Contributive XHTML DocGen

Kitalpha – Capella Studio

THALES

Agenda

Contributive XHTML DocGen

- General Concept
- Contributive XHTML DocGen for Ecore
- Contributive XHTML DocGen for Capella
- Contributive XHTML with Extension point
- Types of competition when multiples contributions
- Pattern substitution principle

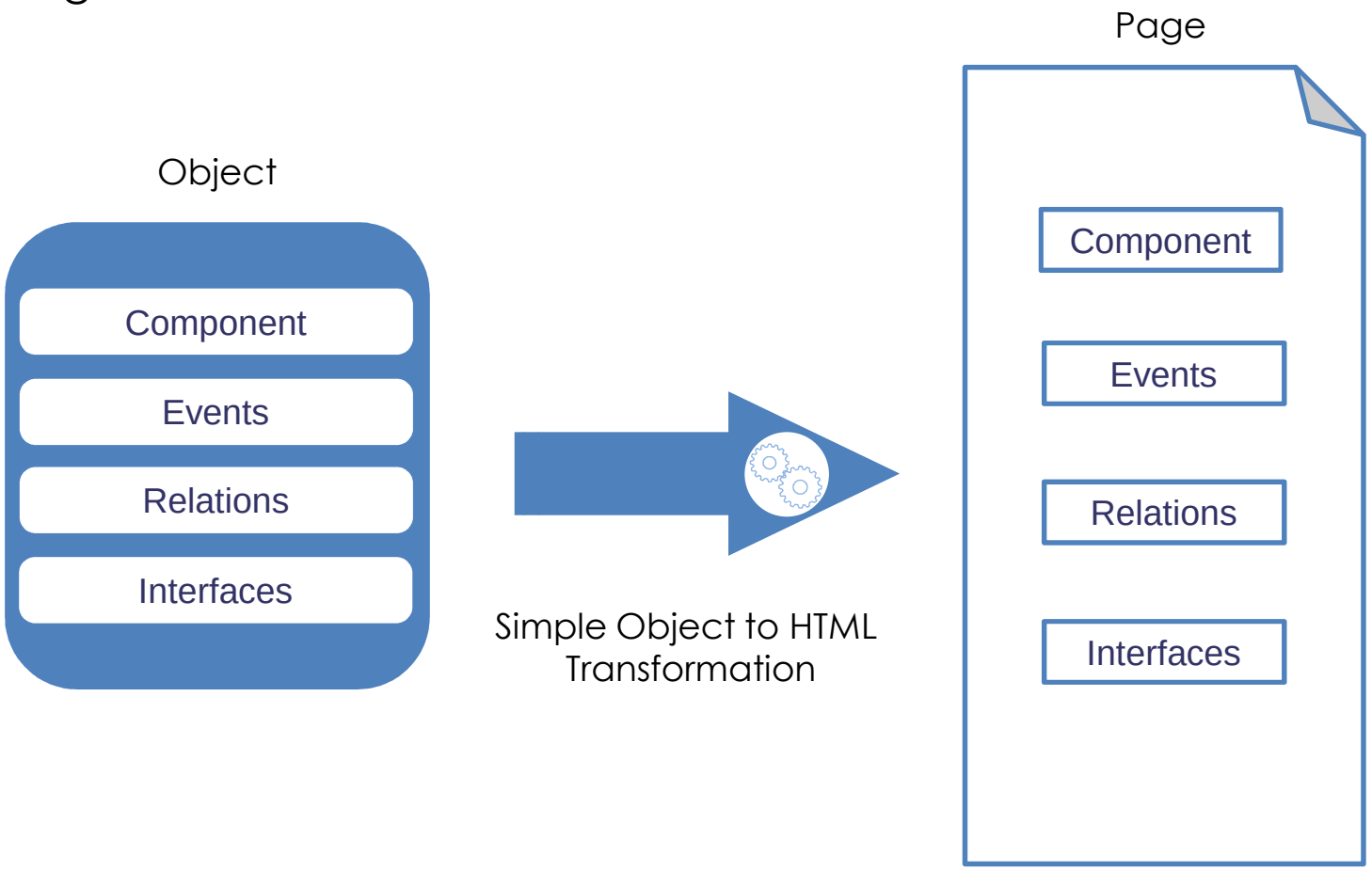
www.thalesgroup.com

General Concept

OPEN

THALES

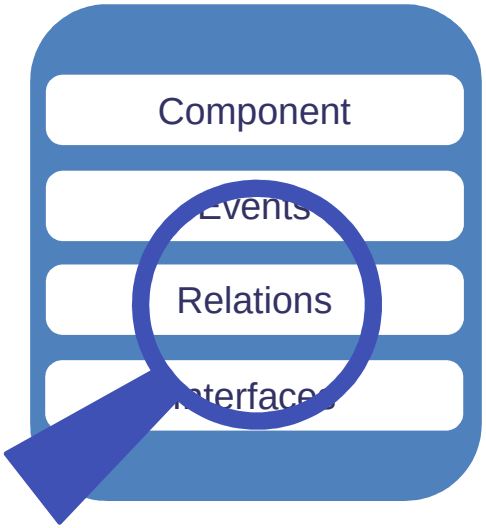
XHTML Page Structure



Piece of XHTML Page

XHTML Page Structure

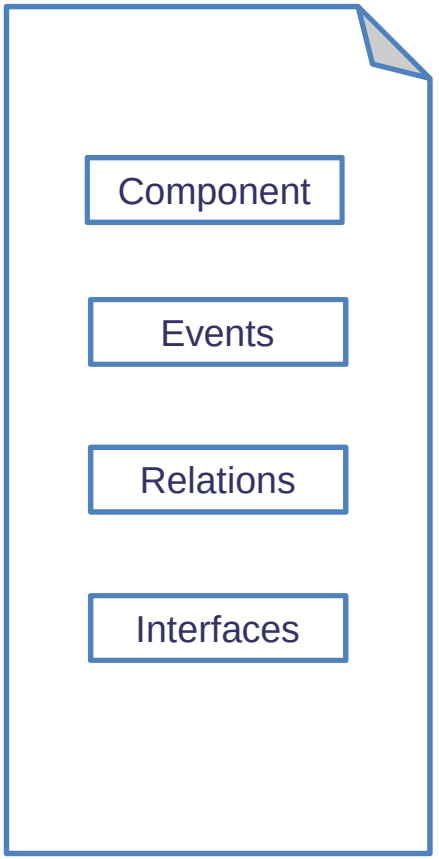
Object



How to split Required & Provided References in generated page

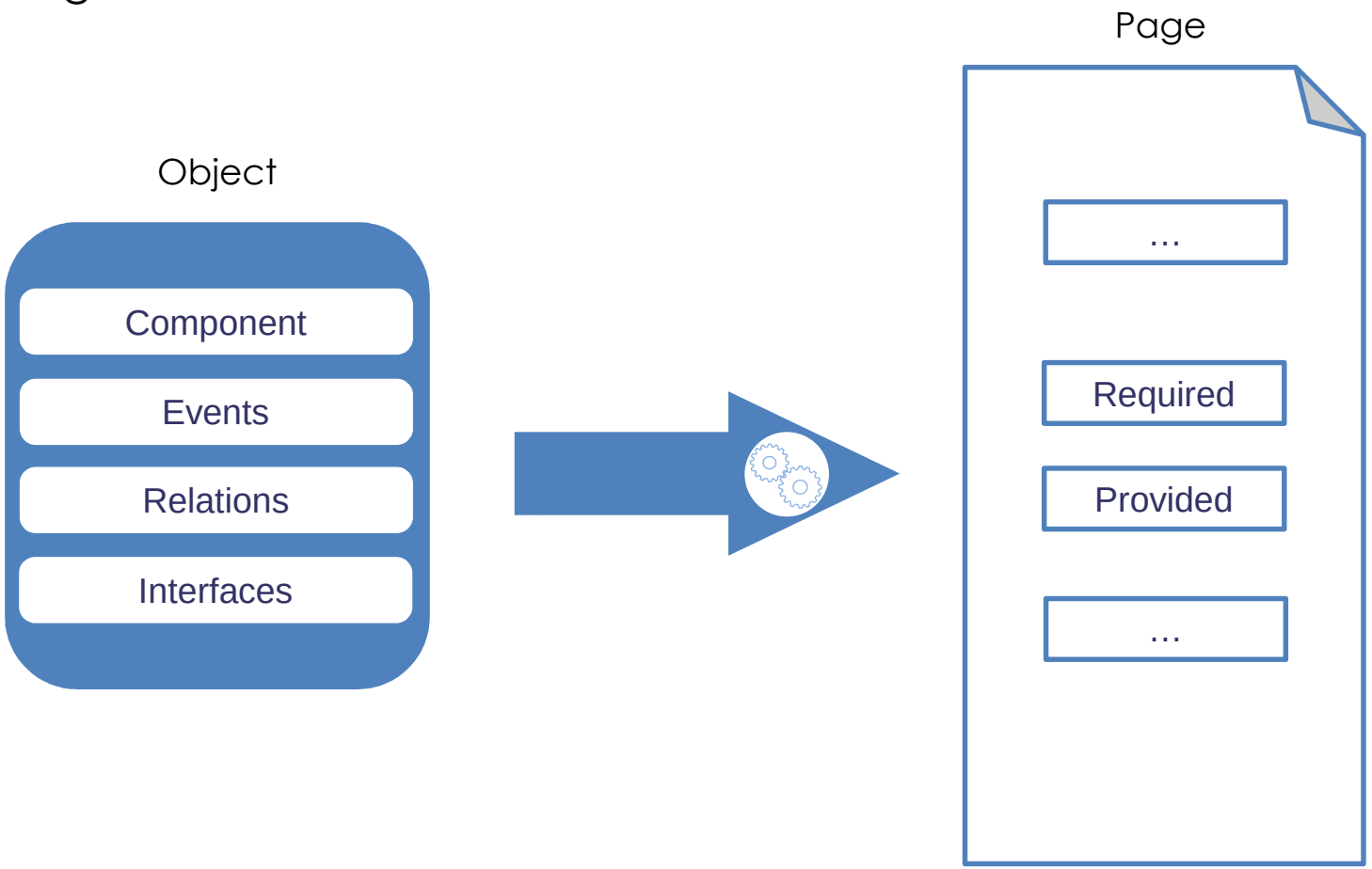
?

Page



Piece of XHTML Page

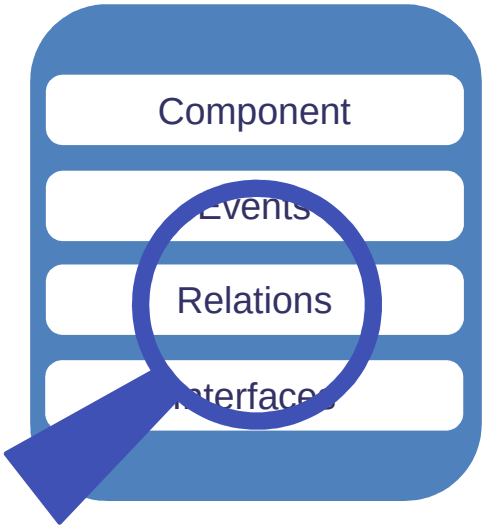
XHTML Page Structure



Piece of XHTML Page

XHTML Page Structure

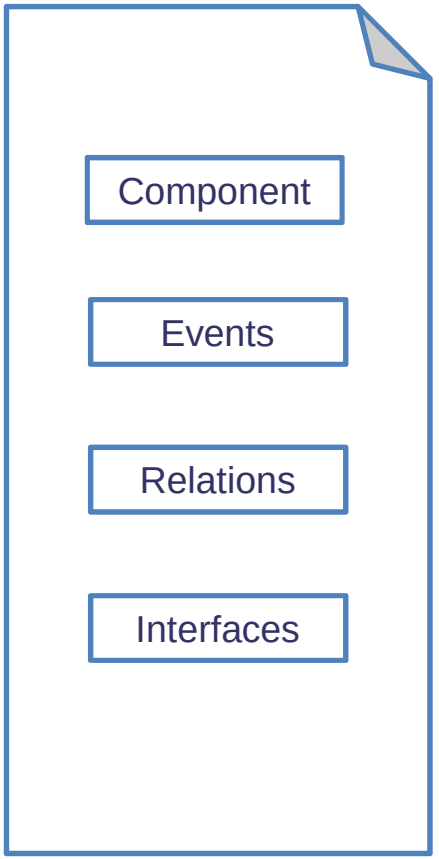
Object



How to add information around relations in generated page

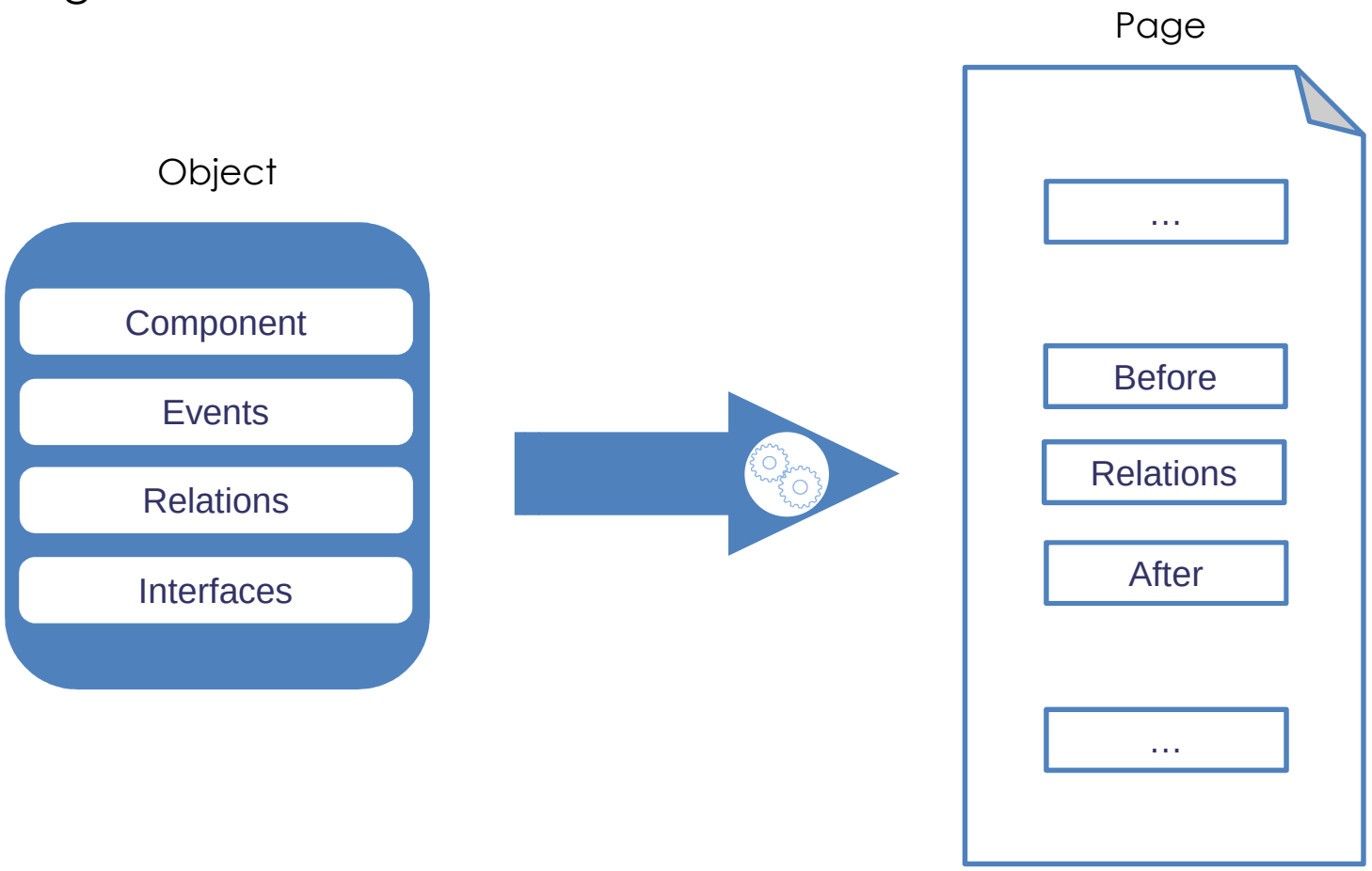
?

Page



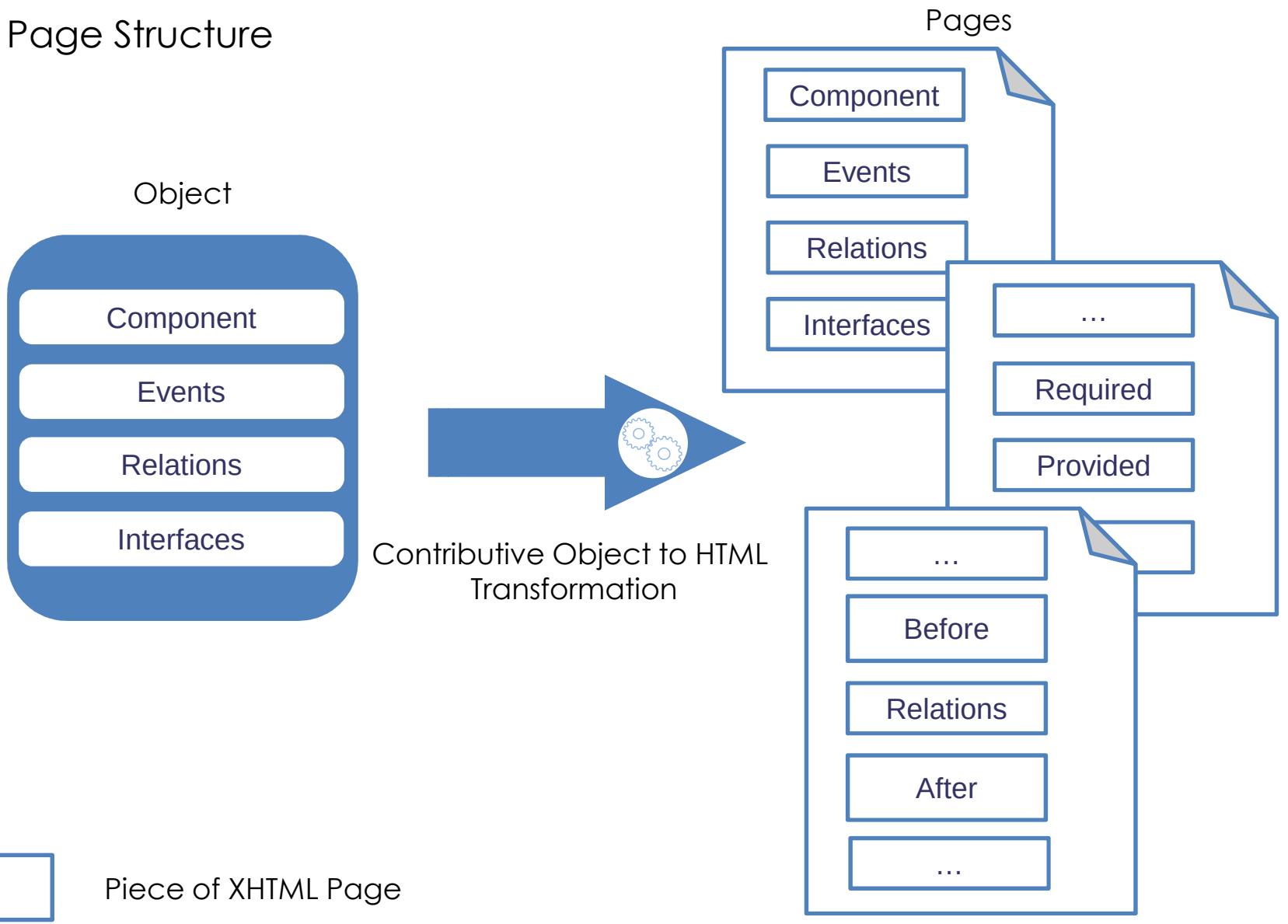
Piece of XHTML Page

XHTML Page Structure



Piece of XHTML Page

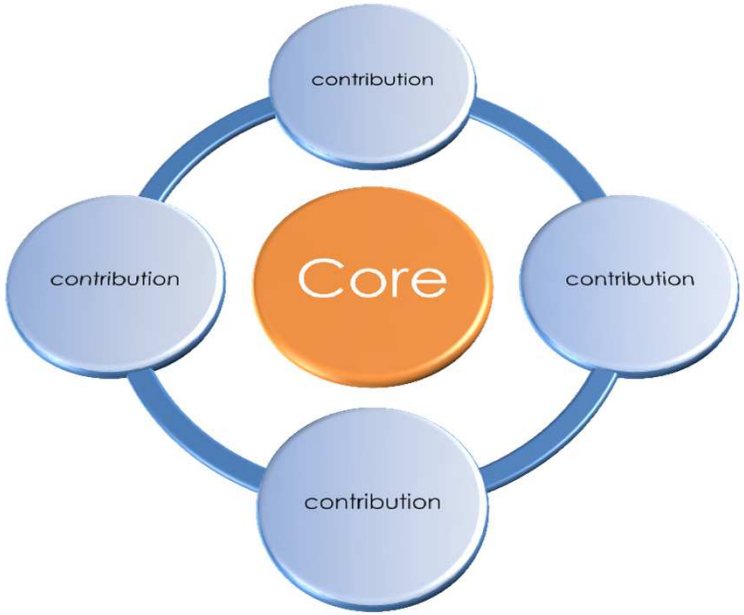
XHTML Page Structure



Introduction – contribution mechanism

Contribution

- Customize functionalities
- Remove functionalities



Contributive XHTML DocGen for Ecore

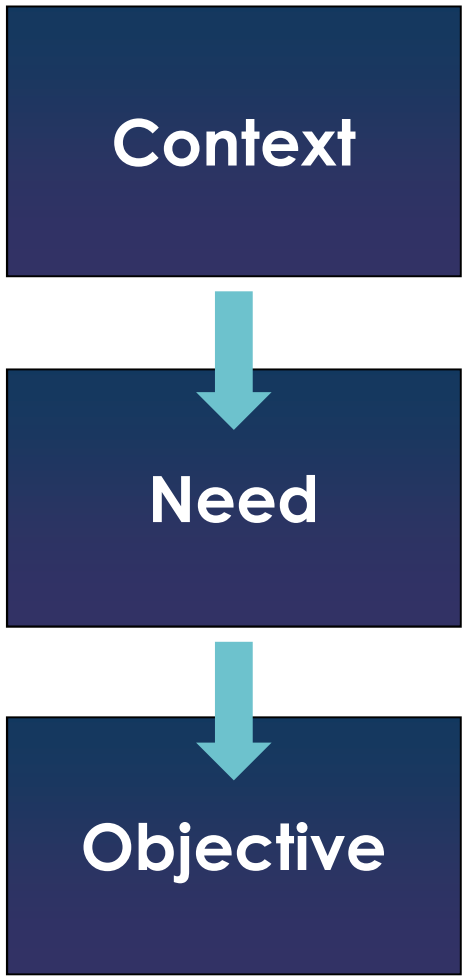
THALES

Agenda

Contributive XHTML DocGen for Ecore

- Purpose
- Concept – Contributive XHTML DocGen based Pattern call
- Example: EDataTypes & EClass EAnnotation

Ecore XHTML DocGen - Purpose



- Models are complex and Basic documentation is insufficient
- Existing Ecore XHTML DocGen is not contributive

- Need to extend XHTML DocGen for customizing and adding or remove information in generated documentation from models

- Rework existing Ecore XHTML DocGen framework to be contributive

Ecore XHTML DocGen - Contributive XHTML DocGen based Pattern calls

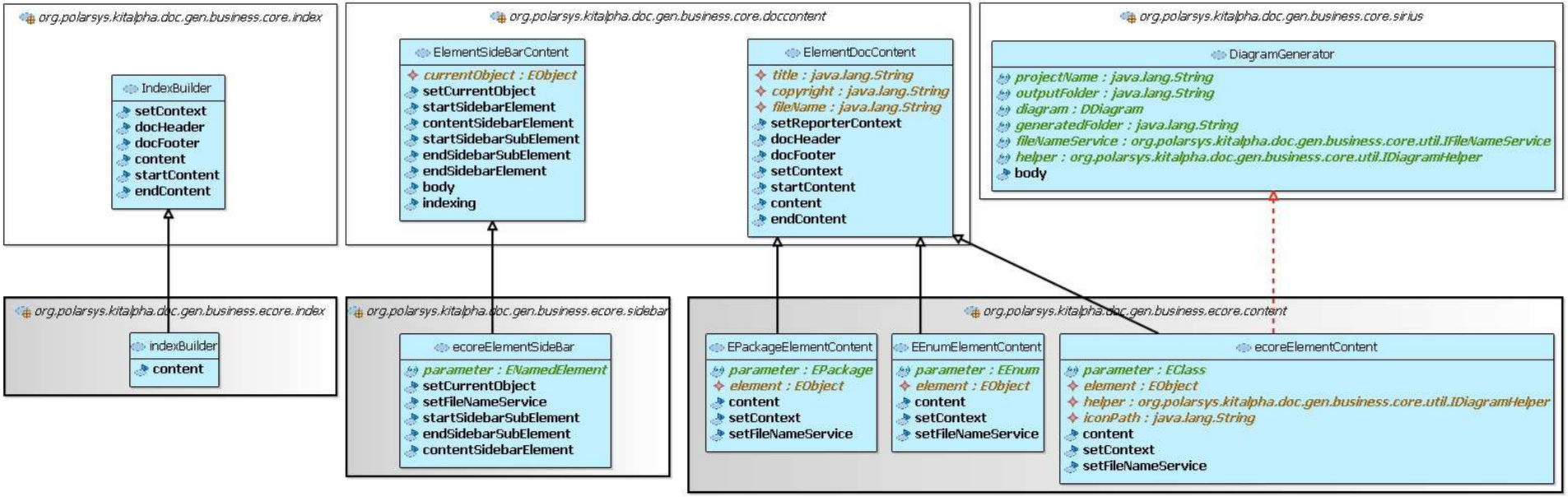
Objective

- Rework existing Ecore XHTML DocGen framework to be contributive by pattern calls and substitution mechanism

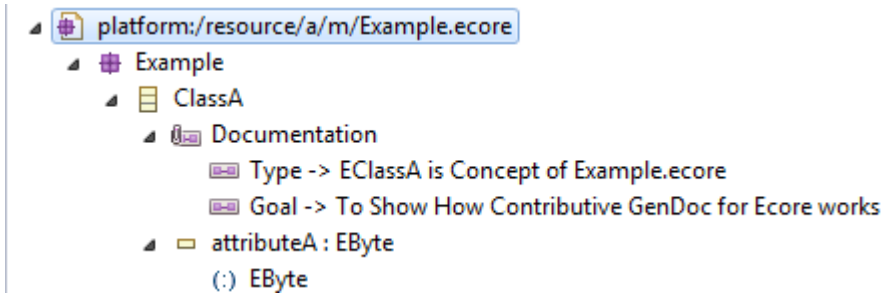
Action

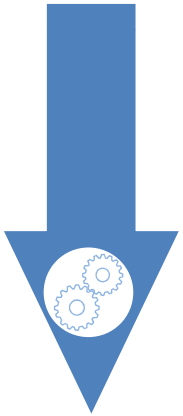
- Rework existing Ecore XHTML DocGen framework to add pattern call and substitution mechanism

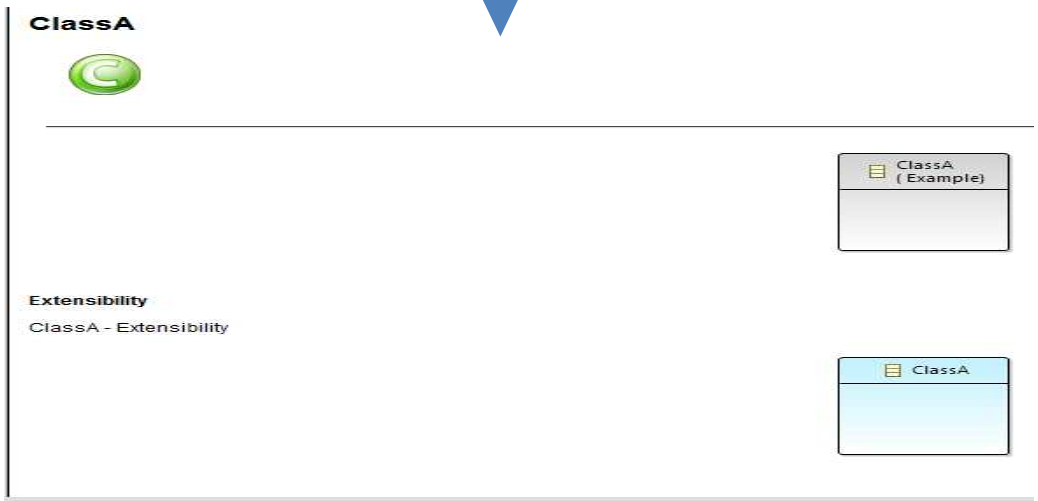
Ecore Generation Pattern



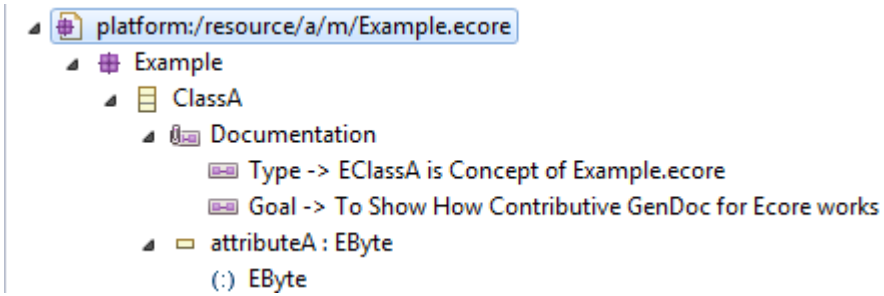
Introduction – Example without contribution

