

Process Aspect: Handling Crosscutting Concerns during Software Process Improvement

Ya-sha Wang

wangys@sei.pku.edu.cn

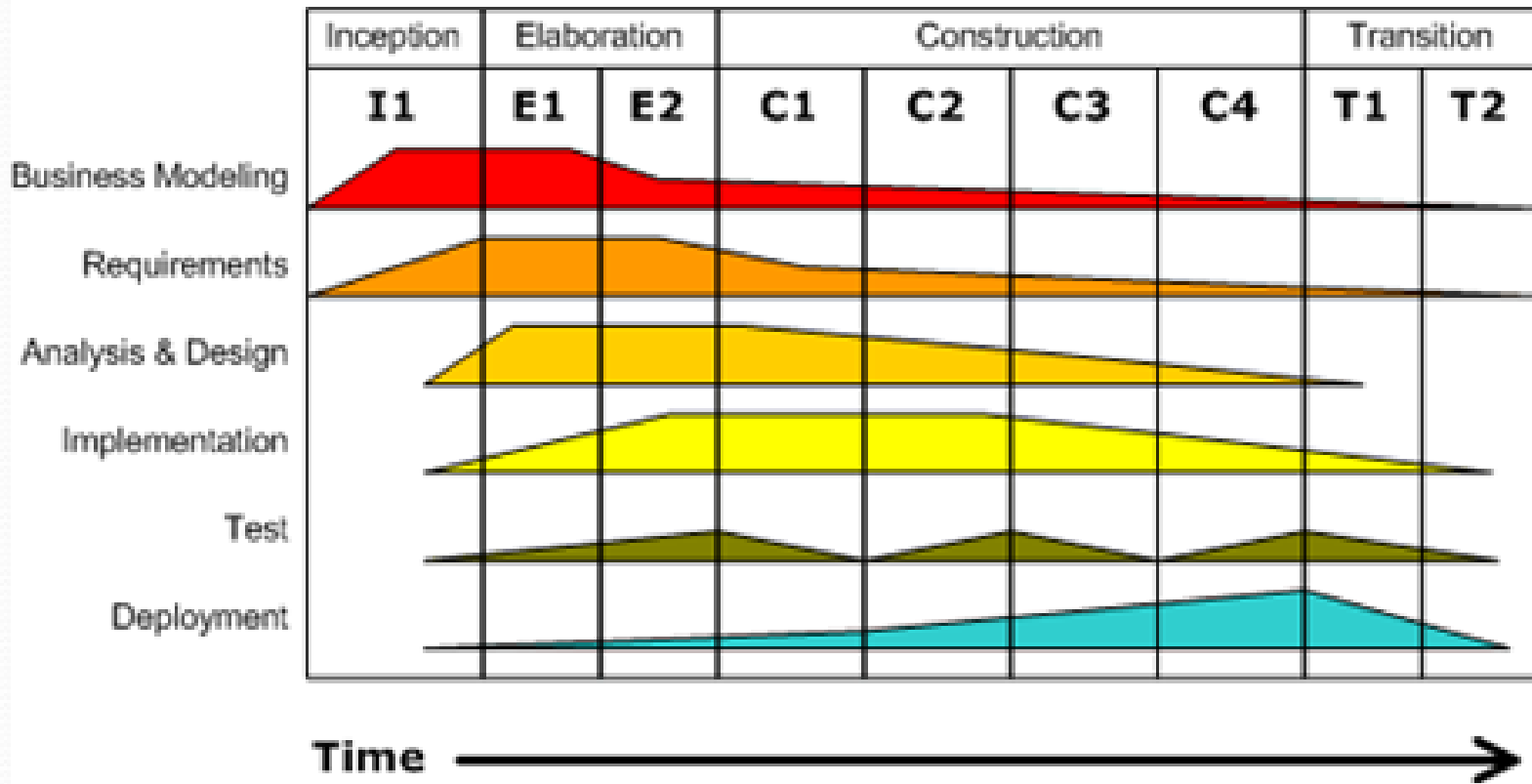
Outline

- **Motivation**
- Our solution
 - Process aspect model
 - Describing a process aspect
- Weaving process aspect into SPEM-based processes
- An example
- Related work

Motivation: New Concerns in SPI

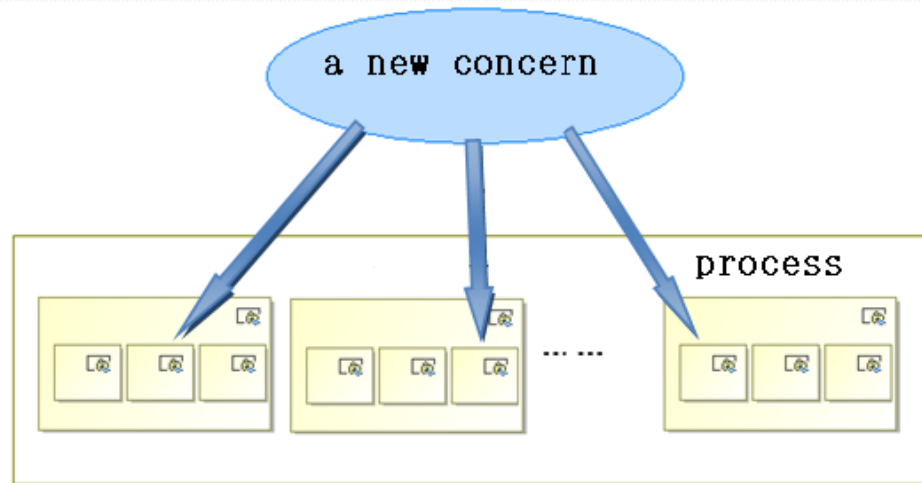
- **Software Process Improvement (SPI)** contains actions taken to **change** an organisation's **processes**, so that they meet the organisation's **business needs** and achieve its **business goals** more effectively - *ISO 15504*
- A frequently emerging situation in SPI...
 - New concerns need to be added
 - Be careful of Intellectual Property (IP) problems
 - Enhance customers' involvement
 - Introduce systematic reuse into process
 - Develop more secure software
 - ...

