Activity Explorer in Eclipse Amalgam
Agenda

1. Introduction
2. User Perspective
3. Developer Perspective
Purpose

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 Page – Description

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

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 Page – Viewer – Example

Button to filter in the Viewer the artefacts matching all the activities of the section

Pattern to filter artefacts in the Viewer

Viewer to list the artefacts matching the activities and pattern
Activity Explorer – Activity Execution – Example 1/3

1. Activity Execution, here diagram creation

2. The diagram is created
Activity Explorer – Activity Execution – Example 2/3

The viewer is updated after the creation of the diagram.
Activity Explorer – Activity Execution – Example 3/3

1. **Operational Analysis**
   - Define Stakeholders Needs
2. **System Analysis**
   - Define Operational Entities and Capabilities
   - Define Operational Activities and describe Interactions
3. **Allocate Operational Activities to Operational Actors, Entities**
   - (A) Create a new Operational Architecture diagram
   - (B) Create a new Operational Role diagram
   - (C) Create a new Operational Entity Scenario
4. **Transverse Modeling**

---

**Actions available on the artefact type**

**Open action for navigation**

---

*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 Page – Description

**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
When the project session is opened, the Activity Explorer is opened on the overview page.
Preferences – Activity Explorer – Management

Allows to activate/deactivate pages, sections 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)
Objective

- Customizing the Activity Explorer with options

User Actions

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

1. Introduction
2. User Perspective
3. Developer Perspective
Foundations
Activity Explorer – Global Architecture & API

Activity Explorer component

Used components

Activity Explorer

Sirius

EMF

Eclipse (UI, Forms, SWT)
Activity Explorer workflow and extension points

Phases

1. Contributing with pages
   - Add a new page to the Activity Explorer

2. Contributing with sections
   - Add a new section to a page

3. Contributing with activities
   - Add a new Activity to a section

Pages Provider

- Add new pages to the Activity Explorer
- Specify if the Page is visible within the Activity Explorer Overview
- Ability to add Sections and Activities to the page

Sections Provider

- Add new Sections to existing page
- Specify if the Page is visible within page
- Ability to add Activities to the page

Activities Provider

- Add new Activities to existing sections
- Specify if the Activity is visible within the section

Extension points

- Activity Explorer workflow and extension points
- Contributing with pages
- Contributing with sections
- Contributing with activities
- Pages Provider
- Sections Provider
- Activities Provider
- Add new page to the Activity Explorer
- Add a new section to a page
- Add a new Activity to a section
- Add new pages to the Activity Explorer
- Add new Sections to existing page
- Add new Activities to existing sections
- Specify if the Page is visible within the Activity Explorer Overview
- Specify if the Page is visible within page
- Specify if the Activity is visible within the section
- Ability to add Sections and Activities to the page
- Ability to add Activities to the page
### Page Contribution

**• Implementation**

<table>
<thead>
<tr>
<th>Plugin name</th>
<th>org.eclipse.amalgam.explorer.activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java Package</td>
<td>org.eclipse.amalgam.explorer.activity.api.editor.pages</td>
</tr>
<tr>
<td>Class name</td>
<td>BasicSessionActivityExplorerPage</td>
</tr>
</tbody>
</table>

**• Extension point(s)**

<table>
<thead>
<tr>
<th>Name</th>
<th>Plugin name</th>
<th>Plugin</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>pagesProvider</td>
<td>org.eclipse.amalgam.explorer.activity</td>
<td>pagesProvider.exsd</td>
<td></td>
</tr>
</tbody>
</table>

**• Default implementation**

<table>
<thead>
<tr>
<th>Description</th>
<th>Empty page, it is used when contribution to the page without class implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java Package</td>
<td>org.eclipse.amalgam.explorer.activity.api.editor.pages</td>
</tr>
<tr>
<td>Class name</td>
<td>BasicSessionActivityExplorerPage</td>
</tr>
</tbody>
</table>
Activity Explorer Extension Point – Page (2/2)

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 page

Implementation class of the activity

Icon of the activity

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
Section Contribution

- **Implementation**

<table>
<thead>
<tr>
<th>Description</th>
<th>Add a section to a page. The section doesn’t need to provide class at the extension.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plugin name</td>
<td>org.eclipse.amalgam.explorer.activity</td>
</tr>
<tr>
<td>Java Package</td>
<td>org.eclipse.amalgam.explorer.activity.api.editor.sections</td>
</tr>
<tr>
<td>Class name</td>
<td>ActivityExplorerSection</td>
</tr>
</tbody>
</table>