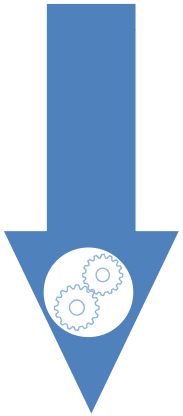






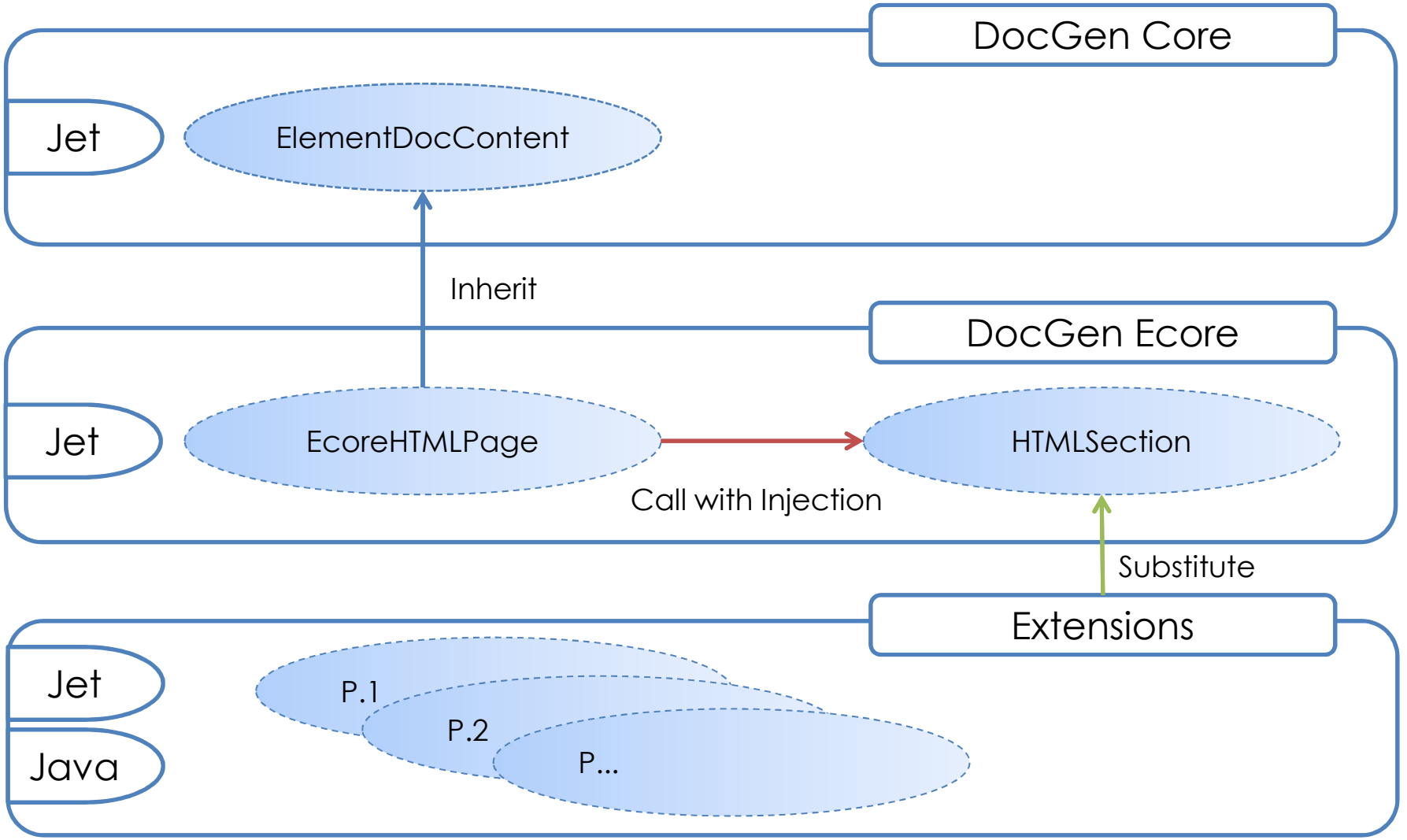
Introduction – Example with contribution





THALES

Concept - Contributive XHTML DocGen based Pattern call



Implementation - Contributive XHTML DocGen based Pattern call

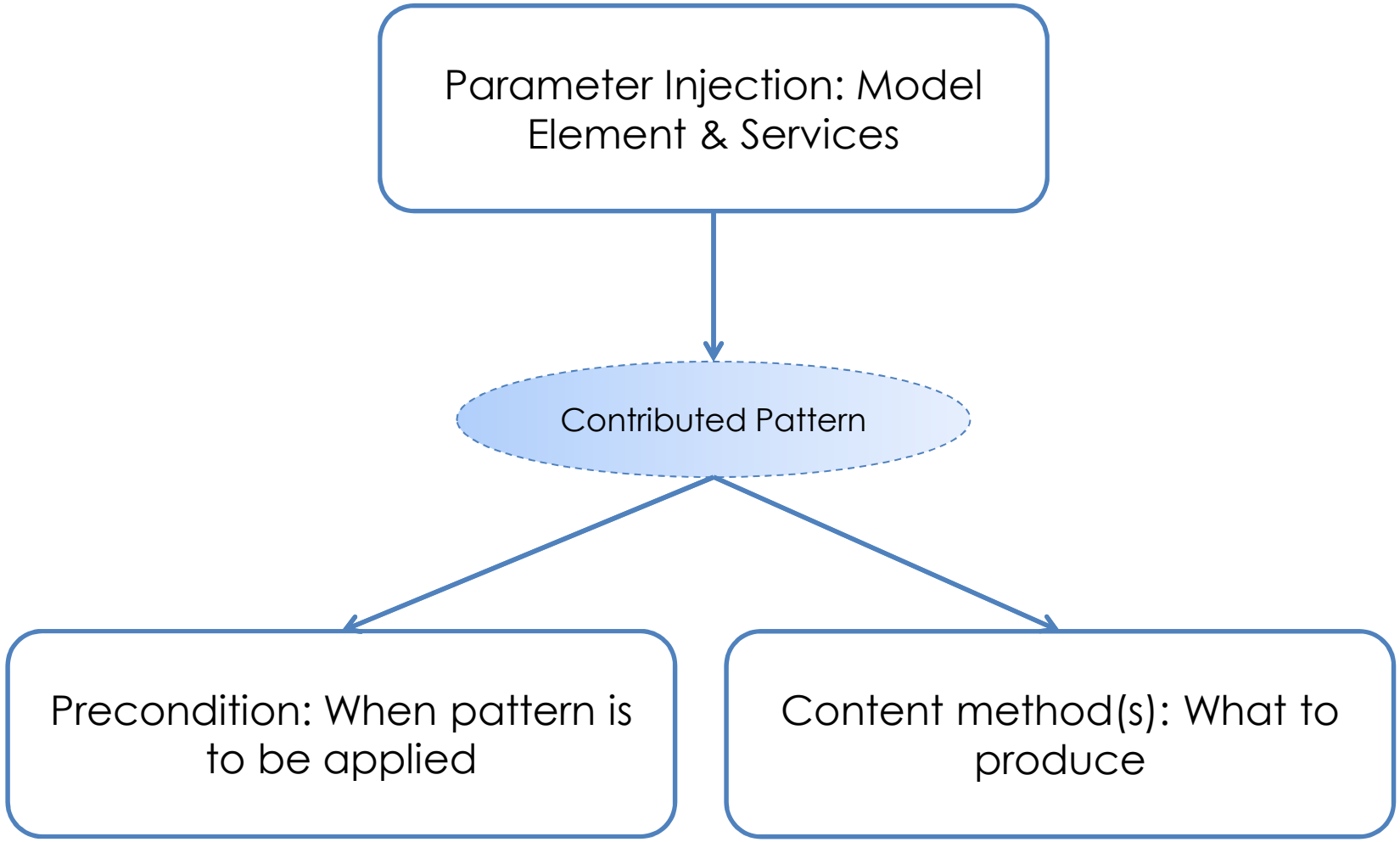
EcoreHTMLPage

- Contribute DocGen Core
- Call "HTMLSection" pattern
- Work on EPackages, EClasses and EDatatypes
- Inject the current model element to "EcoreHTMLPattern"

HTMLSection

- Empty pattern
- Will be substituted by the contributions

Implementation - Contributive XHTML DocGen based Pattern call



Implementation - Contributive XHTML DocGen based Pattern call

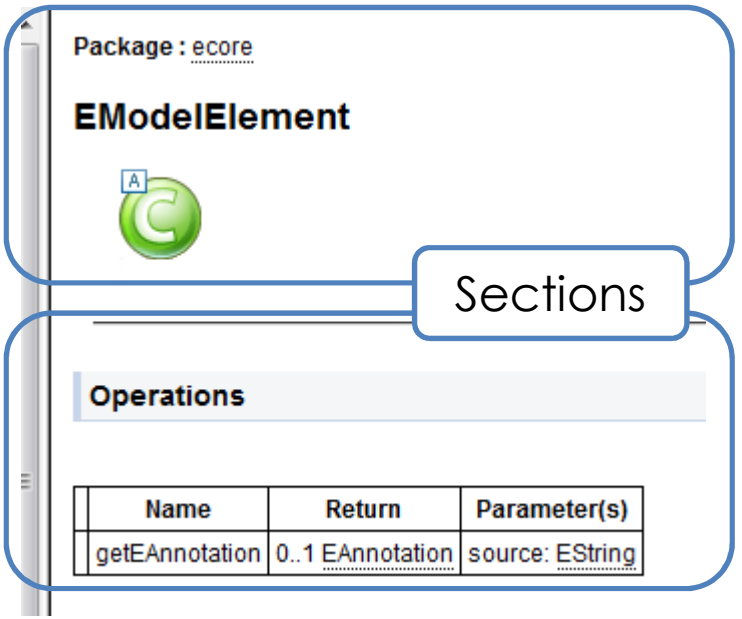
Parameter Injection

- Injection is done in Pattern parameters
- Available queries
  - Ecore DocGen Injected Element: To retrieve the model element
  - Inject Diagram Helper: Inject the diagram helper to manage relations between representation and semantic element
  - Inject File Name Service: Allows to compute file names and links

Ecore DocGen section patterns

Ecore DocGen Patterns

- **EPackage**
  - EPackageHTMLSectionHeader
  - EPackageHTMLSubPackagesSection
  - EPackageDataTypeHTMLSection
- **EClass**
  - EClassHTMLHeader
  - EClassHTMLDescription
  - EClassFocusDiagram
  - EClassSuperType
  - EClassAttributes
  - EClassOperations
  - EClassReferences
  - EClassRepresentation
- **Enumeration**
  - EEnumHTMLHeader
  - EEnumHTMLContent



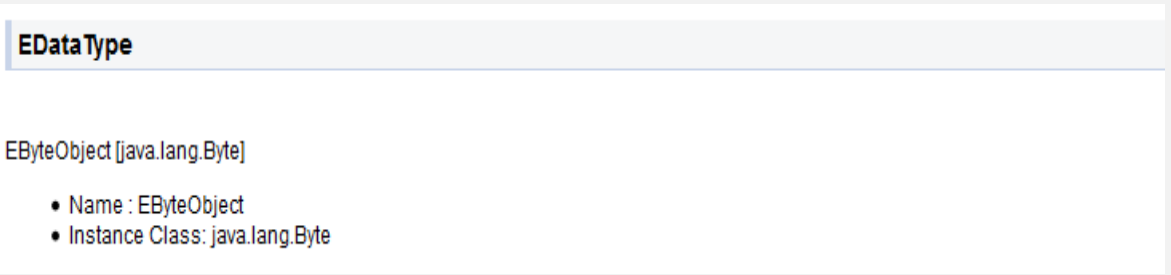
Examples

Example – EDataType

Objective

- Add EDataType page creation by JET Pattern nature
- Format:
  - EDataType Name [EDataType Instance Class]
    - Name: EDataType Name
    - Instance Class: Instance Class Name

Result

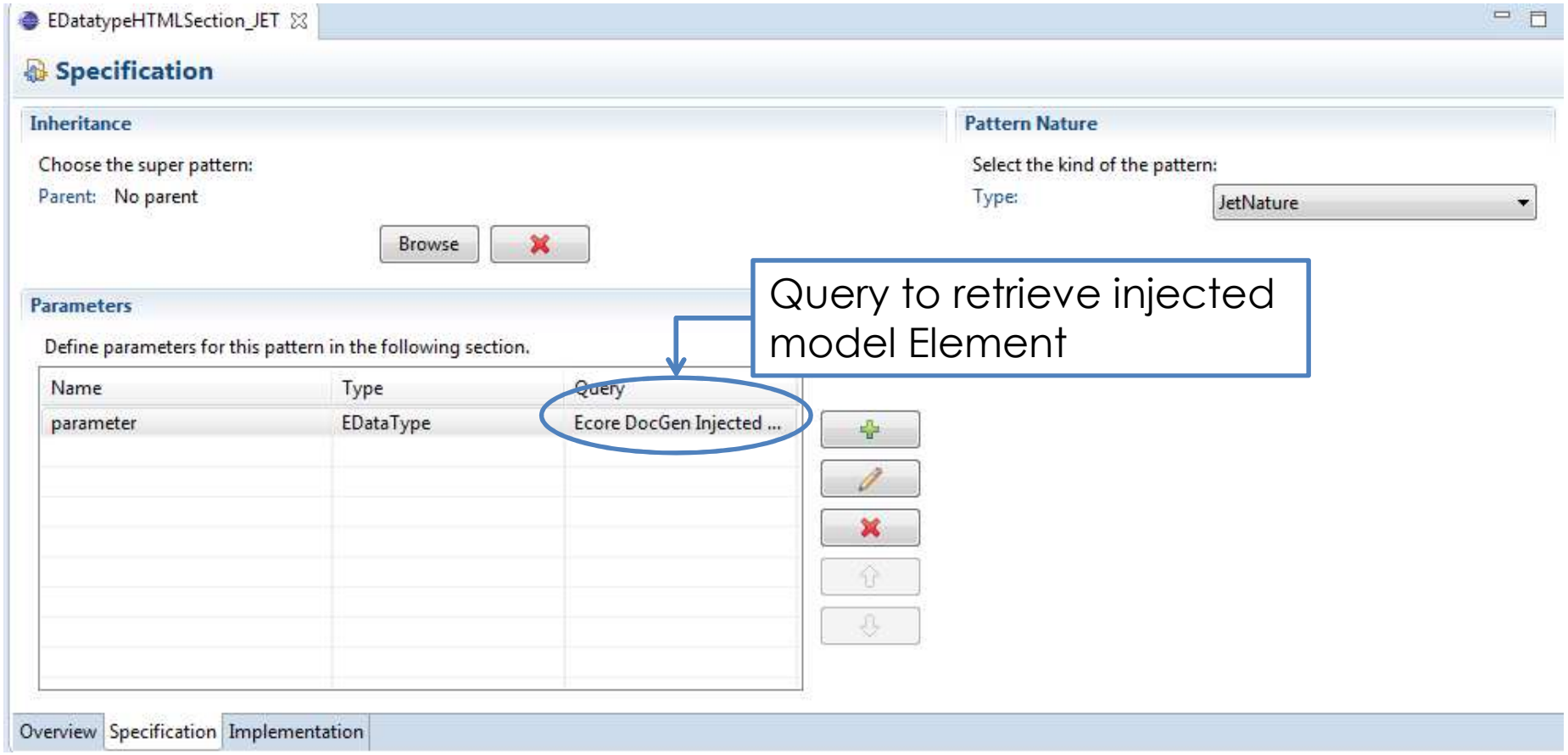


Example – EDataType Steps contribution

- Create a new EGF Factory Component
- Create a Viewpoint Container
- Create a Pattern Viewpoint
- Create a Pattern Library
- Create a new JET or Java Pattern (e.g., EDataTypeSection_JET)
- Create a method(s) (e.g., content) and add it to the orchestration
- Implement the business of the method
- Add preCondition to the pattern to be applied only on EDataTypes
- Add Product Plan to Factory Component
- Invoque EcoreDocGenLauncher
  - Fill invocation contracts
    - ecoreFilePath
    - outputDirectoryPath
    - pattern.substitutions
      - Substitute « HTMLSection » pattern by: EDataTypeSection_JET and HTMLSection
- Run the Factory Component

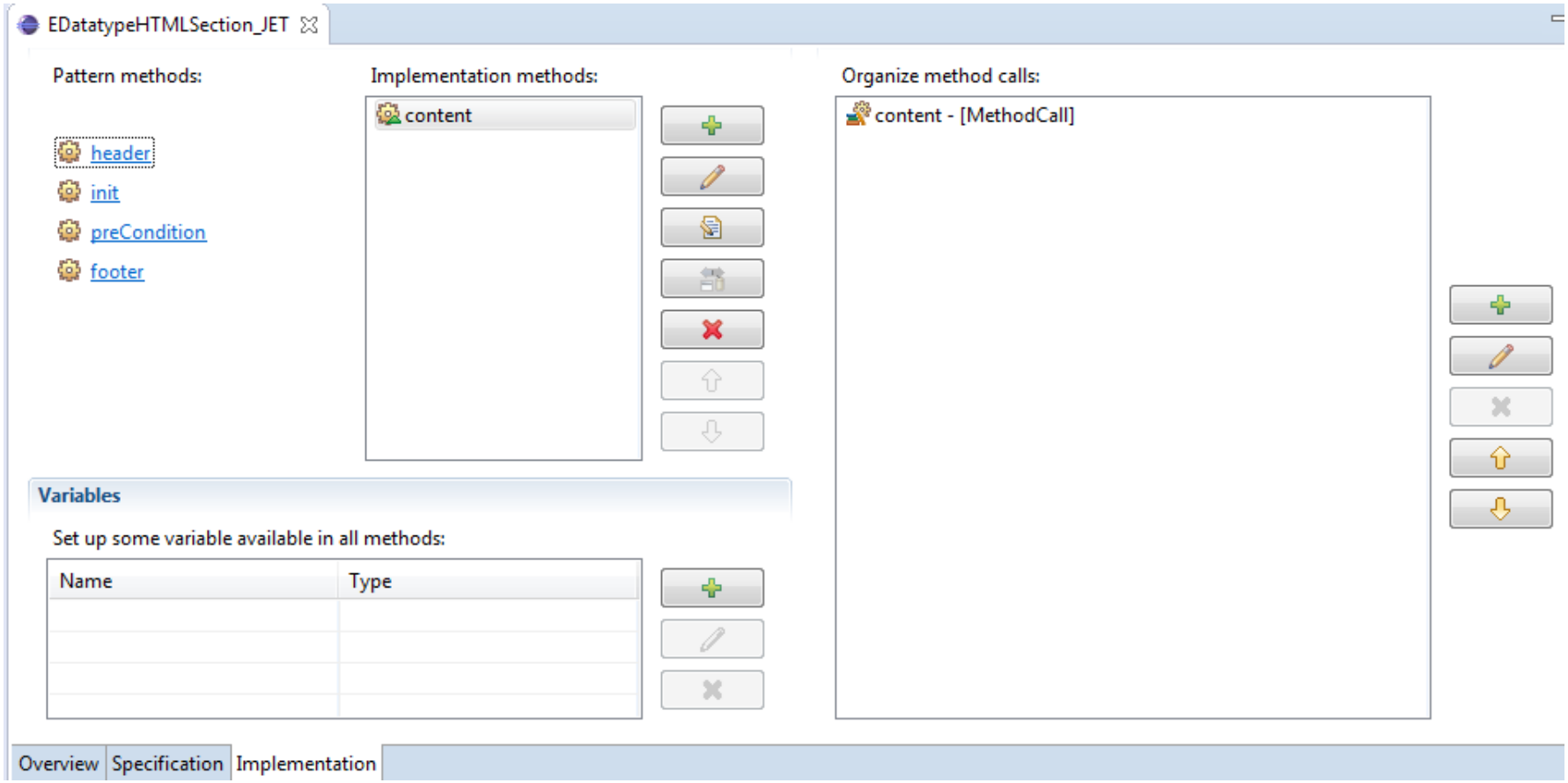
Example – EDataTypeHTMLSection Specification

EDatatypeHTMLSection_JET



Example – EDataTypeHTMLSection Implementation

EDatatypeHTMLSection_JET



2 Example – EDataTypeHTMLSection content and preCondition methods

Content method

EDatatypeHTMLSection_JET

```
<%
String text = LabelProviderHelper.getText(parameter);
%>

<% if (text != null && !text.isEmpty()) { %>
<h2>EDataType</h2>
<%= text %>
<br/>
<ul>
    <li>Name : <%= parameter.getName() %></li>
    <li>Instance Class: <%= parameter.getInstanceClass().getCanonicalName() %></li>
</ul>
<%}%>
```

header | init | preCondition | content | footer

Model Element from pattern parameter

preCondition method

EDatatypeHTMLSection_JET

```
return (parameter instanceof EEnum);
```

header | init | preCondition | content | footer

Example – EClass annotations

Objective

- Add "Class Annotations" section to EClass pages by Java Pattern nature
- Format:
  - Source: annotation source

Key 1	Value 1
Key 2	Value 2

Result

Class Annotations

Source: *http://www.eclipse.org/emf/2002/Ecore*

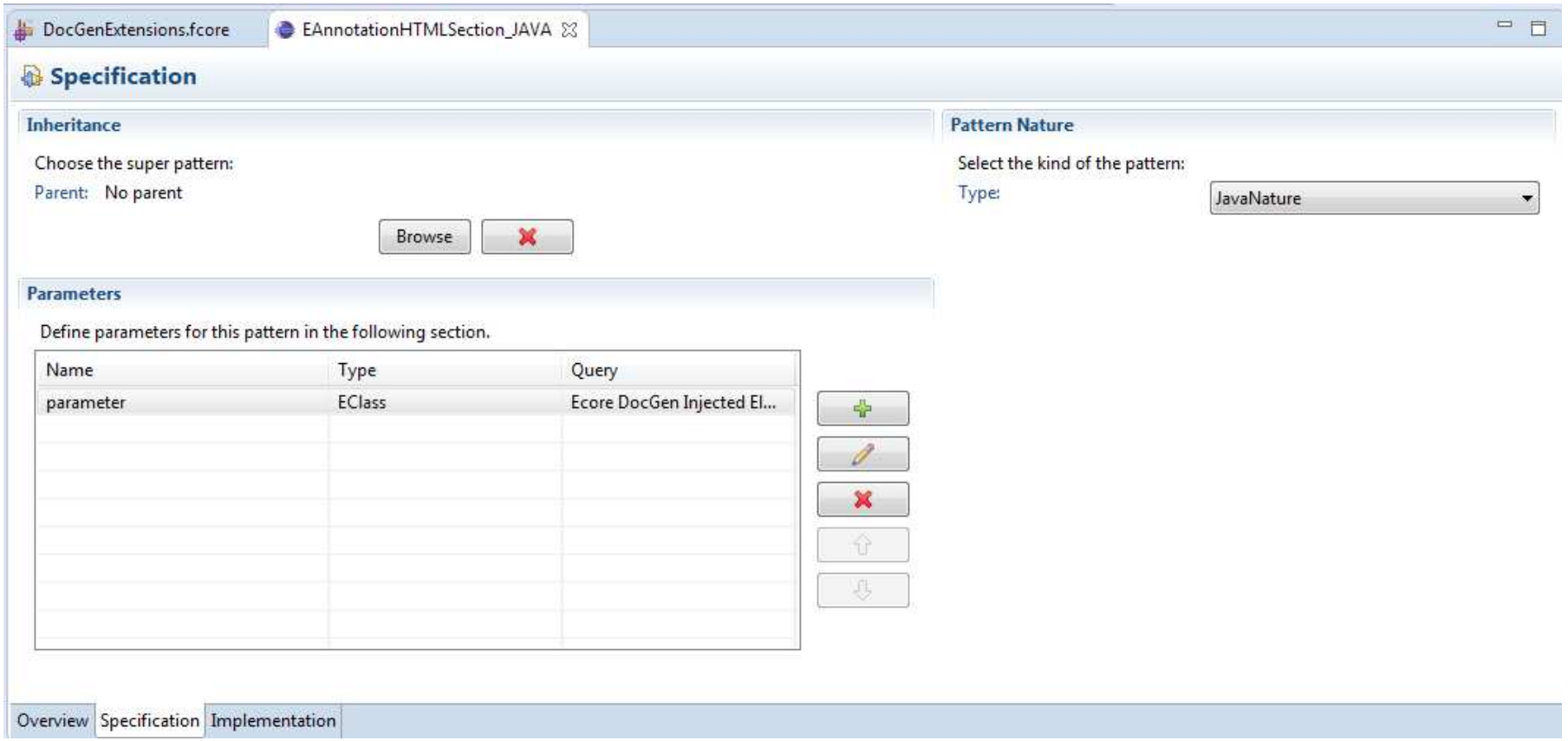
constraints	WellFormedName

Example – EClass Annotation Steps contribution

- Steps are the same as in EDataType contribution example
- Name of pattern: EAnnotationHTMLSection_JAVA
- Add substition pattern in factory component:
  - Substitute « HTMLSection » pattern by: EAnnotationHTMLPattern_JAVA and HTMLSection patterns

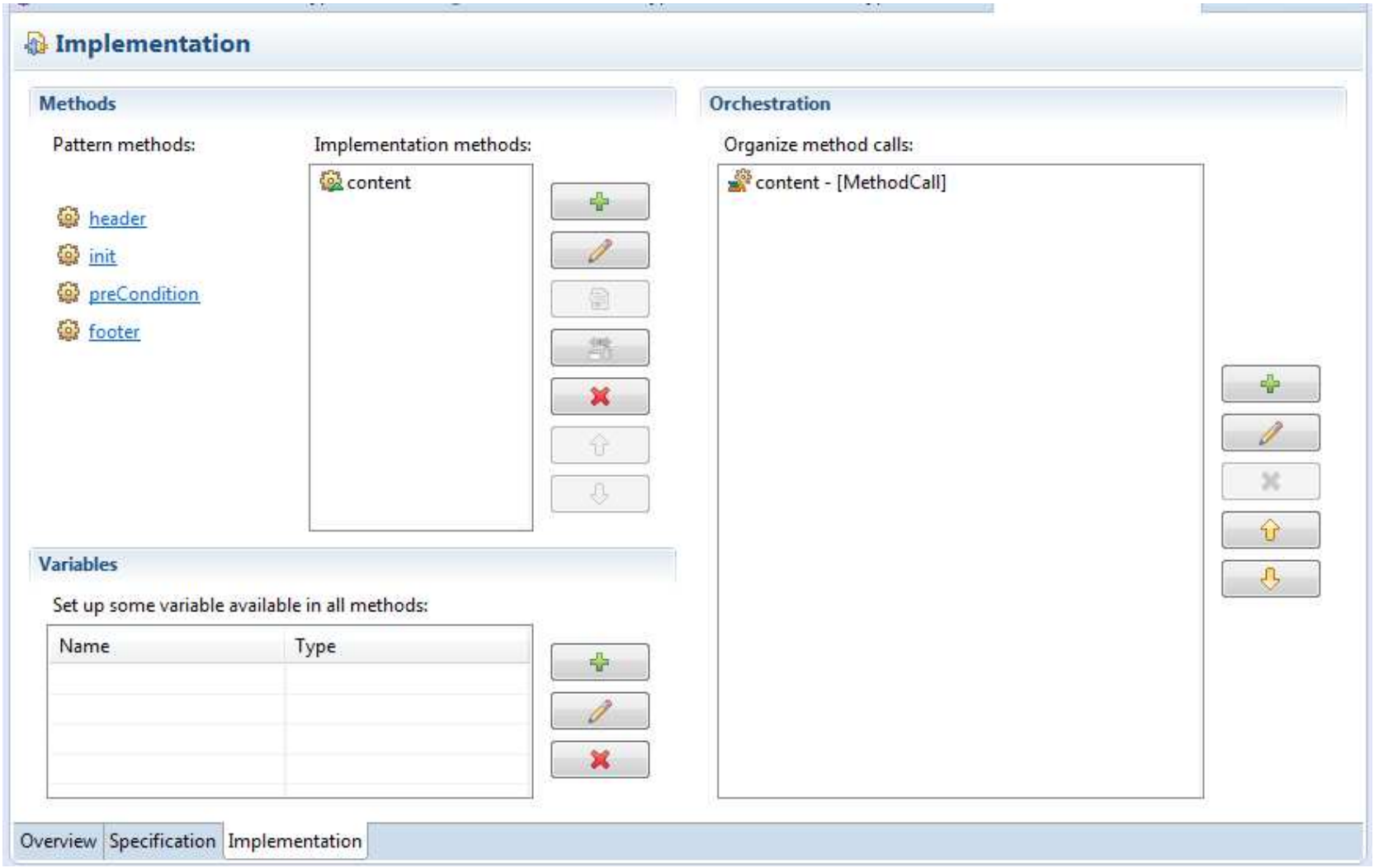
Example – EClass annotations Specification

EAnnotationHTMLSection_Java



Example – EClass annotations Implementation

EAnnotationHTMLSection_Java



Example – EAnnotationHTMLSection content and preCondition methods

Content method

DocGenExtensions.fcore | EAnnotationHTMLSection_JAVA | EAnnotationHTMLSection_JAVA

```
1 EList<EAnnotation> eAnnotations = parameter.getEAnnotations();
2
3 out.append("<h2>Class Annotations</h2>");
4 for (EAnnotation eAnnotation : eAnnotations) {
5
6     EMap<String, String> details = eAnnotation.getDetails();
7     String source = eAnnotation.getSource();
8
9     out.append("<b>Source: <i>").append(source).append("</i></b>");
10    out.append("<table>");
11    for (Entry<String, String> entry : details) {
12        out.append("<tr>");
13        out.append("<td>").append(entry.getKey()).append("</td><td>").append(entry.getValue()).append("</td>");
14        out.append("</tr>");
15    }
16    out.append("</table>");
17 }
```

header | init | preCondition | content | footer

preCondition method

DocGenExtensions.fcore | EAnnotationHTMLSection_JAVA | EAnnotationHTMLSection_JAVA

```
1 return !(parameter.getEAnnotations().isEmpty());
```

header | init | preCondition | content | footer

Contributive XHTML DocGen for Capella

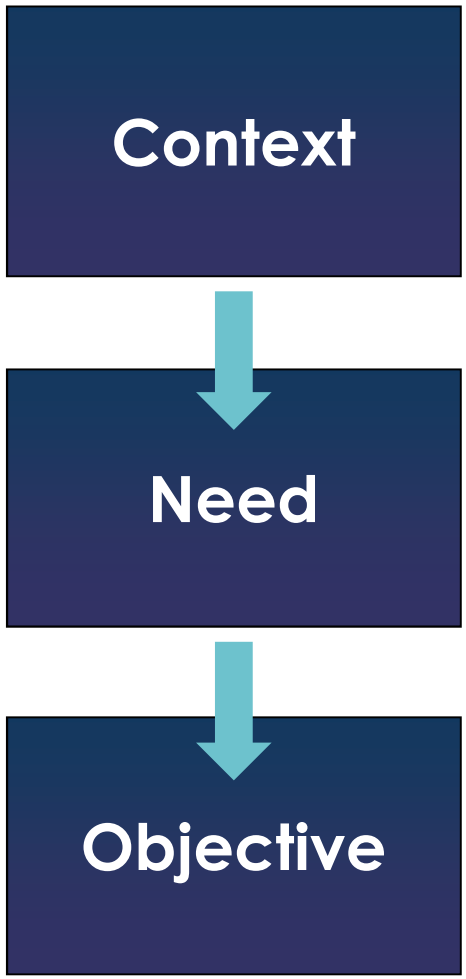
THALES

Agenda

Contributive XHTML DocGen for Capella

- Purpose
- Introduction
- Concept – Contributive Capella XHTML DocGen
- Contributive XHTML with Extension point
- Example: Basic Viewpoints

Purpose



- Capella Models are complex, extensible and Basic documentation is insufficient
- Existing Capella XHTML DocGen is not contributive

- Need to extend Capella XHTML DocGen for customizing, adding or remove information in generated documentation from models

- Allowing selection of multiple extensions when launching a documentation generation

Purpose

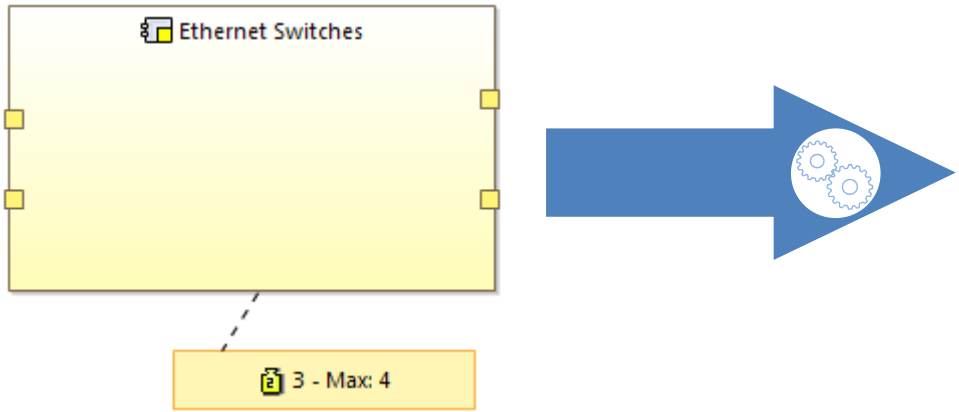
Objective

- Make Capella XHTML DocGen to be contributive by pattern calls and substitution mechanism

Action

- Adding pattern calls and patterns substitutions to make the Capella Docgen Contributive

Introduction – Example without contribution



No description.

