject - Activity Explorer' application. The main interface is titled 'Operational Analysis' and contains a navigation pane with the following sections:

- Operational Analysis** (Define Stakeholders Needs)
- System Analysis**
- Define Operational Entites and Capabilities**
- Define Operational Activities and describe Interaction**
- Allocate Operational Activities to Operational Actors, Entities or Roles**
  - [OAB] Create a new Operational Architecture Diagram
  - [ORB] Create a new Operational Role Diagram
  - [OES] Create a new Operational Entity Scenario
- Transverse Modeling**

A callout box points to a filter icon (two arrows) next to the 'Allocate Operational Activities...' section, stating: "Button to filter in the Viewer the artefacts matching all the activities of the section".

The 'Viewer' window on the right has a search input field labeled 'type filter text'. A callout box points to this field, stating: "Pattern to filter artefacts in the Viewer".

Another callout box points to the main content area of the 'Viewer', stating: "Viewer to list the artefacts matching the activities and pattern".

At the bottom of the application, there is a navigation bar with tabs: Process Map, Operational Analysis, System Analysis, Logical Architecture, Physical Architecture, and EPBS.

This document is not to be reproduced, modified, adapted, published, translated in any material form in whole or in part nor disclosed to any third party without the prior written permission of Thales. © THALES 2013 – All rights reserved.

**Operational Analysis**  
Define Stakeholders Needs

**System Analysis**

- Define Operational Entities and Capabilities
- Define Operational Activities and describe Interactions
- Allocate Operational Activities to Operational Actors, Entities and Capabilities
  - [OAB] Create a new Operational Architecture diagram
  - [ORB] Create a new Operational Role diagram
  - [OES] Create a new Operational Entity Scenario
- Transverse Modeling

**Diagrams Viewer**

Select a name to find  
? = any character, \* = any string

- Common
- Operational Analysis

**1 Activity Execution, here diagram creation**

myProject - Overview

My Operational Analysis Architecture

MyEntity

**2 The diagram is created**

The screenshot shows the 'Operational Analysis' workflow in the Activity Explorer. The workflow consists of several steps: 'Define Operational Entities and Capabilities', 'Define Operational Activities and describe Interactions', 'Allocate Operational Activities to Operational Actors, Entities and Capabilities', and 'Transverse Modeling'. A dialog box titled 'Type representation name' is open, with the text 'My Operational Analysis Architecture' entered. The 'Diagrams Viewer' window is also open, showing a tree view with 'Common' and 'Operational Analysis' folders. A blue callout box with the number '3' and the text 'The viewer is updated after the creation of the diagram' points to the 'Operational Analysis' folder in the viewer.

This screenshot shows a diagram viewer window titled 'My Operational Analysis Architecture'. The diagram contains a single entity represented by a yellow box with the text 'MyEntity' inside.

This screenshot shows the 'Diagrams Viewer' window after the diagram has been created. The tree view now includes 'Operational Architecture Blank' and 'My Operational Analysis Architecture' under the 'Operational Analysis' folder. A blue arrow points from the callout box in the previous screenshot to the 'Operational Analysis' folder in this viewer.

This document is not to be reproduced, modified, adapted, published, translated in any material form in whole or in part nor disclosed to any third party without the prior written permission of Thales. © THALES 2013 – All rights reserved.

The screenshot shows the 'Operational Analysis' workflow with steps: 'Define Operational Entities and Capabilities', 'Define Operational Activities and describe Interactions', and 'Allocate Operational Activities to Operational Actors, Entities and Capabilities'. A dialog box is open for 'Type representation name' with the value 'My Operational Analysis Architecture'. The 'Diagrams Viewer' panel on the right shows a search bar and a tree view with 'Common' and 'Operational Analysis' folders.

**4** Actions available on the artefact type

The diagram viewer shows a single artefact labeled 'MyEntry' on a light yellow background.

**5** Open action for navigation

The 'Diagrams Viewer' shows a search bar and a tree view. A context menu is open over an artefact, listing actions: 'Show in Explorer', 'Open', 'Clone Diagram', 'Delete', 'Rename', 'Patterns', and 'REC / RPL'. The 'Open' action is highlighted.

This document is not to be reproduced, modified, adapted, published, translated in any material form in whole or in part nor disclosed to any third party without the prior written permission of Thales. ©THALES 2013 – All rights reserved.

