Knowledge and Case-Based Reasoning for customization of software processes – A hybrid approach
Contents

- Introduction
- Background
- Related work
- Overall procedure
- Process customization
- Case study
- Conclusion
Introduction (1/2)

- **Motivation**
  - Problems in customizing the software process
    - Depends on the knowledge of experts and manual activities
    - Can be time-consuming, costly and error-prone
  - Traditional approaches to address these problems
    - Tried to use the past experience and knowledge
  - Problems in traditional reuse mechanisms
    - Generates an *incompletely* tailored process
      - When applied independently
Research goal

- Suggests a structured mechanism to customize a process
  - For diverse application domains

- Suggests a hybrid method to achieve this
  - With Case-Based Reasoning (CBR) technique
  - With knowledge-based technique

- Designs a Process Library (PL) as a repository
  - For past cases used by CBR technique
  - For process components used by knowledge-based technique
  - For knowledge needed to integrate two techniques
Background (1/2)

- **Process Library**
  - **Structure**
    - Entity
      - Has knowledge
    - Relationship
      - Require
      - Supported-by
      - Deactivate

[Basic specification syntax of the frame]

[frame] ::= ([frame-identifier], {[slot]})
[slot] ::= ([slot-name], [slot-value]) | ([slot-name], [slot-value] OPT)
[slot-value] ::= [attr.-value] | {'(' [sub-attr.], [attr.-value] ')'} | NULL

[Main structure of PL]
Process Library (cont’d)

- Knowledge model
  - Improvement
    - Improvement goal
  - Domain-specific
    - Domain characteristics
  - Relation of domain factors
- Case
  - Cases of past processes
- Dependency
  - Integrity constraints
Related work (1/2)

- **About the domain analysis that deals experience reuse**
  - Summary of domain analysis by heuristics [SSR’97]
    - Addressed approaches for dealing reusable object
  - Domain modeling for software reuse [CASE’94]
    - Facilitated the process of generating target system specifications

- **About the knowledge-based technique**
  - Knowledge-based process model generation [ISSE-TR’96]
    - Showed a way to customize complete process based on knowledge
Related work(2/2)

- **About the reuse repository**
  - Representing software engineering models [IEEE TSE’92]
    - Utilized the *Experience Factory* concepts
  - Library of process models for software design [CASE’95]
    - Developed a prototype *library* of software process models
  - Organizational learning to domain analysis [ICSE’95]
    - Used *case-based* approach to organize and use the repository
Overall procedure

- **Process customization steps**

![Diagram of process customization steps]

- Flow of information
- Flow of the procedure

New project

Identification of key factors for process design

Process Generation by using Knowledge

Conflicts Resolution

Manual Tailoring

Final Software Process

PL (Process Library)
Process customization(1/5)

- **Step 1. Identification of key factors**
  - Needs to define factors **to differentiate** one project from others
    - Improvement goals, organizational policy types, etc.
  - Defines two groups of characteristics in this paper
    - Improvement initiatives
    - **Domain factors**
      - Factors that identifies the software development process
        - Defined in this paper based on the standards
        - Inserted into the PL if a new one is needed
Process customization (2/5)

Step 2-1. Application of CBR techniques

1. Defines the case
   - Represents requirements in form of a case frame

2. Retrieves the case
   - Calculates similarity between the defined case ‘C’ and the past case ‘P’

   \[
   \text{Similarity}(C, P) = \frac{T \sum f_i(c_i, p_i) \times w_i}{\sum w_i}
   \]

   where
   - \( f_i(c_i, p_i) = 1 \) if \( c_i \) and \( p_i \) are single value factors, and \( value \ of \ c_i = value \ of \ p_i \)
   - \( f_i(c_i, p_i) = \frac{\text{MNV}(c_i, p_i)}{\text{NV}(c_i)} \) if \( c_i \) and \( p_i \) are multiple value factors
   - \( f_i(c_i, p_i) = 0 \) otherwise

   \[* w_i = \text{weight of } i\text{th domain factor} \]
   \[* \text{MNV}(c_i, p_i) : \text{Number of coinciding factor values} \]
   \[* \text{NV}(c_i) : \text{Number of values in factors} \]

- Retrieves the one which has the highest similarity
Step 2-1. Application of CBR techniques (cont’d)