Ports

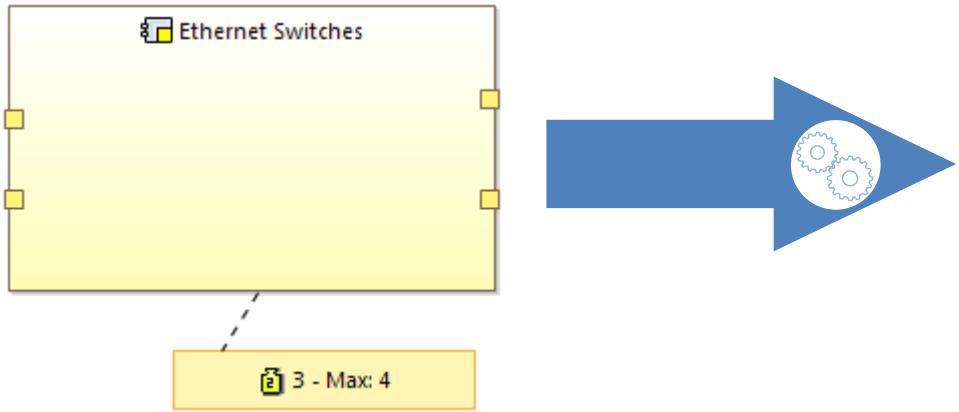
- PP2
- PP 4
- PP 5
- PP 7

Diagrams displaying "Ethernet Switches"

- [PCBD] Implementation Components
- [EAB] Configuration Items and Realized Artefacts
- [PAB] [BUILD] All PCs, PFs, FEs
- [PAB] Focus on Network Setup, Configuration and Tests
- [PAB] Implementation and Behaviour Components
- [PAB] [BUILD] Template
- [PAB] Focus on Network Transport

Page without contributions

Introduction – Example with contribution



PP 5
PP 7

Diagrams displaying "Ethernet Switches"

- [PAB] Focus on Network Setup, Configuration and Tests
- [EAB] Configuration Items and Realized Artefacts
- [PAB] [BUILD] All PCs, PFs, FEs
- [PCBD] Implementation Components
- [PAB] [BUILD] Template
- [PAB] Focus on Network Transport
- [PAB] Implementation and Behaviour Components

Mass Viewpoint

Current mass: 3
Max mass: 2;

Price Viewpoint

Price: 18
Max price: 30

Page with viewpoint contributions

Introduction – Extensions with Substitution

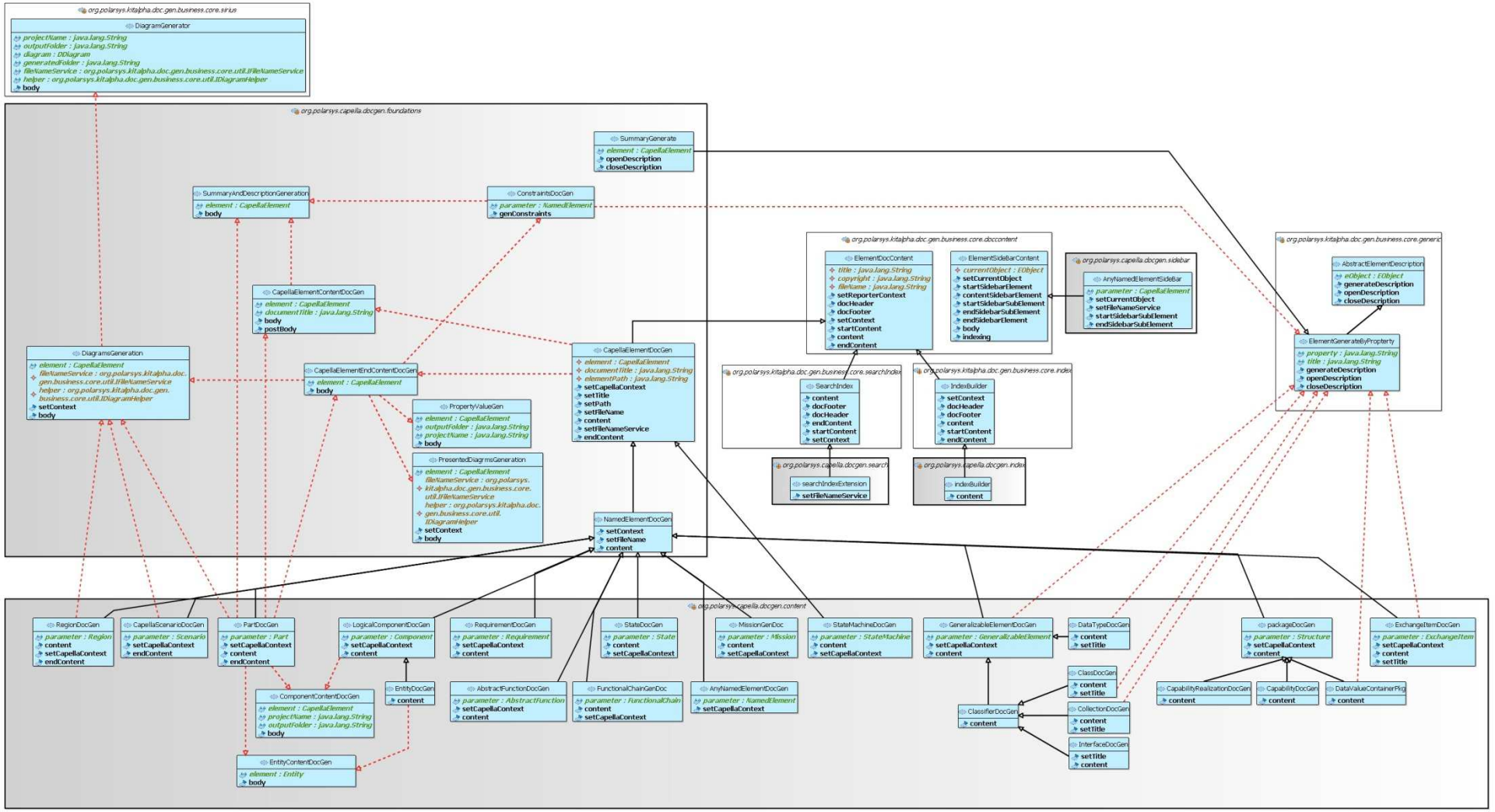
Mechanism

Definition of Pattern extension points where contributions can come with their functionalities

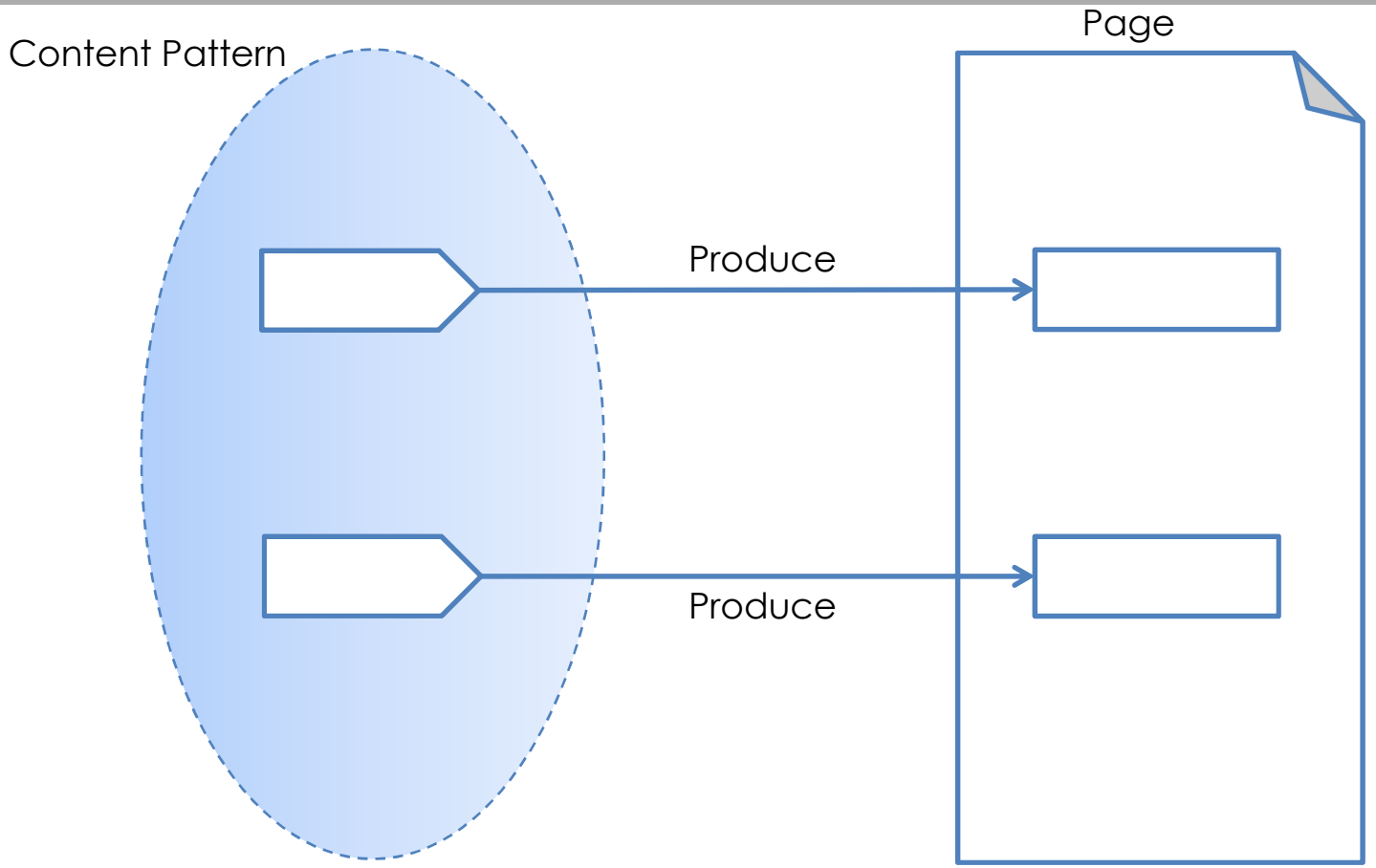
Extensions which aimed to add, customize, replace or extends functionalities

Pattern Extension Point

Concept – Introduction



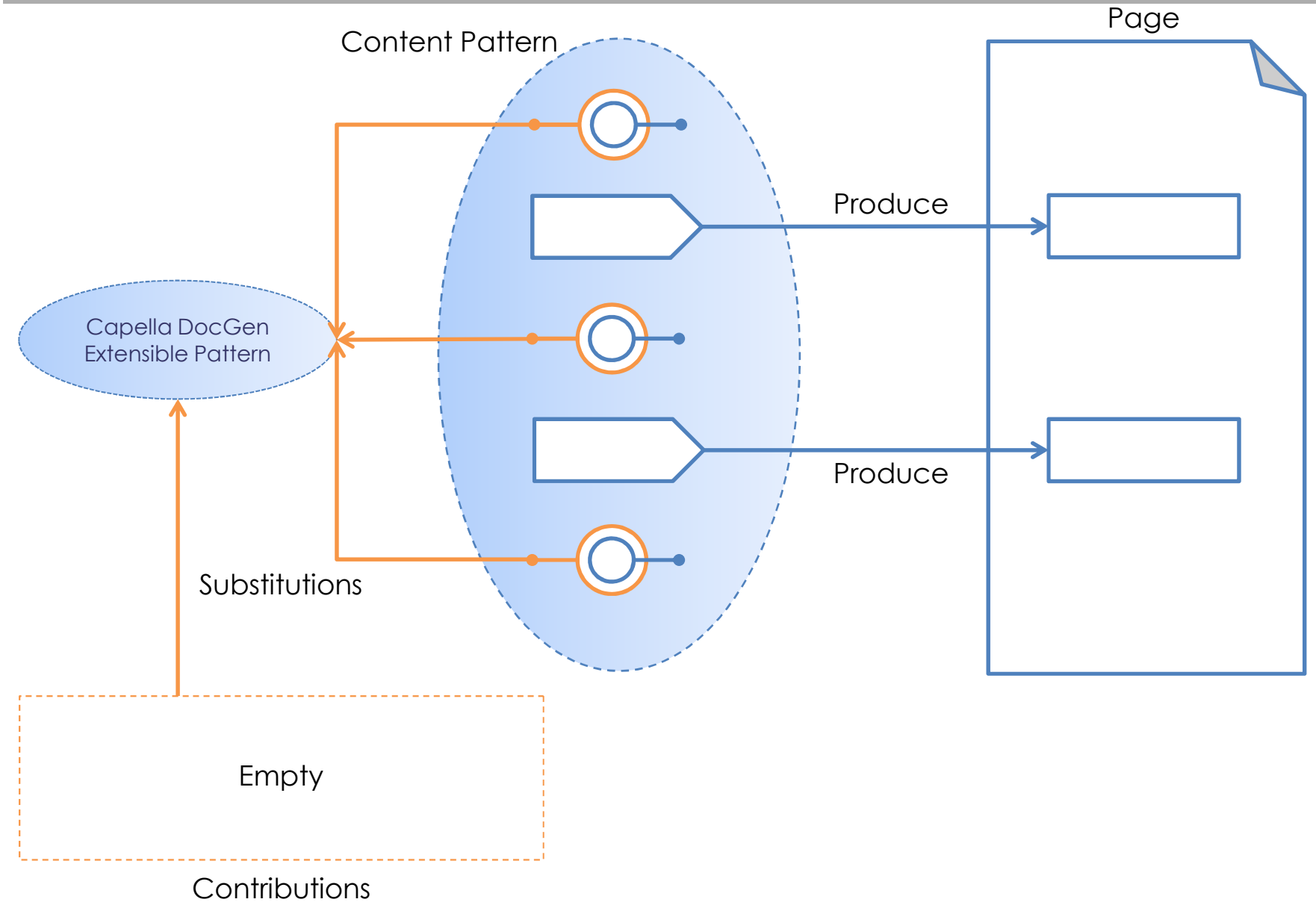
Concept – Existing Capella XHTML DocGen



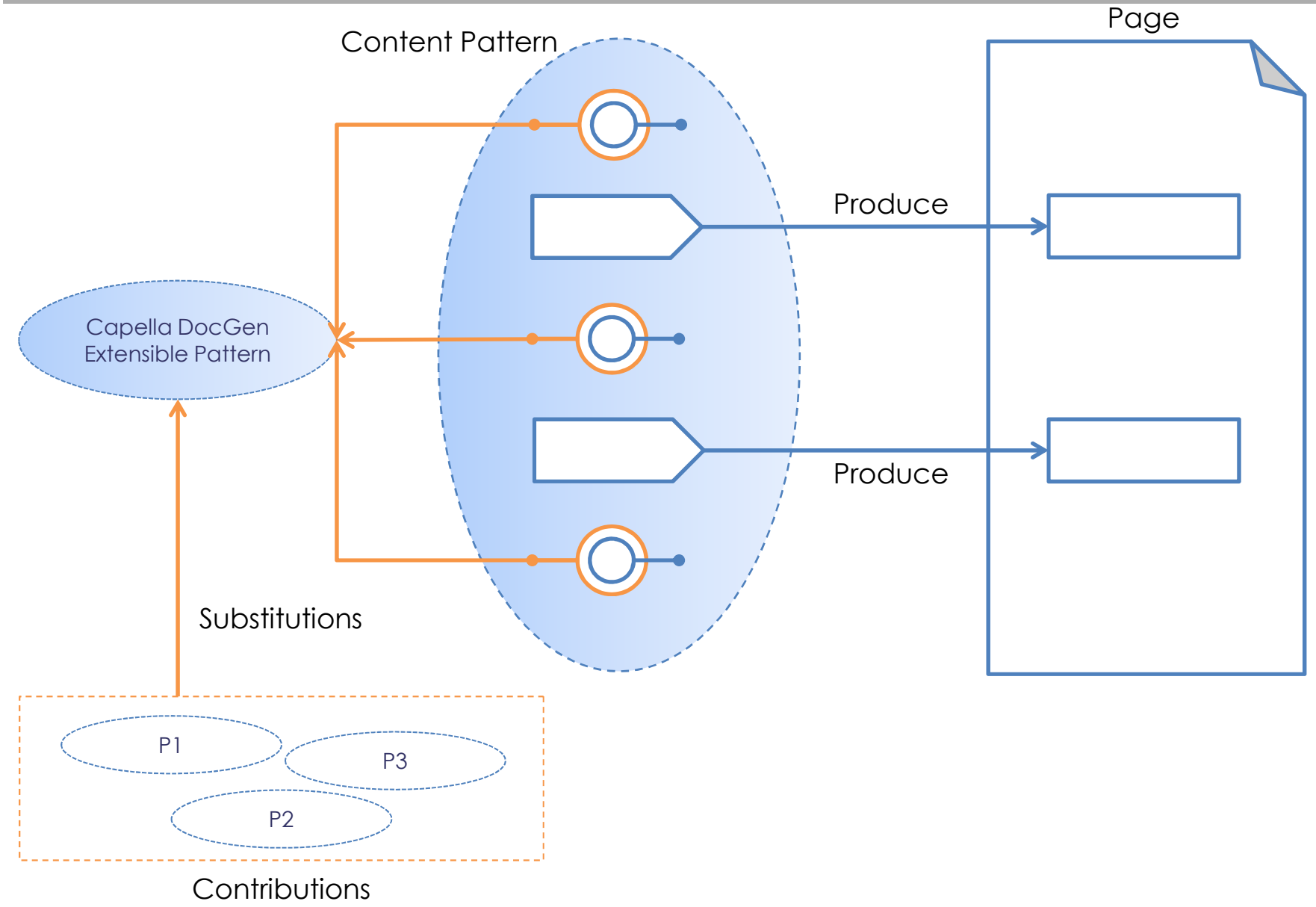
Piece of generation code which produce piece of XHTML page