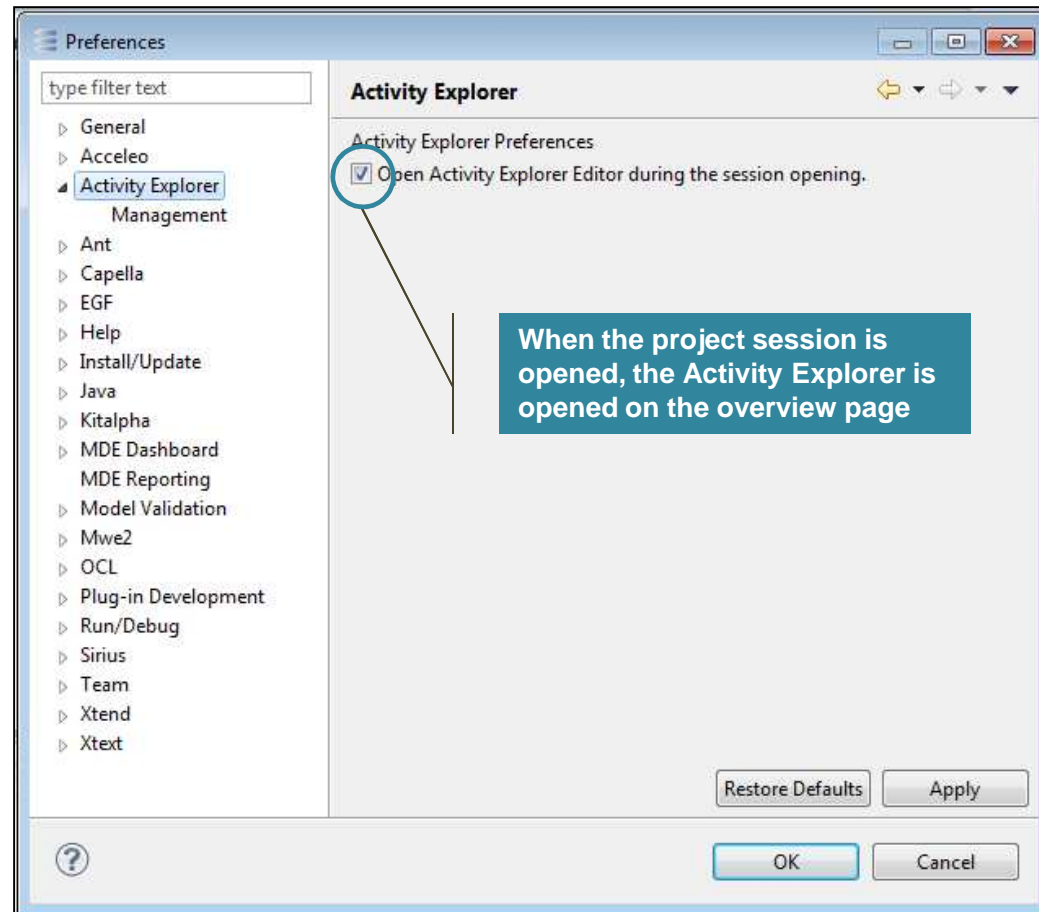
## Objective

- Presentating the activities by page and sections
- Displaying in a viewer the artefacts which match the page / section activities
- Executing an action on an artefact

## User Actions

- Folding/unfolding an activity section
- Navigating between previous and next pages
- Filter and displaying in a viewer artefacts which match activity criteria
- Executing an activity
- Executing an action on an artefact

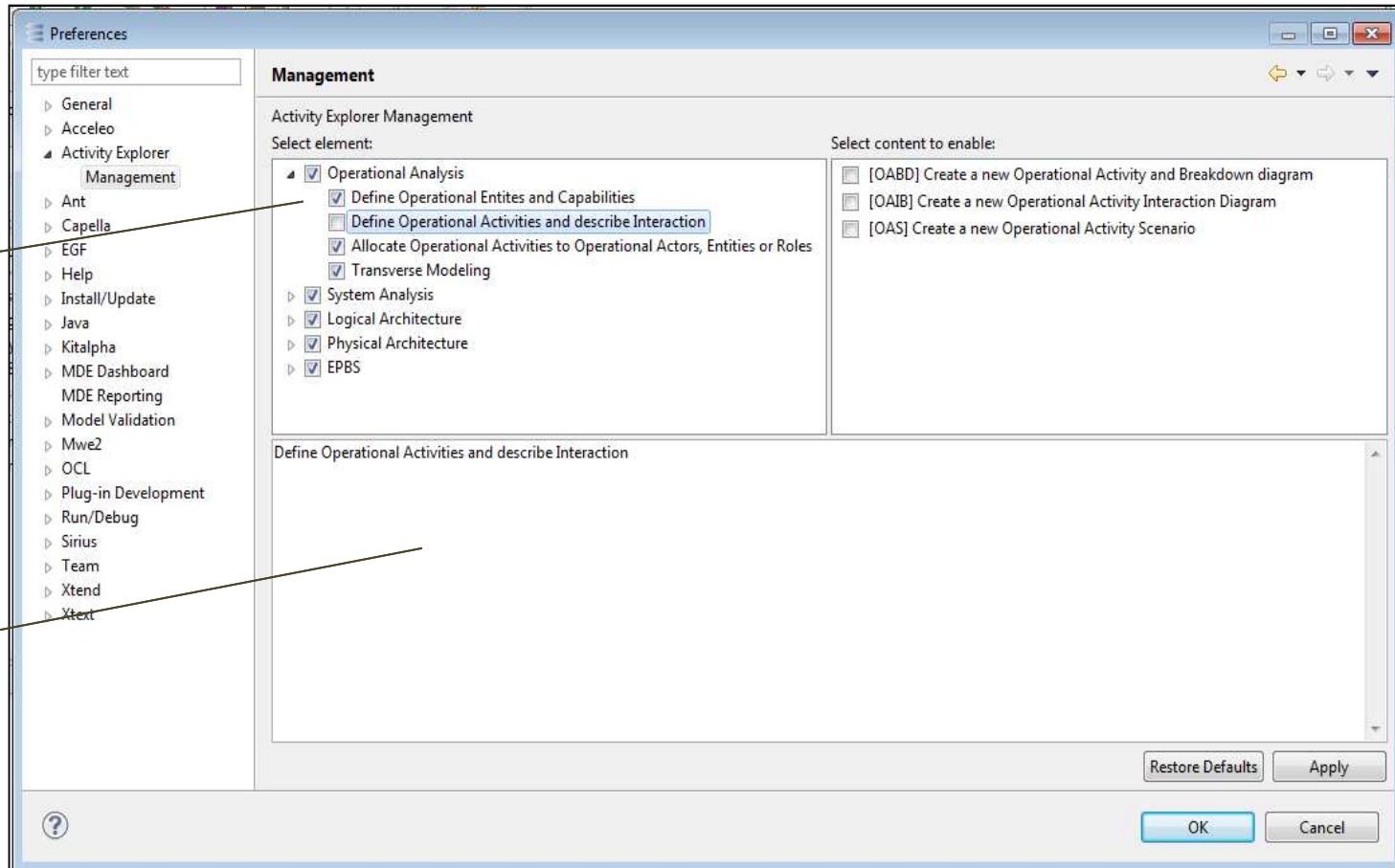
## Preferences – Activity Explorer



## Preferences – Activity Explorer – Management

Allows to activate/deactivate pages, section or activities. When an element is unchecked, it will not be visible within the Activity Explorer

Description of the Activity Explorer elements (pages, sections Activities)



This document is not to be reproduced, modified, adapted, published, translated in any material form in whole or in part nor disclosed to any third party without the prior written permission of Thales. © THALES 2013 – All rights reserved.