Motivation: Building Blocks in Software Process



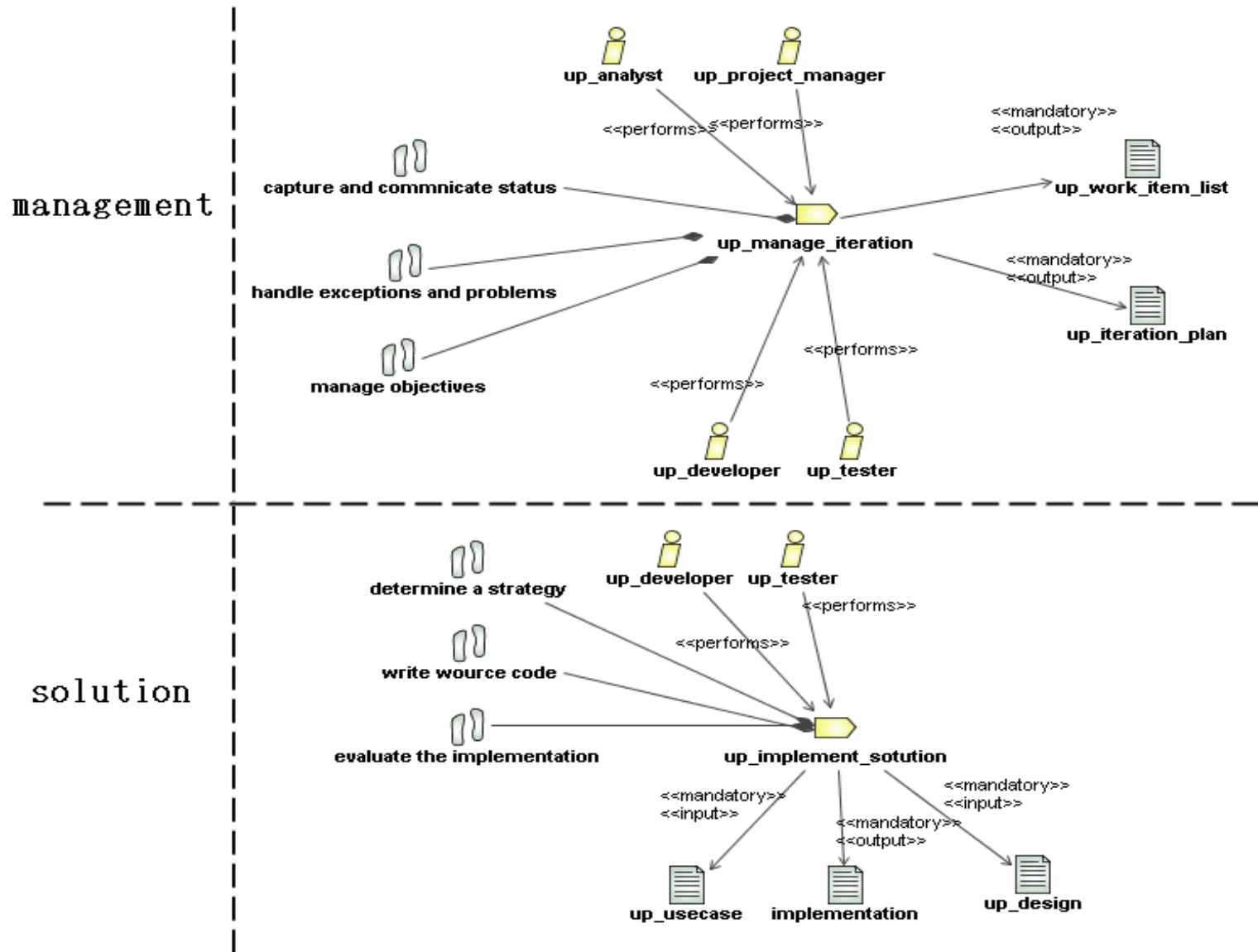
OpenUP

Motivation: Crosscutting Concerns

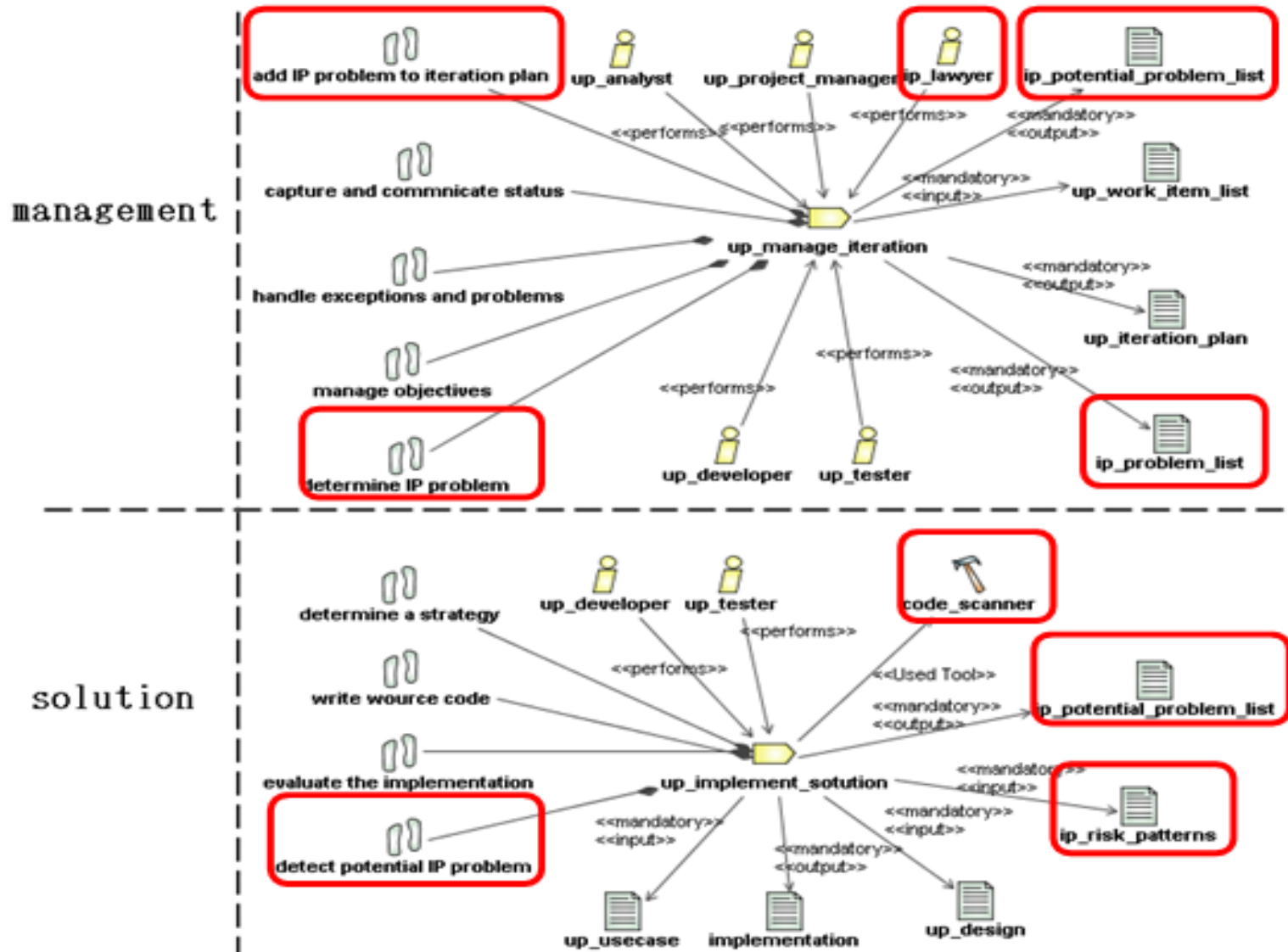


- When implementing certain new concerns...
 - **Changes** usually **spread over many different building blocks** of original processes
 - Be careful of Intellectual Property (IP) problems
 - Enhance customers' involvement
 - Introduce systematic reuse into process
 - Develop more secure software
 - ...

Two Tasks in OpenUP



IP as a Crosscutting Concern



Motivation: What's Wrong

- What if we do NOT handle these crosscutting concerns...
 - How to easily understand the implementation of such a concern
 - What if the **implementation of a concern change** over time ??
 - What if there are **several similar processes** which **also require these concerns** ??
- We need to model the implementation of a crosscutting concern
- Just like it was in software, via Aspect

Outline

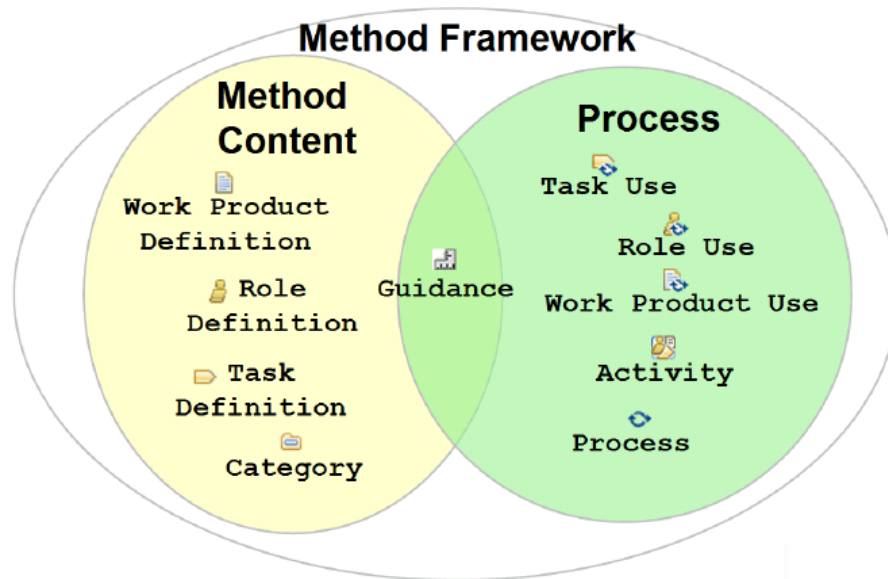
- Motivation
- **Our solution**
 - **Process aspect model**
 - Describing a process aspect
- Weaving process aspect into SPEM-based processes
- An example
- Related work

SPEM 2.0 (2/2)

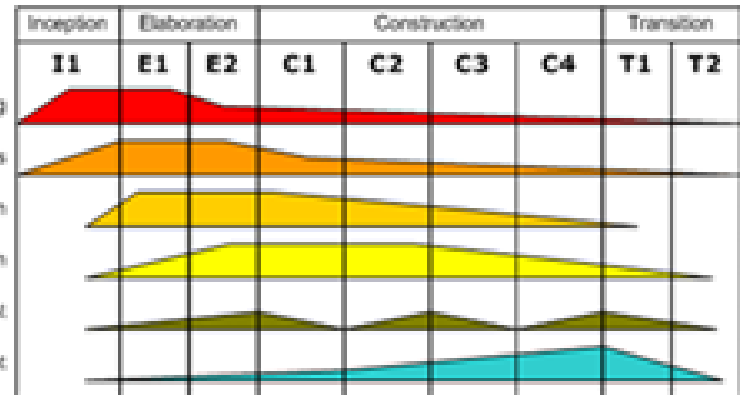
- Software Process Engineering Metamodel (SPEM) 2.0
 - OMG, 2008-4-2
 - 27+ Related Research Papers
 - Strong and Wide Support in Industry



SPEM 2.0 (2/2)



Method Content



Process

Join point

- A join point specifies one possibly changed element in an existing process model when applying a process aspect.
- We define every **task**, **role**, **work product** and **tool** in SPEM2.0 process models can be a join point.