Piece of XHTML Page

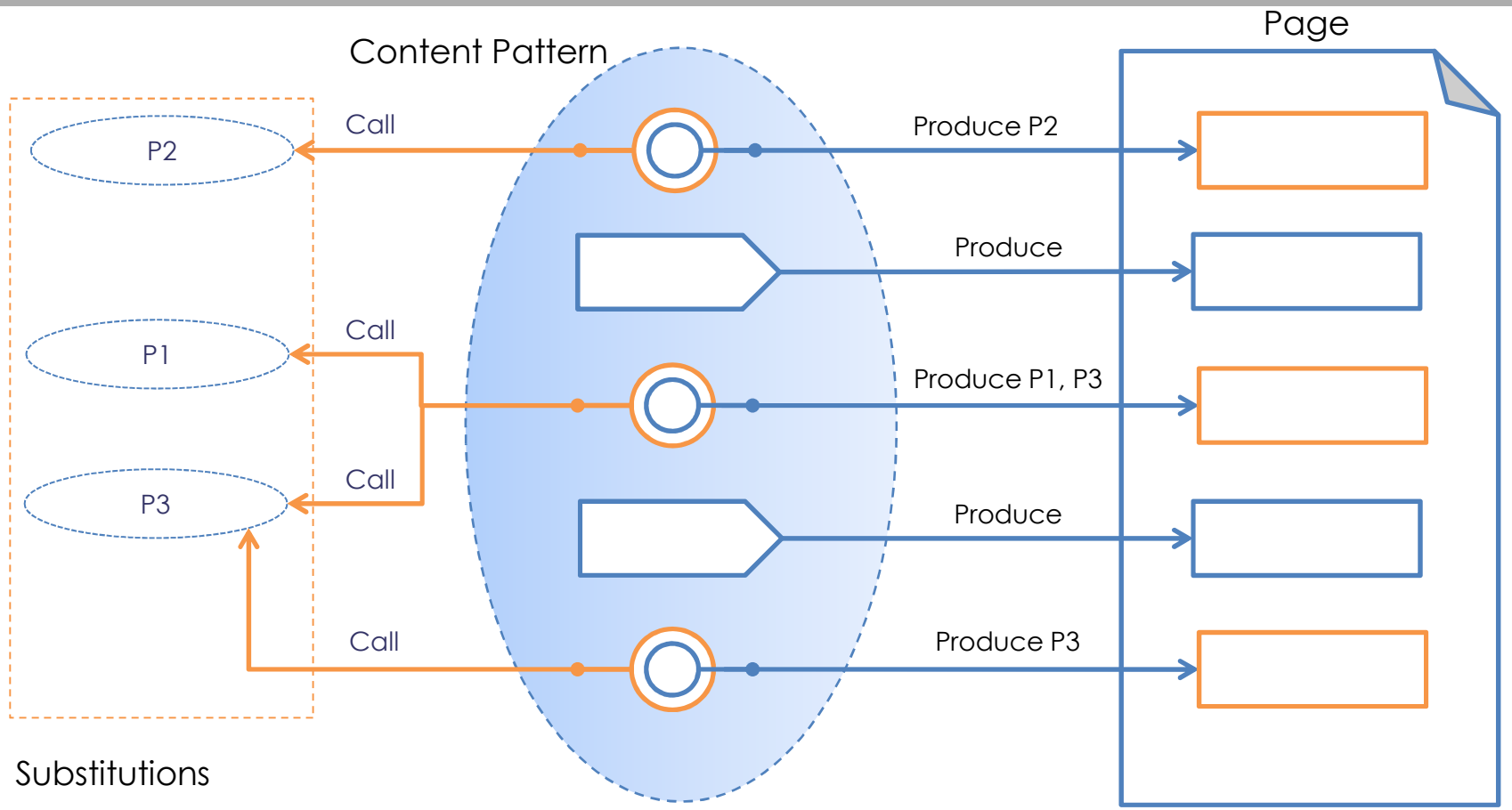
Concept – Capella XHTML DocGen



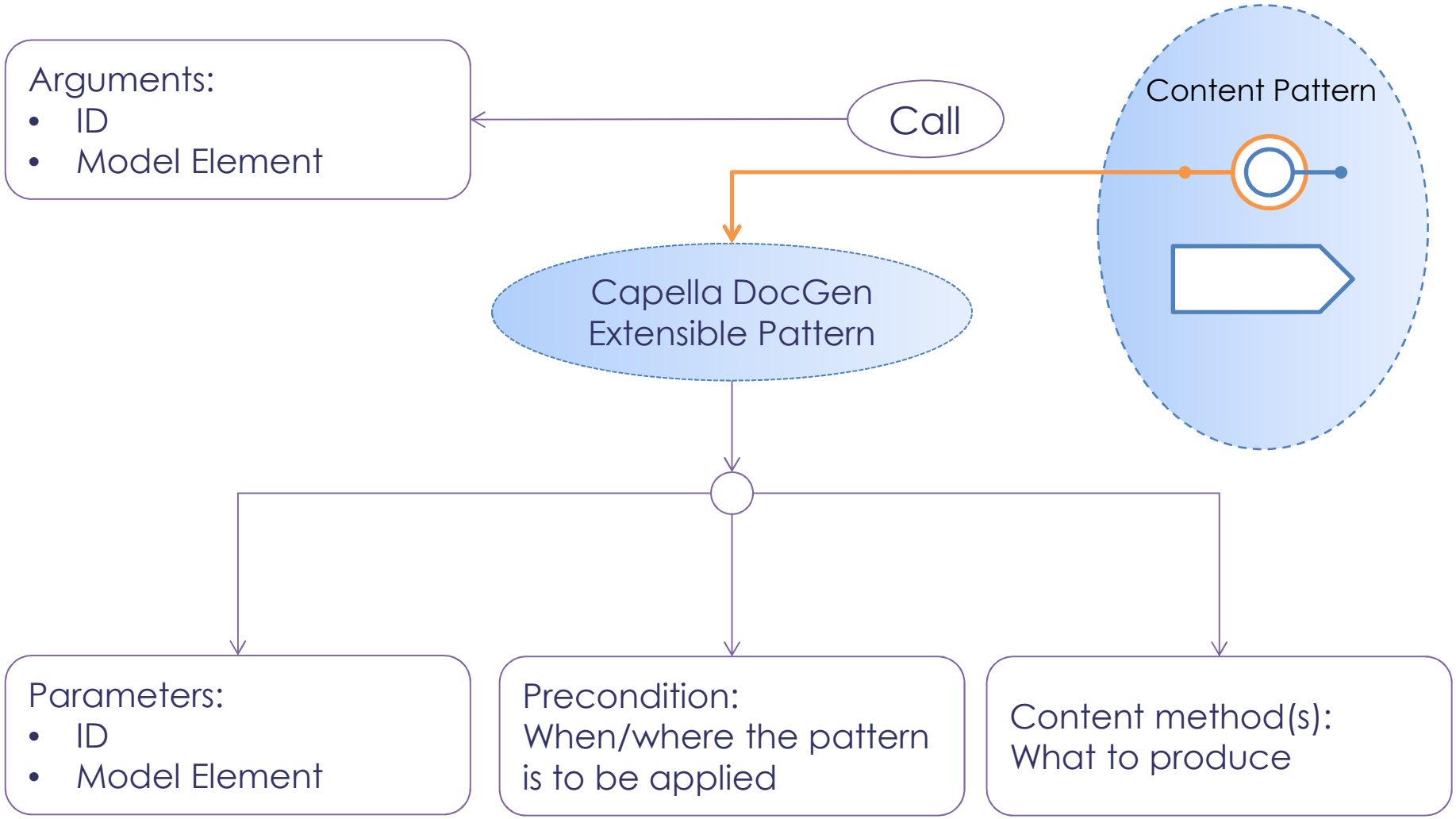
Concept – Capella XHTML DocGen



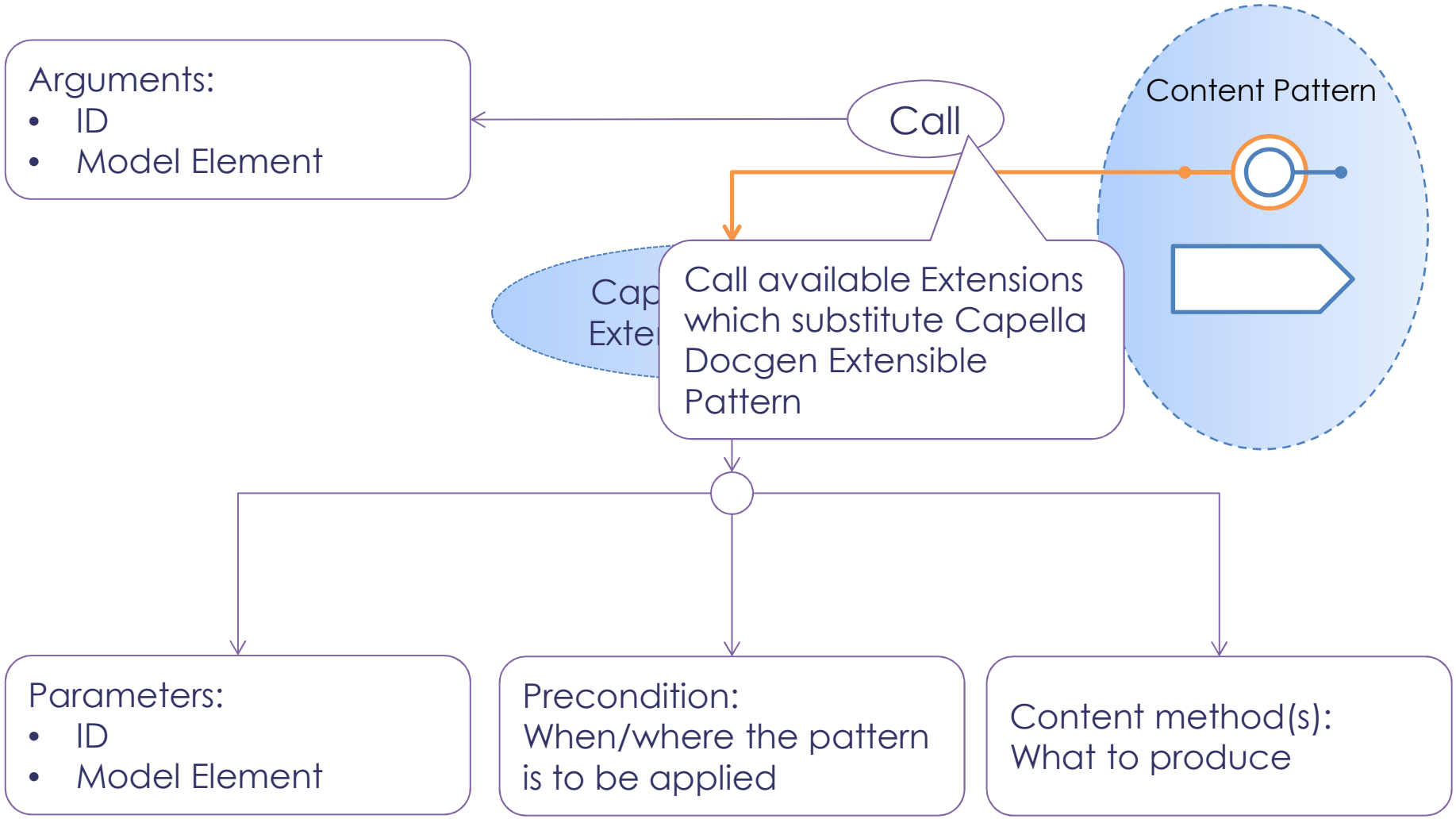
Concept – Capella XHTML DocGen



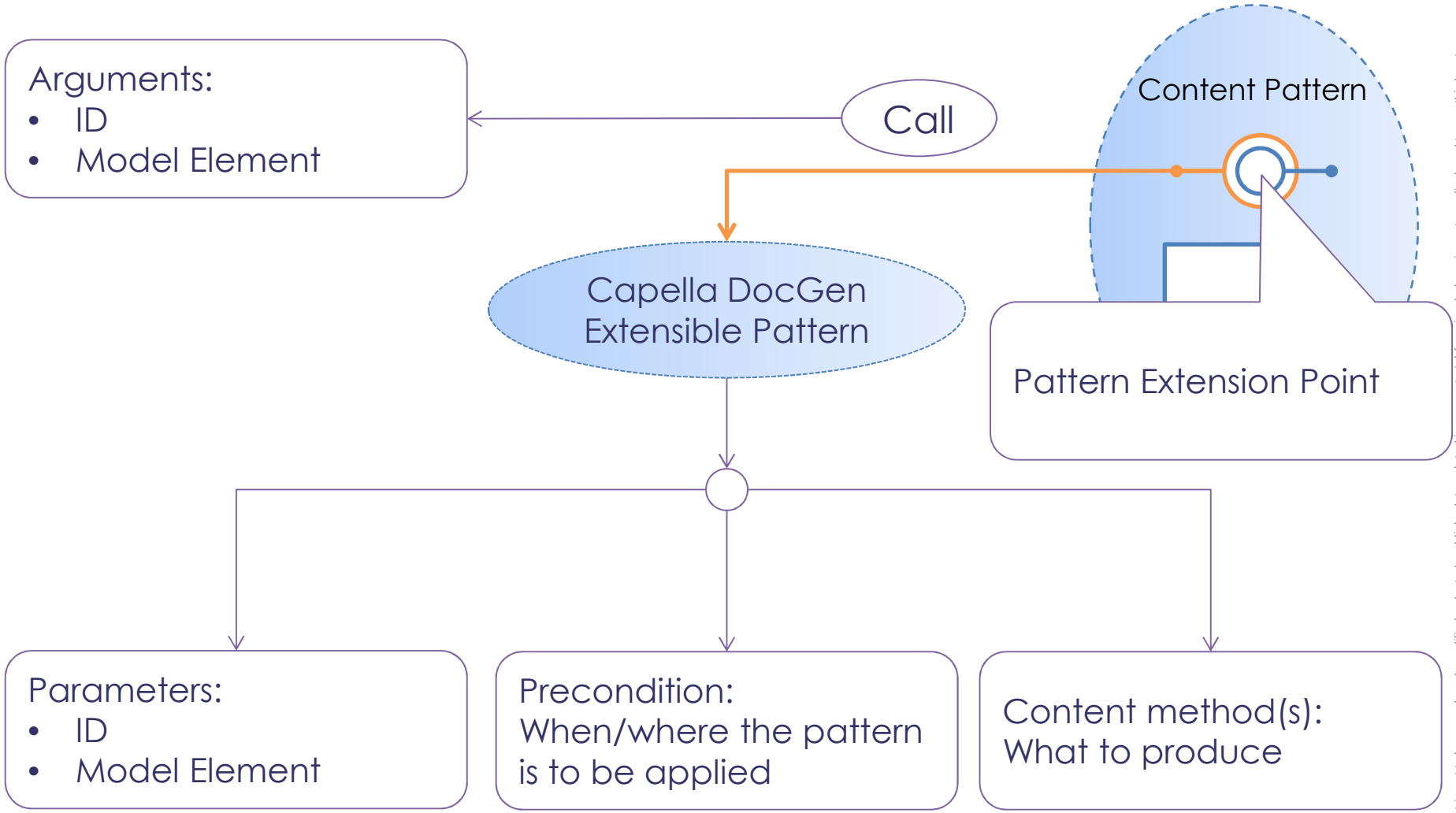
Concept – Content Pattern & Capella DocGen Extensible Pattern



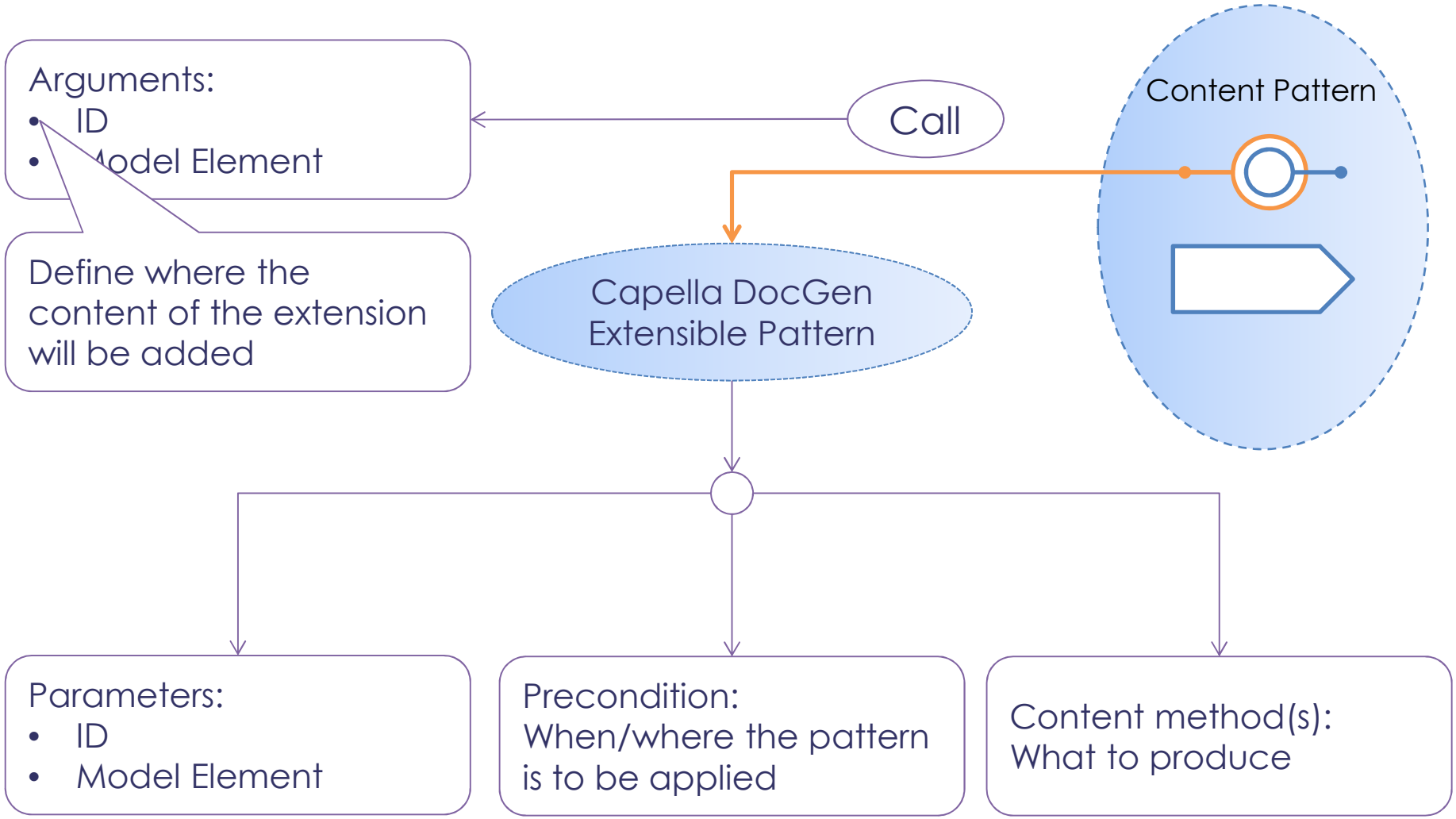
Concept – Content Pattern & Capella DocGen Extensible Pattern



Concept – Content Pattern & Capella DocGen Extensible Pattern



Concept – Content Pattern & Capella DocGen Extensible Pattern



Concept – IDs Definition

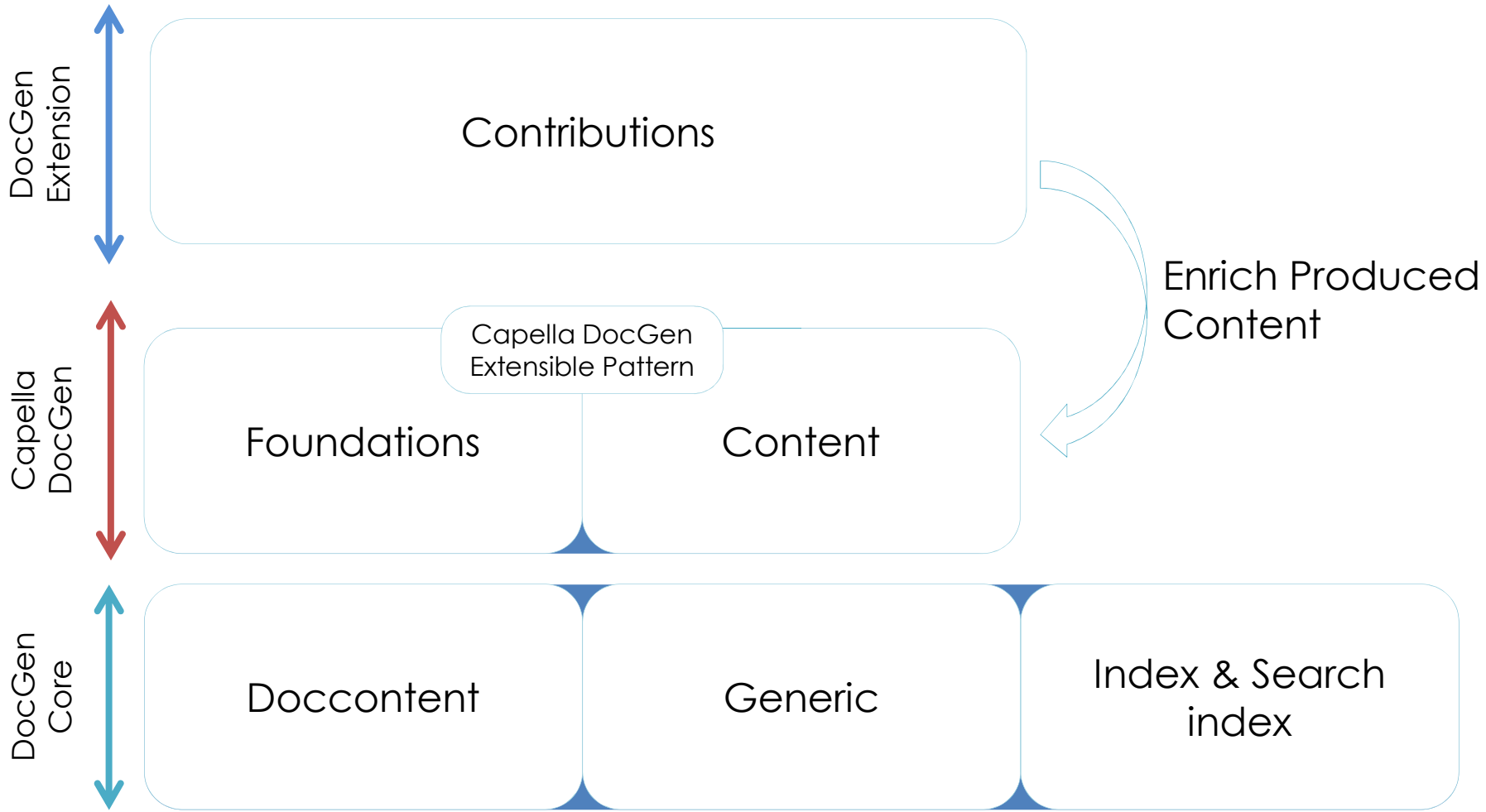
IDs are defined in: org.polarsys.capella.docgen.extensions.ID

```java
package org.polarsys.capella.docgen.extensions;

public interface ID {

    //Parts IDs
    public final String PART_BEFORE_HEADER = "org.polarsys.capella.docgen.content.PartDocGen.part.before.header";
    /**
     * After header & before description
     */
    public final String PART_AFTER_HEADER = "org.polarsys.capella.docgen.content.PartDocGen.part.after.header";
    /**
     * After Description & before Diagram generation
     */
    public final String PART_AFTER_DESCRIPTION = "org.polarsys.capella.docgen.content.PartDocGen.part.after.description";
    /**
     * After Diagram Generation & before type
     */
    public final String PART_AFTER_DIAGRAM_GENERATION = "org.polarsys.capella.docgen.content.PartDocGen.part.after.diagram.generation";
    /**
     * before type Content
     */
    public final String PART_BEFORE_TYPE_CONTENT = "org.polarsys.capella.docgen.content.PartDocGen.part.before.type.content";
    /**
     * After type content and before component content
     */
```

Concept – Capella XHTML DocGen Patterns Overview



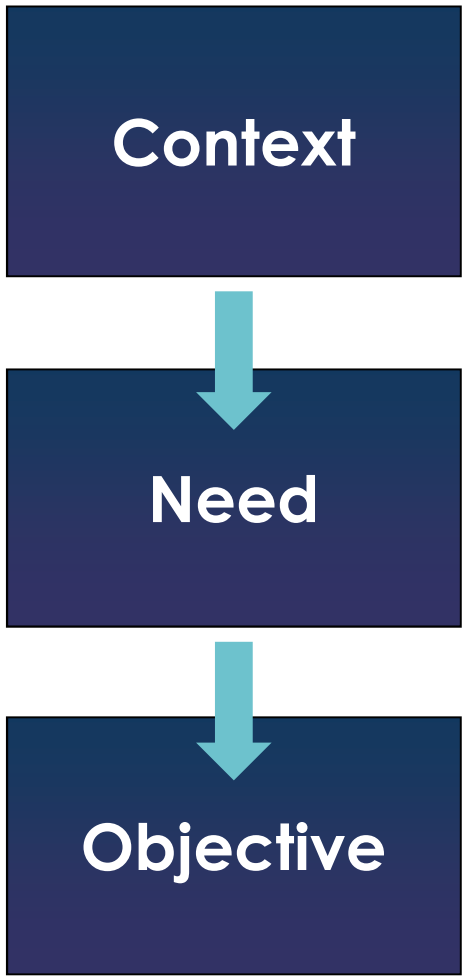
www.thalesgroup.com

Contributive XHTML with Extension point

OPEN

THALES

Core XHTML DocGen - Purpose

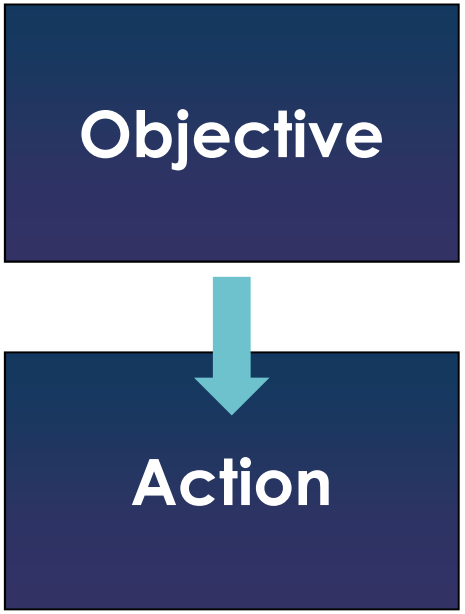


- Models are complex and Basic documentation is insufficient
- Existing Core XHTML DocGen is not contributive

- Need to extend XHTML DocGen for customizing and adding or remove information in generated documentation from models

- Rework existing Core XHTML DocGen framework to be contributive

Core XHTML DocGen - Contributive XHTML DocGen based on Extension Point



- Work on existing Core XHTML DocGen framework to be contributive by Eclipse-base extension point

- Add to existing Core XHTML DocGen framework to be contributive by Eclipse-base extension point

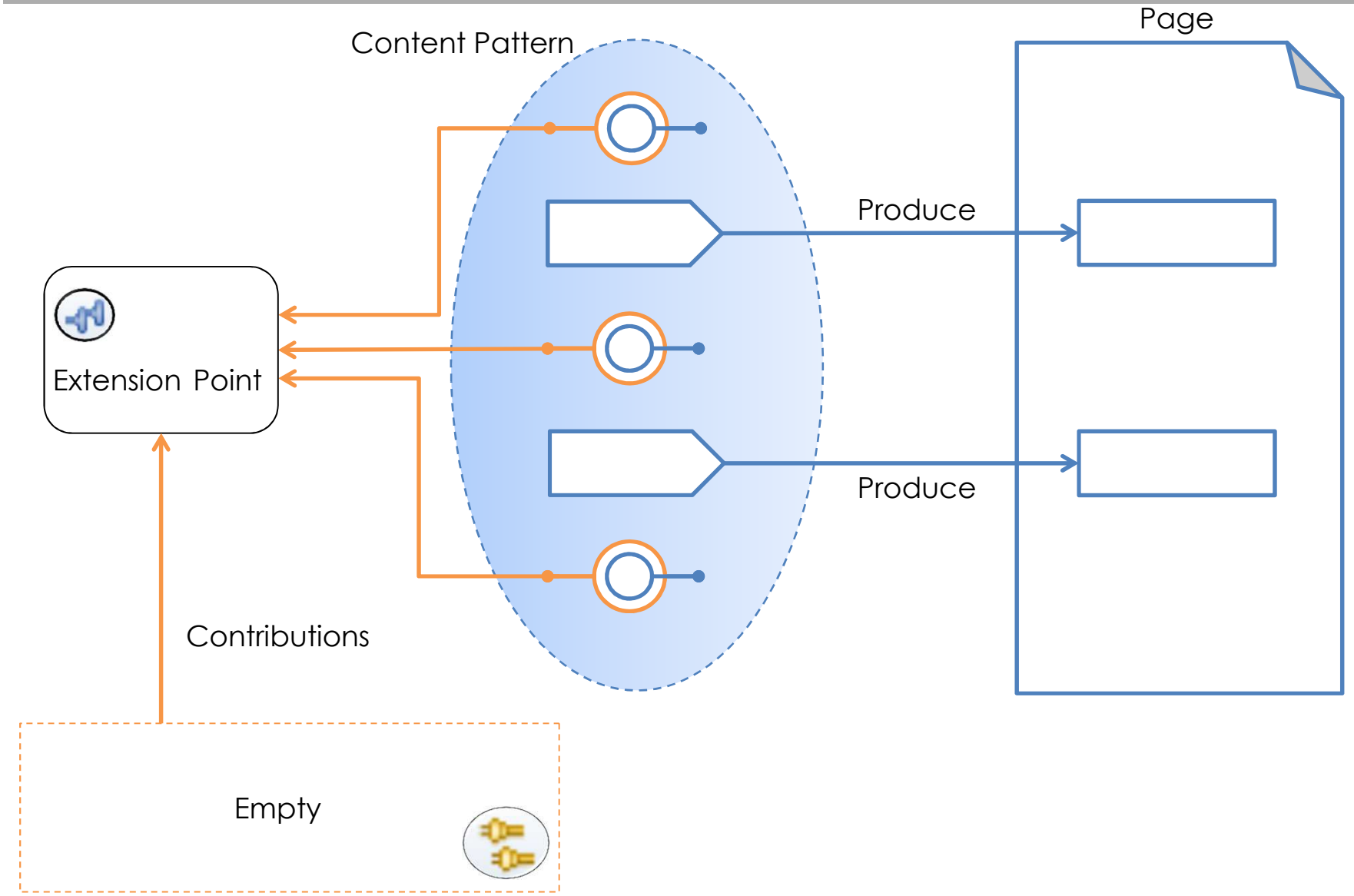
Introduction – Extensions with Eclipse Extension Point

Mechanism

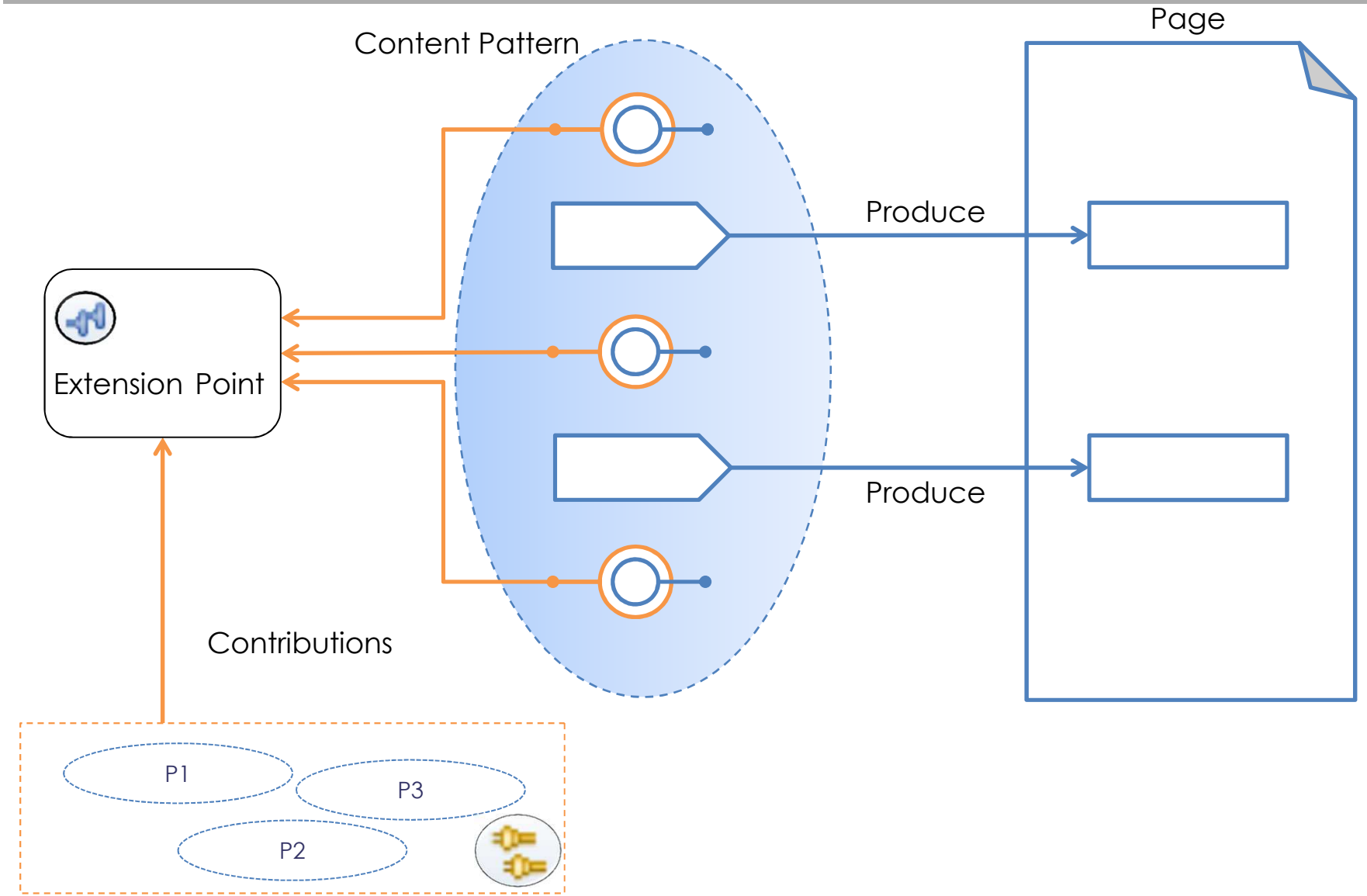
Provide an Eclipse extension point which allows to contribute with Factory Component, Pattern Library or Pattern to add content into the page

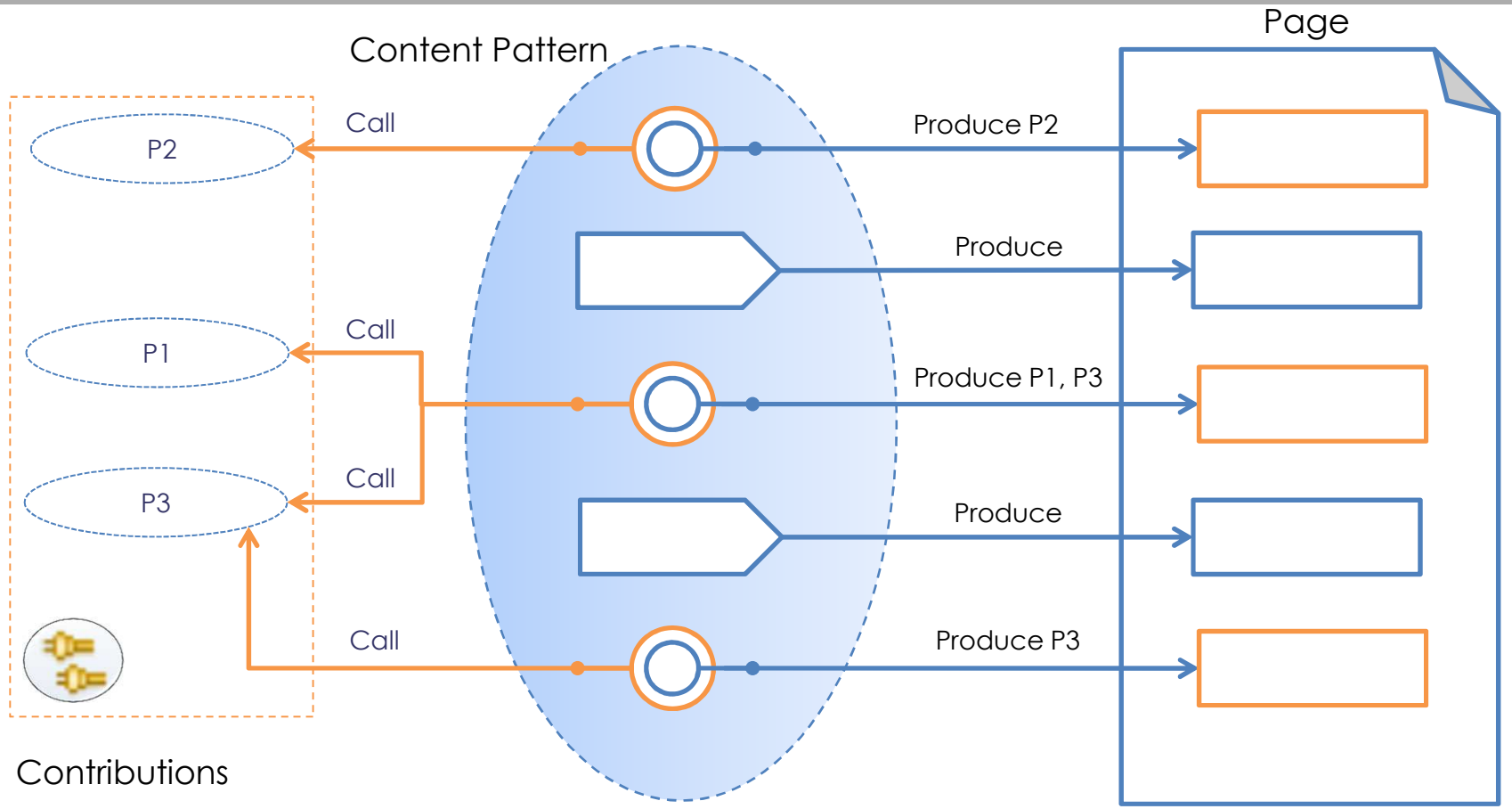
Contributed Factory Component, Pattern Library or Patterns That Add Content into the page

