Large Process Models and Process Model Collections: - Challenges, Methods, Technologies -

Barbara Weber

Victoria Torres

Manfred Reichert
Barbara Weber  
Barbara.Weber@uibk.ac.at  
University of Innsbruck

Victoria Torres  
vtorres@pros.upv.es  
Polytechnic University of Valencia

Manfred Reichert  
manfred.reichert@uni-ulm.de  
University of Ulm
Agenda

- Challenges & Basic Notions
- Part I: Large Process Models
- Part II: Large Process Model Collections
- Part III: Large Process Structures
- References
Process Model

- Process model contains
  - Node
    - Control connector
    - Event
    - Activity
      - Atomic activity
      - Complex activity
    - Data object
      - input
      - output
    - Resource executes
      - Atomic activity contains
      - Operation
Process Model

- **Process model**
  - contains
    - **Node**
      - contains
        - **Control edge**
        - **Event**
        - **Activity**
          - contains
            - **Resource**
            - **Atomic activity**
              - contains
                - **Operation**
            - **Complex activity**
              - contains
                - **Data object**

- connector
- connect
- contains
- is a
- connects
- input
- output
- operational
- structural
- behavioural
- informational
- organizational
Process Model

- Process model
  - contains
  - Node
    - contains
    - Event
      - is a
        - Control connector
      - is a
        - Activity
          - input
          - output
        - Data object
          - describes
          - contains
            - Atomic activity
              - contains
              - Operation
          - Complex activity
            - operational
            - behavioural
            - structural
          - informational
            - organizational
            - operational

- contains
- connects
Process Model Collections

- Sets of process models
  - Sharing goals
    - Collections of process model variants
  - Targeted at different stakeholders
    - Collections of process model user views
  - Described at different abstraction levels
    - Collections of process model at different level of detail
  - Dealing with process model evolution
    - Collections of process model versions
  - Stored within the same repository
    - Collections of enterprise process models
Process Structures
Process Structures

System Entwurf

System Spezifikation

System Integration

Komponenten Spezifikation

Komponenten Test

Software Design

Software Modultest

Software Implementierung

Fahrzeug Erprobung

Testfälle

Testfälle
Lifecycle Phases
Lifecycle Phases

- **Analysis & Design**
  - BP identification and modelling
    - Based on domain requirements
    - BP Modelling notation and languages
  - Validation & Verification
    - Simulation techniques support Validation

Resulting artefact: BP model

Fostering communication between different stakeholders
Lifecycle Phases

- **Configuration**
  - Implementation of BP models
  - Implementation platform has to be chosen
  - BP model preparation for enactment
    - Interaction with the enterprise eco-system
      - users & existing systems
    - Tests to check desired behaviour

» Resulting artefact: Ready-to-enact BP model
Enactment

- BP instance execution
  - Guaranteeing BP model constraints compliance
- Monitoring & Visualizing techniques
  - Allow discovering the status of active BP cases

» Resulting artefacts:
  » Business Process instances
  » Execution logs
Lifecycle Phases

- **Diagnosis**
  - Analysis of execution logs
    - Identification of poor quality designs
      - Fragments that are not used at all
    - Identification of problems regarding execution environment adaptation

  » Resulting artefact: Process model and configuration changes report
Lifesycle Phases

- Evolution
  - Application of changes to BP models based on
    - New requirements
    - Improvement opportunities

» Resulting artefact: BP Model more accurate to the BP and its environment
Agenda

- Challenges & Basic Notions
- Part I: Large Process Models
- Part II: Large Process Model Collections
- Part III: Large Process Structures
- References
Challenge: Large Process Models

- Challenge:
  Understanding process models of large size
## Process Model Smells

<table>
<thead>
<tr>
<th>PMS1: Non-intention Revealing Naming of Activities / Process Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMS2: Contrived Complexity</td>
</tr>
<tr>
<td>PMS3: Redundant Process Fragment</td>
</tr>
<tr>
<td>PMS4: Large Process Models</td>
</tr>
<tr>
<td>PMS5: Lazy Process Models</td>
</tr>
<tr>
<td>PMS6: Unused Branches</td>
</tr>
<tr>
<td>PMS7: Frequently Occurring Instance Changes</td>
</tr>
<tr>
<td>PMS8: Frequently Occurring Variant Changes</td>
</tr>
</tbody>
</table>