- *AspectJ: The join point model provides the common frame of reference that makes it possible for execution of a program's aspect and non-aspect code to be properly coordinated*
 - *Method Call*
 - *Event*
 - *Reading / Writing a Field*

Pointcut

- Pointcut is a set of join points on which a certain advice should apply, which results from certain filtering among join points
- We define 4 designators to specify the filtering rules, for example:
 - *tasks_designators*: $tasks_criteria \rightarrow \{e \mid e.type = Task\}$
 - $tasks_criteria = \{(f,v) \mid (f, v.type) \in \{(name, string), (description, string), (step, Step), (performer, Role), (input, WorkProduct), (output, WorkProduct), (used\ tool, Tool)\}\}$

$\forall e \in tasks_designators((f_1, v_1), (f_2, v_2), \dots, (f_n, v_n)),$ we have $e.f_i = v_i, i \in [1, n]$
- *AspectJ*: A pointcut is a set of join points, plus, optionally, some of the values in the execution context of those join points
 - by parameter
 - by return value
 - by literal

Advice

- Advice defines which operations should be taken.
- An operation is represented as (a,p) , where a denotes the action to be taken, and p denotes the parameter for the action. Formally, we can define Advice as:

$Advice = \{(a,p) \mid (a, p.type) \in \{(add_attribute, string), (add_role, Role), (add_workproduct, WorkProduct), (add_tool, Tool), (add_task_before, Task), (add_task_after, Task), (add_unordered_task, Task)\}\}$

- *AspectJ: Advice is a method-like mechanism used to declare that certain code should execute at each of the join points*
 - *Jave Code Snippet*

Correspondence Cases

Meaningful correspondence cases between a pointcut type and operations in its advice.