Eclipse Extension Point

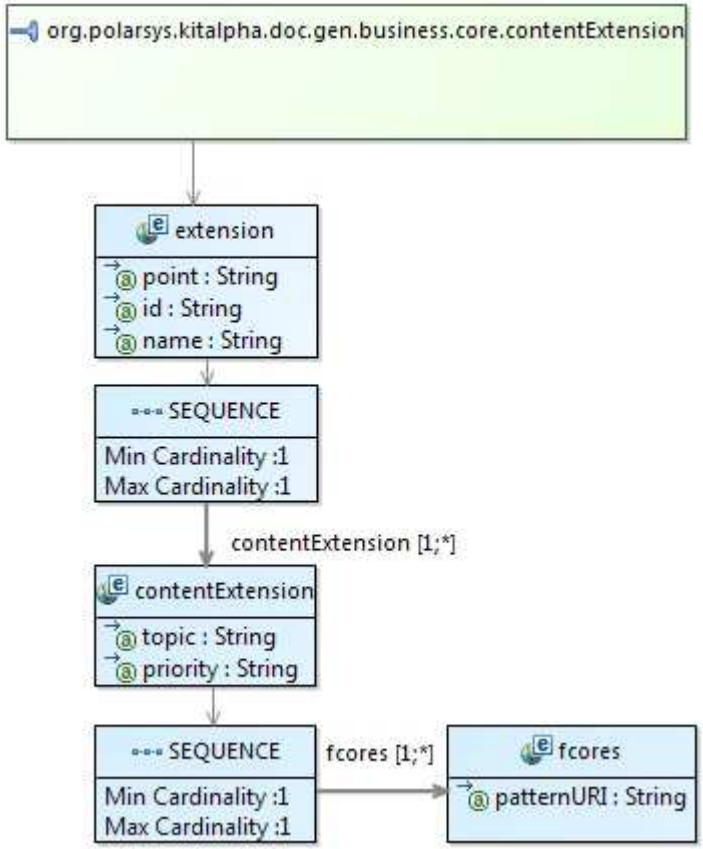


Concept – Capella XHTML DocGen



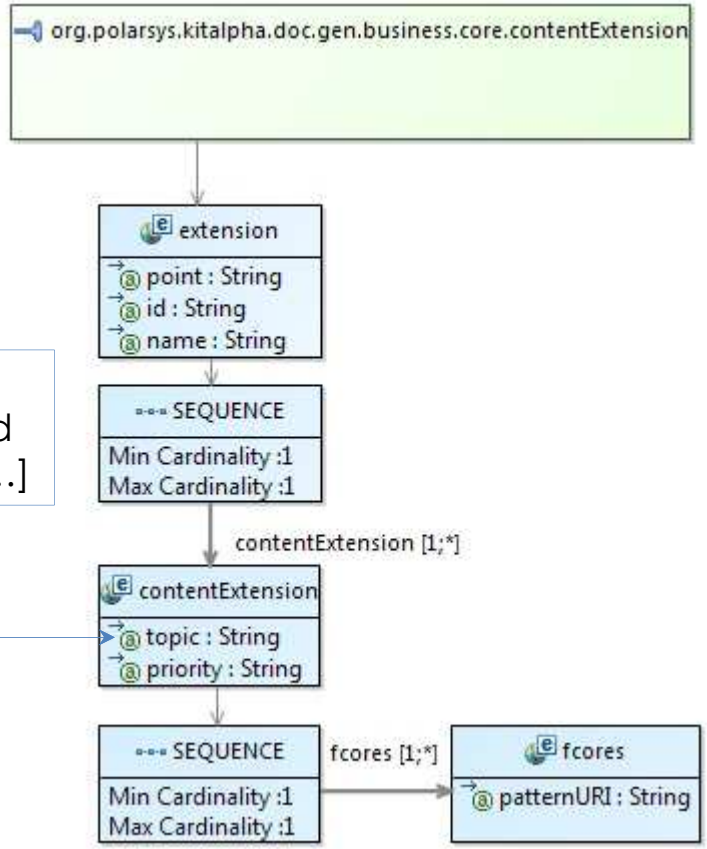


Implementation – Extension point



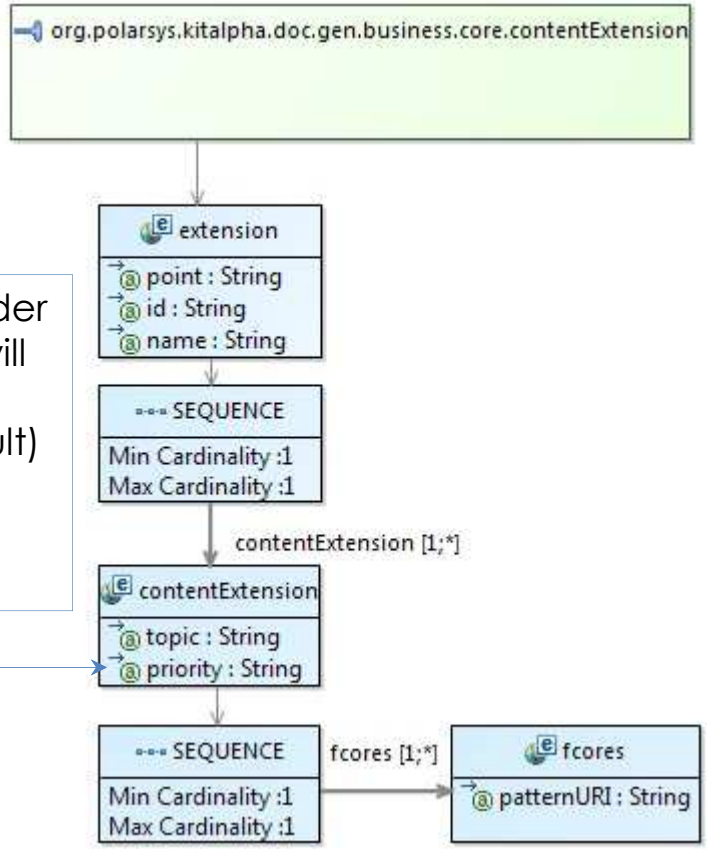
Implementation – Extension point

Topic indicate where the Contributions will be applied
Format: TOPIC_1 [, TOPIC2,…]

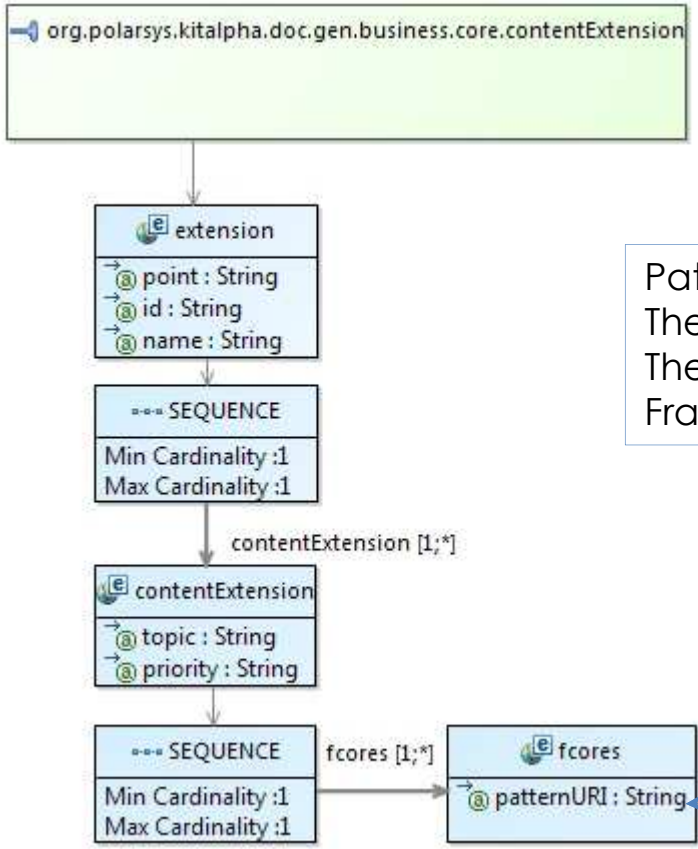


Implementation – Extension point

Priority indicates in which order
The contributions to topics will
Be executed.
It's an integer from -1 (default)
To n.
-1 : is the lowest priority
0 : is the highest priority



Implementation – Extension point



PatternURI indicates the path to The patterns.
The path must contains the Fragment

Implementation – Topics

- **Topic**
  - Declaration of execution point where a set of patterns will be applied
  - Dispatching a set of patterns on execution points on which they will be applied

- **Execution point**
  - Point in the flow execution where external behavior will be applied

- **Dispatching**
  - Association of a set of execution with a set of patterns
  - The patterns have a priority (lowest by default)

Implementation – Execution point definition

- Services to declare execution point are defined in:
  org.polarsys.kitalpha.doc.gen.business.core.extension.intf.IDocGenExtensionEngine

- Example to declare Execution point "MY_EXECUTION_POINT"

```
DocGenExtensionFactory.newDocGenExtensionEngine().apply(
        « MY_EXECUTION_POINT",          //the execution point
        ctx,                            //the context of the pattern
        getParameters(),                //the parameters of the pattern
        stringBuffer);                  //a buffer
```

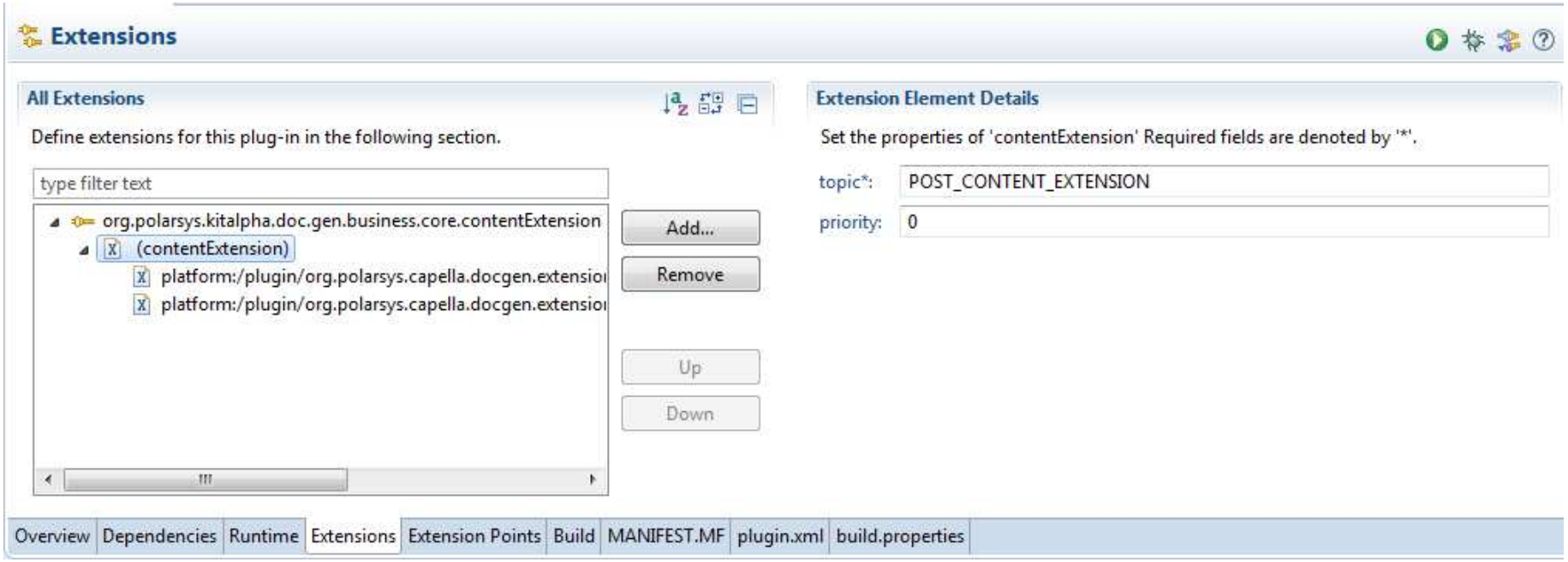
- Example to declare generic execution point

```
DocGenExtensionFactory.newDocGenExtensionEngine().execute(
        ctx,                            //the context of the pattern
        getParameters());               //the parameters of the pattern
```

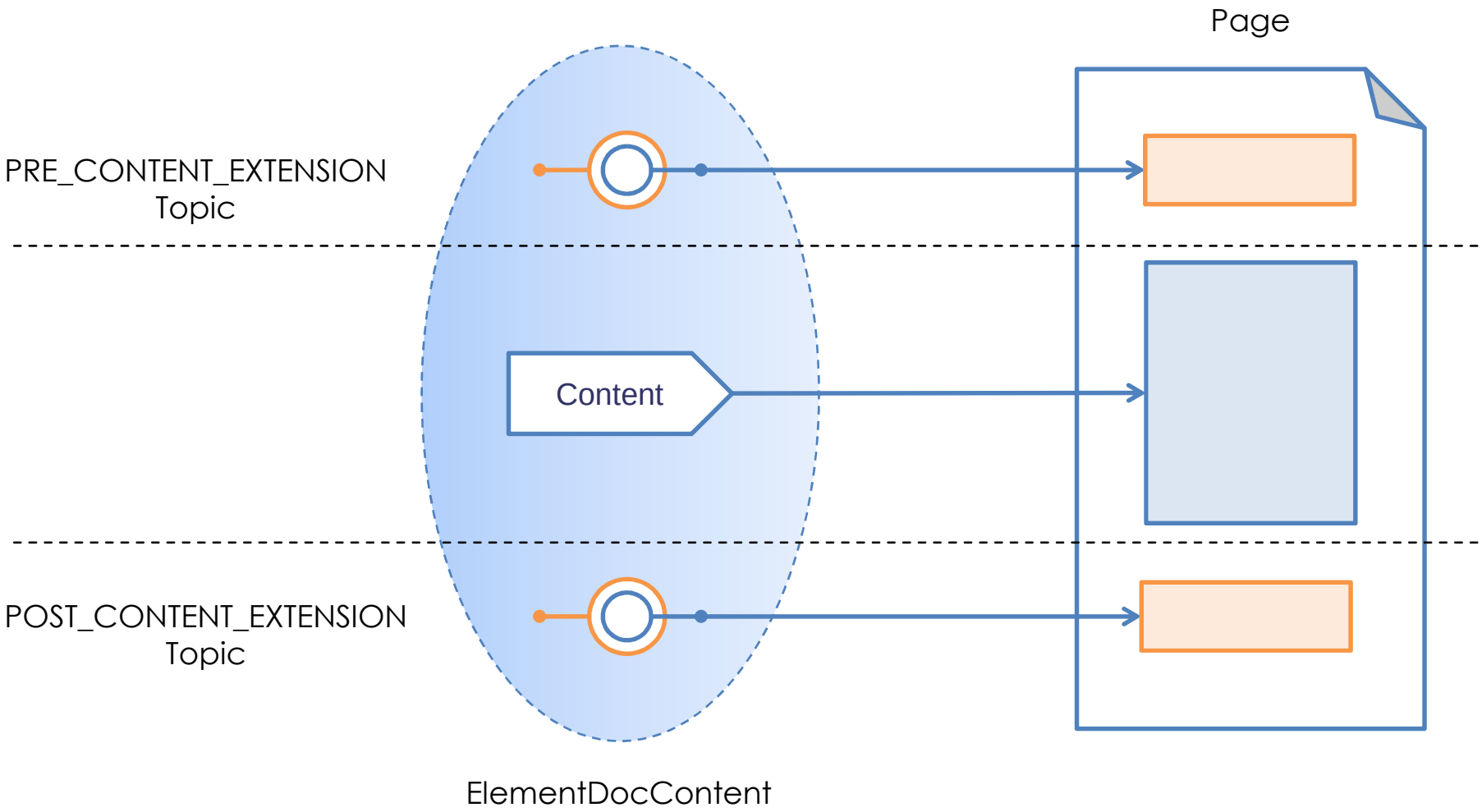
Implementation – Dispatching

- **Dispatching**
  - Contribution to: org.polarsys.kitalpha.doc.gen.business.core.extensionContent

- Example



Implementation – Provided Topics



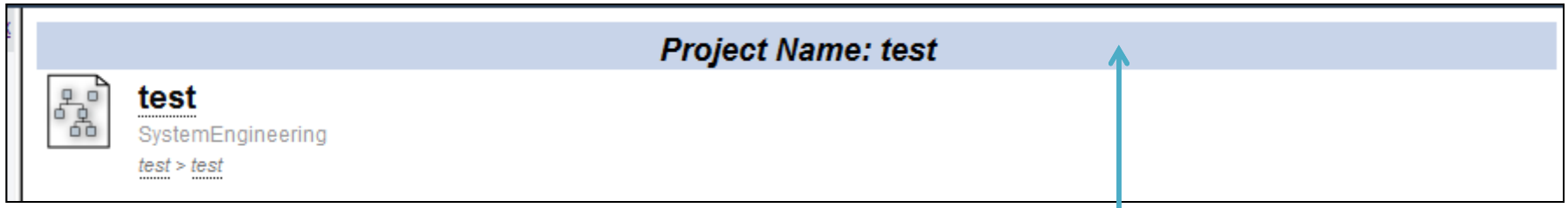
Examples

Example

- Use case
  - PRE_CONTENT_EXTENSION
    - Display the project name in center with changing the background color on all pages
  - POST_CONTENT_EXTENSION
    - Display the information of Price and Mass viewpoints in that order and apply the same background color as project name
  - The background color settings must be contributed

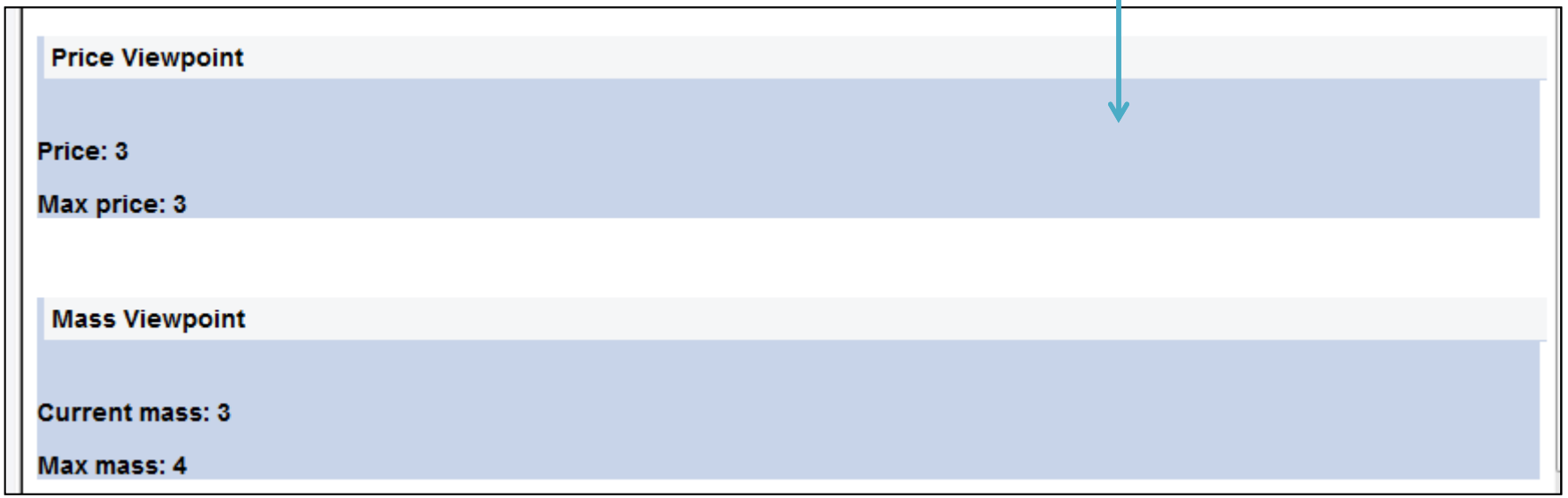
Example

PRE_CONTENT_EXTENSION



Same background

POST_CONTENT_EXTENSION



Example – Project name

Steps to contribute to PRE_CONTENT_EXTENSION execution point

1. Create fcore with pattern named ProjectName
2. Add parameter named “parameter” typed Eobject
3. Add precondition which check if the root is Capella project
4. Implement and orchestrate the body method (cf. code after)
5. Contribute to dispatch the pattern on PRE_CONTENTE_EXTENSION with priority 0

Pattern code

```
If (parameter != null){
    Map<String, Object> callParameters = new HashMap<String, Object>();
    callParameters.put("parameter", this.parameter);
    DocGenExtensionFactory.newDocGenExtensionEngine().apply("BEFORE_PROCESSING_CONTENT",
                                        ctx, callParameters, out);
    String name = ((Project) parameter).getName();
    out.append("<div style='color:black'><center><h1><i>Project Name: ").
                append(name).
                append("</i></h1></center></div>");
    DocGenExtensionFactory.newDocGenExtensionEngine().apply("AFTER_PROCESSING_CONTENT",
                                        ctx, callParameters, out);
}
```

Notice this contribution declares 2 execution point named BEFORE_PROCESSING_CONTENT and AFTER_PROCESSING_CONTENT. These execution point will served to apply a background color

Example – Price Viewpoint

Steps to contribute to POST_CONTENT_EXTENSION execution point

1. Create fcore with pattern named pricevp
2. Add parameter named “parameter” typed EObject
3. Add precondition which check if the parameter is part
4. Implement and orchestrate the body method (cf. code next slide)
5. Contribute to dispatch the pattern on POST_CONTENTE_EXTENSION with Priority 0

Example – Price Viewpoint

Pattern code

```
if (parameter != null && ((Part) parameter).getAbstractType() instanceof PhysicalComponent) {
            PhysicalComponent p = (PhysicalComponent) ((Part) parameter).getAbstractType();
            EList<AbstractTypedElement> abstractTypedElements = p.getAbstractTypedElements();
            for (AbstractTypedElement ac : abstractTypedElements) {
                        EList<ElementExtension> ownedExtensions = ac.getOwnedExtensions();
                        for (ElementExtension e : ownedExtensions) {
                          if (e instanceof PartPrice) {
                          Map<String, Object> callParameters = new HashMap<String, Object>();
                          callParameters.put("parameter", this.parameter);
                          DocGenExtensionFactory.newDocGenExtensionEngine()
                                    .apply("BEFORE_PROCESSING_CONTENT", ctx, callParameters, out);
                          PartPrice price = ((PartPrice) e);
                          int value = price.getValue();
                          int maxValue = price.getMaxValue();
                          out.append("<h2>Price Viewpoint</h2>");
                          out.append("<h3>Price: ").append(value).append("<br/></h3>\n");
                          out.append("<h3>Max price: ").append(maxValue).append("</h3>");
                          DocGenExtensionFactory.newDocGenExtensionEngine()
                                    .apply("AFTER_PROCESSING_CONTENT", ctx, callParameters, out);
                        }
            }
}
```

Notice this contribution declares 2 execution point named BEFORE_PROCESSING_CONTENT and AFTER_PROCESSING_CONTENT. These execution point will served to apply a background color

Example – Mass Viewpoint

Steps to contribute to POST_CONTENT_EXTENSION execution point

1. Create fcore with pattern named massvp
2. Add parameter named “parameter” typed EObject
3. Add precondition which check if the parameter is part
4. Implement and orchestrate the body method (cf. code next slide)
5. Contribute to dispatch the pattern on POST_CONTENTE_EXTENSION with Priority 1

Example – Price Viewpoint

Pattern code

```
if (parameter != null && ((Part) parameter).getAbstractType() instanceof PhysicalComponent) {
            PhysicalComponent p = (PhysicalComponent) ((Part) parameter).getAbstractType();
            EList<AbstractTypedElement> abstractTypedElements = p.getAbstractTypedElements();
            for (AbstractTypedElement ac : abstractTypedElements) {
              EList<ElementExtension> ownedExtensions = ac.getOwnedExtensions();
              for (ElementExtension e : ownedExtensions) {
                if (e instanceof PartMass) {
                        Map<String, Object> callParameters = new HashMap<String, Object>();
                        callParameters.put("parameter", this.parameter);
                        DocGenExtensionFactory.newDocGenExtensionEngine()
                          .apply("BEFORE_PROCESSING_CONTENT", ctx, callParameters, stringBuffer);
                        PartMass p1 = (PartMass) e;
                        currentMass = p1.getValue();
                        maxValue = p1.getMaxValue();

                        stringBuffer.append("<h2>Mass Viewpoint</h2>");
                        stringBuffer.append(currentMass);
                        stringBuffer.append("<h3>Current mass: ");
                        stringBuffer.append(maxValue).append("<br/></h3><h3>");
                        stringBuffer.append("<h3>Max mass: ").append("</h3>");

                        DocGenExtensionFactory.newDocGenExtensionEngine()
                          .apply("AFTER_PROCESSING_CONTENT", ctx, callParameters, stringBuffer);
                        }
            }
}
```

Notice this contribution declares 2 execution point named BEFORE_PROCESSING_CONTENT and AFTER_PROCESSING_CONTENT. These execution point will served to apply a background color

Example – Background color

Steps to contribute to BEFORE_PROCESSING_CONTENT execution point

1. Create fcore with pattern named backgroundBefore
2. Add parameter named “parameter” typed EObject
3. Implement and orchestrate the body method (cf. code after)
4. Contribute to dispatch the pattern on BEFORE_PROCESSING_CONTENT with Priority 1

Pattern code (Jet pattern)

```
<div style='background-color:#C8D4E9;color:black;'>
```

Steps to contribute to AFTER_PROCESSING_CONTENT execution point

1. Create fcore with pattern named backgroundAFTER
2. Add parameter named “parameter” typed EObject
3. Implement and orchestrate the body method (cf. code after)
4. Contribute to dispatch the pattern on AFTER_PROCESSING_CONTENT with Priority 1

Pattern code (Jet pattern)

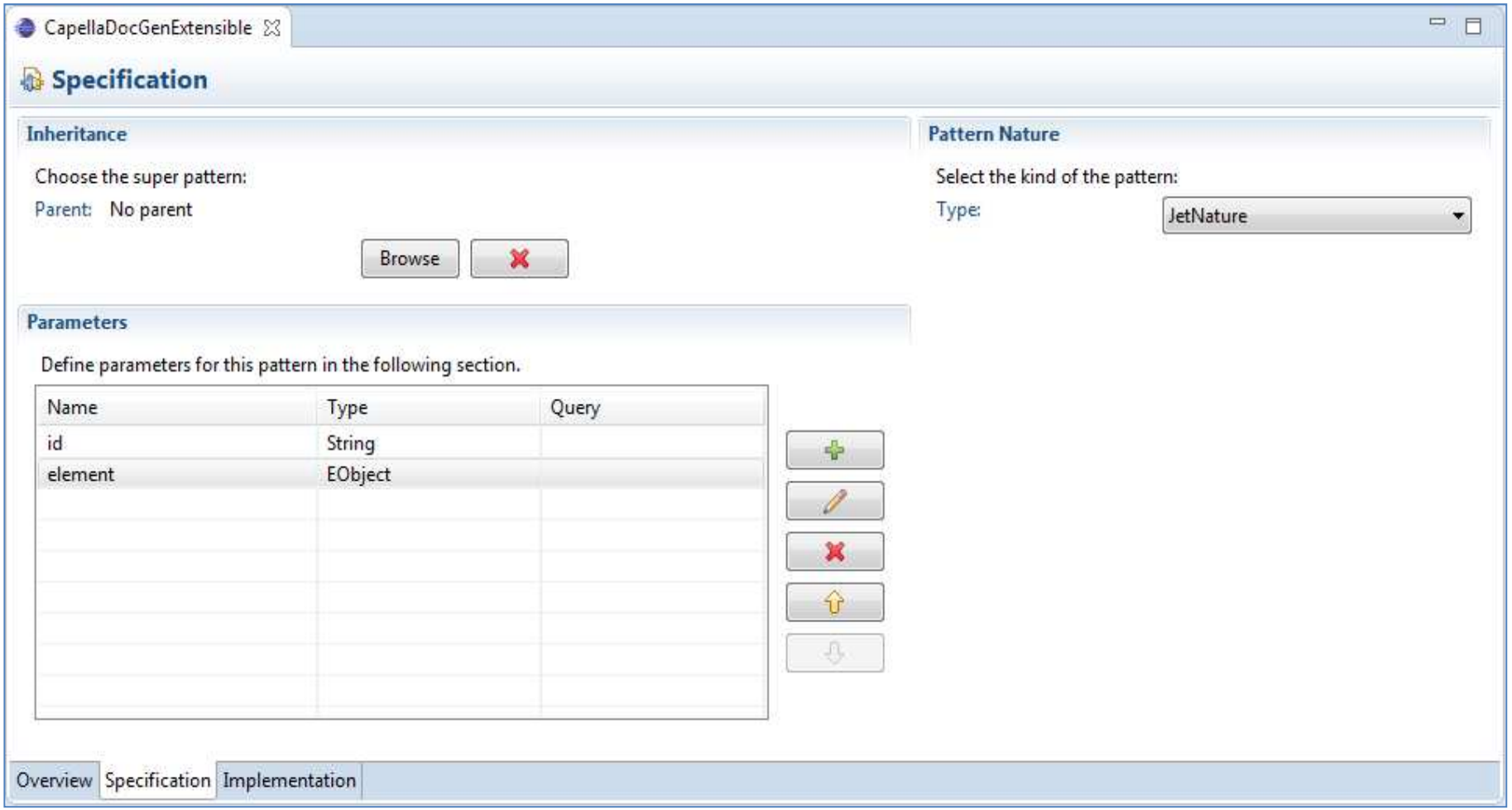
```
</div>
```

Pattern substitution principle

Capella DocGen Extensible Pattern Implementation details

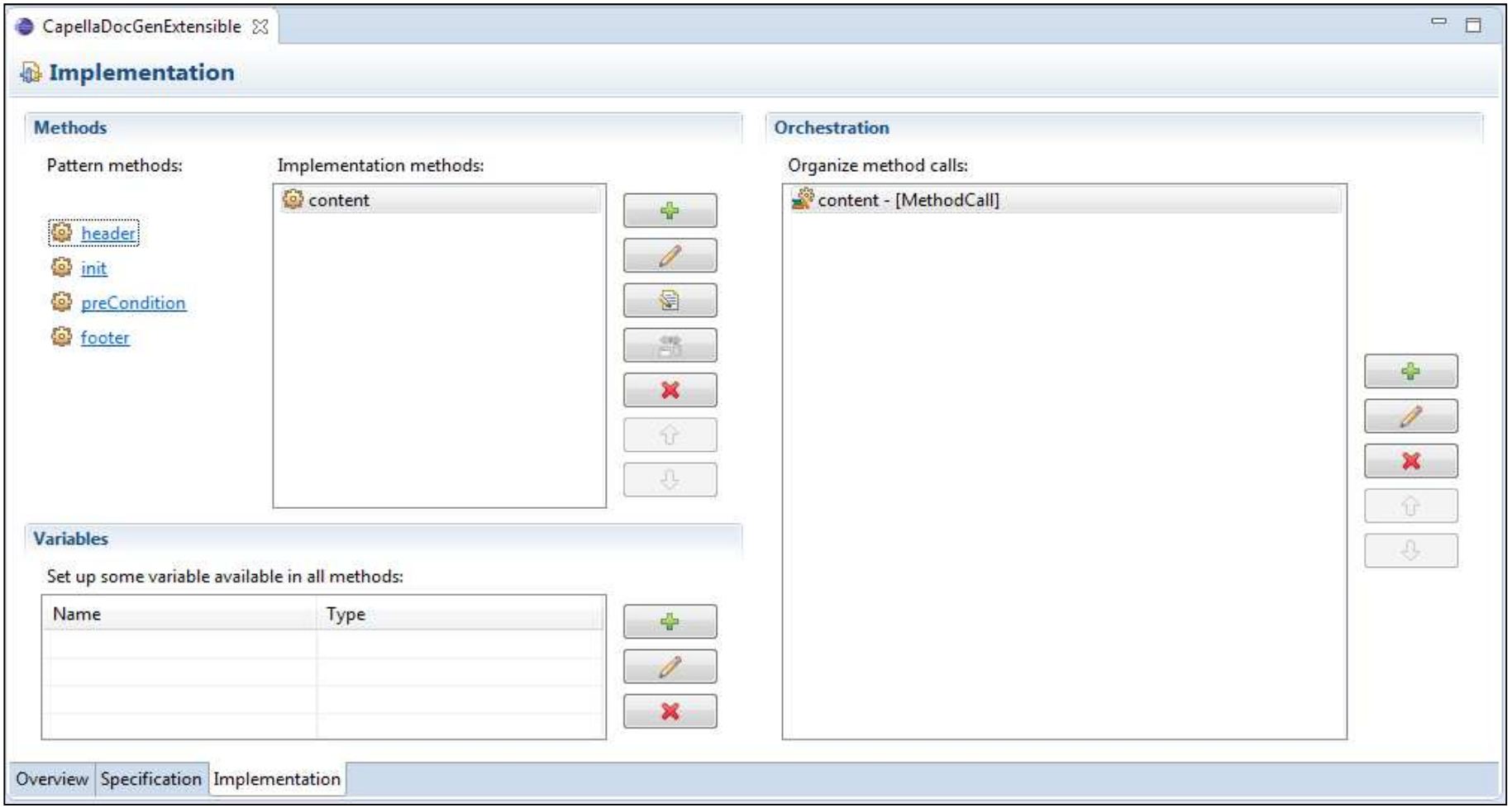
Specification – Capella DocGen Extensible Pattern