## Objective

- Customizing the Activity Explorer with options

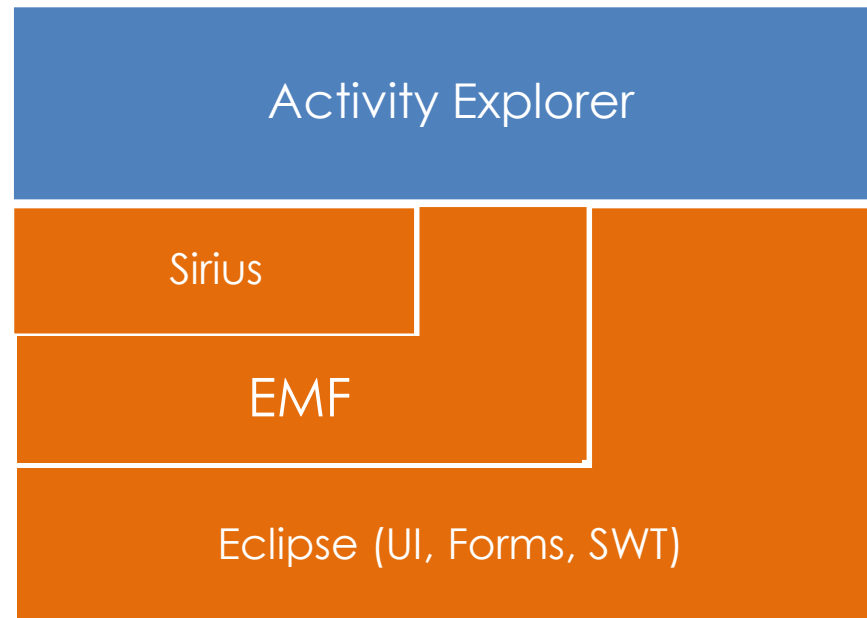
## User Actions

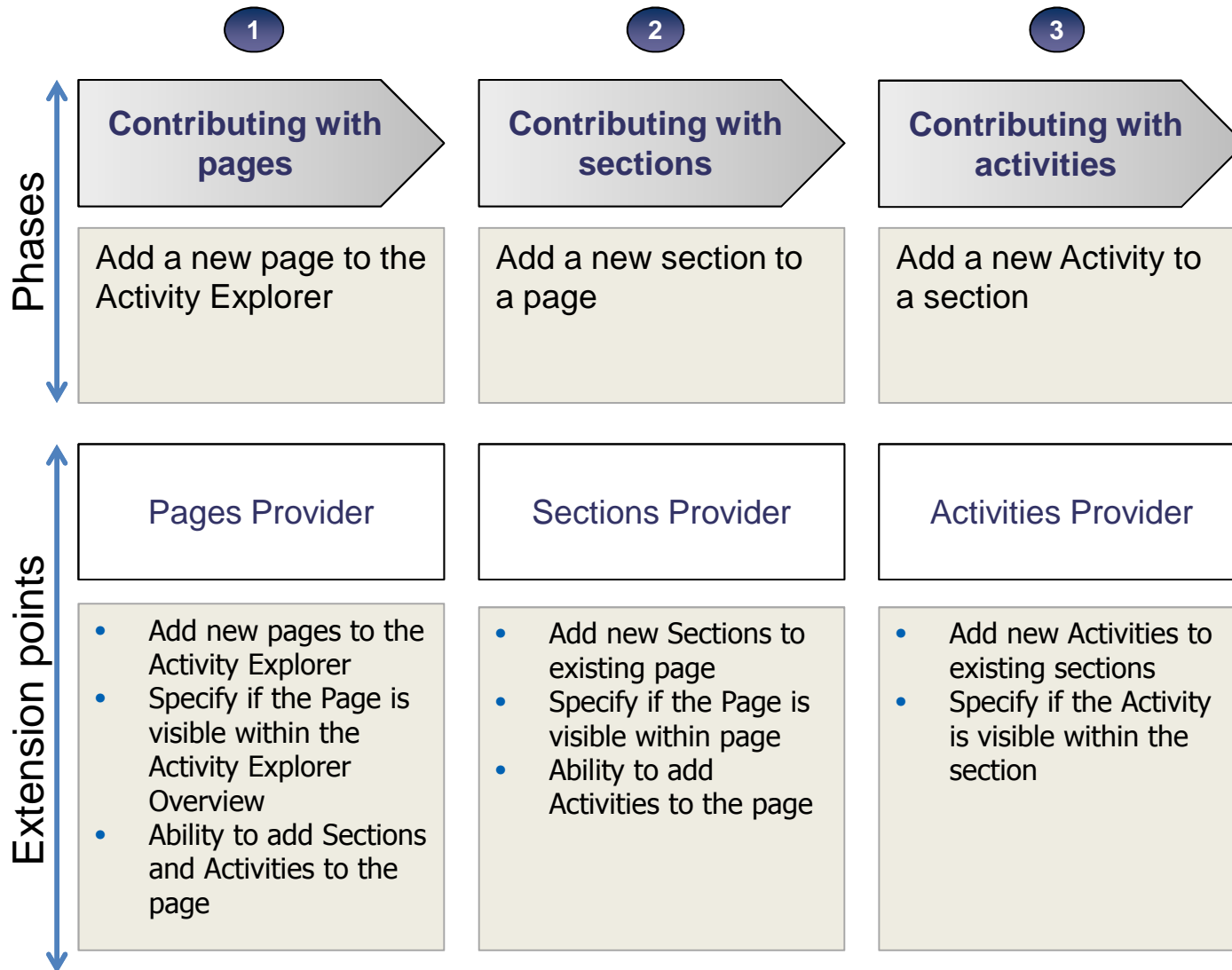
- In the Preferences view, proposing options to active/deactivate pages, sections