Pointcut Type	Operation	Meaning
Task	(add_attribute,x)	Add a new attribute x for an existing task
Task	(add_role,x)	Relate a new role x with an existing task
Task	(add_workproduct, x)	Relate a new work product x with an existing task
Task	(add_tool, x)	Relate a new tool x with an existing task
Task	(add_task_before, x)	Add a new task x before an existing task
Task	(add_task_after, x)	Add a new task x after an existing task
Task	(add_unordered_task, x)	Add a new unordered task x into an existing task
Role	(add_attribute,x)	Add a new attribute x for an existing role
Role	(add_task, x)	Relate a new role x with an existing role
Role	(add_workproduct, x)	Relate a new work product x with an existing role
WorkProduct	(add_attribute x)	Add a new attribute x for an existing work product
WorkProduct	(add_task, x)	Relate a new task x with an existing work product
WorkProduct	(add_role, x)	Relate a new role x with existing work product
WorkProduct	(add_workproduct, x)	Relate a new work product x with an existing work product
Tool	(add_attribute,x)	Add a new attribute x for an existing tool
Tool	(add_task, x)	Relate a new task x with an existing tool

Outline

- Motivation
- **Our solution**
 - Process aspect model
 - **Describing a process aspect**
- Weaving process aspect into SPEM-based processes
- An example
- Related work

XML Schema for Process Aspect

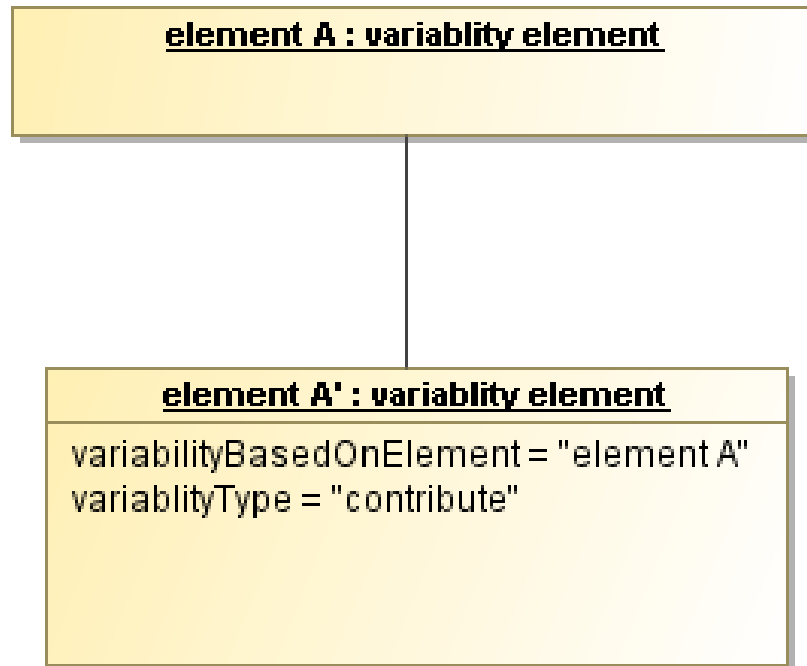
```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:spem="ProcessDefinitionSchema"
targetNamespace="http://www.w3school.com.cn" xmlns="http://www.w3school.com.cn">
<xs:element name="ProcessAspect">
  <xs:complexType>
    <xs:element name="PointcutAndAdvice" maxOccurs="unbounded">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="pointcut">
            <xs:complexType>
              <xs:element name="_name" type="xs:string"/>
              <xs:element name="_type" type="xs:string"/>
              <xs:element name="_attribute" type="xs:string"/>
              <xs:element name="_parent" type="spem:element"/>
              <xs:element name="_child" type="spem:element"/>
            </xs:complexType>
          </xs:element>
          <xs:element name="advice">
            <xs:complexType>
              <xs:element name="add_attribute" type="spem:attribute"/>
              <xs:element name="add_role" type="spem:role"/>
              <xs:element name="add_artifact" type="spem:artifact"/>
              <xs:element name="add_tool" type="spem:tool"/>
              <xs:element name="add_task_before" type="spem:task"/>
              <xs:element name="add_task_after" type="spem:task"/>
              <xs:element name="add_unordered_task" type="spem:task"/>
              <xs:element name="add_relation" type="spem:relation"/>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:complexType>
</xs:element></xs:schema>
```

Outline

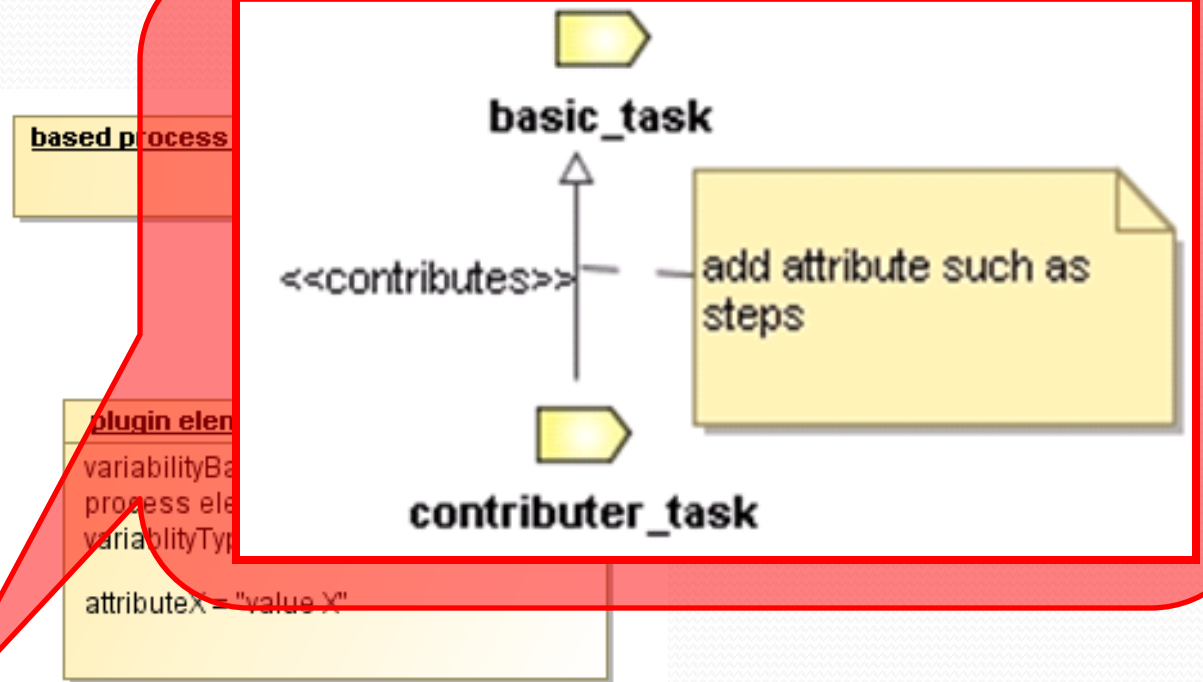
- Motivation
- Our solution
 - Process aspect model
 - Describing a process aspect
- **Weaving process aspect into SPEM-based processes**
- An example
- Related work

Leveraging Process Aspect into SPEM-based Processes

We leverage the VariabilityElement class (in the method plugin package of SPEM2.0) to serve as the infrastructure for weaving process aspect.