Capella DocGen Extensible Pattern



Implementation – Capella DocGen Extensible Pattern

Capella DocGen Extensible Pattern



Capella DocGen – Contribution steps

Contribution Steps – Step 1

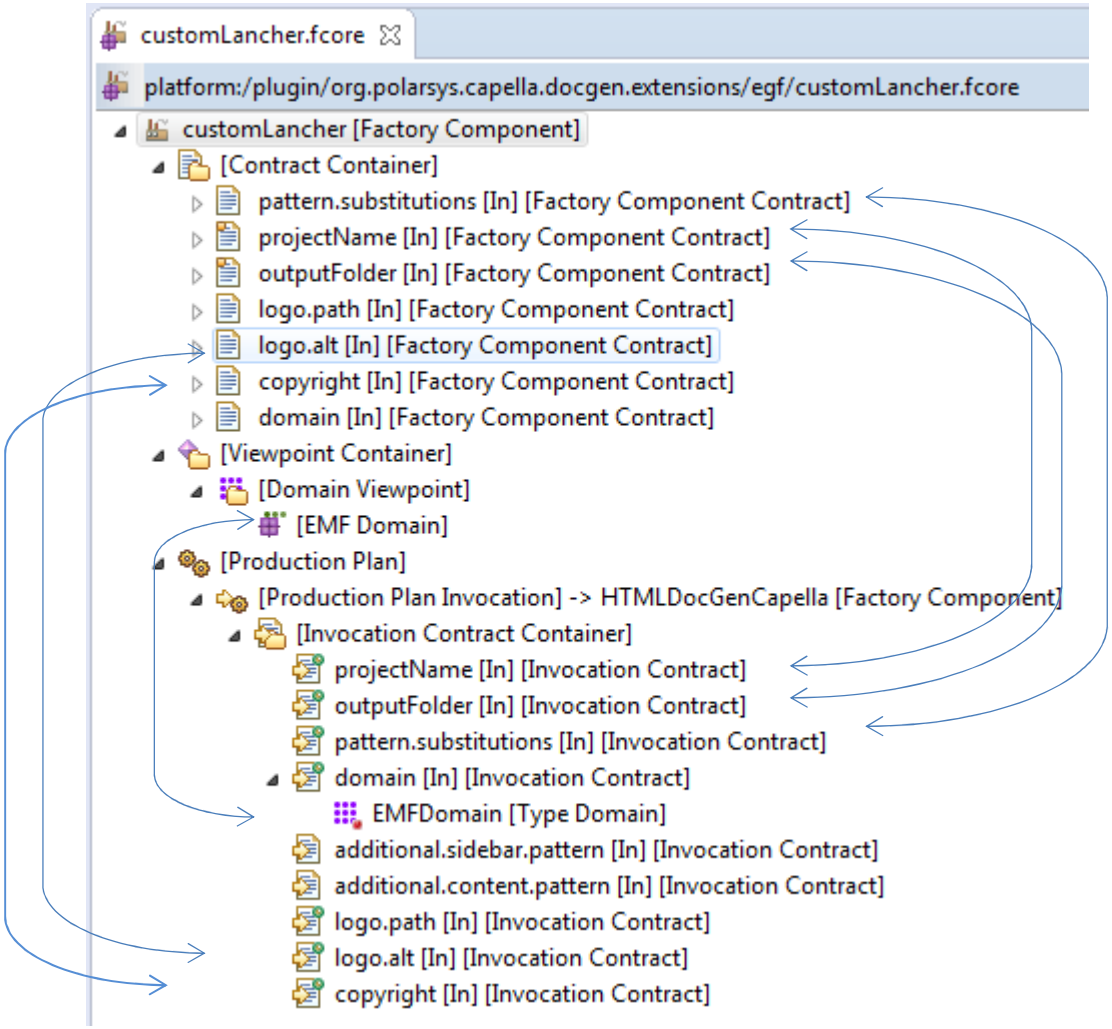
Step 1: Content Pattern Specification

- Create a new EGF Factory Component
- Create a Viewpoint Container
- Create a Pattern Viewpoint
- Create a Pattern Library
- Create a new JET or Java Pattern (e.g., PartMassViewpoint)
- Create a method(s) (e.g., content) and add it to the orchestration
- Implement the business of the methods
- Add preCondition specifying where/when to contribute

Step 2: Launcher Definition

- Create a new EGF Factory Component (e.g, CustomLauncher)
- Create contract Container
- Create Factory Component Contracts, at least for:
  - Project Name (projectName)
  - Output Folder (outputFolder)
  - Pattern substitutions (pattern.substitutions)
- Substitute CapellaDocGenExtensible by pattern previously created and with itself
- Create a Viewpoint Container
- Create a Domain Viewpoint
- Create a EMF Domain
- Add Production Plan
- Invoke HTMLDocGenCapella factory
- Connect contracts of HTMLDocGenCapella with launcher's contracts
- Contribute to "org.polarsys.capella.docgen.launcher" to register the new launcher

Contribution Steps – Step 2



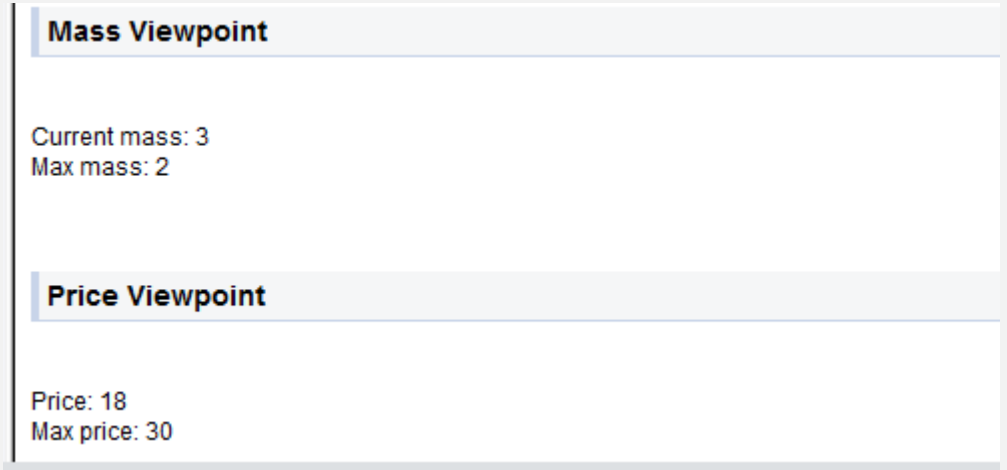
Examples

Example – Basic Mass & Basic Price

Objective

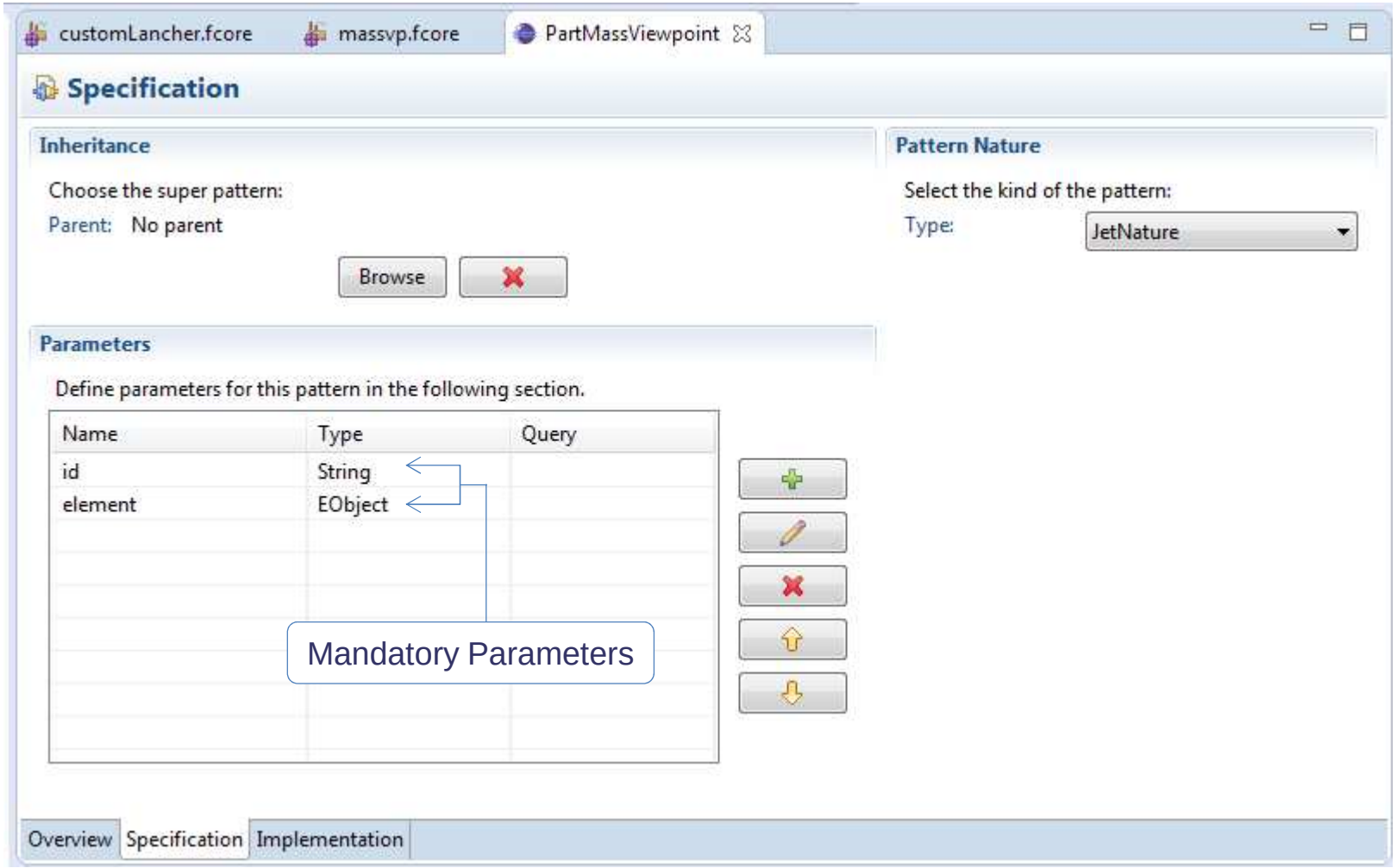
- Add Basic Mass and Basic Price to Physical Component

Result

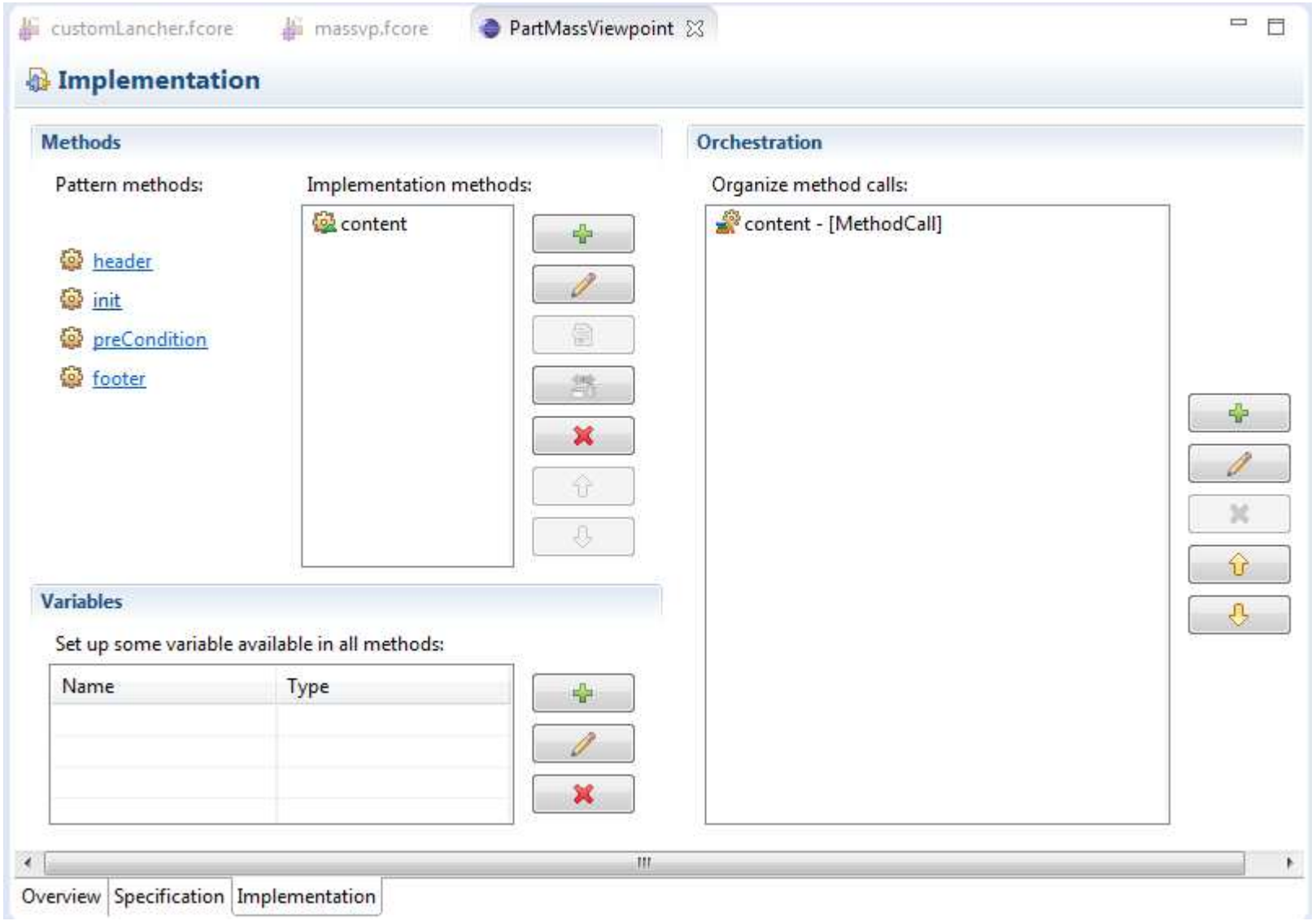


Example – Mass Viewpoint Pattern specification

PartMassViewpoint



Example – Mass Viewpoint Pattern Implementation



Example – Mass Viewpoint Pattern

preCondition method

customLancher.fcore | massvp.fcore | PartMassViewpoint | PartMassViewpoint

```
return ID.PART_AFTER_END_CONTENT.equals(id);
```

header | init | preCondition | content | footer

Content method

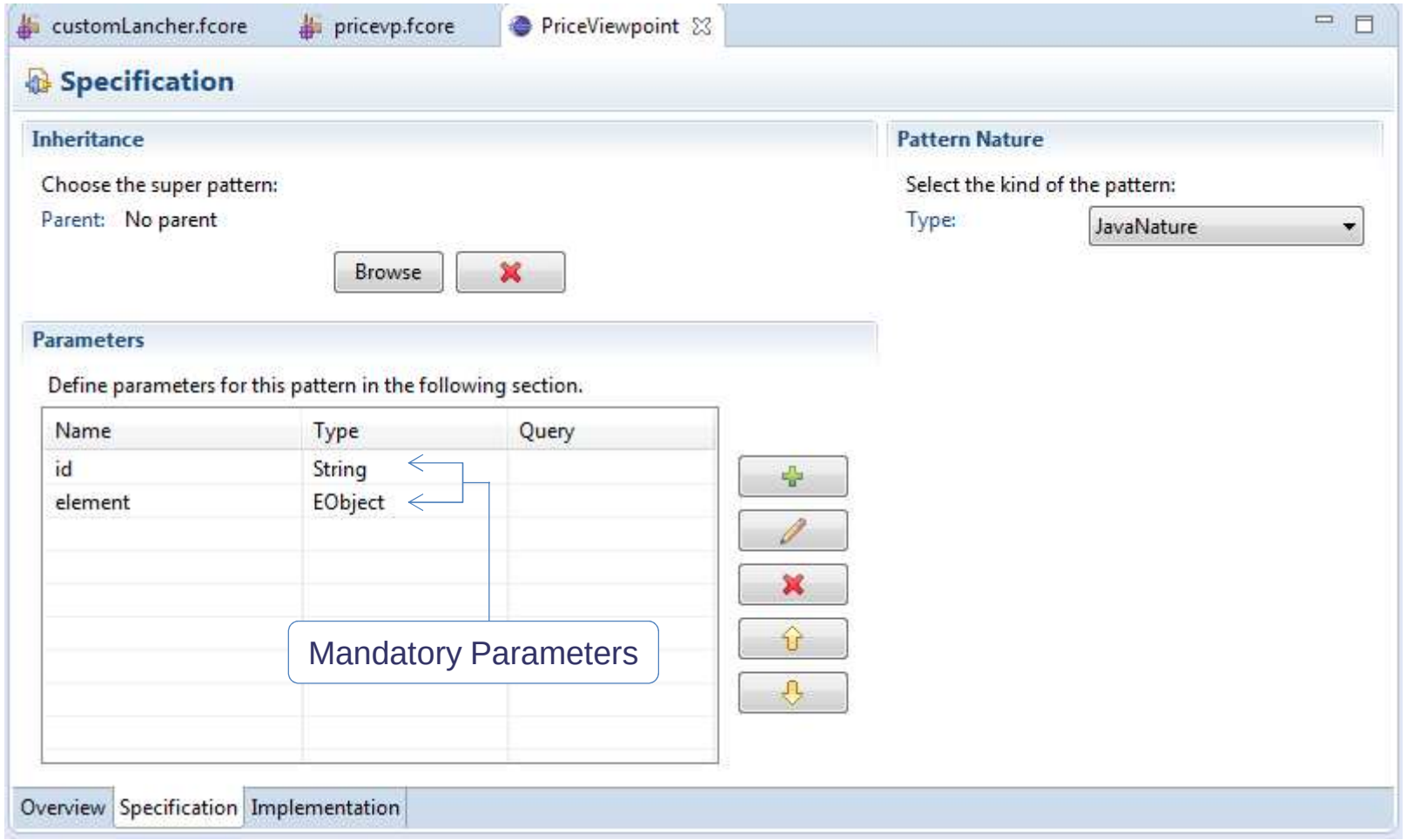
PriceViewpoint | PriceViewpoint | PartMassView... | org.polarsy... | PartMassView...

```
<%
int currentMass = 0;
int maxValue = 0;
if (element instanceof PhysicalComponent) {
            PhysicalComponent p = (PhysicalComponent) element;
            EList<AbstractTypedElement> abstractTypedElements = p.getAbstractTypedElements();
            for (AbstractTypedElement ac : abstractTypedElements) {
                EList<ElementExtension> ownedExtensions = ac.getOwnedExtensions();
                for (ElementExtension e : ownedExtensions) {
                    if (e instanceof PartMass) {
                        PartMass p1 = (PartMass) e;
                        currentMass = p1.getValue();
                        maxValue = p1.getMaxValue();
                        %>
                        <h2>Mass Viewpoint</h2>
                        Current mass: <%=currentMass%><br/>
                        Max mass: <%=maxValue%>;
                    <%}
                }
            }
        }
%>
```

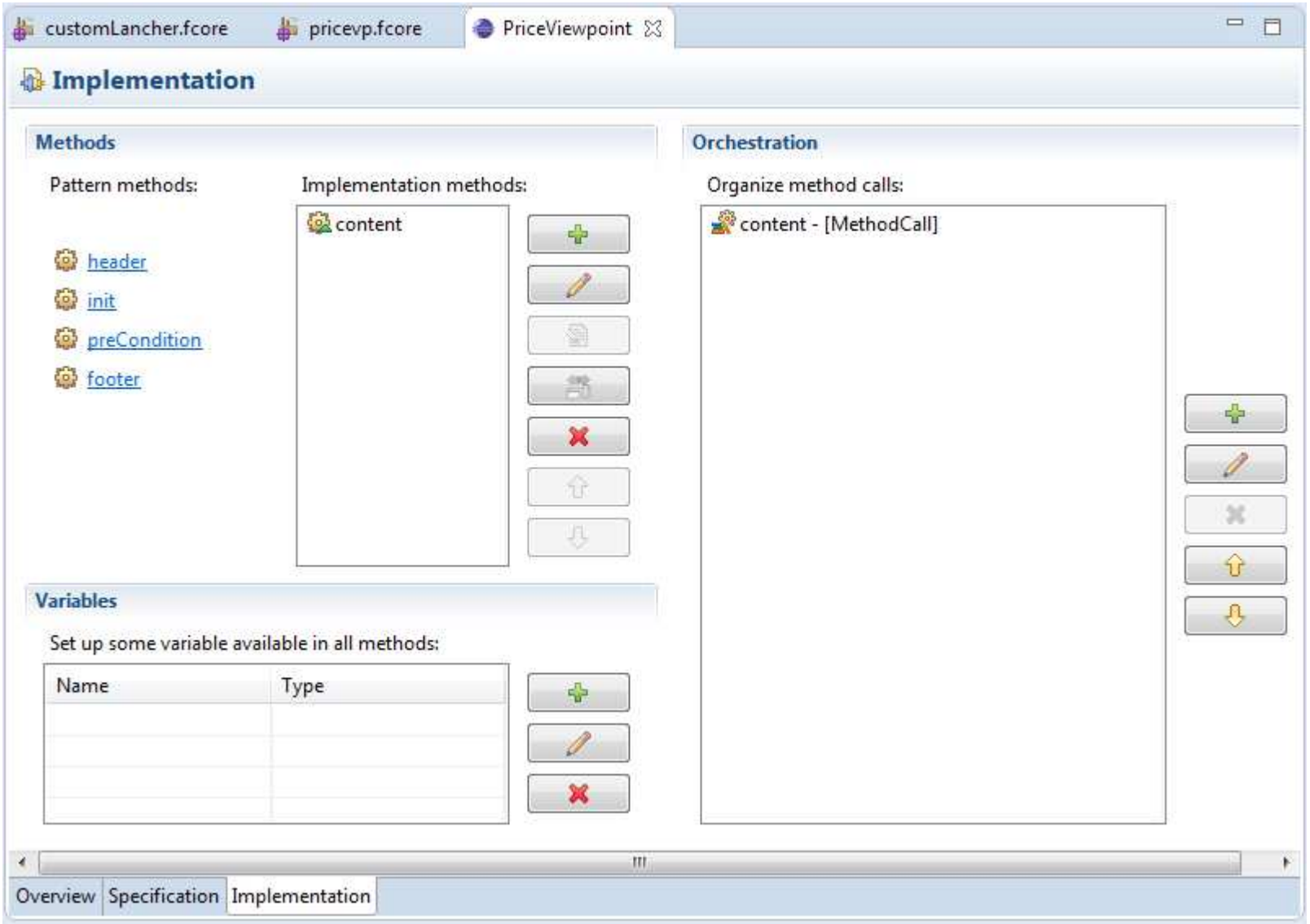
header | init | preCondition | content | footer

Example – Price Viewpoint Pattern Specification

PriceViewpoint



Example – Price Viewpoint Pattern Implementation



Example – Price Viewpoint Pattern

preCondition method

customLancher.fcore | pricevp.fcore | PriceViewpoint | PriceViewpoint

```
return ID.PART_AFTER_END_CONTENT.equals(id);
```

header | init | preCondition | content | footer

Content method

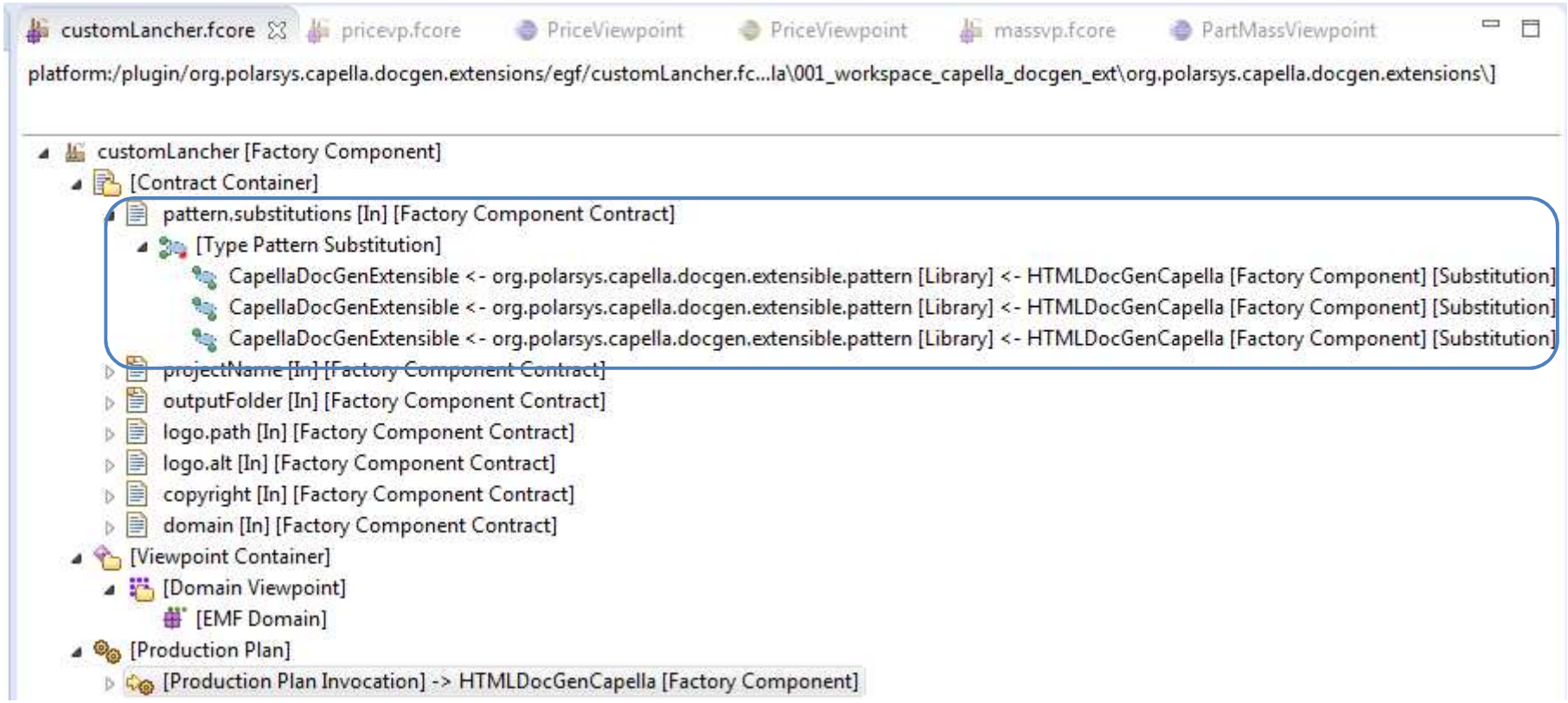
customLancher.fcore | pricevp.fcore | PriceViewpoint | PriceViewpoint

```
if (element instanceof PhysicalComponent) {
            PhysicalComponent p = (PhysicalComponent) element;
            EList<AbstractTypedElement> abstractTypedElements = p.getAbstractTypedElements();
            for (AbstractTypedElement ac : abstractTypedElements) {
                EList<ElementExtension> ownedExtensions = ac.getOwnedExtensions();
                for (ElementExtension e : ownedExtensions) {
        if (e instanceof PartPrice) {
            PartPrice price = ((PartPrice)e);
            int value = price.getValue();
            int maxValue = price.getMaxValue();
            out.append("<h2>Price Viewpoint</h2>");
            out.append("Price: ").append(value).append("<br/>\n");
            out.append("Max price: ").append(maxValue);
        }
        }
    }
}
```