- 1 Introduction
- 2 User Perspective
- 3 Developer Perspective

# Foundations





OPEN

## Page Contribution

- Implementation

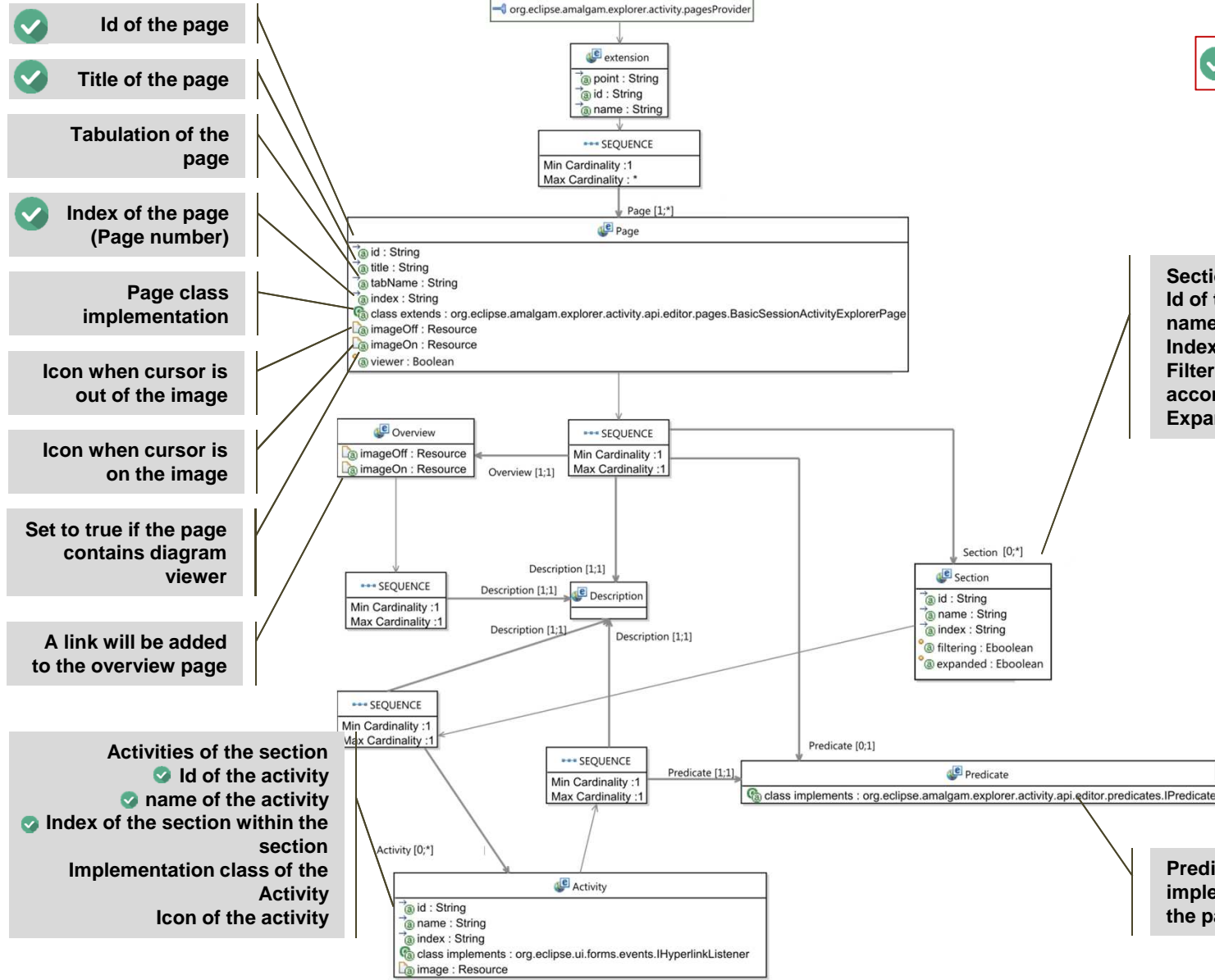
<b>Plugin name</b>	org.eclipse.amalgam.explorer.activity
<b>Java Package</b>	org.eclipse.amalgam.explorer.activity.api.editor.pages
<b>Class name</b>	BasicSessionActivityExplorerPage

- Extension point(s)

Name	Plugin	Schema
pagesProvider	org.eclipse.amalgam.explorer.activity	pagesProvider.exsd

- Default implementation

<b>Description</b>	Empty page, it is used when contribution to the page without class implementation
<b>Java Package</b>	org.eclipse.amalgam.explorer.activity.api.editor.pages
<b>Class name</b>	BasicSessionActivityExplorerPage



✓ Id of the page

✓ Title of the page

Tabulation of the page

✓ Index of the page (Page number)

Page class implementation

Icon when cursor is out of the image

Icon when cursor is on the image

Set to true if the page contains diagram viewer

A link will be added to the overview page

Activities of the section

✓ Id of the activity

✓ name of the activity

✓ Index of the section within the section

Implementation class of the Activity

Icon of the activity

✓ Mandatory field

Sections of the page  
 Id of the section ✓  
 name of the section ✓  
 Index of the section within the page ✓  
 Filtering the diagram viewer according to the section  
 Expand the section at the opening

Predicate of the page which implements precondition to add the page to the Activity Explorer

OPEN

## Section Contribution

- Implementation

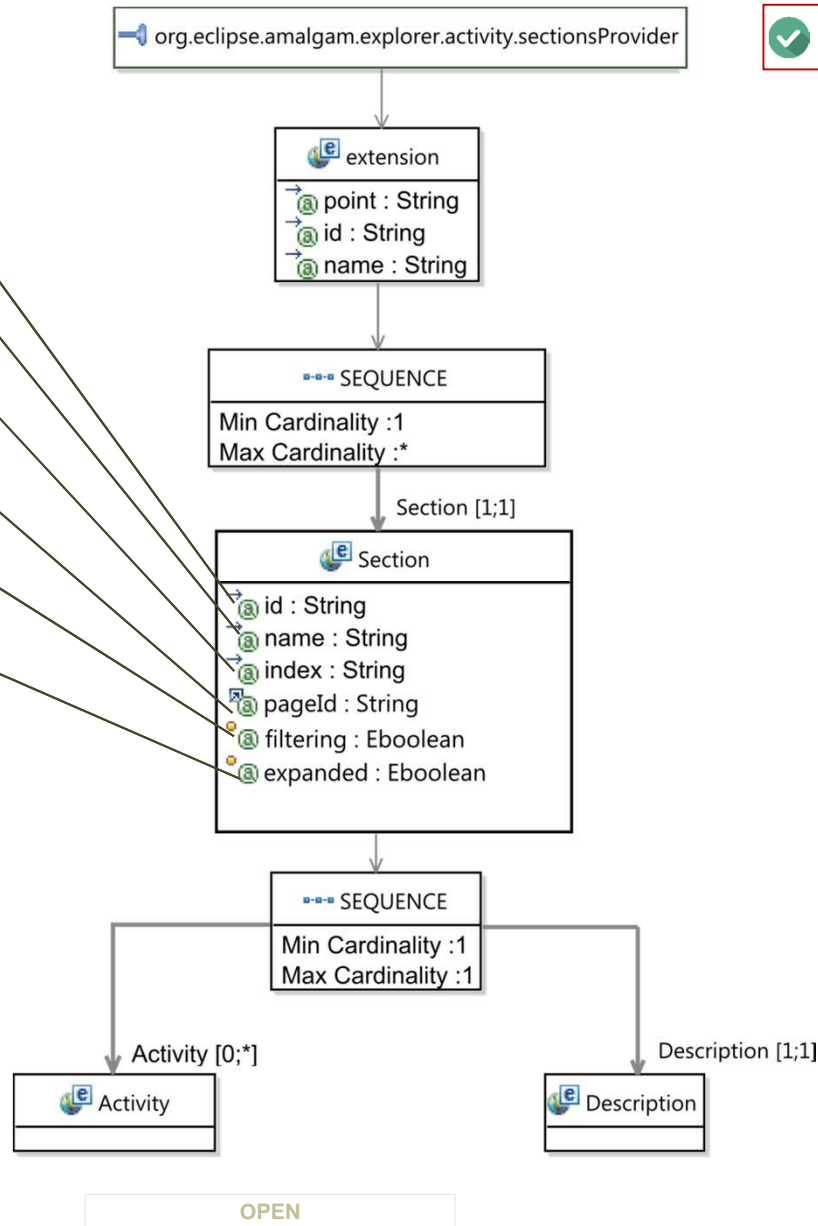
<b>Description</b>	Add a section to a page. The section doesn't need to provide class at the extension.
<b>Plugin name</b>	org.eclipse.amalgam.explorer.activity
<b>Java Package</b>	org.eclipse.amalgam.explorer.activity.api.editor.sections
<b>Class name</b>	ActivityExplorerSection