Mapping Pattern 1



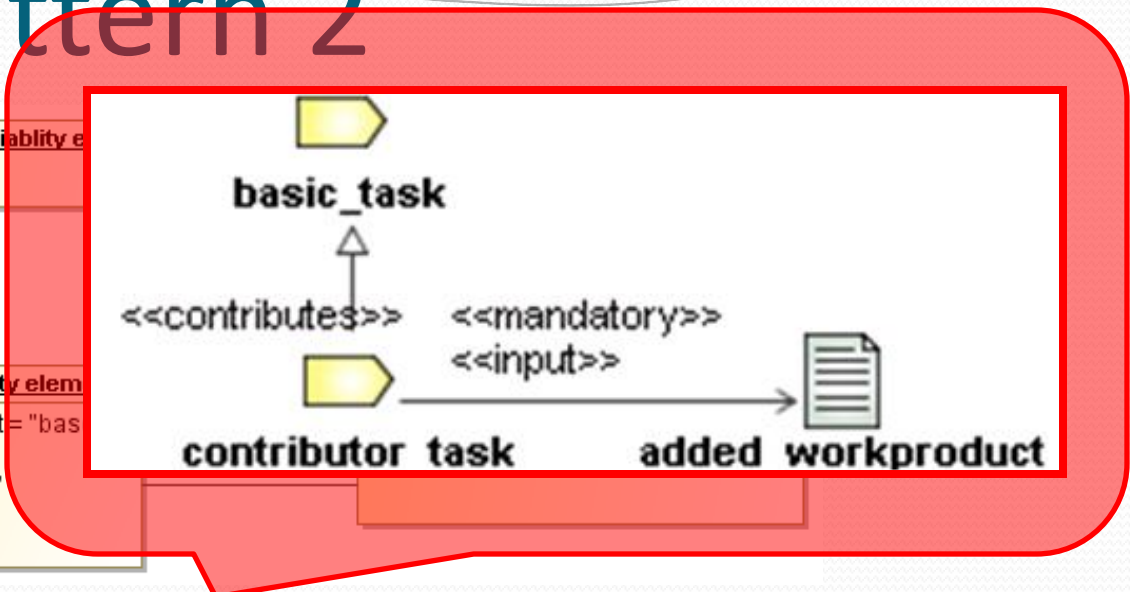
Add new attribute for existing task .	Add new attribute for existing role .	Add new attribute for existing product .	Add new attribute for existing tool .

Mapping Pattern 2

based process element A : variability element

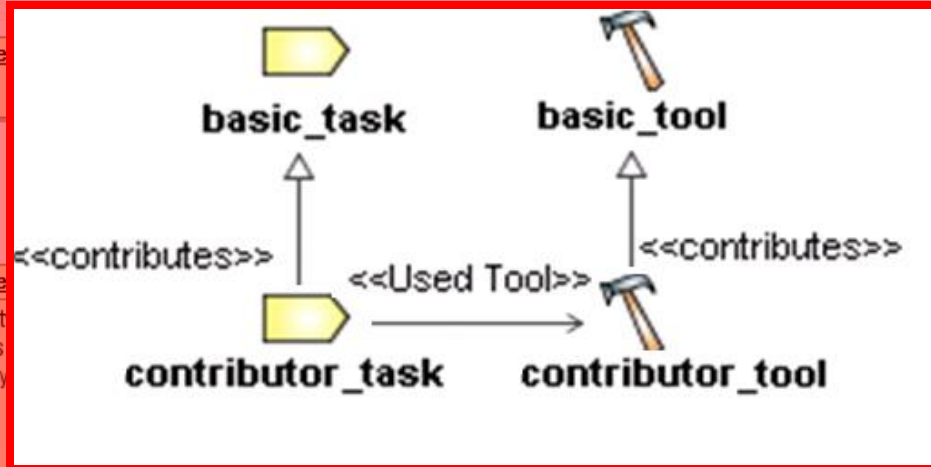
plugin element A' : variability element

variabilityBasedOnElement = "based process element A"
variabilityType = "contribute"



Relate new role with existing task .	Relate new artifact with existing task .	Relate new tool with existing task .	Relate new role with existing artifact .
Relate new task with existing role .	Relate new task with existing artifact .	Relate task role with existing tool .	Relate new role with existing artifact .
Add task after existing task .	Add task before existing task .	Add task in existing activity .	Relate new role with existing task .

Composing Basic Patterns

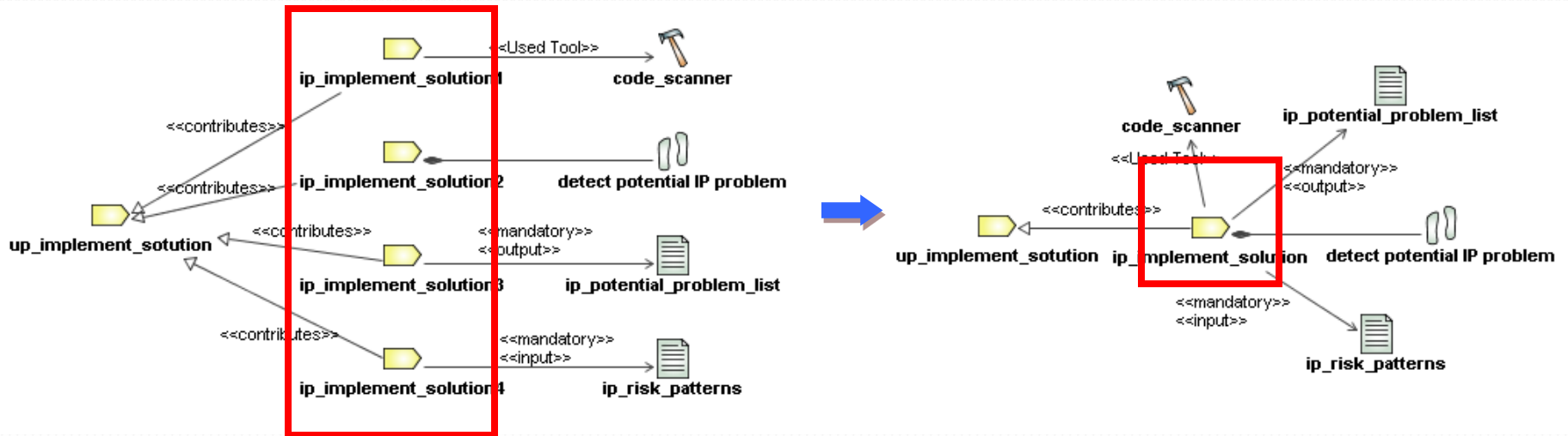


= Pattern 1 + 2

<p>Add new relation between existing task and role . .</p>	<p>Add new relation between existing task and artifact . .</p>	<p>Add new relation between existing task and tool . .</p>
<p>Add new relation between existing role and artifact . .</p>	<p>Add new relation between two existing artifact . .</p>	<p>Add new relation between two existing tasks . .</p>