header | init | preCondition | content | footer

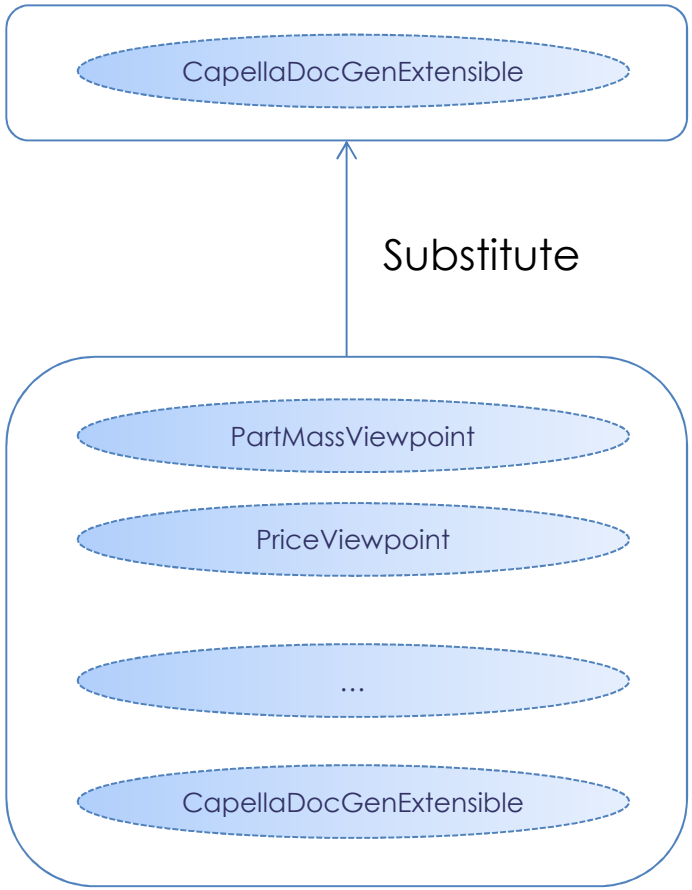
Example – Custom Launcher

Substitution of CapellaDocGenExtensible Pattern



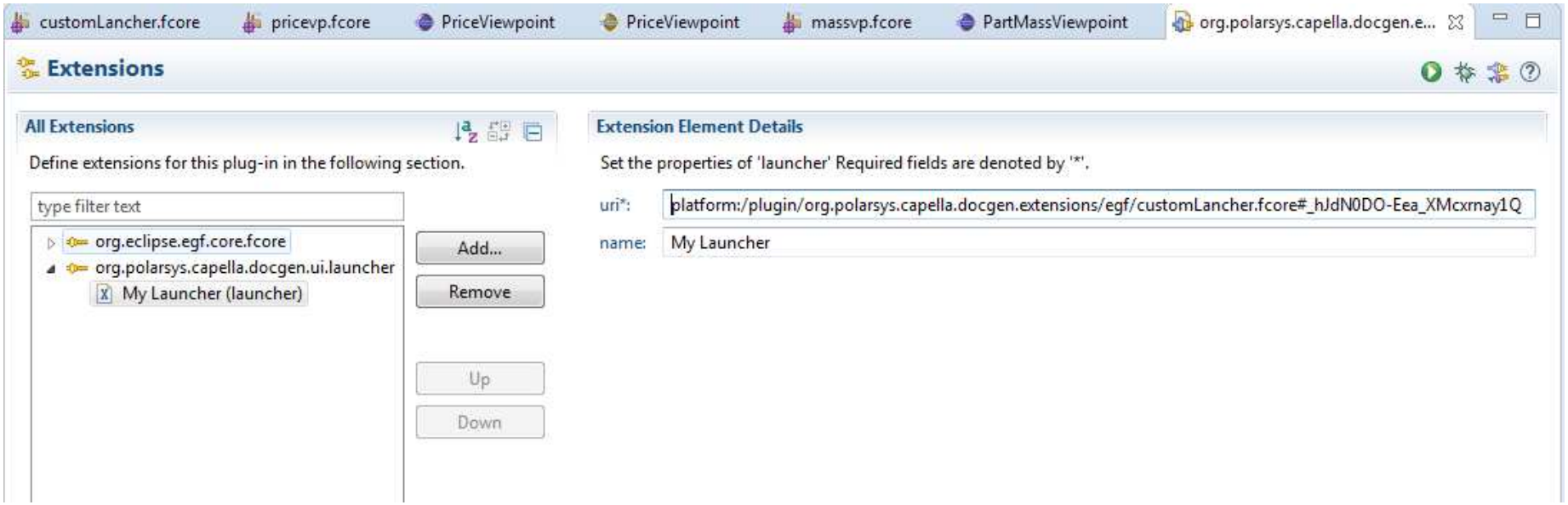
Example – Custom Launcher

Substitution of CapellaDocGenExtensible Pattern

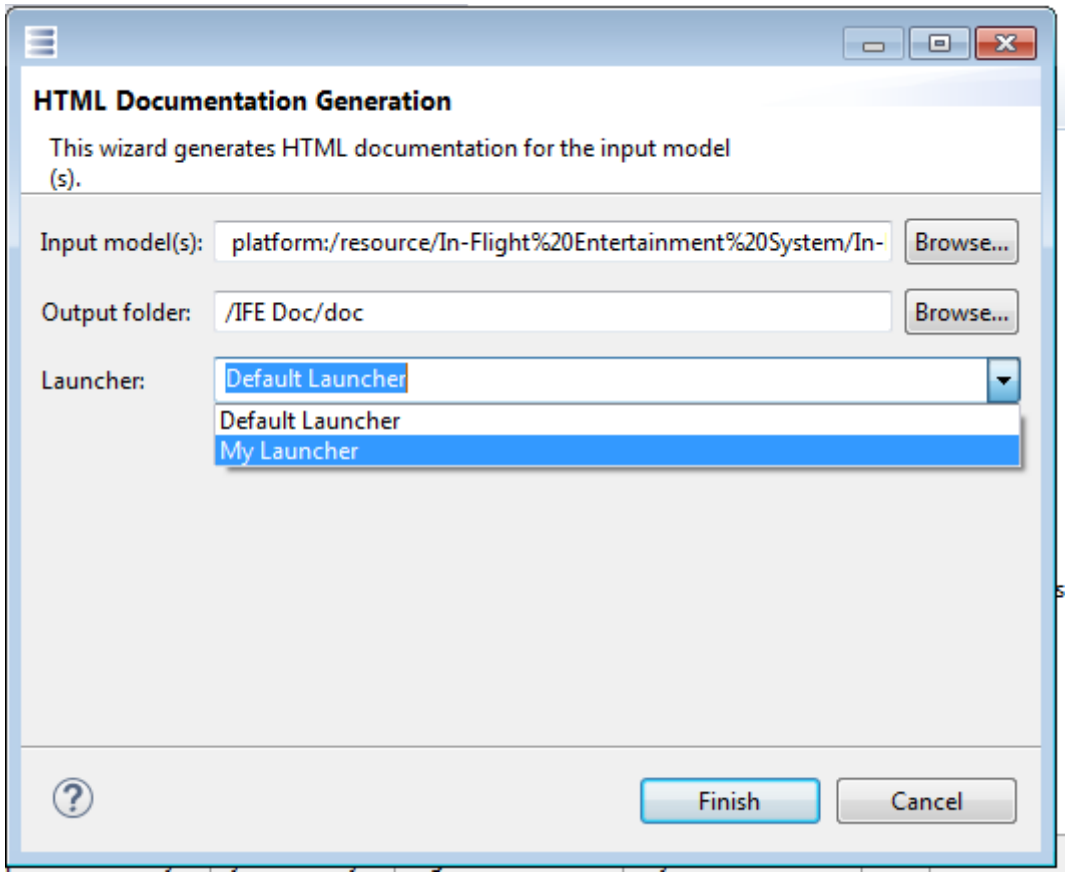


Example – Custom Launcher

Contribution to Capella docGen Launcher



Example – Custom Launcher



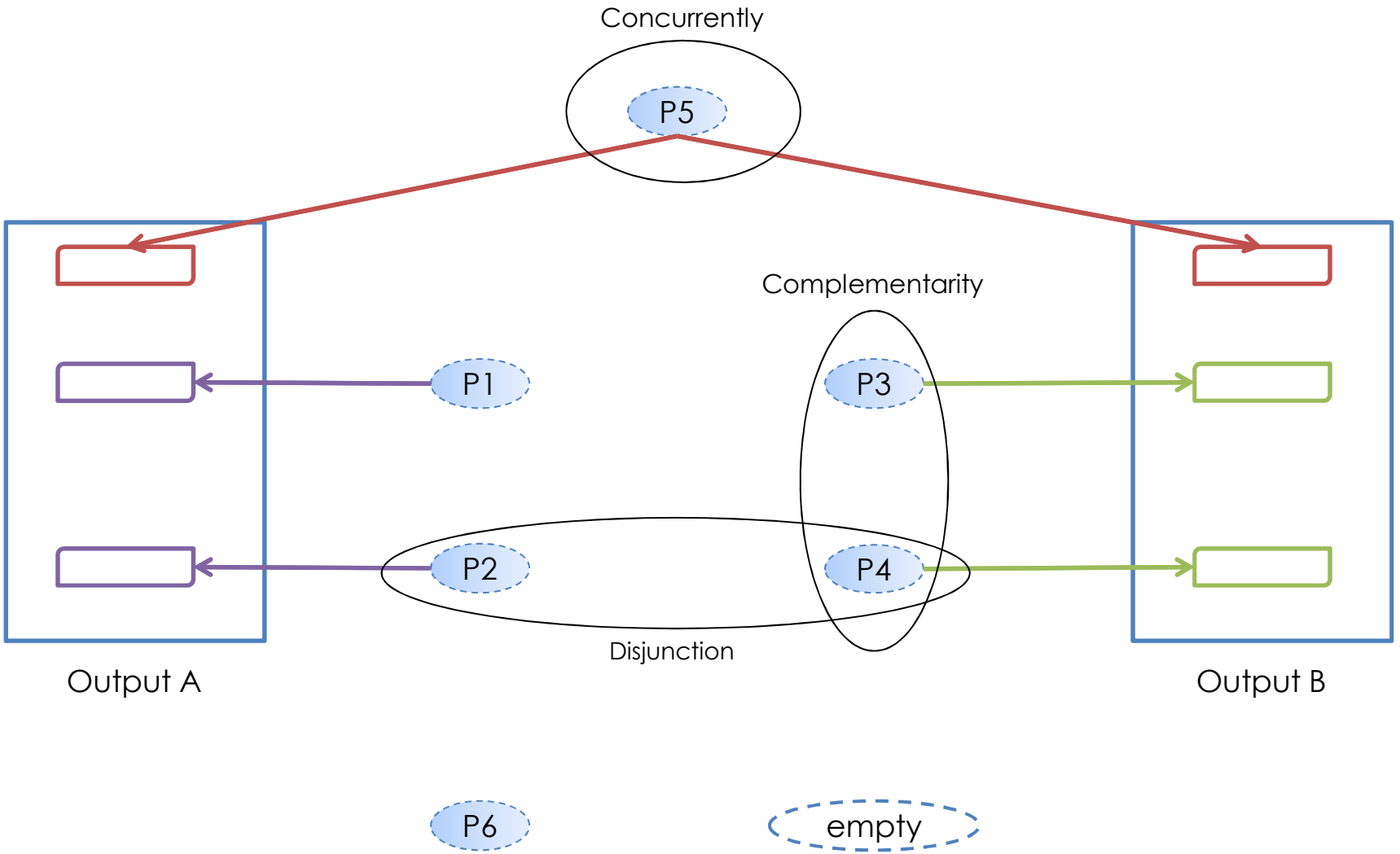
Types of competition when multiples contributions

THALES

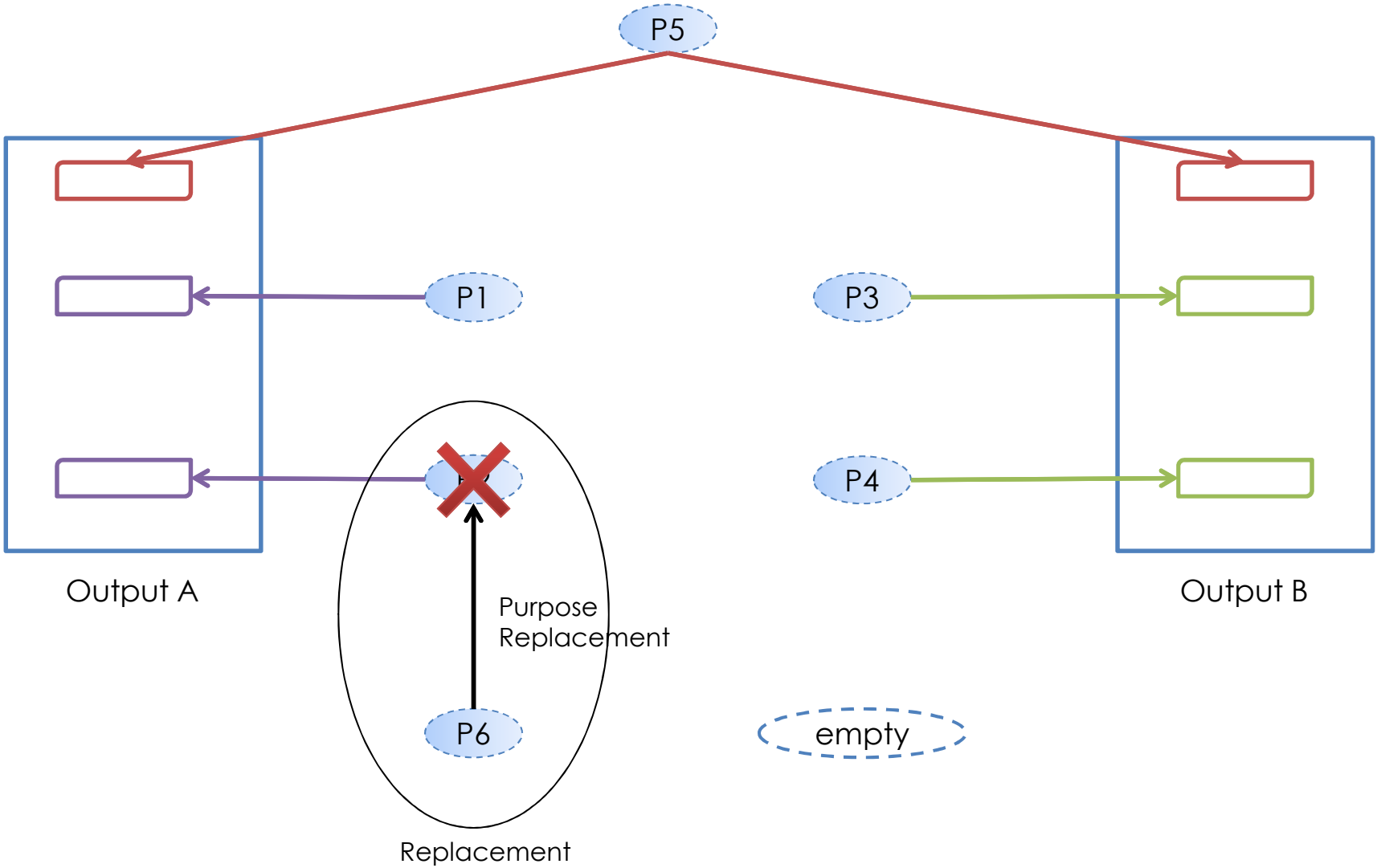
Types of competition when multiples contributions

Disjunction	Two contributions work on different purposes for different outputs. Example: Class and Package contributions
Complementarity	Two contributions work on the different purposes for the same output. Example: Class Attributes and Class Operations contributions
Concurrently	A Contribution works on a purpose for subset or whole of output. Example: adding copyright before header of all output pages
Replacement	A Contribution replaces another one. Replacement may be the order of the production or the purpose of the production Example: adding copyright after last section of Class pages
Annihilation	A contributions erases the effect of another one. Example : the body of a Section is empty

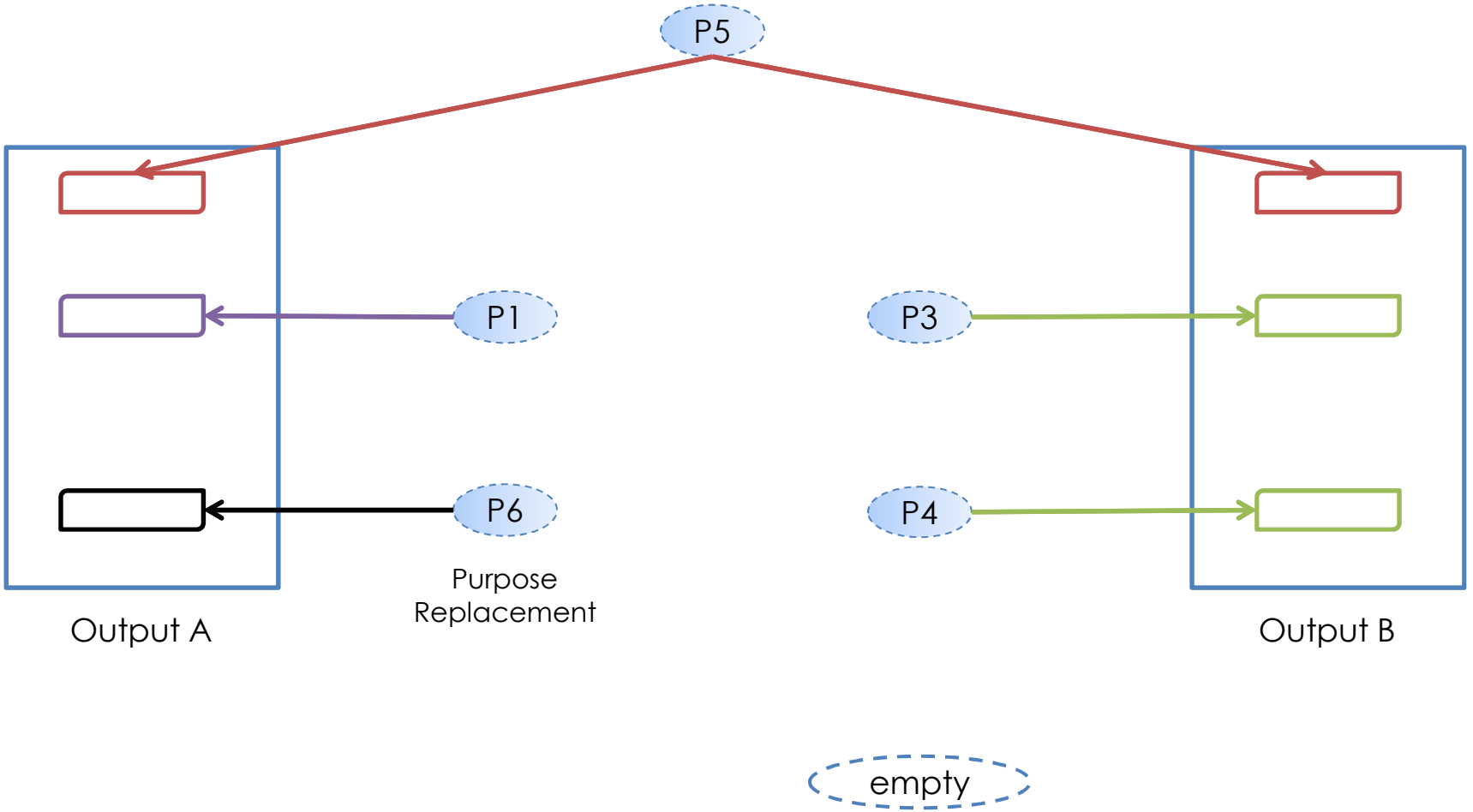
Types of competition when multiples contributions



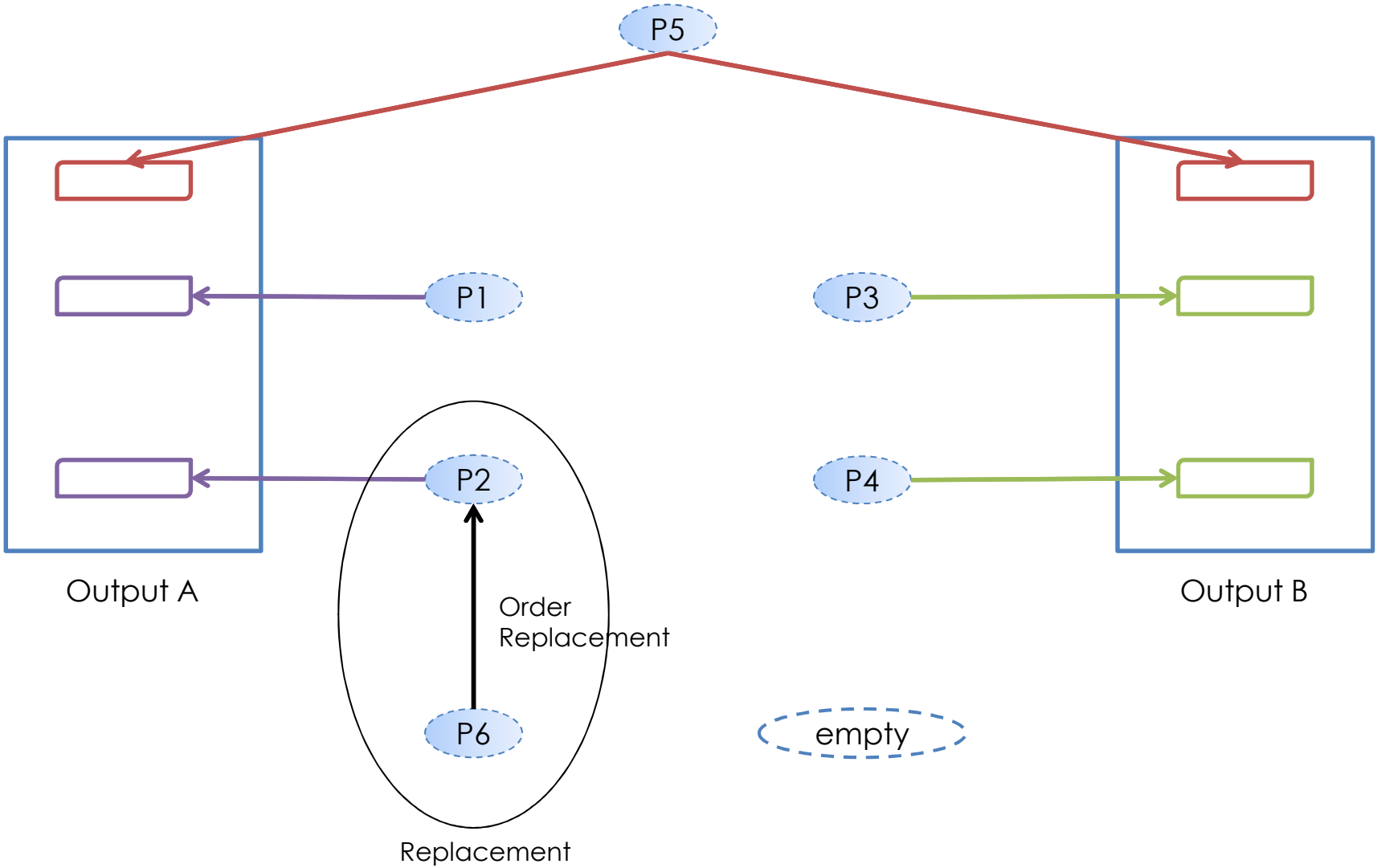
Types of competition when multiples contributions



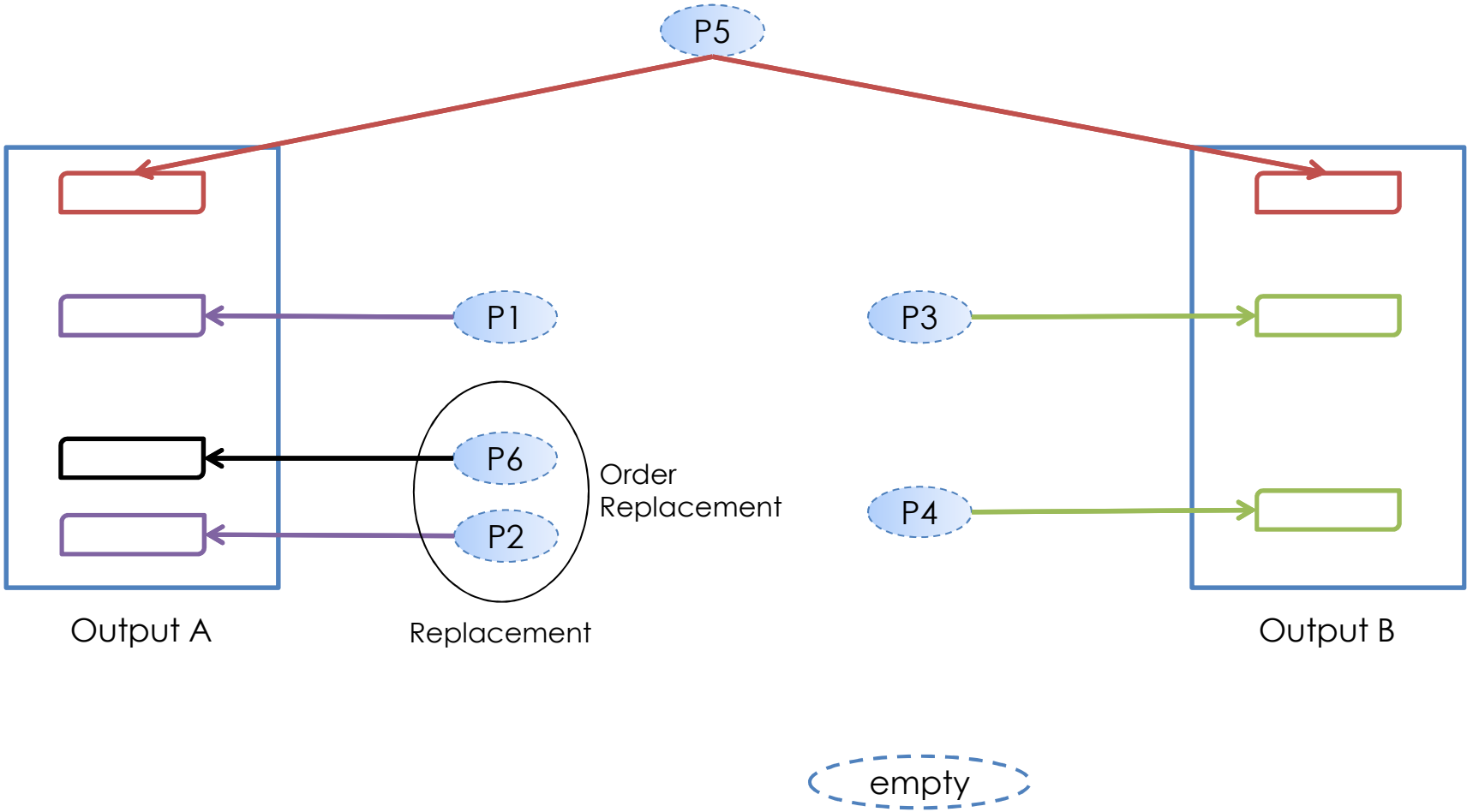
Types of competition when multiples contributions



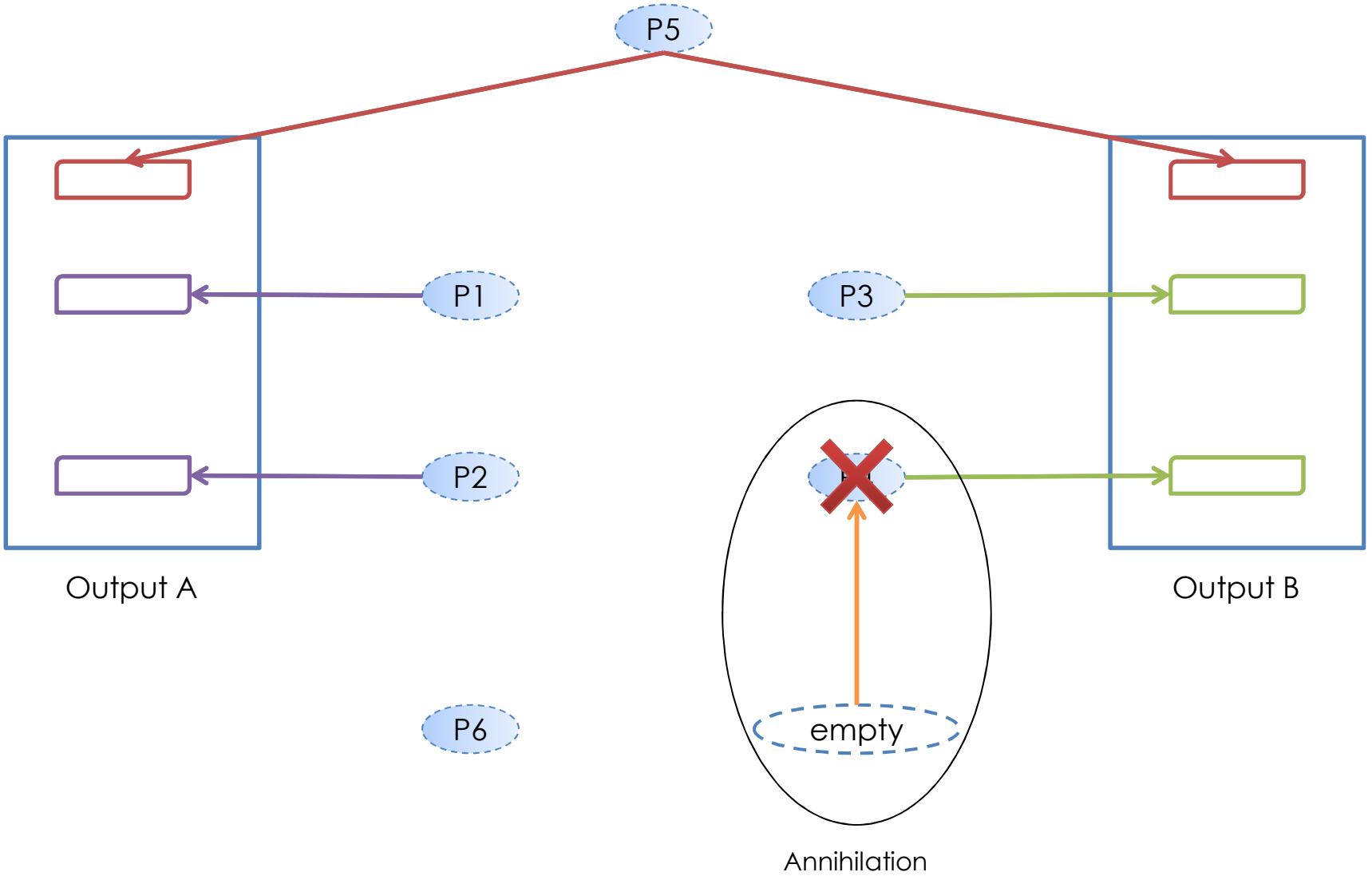
Types of competition when multiples contributions