3. Adapts the case
   - Obsolete factors are deleted
   - Required factors are added

---

**Procedure after retrieving**

[case adaptation algorithm]

```plaintext
FOR EACH d IN DF_deleted
    PUT d INTO case_new.CASE_DOMAIN_SPECIFIC_DRIVER
FOR EACH activity a that supports d
    IF a is in case_new.CASE_DEACTIVATE
       THEN
           DELETE a FROM case_new.CASE_SUPPORTED_BY
           PUT a INTO case_new.CASE_DEACTIVATE
    ELSE
       PUT a INTO case_new.CASE_SUPPORTED_BY
   END IF
END FOR
FOR EACH driver d2 required by d
    PUT d2 INTO DF_added /* to recursively find drivers */
END FOR
```

---

**[Procedure after retrieving]**
Process customization (4/5)

- **Step 2-2. Application of knowledge-based techniques**
  
  ✓ 1. Constructs a set of domain factors
  ✓ 2. Includes drivers and activities for each domain factor
  ✓ 3. Deactivate activities if needed

```java
FOR EACH domain_factor d IN D_{dir}
    IF d.precondition_driver is satisfied AND d.domain_value is satisfied THEN
        PUT d.activities INTO A_{dir}
        PUT d.drivers INTO D_{dir} /* to recursively fire drivers */
        PUT d INTO D_{dir}
    END IF
    FOR EACH a IN A_{dir}
        IF a EQUALS d.deactivate THEN
            IF user confirms deletion THEN
                DELETE a FROM A_{dir}
            END IF
        END IF
    END FOR
END FOR
```

[Algorithm for process generation]
Process customization (5/5)

- **Step 3. Conflict resolution**
  - Occurs when applying two techniques together
  - Uses dependency knowledge to resolve inconsistencies

  ```
  FOR EACH (dependency knowledge K IN Process Library)
  IF there is conflict regarding K in Dprev and/or Aprev
  THEN
    solve conflict under user’s confirmation
  END IF
  END FOR
  
  [Algorithm for conflict resolution]
  
- **Step 4. Manual tailoring**
  - Has the need due to many project-specific requirement
  - Lets users apply their own knowledge
    - Add or delete some activities manually
Illustration of knowledge-based technique

✓ Web-based Executive Information System (EIS)
  - Information management system for decision-making of executive

[Representation of the case ‘Web-based EIS’]

[Representation of the relationship]
Illustration of knowledge-based technique (cont’d)

[Example of the customized process]
Illustration of CBR technique

Intranet-based total Management Information System (MIS)

[The case ‘Intranet-based MIS’ vs the case ‘Web-based EIS’]

Similarity \( \left( Web\_based\_EIS\_case, Intranet\_based\_total\_MIS \right) \)

\[
= \sum_{i} \left( f_i(\text{Web\_based\_EIS\_case}_i, \text{Intranet\_based\_total\_MIS}_i) \times w_i \right) = 0.72
\]

[Calculation of similarity between two cases]

[Comparison result of two cases]
Illustration of CBR technique (cont'd)

[Example of parts of the adapted process]
Conclusion

- **Contribution**
  - Suggests a hybrid approach
    - Efficient customization by CBR technique
    - Easy to model knowledge by knowledge-based technique
  - Overcomes the limitations
    - Possible to apply various application domain

- **Future work**
  - Needs to accumulate the actual project cases
  - Needs to extend process models and implement completely
Discussion

- **Characteristic**
  - Utilized existing works effectively
  - Intuitive and easy to understand and use

- **Limitation**
  - Couldn’t show the overcome of other works’ limitation clearly
  - Can’t be used without having lots of past experience
## Domain factors

- Defined ten factors in this paper

<table>
<thead>
<tr>
<th>Domain factor</th>
<th>Type</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lifecycle model</td>
<td>Single value</td>
<td>Waterfall, Incremental, Evolutionary, ...</td>
</tr>
<tr>
<td>Software development type</td>
<td>Single value</td>
<td>New development, Prototyping, Re-use, ...</td>
</tr>
<tr>
<td>Software type</td>
<td>Single value</td>
<td>Web-based, Intranet, Real-time system, ...</td>
</tr>
<tr>
<td>Project size</td>
<td>Single value</td>
<td>Small, Medium, Large</td>
</tr>
<tr>
<td>Organizational Policy</td>
<td>Multiple value</td>
<td>Security policy, Safety policy, ...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>