Dealing with Redundancy

- A simple example



The Merge Algorithm

Merge Algorithm

Let E : list of elements in the aspect-method-plugin-package

Let $e.contributee$: the element pointed by e with a 'contribute' relation

Let $e.addProperty(p)$: adding property p into e

Let $relation(e1, e2)$: relation between $e1$ and $e2$

Let $e1.addRelatedElement(e2, r)$: adding relation r between $e1$ and $e2$

Let $E.deleteElement(e)$: deleting element e itself and all the links from e or to e

s1: For each $e1$ of E

s2: For each $e2$ of E

s3: If $(e1 \neq e2)$ and $(e1.contributee = e2.contributee)$

s4: For each p of $e2.properties$

s5: $e1.addProperty(p)$

s6: For each re of $e2.relatedElements$

s7: $e1.addRelatedElement(re, relation(e2, re))$

s8: $E.deleteElement(e2)$

Proof 1: Semantics of E equal Semantics of E'

Let E': list of elements in the aspect-method-plugin-package after Merge.

Proof 1: Semantics of E equal Semantics of E'

For \forall property p contributed from E to M, let $p \in x$, x is an element of E.

During Merge,

- (a) If x acts as $e1$ in $s3$, obviously $s4$ to $s8$ do not touch p ;
- (b) If x acts as $e2$ in $s3$, $s5$ will add p into $e1$ and $e1.contributee = M$, p is still contributed from E' to M;
- (c) If x does not satisfy $s3$, obviously there is no other place in Merge that changes p

So, p is still contributed from E' to M

In the same way, we can have:

for \forall related element re contributed from E to M, re is still contributed from E' to M

Meanwhile,

E' does not introduce any new property or related element that does not belong to E

As a result, Semantics of E equal Semantics of E'

Proof 2: E' has the least simplicity

There are two types of elements in E: some that directly contribute to the original process (let them be contributors), and others that act as related elements of contributors (let them be related_elements). If we define the simplicity of E as number of contributors + number of related elements, we can prove that E' has the least simplicity

Proof 2: E' has the least simplicity

Proof 2: E' has the least simplicity

In E', the number of contributor = the number of contributee, which is the possible least. Otherwise there will exist one contributee without any contributor, which will change the semantics of the whole Aspect.

Meanwhile, s7 and s8 do not change the number of related_elements, so the number of related_elements stays unchanged after Merge.

Therefore, we can get that the sum of contributor number and related_elements number, which is the simplicity defined above, has the least possible value in E'.