- Extension point(s)

Name	Plugin	Schema
sectionsProvider	org.eclipse.amalgam.explorer.activity	sectionsProvider.exsd

- ✓ Id of the section
- name of the section
- ✓ Index where the section must appear within Activity Explorer. Must be an Integer
- ✓ The Id of the page where the section will be appear
- Indicate if Viewer filtering is activated
- Indicate if Section is expanded at the opening

✓ Mandatory field



This document is not to be reproduced, modified, adapted, published, translated in any material form in whole or in part nor disclosed to any third party without the prior written permission of Thales. © THALES 2013 – All rights reserved.



## Activity Contribution

- Implementation

<b>Plugin name</b>	org.eclipse.amalgam.explorer.activity
<b>Java Package</b>	org.eclipse.ui.forms.events
<b>Interface name</b>	org.eclipse.ui.forms.events.IHyperlinkListener

- Extension point(s)

Name	Plugin	Schema
activitiesProvider	org.eclipse.amalgam.explorer.activity	activitiesProvider.exsd

✓ Mandatory field

✓ Id of the activity

name of the activity

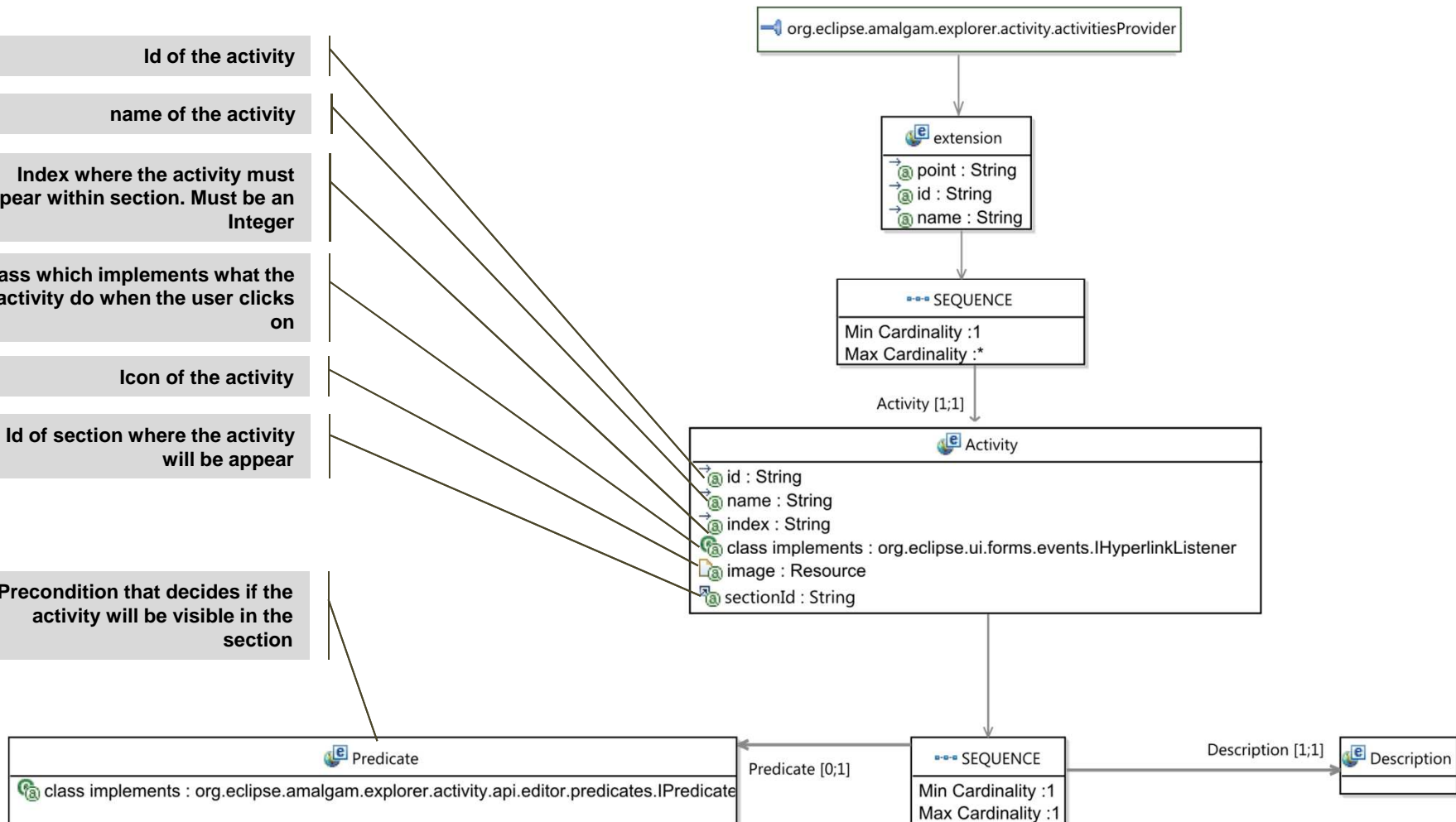
✓ Index where the activity must appear within section. Must be an Integer

Class which implements what the activity do when the user clicks on

Icon of the activity

✓ Id of section where the activity will be appear

Precondition that decides if the activity will be visible in the section



This document is not to be reproduced, modified, adapted, published in any material form in whole or in part nor disclosed to any third party without the prior written permission of Thales © THALES 2013 – All rights reserved.

- Predicate
  - A page or an activity can be associated to a predicate
  - Allows to put visibility conditions on the page/activity within the Activity Explorer

- Implementation

<b>Plugin name</b>	org.eclipse.amalgam.explorer.activity
<b>Java Package</b>	org.eclipse.amalgam.explorer.activity.api.editor.predicates
<b>Interface name</b>	org.eclipse.amalgam.explorer.activity.api.editor.predicates.IPredicate

- Extension point(s)

<b>Name</b>	<b>Plugin</b>	<b>Schema</b>
<b>PagesProvider</b>	org.eclipse.amalgam.explorer.activity	pageProvider.exsd

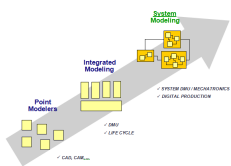
# Example

This document is not to be reproduced, modified, adapted, published, translated in any material form in whole or in part nor disclosed to any third party without the prior written permission of Thales. © THALES 2013 – All rights reserved.