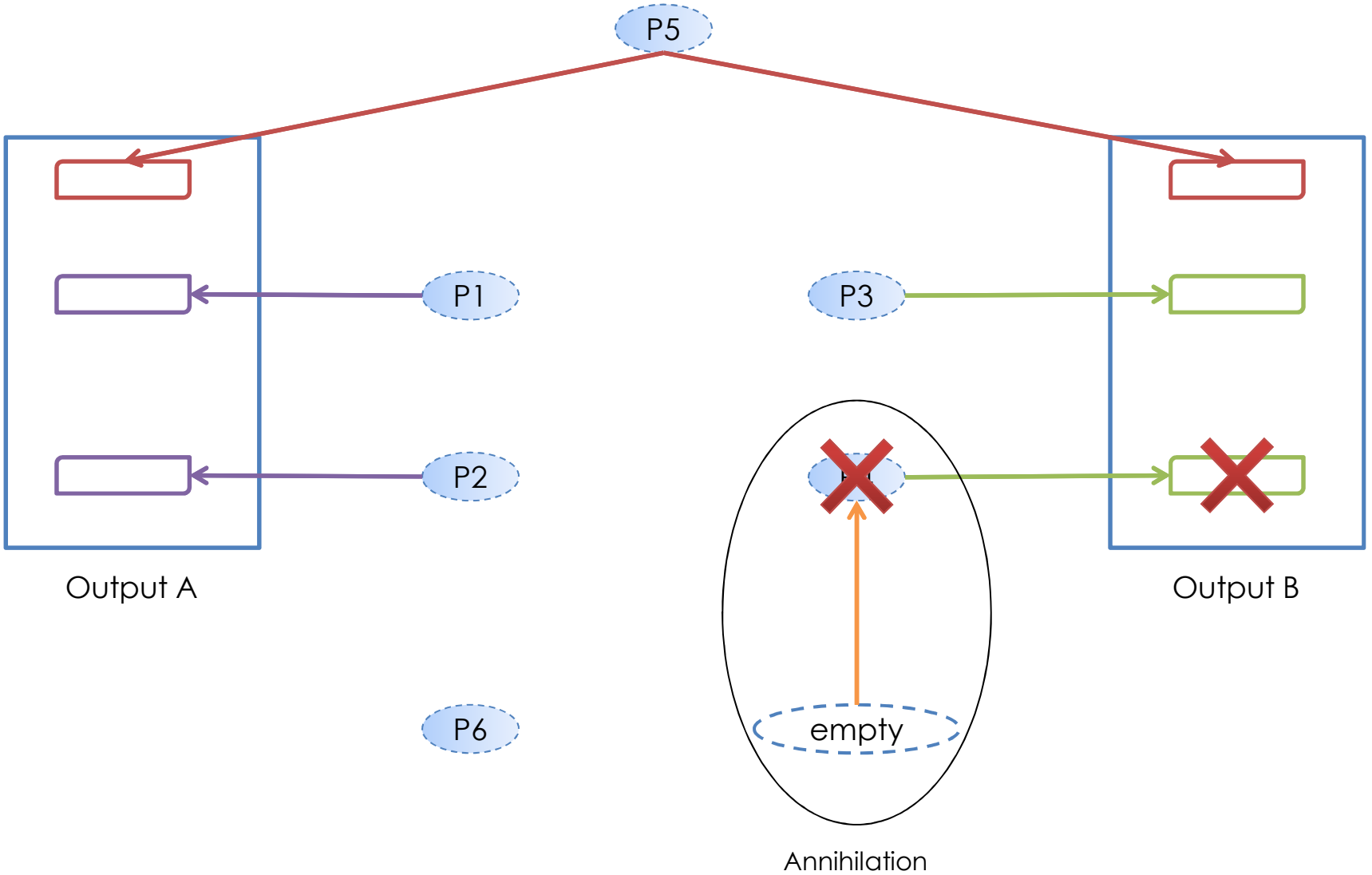
Types of competition when multiples contributions



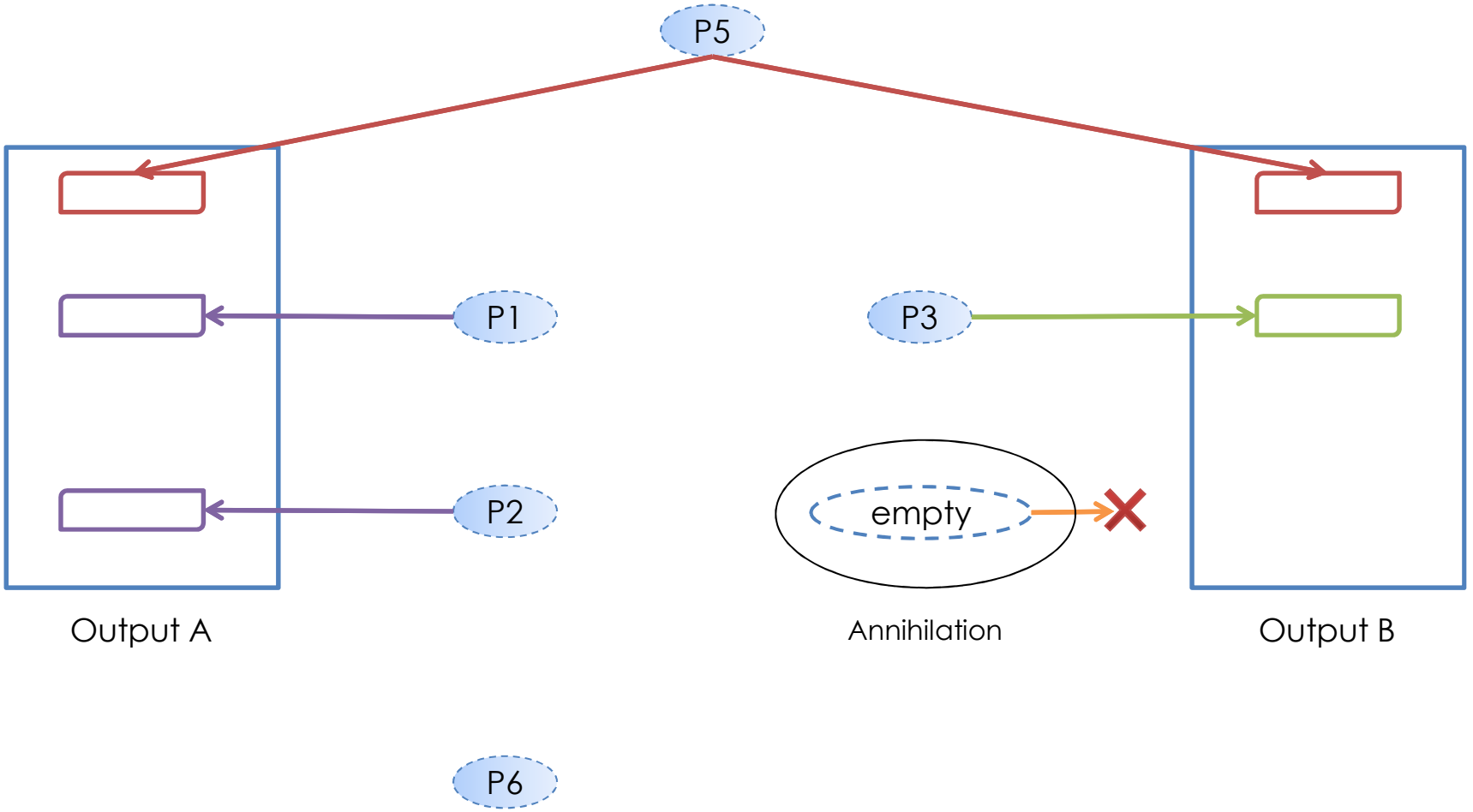
Types of competition when multiples contributions



Types of competition when multiples contributions



Types of competition when multiples contributions



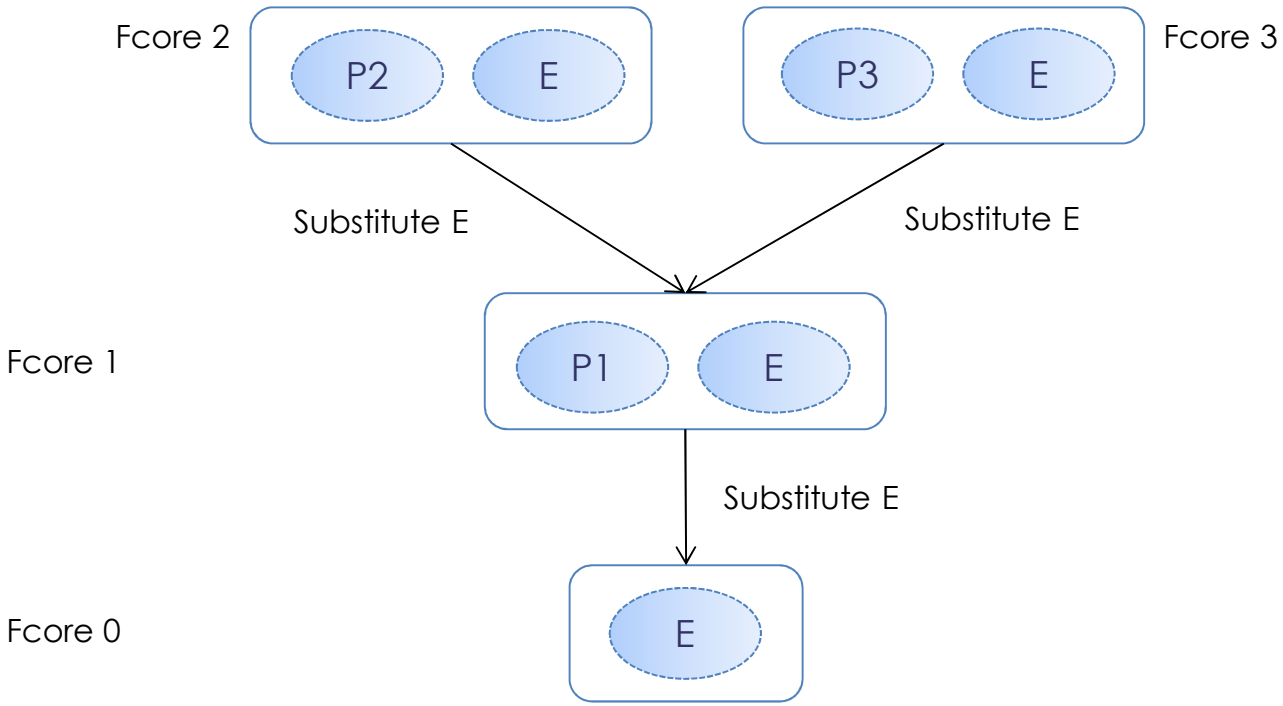
www.thalesgroup.com

Pattern substitution principle

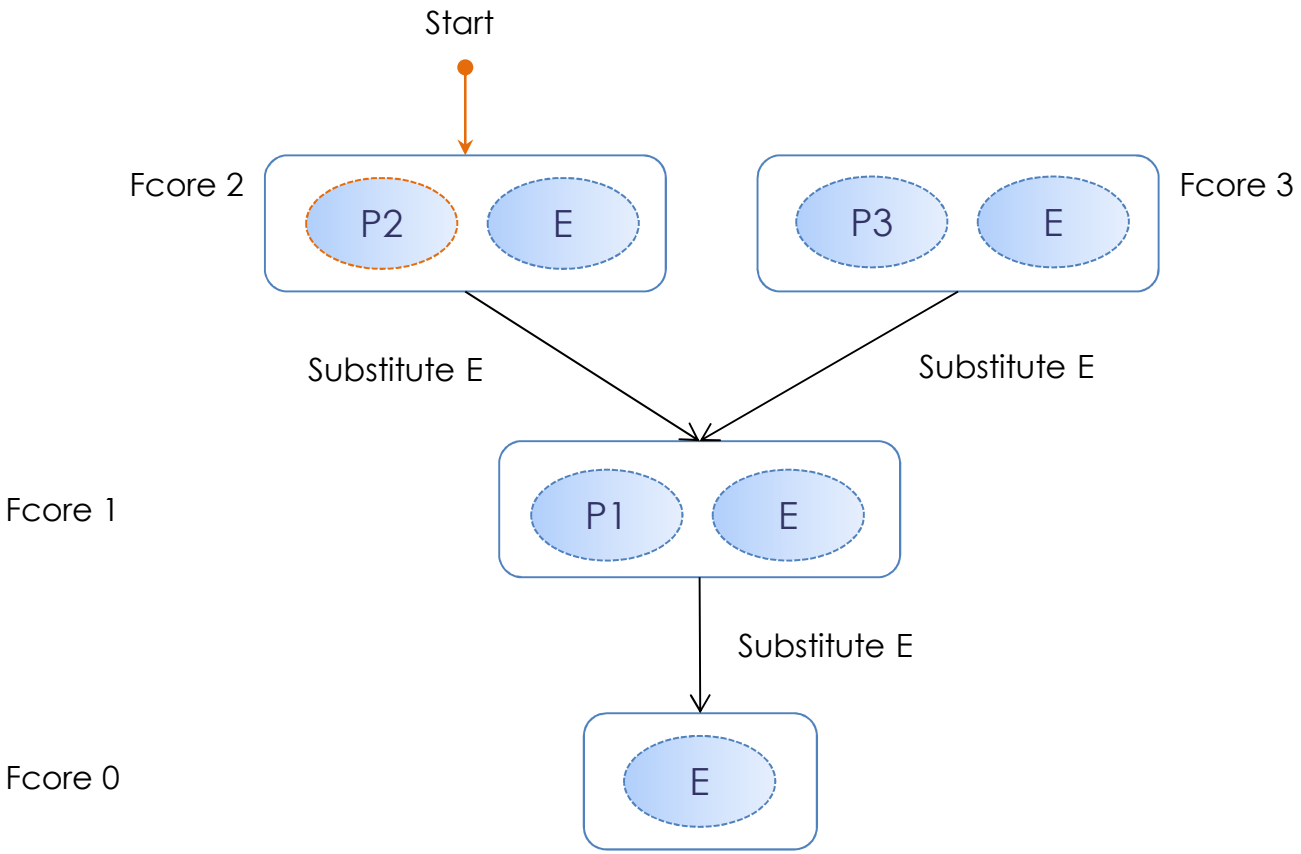
THALES

Concept – Substitution principle

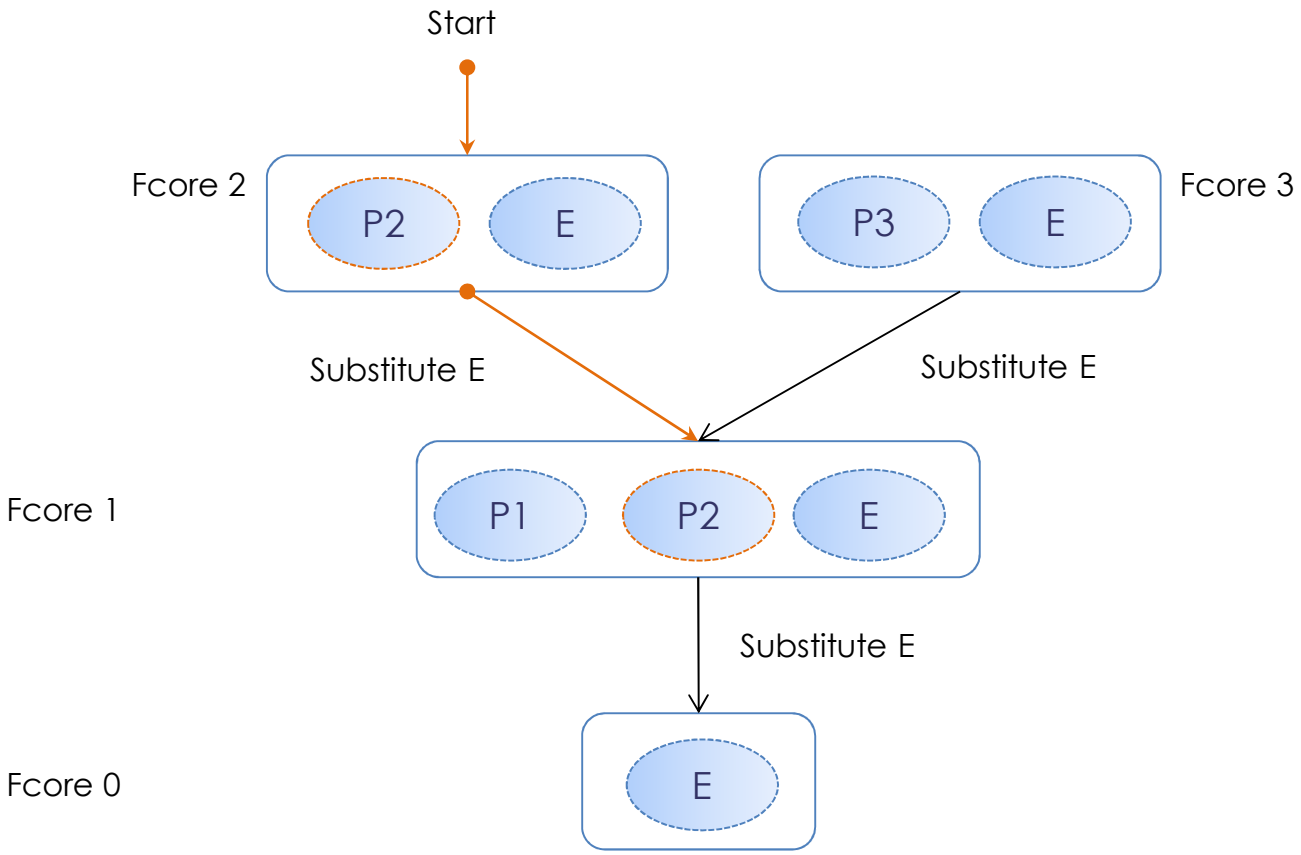
- Add always the empty pattern in the list of replacement patterns



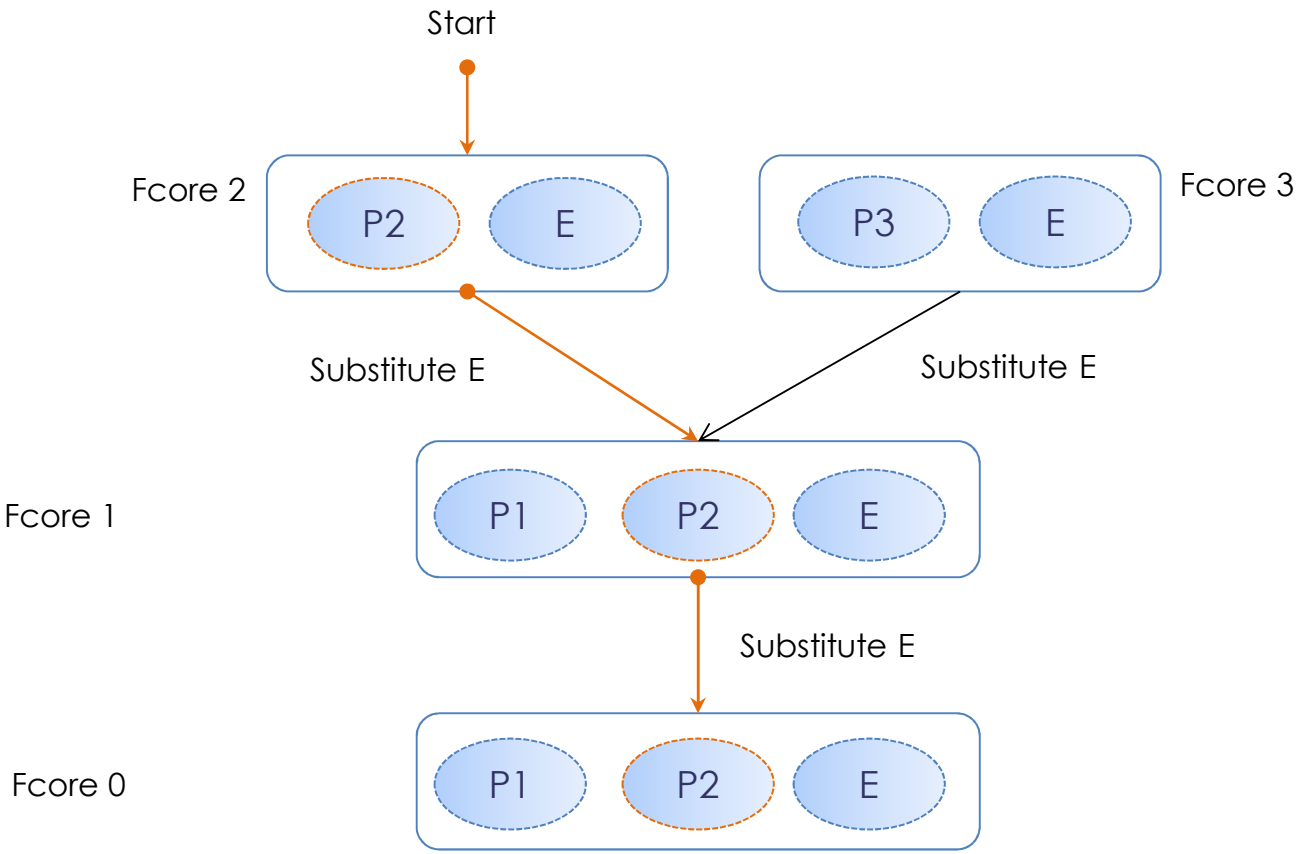
Concept – Substitution principle



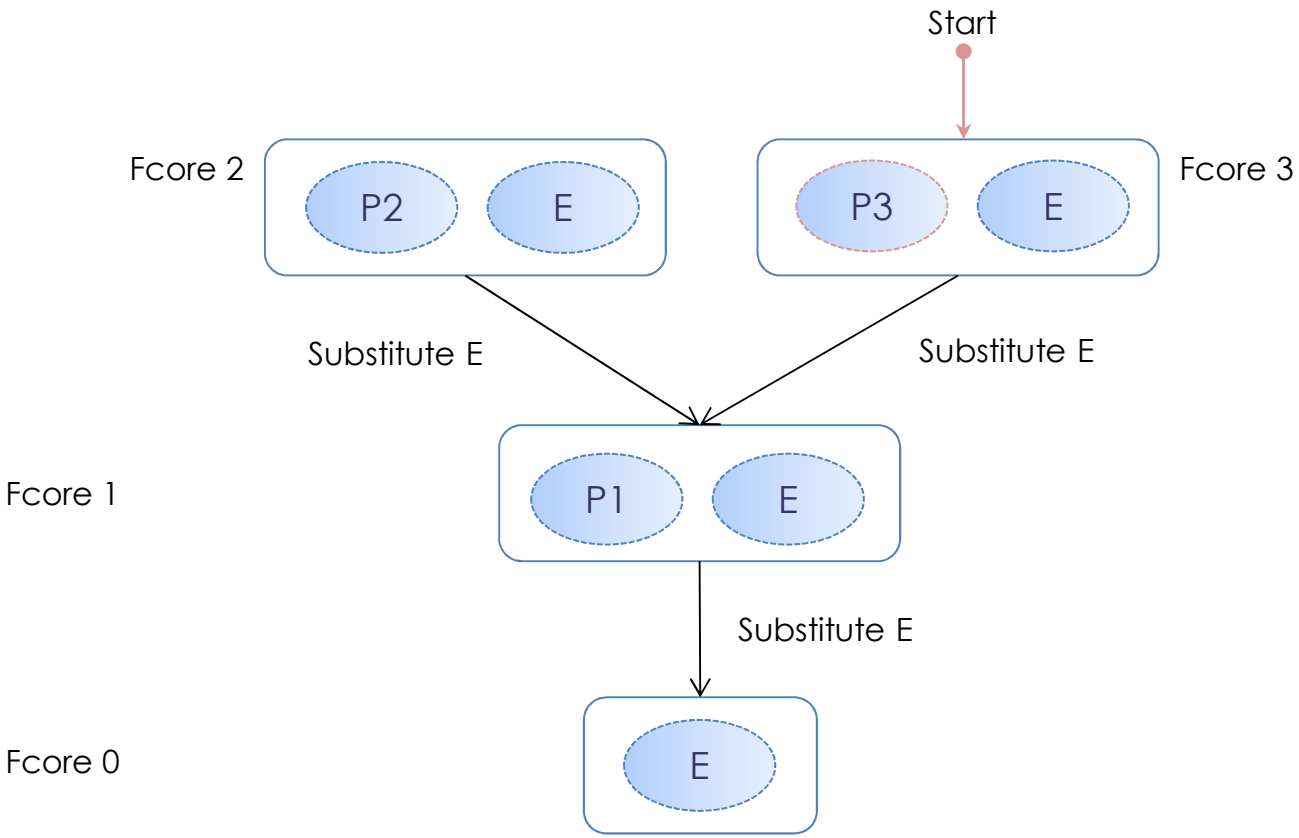
Concept – Substitution principle



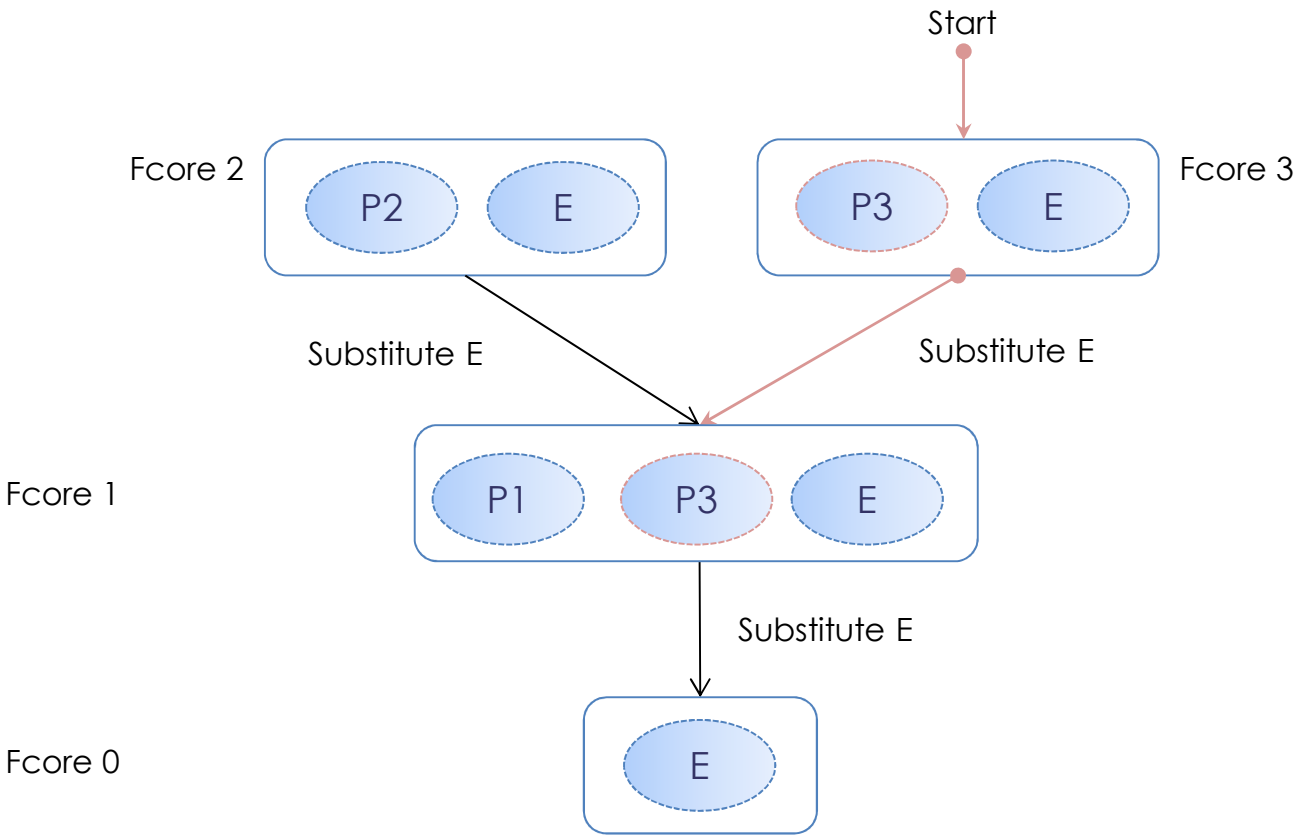
Concept – Substitution principle



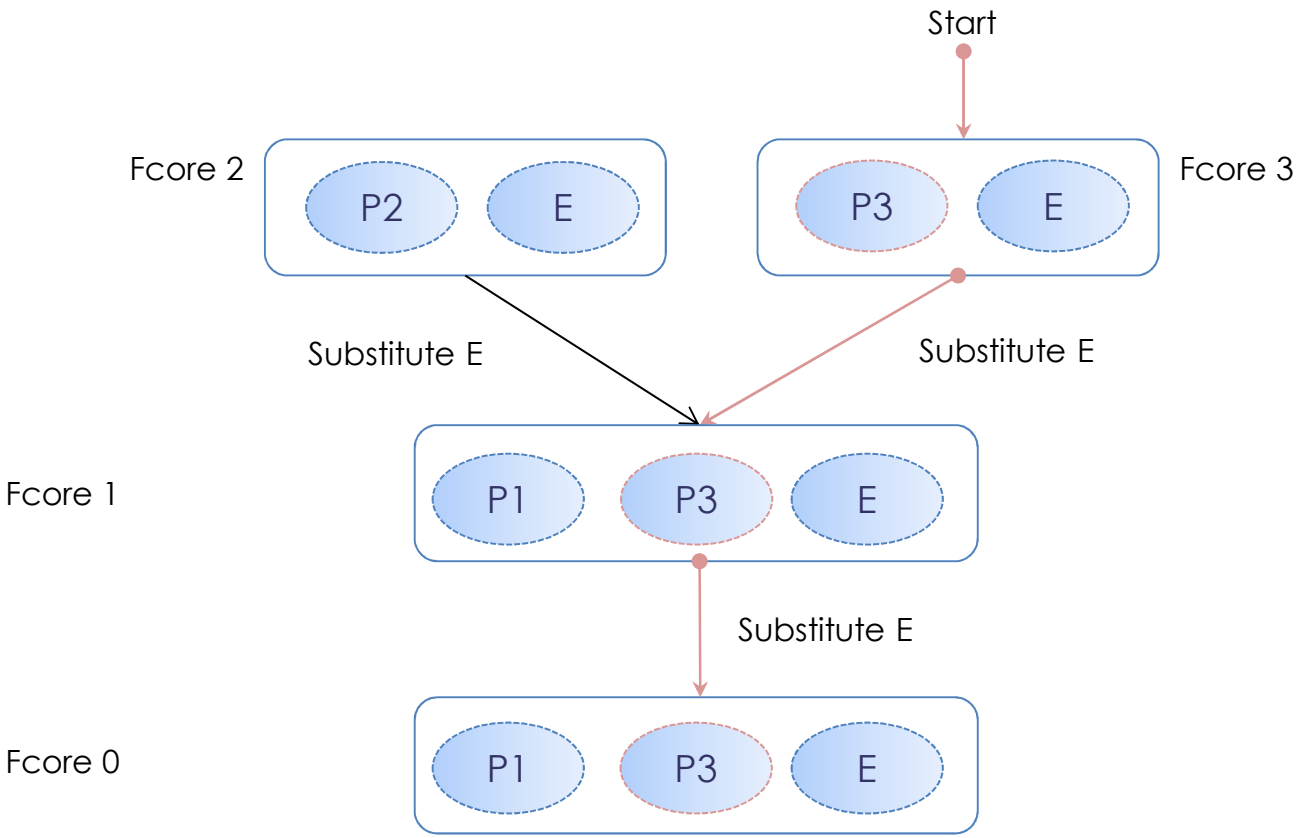
Concept – Substitution principle



Concept – Substitution principle



Concept – Substitution principle



Initiative supported by **Clarity**,
a French collaborative project
http://www.clarity-se.org/

http://polarsys.org/kitalpha/

http://polarsys.org/capella/