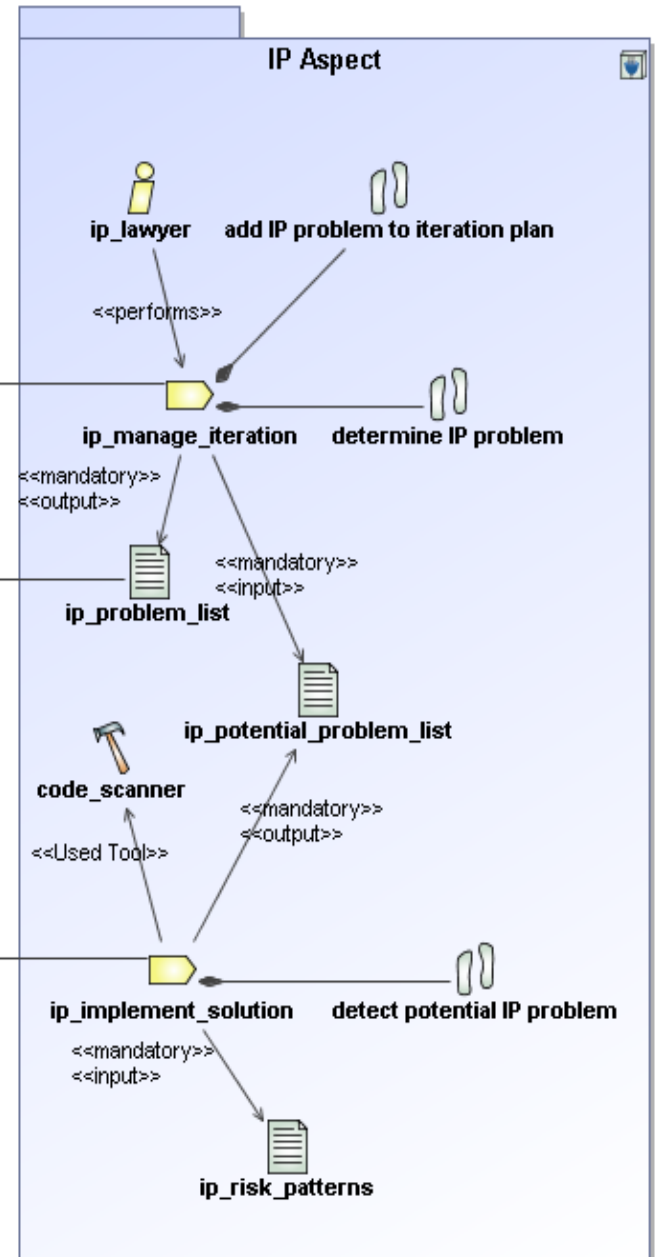
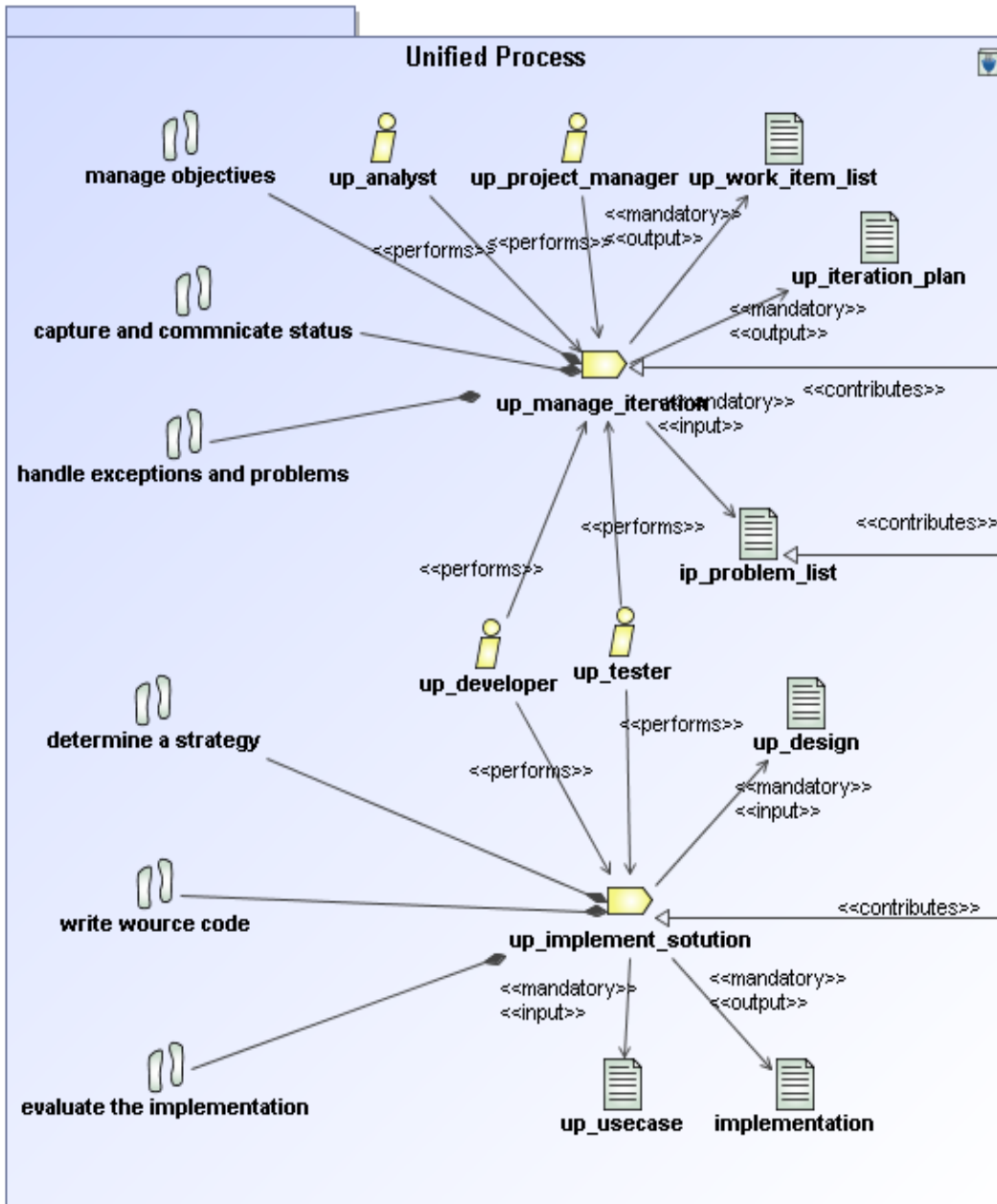
That is to say, E' has the least simplicity

Outline

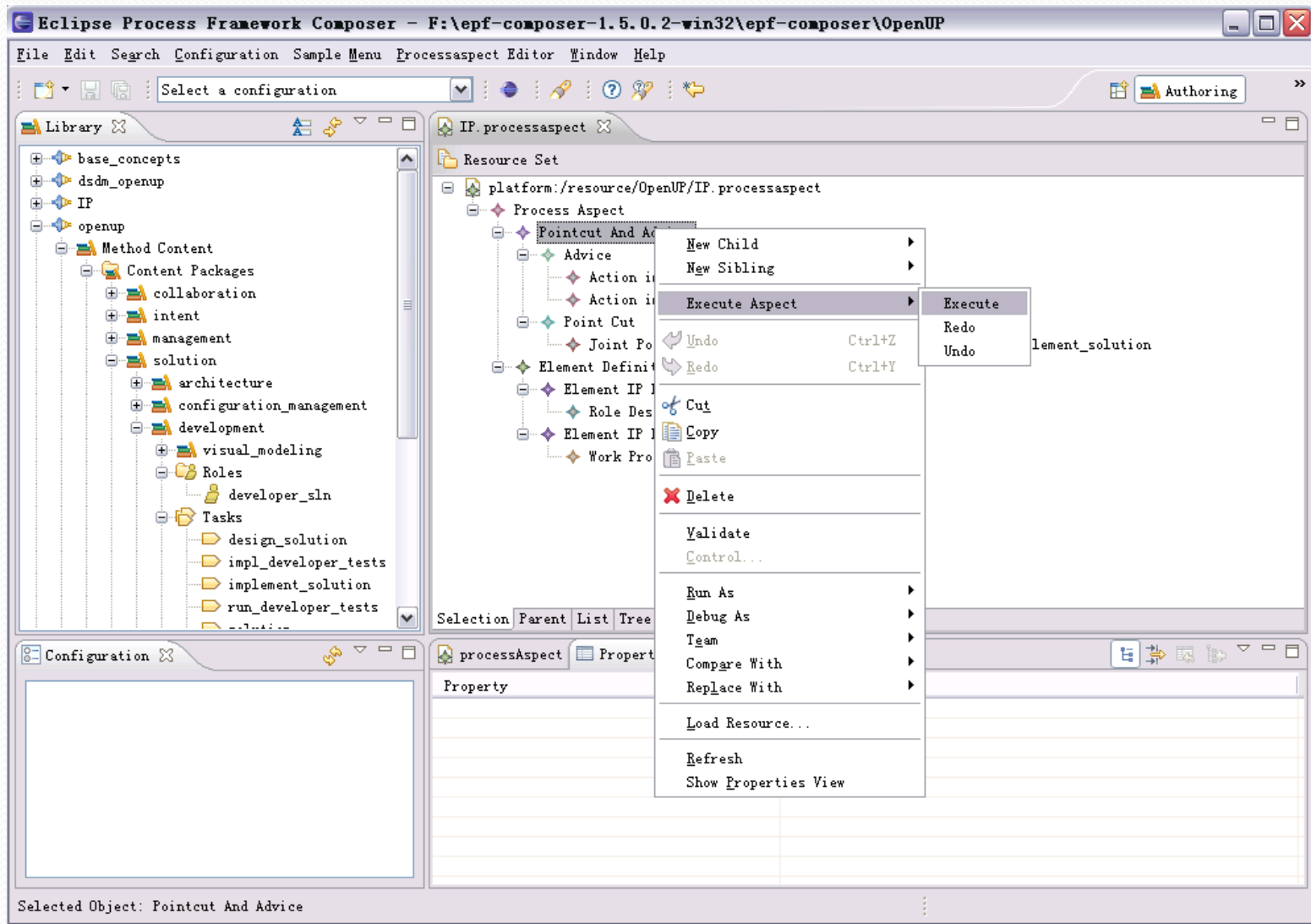
- Motivation
- Our solution
 - Process aspect model
 - Describing a process aspect
- Weaving process aspect into SPEM-based processes
- **An example**
- Related work