# The Activity Explorer has been developed in the context of PolarSys by Capella and Kitalpha

 **Capella**  
<http://polarsys.org/capella/>

 **Kitalpha**  
<http://polarsys.org/kitalpha/>



Capella has been supported by **Clarity**, a French collaborative project

Kitalpha has been supported by **Sys2Soft**, **Crystal**, and **Clarity**, French and European collaborative projects

## Purpose

- Implement a partial Capella-like activity Explorer
  - Operational Analysis page with all functionalities (**Sections, Activities**)
    - **Define Operational Entities and Capabilities**
      - [OEBD] Create a new Operational Entity Breakdown
      - [OCB] Create a new Operational Capability diagram
    - **Allocate Operational Activities to Operaitonal Actors, Entities or Roles**
      - [OAB] Create a new Operational Architecture Diagram
      - [ORB] Create a new Operaitonal Role diagram
      - [OES] Create a new Operational Entitiy Scenario
    - **Transverse Modeling**
      - [CDB] Create a new Class Diagram
      - [M&S] Create a new Modes & States Machine
      - Create a new State & Mode / Operational Activities matrix
  - Other pages without functionalities
    - System Analysis page
    - Logical Architecture Page
    - Physical Architecture Page
    - EPBS page

## Steps

1. Open Capella studio
2. Create a new plugin: `org.polarsys.capella.core.activity.explorer`
3. Add dependencies to:
  - `org.eclipse.amalgam.explorer.activity`
  - `org.eclipse.sirius`
  - `org.eclipse.sirius.ui`
  - `org.eclipse.sirius.diagram`
  - `org.polarsys.capella.core.ui.toolkit`
  - `org.polarsys.capella.core.model.helpers`

4. Go to the extensions tab
5. Add extension to: org.eclipse.amalgam.explorer.activity.pageProvider
6. Set values of created first page as in the capture (Operational Analysis page)

**Extension Element Details**

Set the properties of "Page". Required fields are denoted by "\*".

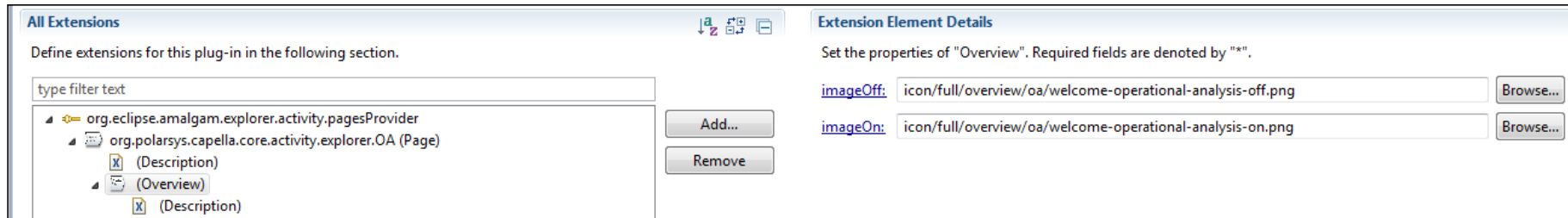
id*:	<input type="text" value="org.polarsys.capella.core.activity.explorer.OA"/>
title*:	<input type="text" value="Operational Analysis"/>
index*:	<input type="text" value="1"/>
tabName:	<input type="text" value="Operational Analysis"/>
class:	<input type="text"/> <input type="button" value="Browse..."/>
imageOff:	<input type="text" value="icon/full/overview/oa/operationalanalysis_overview_01.png"/> <input type="button" value="Browse..."/>
imageOn:	<input type="text" value="icon/full/overview/oa/operationalanalysis_overview_02.png"/> <input type="button" value="Browse..."/>
viewer:	<input type="text" value="true"/>



We don't need to provide implementation in our case because the default implementation is enough



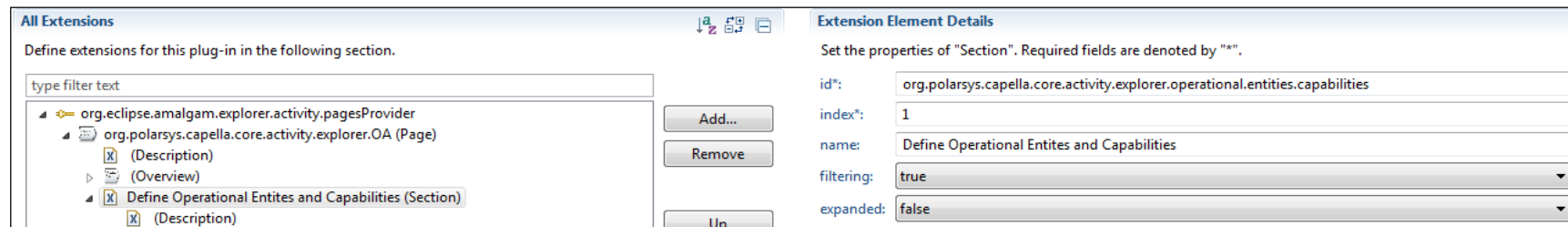
## 7. Specify the icons of the page in Overview section



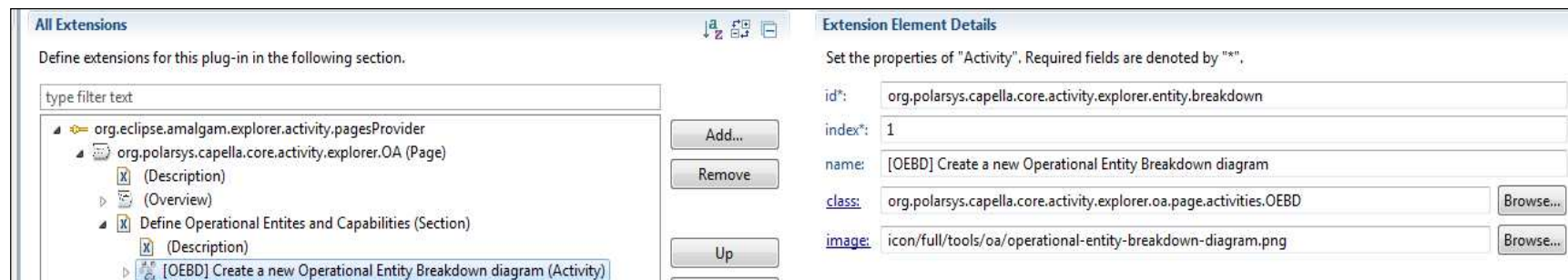
## 8. Specify the description below of the page in Overview section

```
<p>
<b>Define Stakeholder Needs and Environment</b><br/><br/>
Capture and consolidate operational needs from stackholders<br/>
Define what the users of the system have to accomplish<br/>
Identify entities, actors, roles, activities, concepts
</p>
```