- **Extension point(s)**

<table>
<thead>
<tr>
<th>Name</th>
<th>Plugin</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>sectionsProvider</td>
<td>org.eclipse.amalgam.explorer.activity</td>
<td>sectionsProvider.exsd</td>
</tr>
</tbody>
</table>
Activity Explorer Extension Point – Section (2/2)

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

org.eclipse.amalgam.explorer.activity.sectionsProvider

extension

point : String
id : String
name : String

SEQUENCE
Min Cardinity : 1
Max Cardinity : *

Section [1:1]

id : String
name : String
index : String
pageld : String
filtering : EBoolean
expanded : EBoolean

SEQUENCE
Min Cardinity : 1
Max Cardinality : *

Activity [0:*]

Description [1:1]
Activity Explorer Extension Point – Activity (1/2)

Activity Contribution

• Implementation

<table>
<thead>
<tr>
<th>Plugin name</th>
<th>org.eclipse.amalgam.explorer.activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java Package</td>
<td>org.eclipse.ui.forms.events</td>
</tr>
<tr>
<td>Interface name</td>
<td>org.eclipse.ui.forms.events.IHyperlinkListener</td>
</tr>
</tbody>
</table>

• Extension point(s)

<table>
<thead>
<tr>
<th>Name</th>
<th>Plugin</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>activitiesProvider</td>
<td>org.eclipse.amalgam.explorer.activity</td>
<td>activitiesProvider.exsd</td>
</tr>
</tbody>
</table>
Activity Explorer Extension Point – Activity (2/2)

- **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**

**Mandatory field**

```
org.eclipse.amalgam.explorer.activity.activitiesProvider
```

```
extension
point : String
id : String
name : String
```

- **SEQUENCE**
  - Min Cardinality : 1
  - Max Cardinality : *

```
Activity
id : String
name : String
index : String
class implements : org.eclipse.ui.forms.events.IHyperlinkListener
image : Resource
sectionId : String
```

```
Predicate
class implements : org.eclipse.amalgam.explorer.activity.api.editor.predicates.IPredicate
```

- **SEQUENCE**
  - Min Cardinality : 1
  - Max Cardinality : 1

```
Description
id : String
title : String
```

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 prior written permission of Thales

© THALES 2013 – All rights reserved.
Activity Explorer Extension Point – Predicate

• 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

<table>
<thead>
<tr>
<th>Plugin name</th>
<th>org.eclipse.amalgam.explorer.activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java Package</td>
<td>org.eclipse.amalgam.explorer.activity.api.editor.predicates</td>
</tr>
<tr>
<td>Interface name</td>
<td>org.eclipse.amalgam.explorer.activity.api.editor.predicates.IPredicate</td>
</tr>
</tbody>
</table>

• Extension point(s)

<table>
<thead>
<tr>
<th>Name</th>
<th>Plugin</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>PagesProvider</td>
<td>org.eclipse.amalgam.explorer.activity</td>
<td>pageProvider.exsd</td>
</tr>
</tbody>
</table>
Example
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 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
      - [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)

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>Capture and consolidate operational needs from stakeholders<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

```java
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

```java
public class OEBD extends AbstractCapellaNewDiagram {
    public OEBD() {
        /* Get the right level in Capella project where create the element */
        super(ModelQueryHelper.getOperationalContext(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)

```java
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";
    }
}
```

```java
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)

```java
public class OAIB extends AbstractCapellaNewDiagram {
    public OAIB () {
        super(ModelQueryHelper.getRootOperationalActivity((Project) ActivityExplorerManager.INSTANCE.getRootSemanticModel()));
    }

    @Override
    public String getRepresentationName() {
        return "Operational Capabilities Blank";
    }
}
```

```java
public class OAS extends AbstractCapellaNewDiagram {
    public OAS () {
        super(ActivityExplorerManager.INSTANCE.getRootSemanticModel());
    }

    @Override
    public String getRepresentationName() {
        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)

```java
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";
    }
}
```

```java
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)

```java
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);
        }
    }
}
```

```java
public class CDB extends AbstractCapellaNewDiagram {
    public CDB () {
        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)

```java
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.selectOperationalAndCreateStateMachineRegion((Project) root_p);
        if (!createDiagram(root_p, session_p)) {
            handleDiagramCreationError(event_p, root_p);
        }
    }
}
```

```java
public class CreateMatix extends AbstractCapellaNewDiagram {
    public CreateMatix () {
        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

The implementation class of the predicate

```java
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 on other pages than Capella.
Final Extensions of Activity Explorer for Capella Example

The resulting Operational Analysis page

The resulting Process Map page