An Example

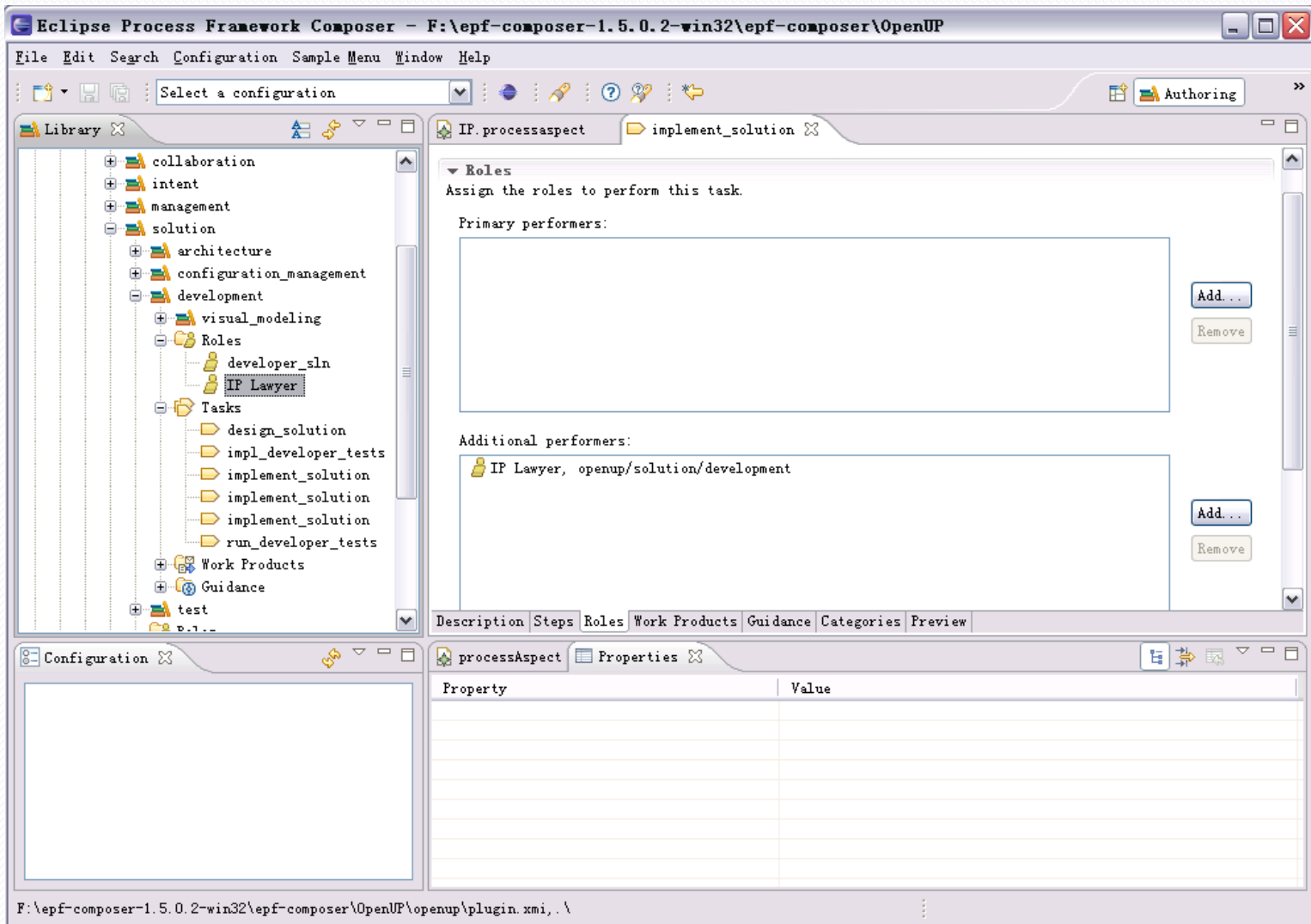
```
<ProcessAspect>
  <!--...-->
  <pointcutAndAdvice>
    <pointcut>
      <_name>implement solution</_name>
      <_type>OpenUP::Task</_type>
    </pointcut>
    <advice>
      <add_attribute>
        <spem::Step>
          <spem::Step::Name>detect potential IP problem</spem::Step::Name>
          <spem::Step::Description>
            Detecting potential IP problem is carried out by a code scanner...
          </spem::Step::Description>
        </spem::Step>
      </add_attribute>
      <add_artifact>risk patterns</add_artifact>
      <add_artifact>potential problem list</add_artifact>
      <add_tool>code scanner</add_tool>
    </advice>
  </pointcutAndAdvice>
  <!--...-->
</ProcessAspect>
```



Supporting Tool (1/2)



Supporting Tool (2/2)



Outline

- Motivation
- Our solution
 - Process aspect model
 - Describing a process aspect
- Weaving process aspect into SPEM-based processes
- An example
- **Related work**

Related work

- **General position paper on software process aspect**
 - M. Sutton Jr. Aspect-Oriented Software Development and Software Process. SPW 2005: 177-191
- **Software process aspect at execution time**
 - Q., C.A. Lima Reis, “Towards an Aspect-Oriented Approach to Improve the Reusability of Software Process Models,” In Proceedings of the IWEA, New York.
 - Oren Mishali , Shmuel Katz, Using aspects to support the software process: XP over Eclipse, Proceedings of the 5th international conference on Aspect-oriented software development, March 20-24, 2006, Bonn, Germany
- **Workflow aspect**
 - Anis Charfi, Mira Mezini. Aspect-Oriented Workflow Languages. OTM Workshops 2006: Montpellier, France

Future work

- advice.operation
 - replace_xxx
 - delete_xxx
- Dealing with conflict