9. Add a new Section by right click on the page, new menu, then Section
10. Fill the fields of the new section as in the capture below



11. Add a new Activity to the Section by right click on section, new menu, then Activity
12. Fill the fields of the new Activity as in the capture below



### 13. Create an abstract class: AbstractCapellaNewDiagram which extends AbstractNewDiagramHyperLinkAdapter

```
public abstract class AbstractCapellaNewDiagram extends AbstractNewDiagramHyperlinkAdapter {  
  
    public AbstractCapellaNewDiagram(EObject project_p) {  
        super(project_p, ActivityExplorerManager.INSTANCE.getSession());  
    }  
}
```

### 14. For The activity “Create a new Operational Entity Breakdown diagram”, the implementation class look like below

```
public class OEBD extends AbstractCapellaNewDiagram {  
  
    public OEBD() {  
        /*  
        * Get the right level in Capella project where create the element  
        */  
        super(ModelQueryHelper.getOperationalContext((Project) ActivityExplorerManager.INSTANCE.getRootSemanticModel()));  
    }  
  
    @Override  
    public String getRepresentationName() {  
        //The name of the visual description that allows to get the right diagram for the element  
        return "Operational Entity Breakdown";  
    }  
}
```

## 15. Do the same steps to add sections and activities to Operational Analysis Pages

### Implementation classes for Operational Analysis' activities (1/5)

```
public class OCB extends AbstractCapellaNewDiagram {  
  
    public OCB () {  
        /*  
        * Get the right level in Capella project where create the element  
        */  
        super(ModelQueryHelper. getRootOperationalCapability((Project) ActivityExplorerManager.INSTANCE.getRootSemanticModel()));  
    }  
  
    @Override  
    public String getRepresentationName() {  
        //The name of the visual description that allows to get the right diagram for the element  
        return "Operational Capabilities Blank" ;  
    }  
}
```

```
public class OABD extends AbstractCapellaNewDiagram {  
  
    public OABD () {  
        /*  
        * Get the right level in Capella project where create the element  
        */  
        super(ModelQueryHelper. getRootOperationalActivity((Project) ActivityExplorerManager.INSTANCE.getRootSemanticModel()));  
    }  
  
    @Override  
    public String getRepresentationName() {  
        //The name of the visual description that allows to get the right diagram for the element  
        return "Operational Activity Breakdown" ;  
    }  
}
```

## Implementation classes for Operational Analysis' activities (2/5)

```
public class OAIB extends AbstractCapellaNewDiagram {  
  
    public OAIB () {  
        /*  
        * Get the right level in Capella project where create the element  
        */  
        super(ModelQueryHelper. getRootOperationalActivity((Project) ActivityExplorerManager.INSTANCE.getRootSemanticModel()));  
    }  
  
    @Override  
    public String getRepresentationName() {  
        //The name of the visual description that allows to get the right diagram for the element  
        return "Operational Capabilities Blank";  
    }  
}
```

```
public class OAS extends AbstractCapellaNewDiagram {  
  
    public OAS () {  
        super(ActivityExplorerManager.INSTANCE.getRootSemanticModel());  
    }  
  
    @Override  
    public String getRepresentationName() {  
        //The name of the visual description that allows to get the right diagram for the element  
        return "Activity Interaction Scenario";  
    }  
  
    @Override  
    protected void linkPressed(HyperlinkEvent event_p, EObject root_p, Session session_p) {  
  
        root_p = ModelCreationHelper.selectOperationalCapabilityAndCreateInteractionScenario((Project) root_p);  
  
        if (!createDiagram(root_p, session_p)) {  
            handleDiagramCreationError(event_p, root_p);  
        }  
    }  
}
```

## Implementation classes for Operational Analysis' activities (3/5)

```
public class OAB extends AbstractCapellaNewDiagram {  
  
    public OAB () {  
        /*  
        * Get the right level in Capella project where create the element  
        */  
        super(ModelQueryHelper. getOperationalContext((Project) ActivityExplorerManager.INSTANCE.getRootSemanticModel()));  
    }  
  
    @Override  
    public String getRepresentationName() {  
        //The name of the visual description that allows to get the right diagram for the element  
        return "Operational Capabilities Blank ";  
    }  
}
```

```
public class ORB extends AbstractCapellaNewDiagram {  
  
    public ORB () {  
        /*  
        * Get the right level in Capella project where create the element  
        */  
        super(ModelQueryHelper. getOperationalContext((Project) ActivityExplorerManager.INSTANCE.getRootSemanticModel()));  
    }  
  
    @Override  
    public String getRepresentationName() {  
        //The name of the visual description that allows to get the right diagram for the element  
        return "Operational Role Blank";  
    }  
}
```

## Implementation classes for Operational Analysis' activities (4/5)

```
public class OES extends AbstractCapellaNewDiagram {

    public OES () {
        super(ActivityExplorerManager.INSTANCE.getRootSemanticModel());
    }

    @Override
    public String getRepresentationName() {
        //The name of the visual description that allows to get the right diagram for the element
        return "Operational Interaction Scenario";
    }

    @Override
    protected void linkPressed(HyperlinkEvent event_p, EObject root_p, Session session_p) {
        root_p = ModelCreationHelper.selectOperationalCapabilityAndCreateInteractionScenario((Project) root_p);
        if (!createDiagram(root_p, session_p)) {
            handleDiagramCreationError(event_p, root_p);
        }
    }
}
```

```
public class CDB extends AbstractCapellaNewDiagram {

    public CDB () {
        /*
         * Get the right level in Capella project where create the element
         */
        super(ModelQueryHelper.getOaDataPkg((Project) ActivityExplorerManager.INSTANCE.getRootSemanticModel()));
    }

    @Override
    public String getRepresentationName() {
        //The name of the visual description that allows to get the right diagram for the element
        return "Class Diagram Blank";
    }
}
```

## Implementation classes for Operational Analysis' activities (5/5)

```
public class MandS extends AbstractCapellaNewDiagram {  
  
    public MandS () {  
        super(ActivityExplorerManager.INSTANCE.getRootSemanticModel());  
    }  
  
    @Override  
    public String getRepresentationName() {  
        //The name of the visual description that allows to get the right diagram for the element  
        return "Modes & States ";  
    }  
  
    @Override  
    protected void linkPressed(HyperlinkEvent event_p, EObject root_p, Session session_p) {  
        root_p = ModelCreationHelper. selectOperationalEntityAndCreateStateMachineRegion((Project) root_p);  
        if (!createDiagram(root_p, session_p)) {  
            handleDiagramCreationError(event_p, root_p);  
        }  
    }  
}
```

```
public class CreateMatix extends AbstractCapellaNewDiagram {  
  
    public CreateMatix () {  
        /*  
        * Get the right level in Capella project where create the element  
        */  
        super(ModelQueryHelper. getOperationalAnalysis((Project) ActivityExplorerManager.INSTANCE.getRootSemanticModel()));  
    }  
  
    @Override  
    public String getRepresentationName() {  
        //The name of the visual description that allows to get the right diagram for the element  
        return "State And Mode - Matrix";  
    }  
}
```



## 16. Add predicate on Operational Analysis page and check if the semantic root model is Capella Project

Define extensions for this plug-in in the following section.

type filter text

- org.eclipse.amalgam.explorer.activity.pagesProvider
  - org.polarsys.capella.core.activity.explorer.OA (Page)
    - (Description)
    - (Overview)
    - Define Operational Entites and Capabilities (Section)
    - Define Operational Activities and describe Interaction (Section)
    - Allocate Operational Activities to Operational Actors, Entities or Roles (Section)
    - Transverse Modeling (Section)
    - CapellaProjectPredicate (Predicate)

Add...

Remove

Up

Down

Set the properties of "Predicate". Required fields are denoted by "\*\*".

class\*:

## The implementation class of the predicate

```
public class CapellaProjectPredicate implements IPredicate {

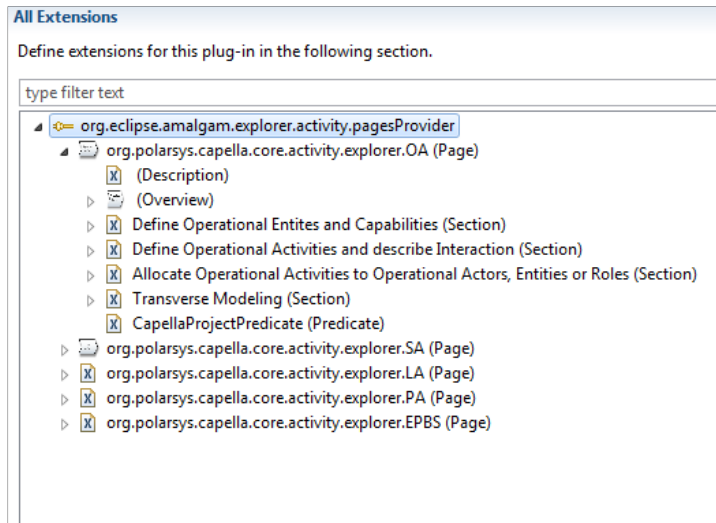
    public CapellaProjectPredicate() {
    }

    @Override
    public boolean isOk() {
        return ActivityExplorerManager.INSTANCE.getRootSemanticModel() != null &&
            ActivityExplorerManager.INSTANCE.getRootSemanticModel() instanceof Project;
    }

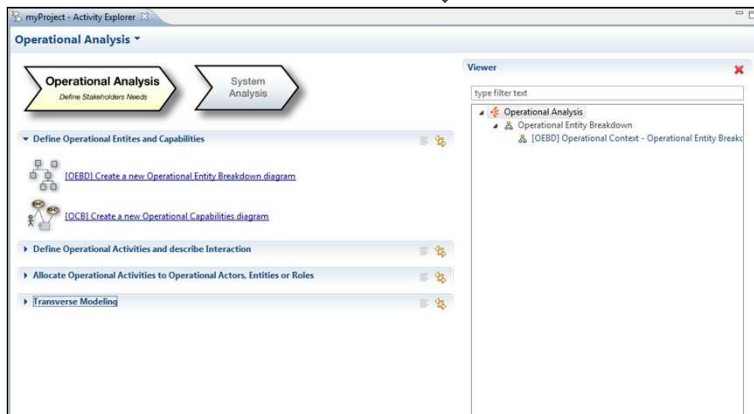
}
```



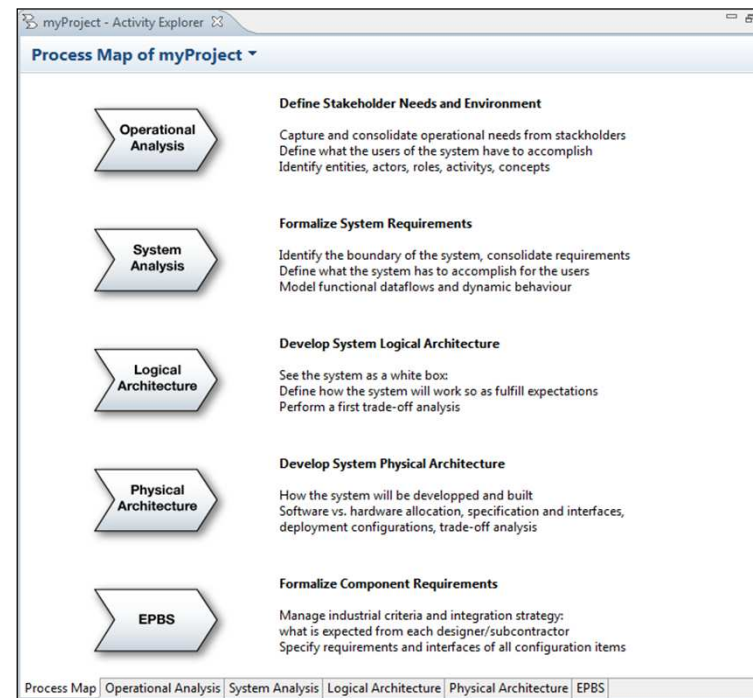
When the Activity Explorer is opened on not Capella project, the page will be never be visible. Do the same the same on other pages than Capella.



Final Extensions of Activity Explorer for Capella Example



The resulting Operational Analysis page



The resulting Process Map page