(Weber and Reichert 2008, Weber et al. 2011)
Refactoring Large Process Models

Identification of Process Model Smells

Application of Refactoring Techniques
# Process Model Refactorings

<table>
<thead>
<tr>
<th>Refactoring</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RF1</td>
<td>Rename Activity</td>
</tr>
<tr>
<td>RF2</td>
<td>Rename Process Schema</td>
</tr>
<tr>
<td>RF3</td>
<td>Substitute Process Fragment</td>
</tr>
<tr>
<td>RF4</td>
<td>Extract Process Fragment</td>
</tr>
<tr>
<td>RF5</td>
<td>Replace Process Fragment by Reference</td>
</tr>
<tr>
<td>RF6</td>
<td>Inline Process Fragment</td>
</tr>
<tr>
<td>RF7</td>
<td>Re-label Collection</td>
</tr>
<tr>
<td>RF8</td>
<td>Remove Redundancies</td>
</tr>
</tbody>
</table>

(Weber and Reichert 2008, Weber et al. 2011)
Labeling of Process Models

- PMS1. Non intention revealing naming of activities / process models
  - Ambiguous or non intention revealing labels
  - Inconsistent use of labeling styles

(Weber et al. 2011)
Poor labeling may lead to ambiguities

- Plan a data transfer?
  or
- Transfer plan data?

Leopold et al. 2011
Insights from a process modeling experiment with 113 students.

The following sentence in the process description resulted into 84 different ways for naming this particular activity.

“Afterwards the scouting team attends games of the player they are interested in live in the football stadium.”

Fahland 2012
Guidelines for Labeling of Process Models

- Many guidelines stress the importance of proper labeling
  (Malone 2003, Sharp and McDermott 2008)

- Empirical insights show that verb-object style is best understandable
  (Mendling et al. 2010)
Automatic Refactoring of Labels

Labels are automatically refactored from action-noun style to labels in verb-object style

(Leopold, Smirnov, and Mendling 2012)

Plan data transfer to EPC-PCA from profitability analysis

Transfer plan data to EPC-PCA from profitability analysis

Refactorings RF1: Rename Activity

RF7: Re-label Collection
Structured versus Unstructured Business Process Models

- PMS2. Contrived Complexity
  - Unstructured process fragments are more difficult to understand than well-structured ones
  - Mendling, Reijers, van der Aalst 2010

Unstructured Process Model
Structured versus Unstructured Business Process Models

- Automatic transformation of unstructured into structured model (Polyvyanyy et al. 2012)

Unstructured Process Model

Well-structured Process Model

- Refactoring RF2: Substitute Process Fragment
PMS2. Contrived Complexity
- Process Fragments are syntactically often different, but semantically equivalent (Gert et al. 2010)
Structured versus Unstructured Business Process Models

- Automatic detection of semantically equivalent process fragments (Gert et al. 2010)

Refactoring RF2: Substitute Process Fragment
Redundant Process Fragments

- PMS3: Redundant Process Fragments
  - Clones can be commonly found in existing process models
    (Dumas et al. 2012, Weber et al. 2011)
  - More than 560 clones in the SAP reference model
    (Dumas et al. 2012, Weber et al. 2011)
Redundant Process Fragments

(Dumas et al. 2012)
Redundant Process Fragments

(Dumas et al. 2012)
Redundant Process Fragments

- Method for automatically detecting exact clones (Dumas et al. 2012)
- Detected clones can be automatically extracted to subprocesses
- Refactoring RF4: Extract Process Fragment
  RF5: Replace Process Fragment by Reference
  RF8: Remove Redundancies
PMS4: Large Process Models

- Literature reports about process models with several hundred activities (Soto et al. 2008)
- Large process models tend to comprise more formal flaws than smaller ones (Mendling et al. 2008)
Large Process Models

- Method for automatic modularization of business process models (Reijers et al. 2011)

- Method for the automatic labeling of process models (Leopold et al. 2011)

- Refactoring RF4: Extract Process Fragment
  RF5: Replace Process Fragment by Reference
  RF8: Remove Redundancies
- Modularization is a widely used strategy to reduce complexity in conceptual models.

- Hierarchical structures supported by many conceptual modeling languages.

- Empirical evidence on the benefits of hierarchy are contradictory.
Hierarchisation in Conceptual Models
A Systematic Literature Review

<table>
<thead>
<tr>
<th>Work</th>
<th>Findings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Moody [15]</td>
<td>Positive influence on accuracy, no influence / negative influence on time</td>
</tr>
<tr>
<td>Domain: ER-Models</td>
<td></td>
</tr>
<tr>
<td>Reijers et al. [16, 17]</td>
<td>Positive influence on understandability for one out of two models</td>
</tr>
<tr>
<td>Domain: Business Process Models</td>
<td></td>
</tr>
<tr>
<td>Cruz-Lemus et al. [9, 18]</td>
<td>Series of experiments, positive influence on understandability in last experiment</td>
</tr>
<tr>
<td>Domain: UML Statecharts</td>
<td></td>
</tr>
<tr>
<td>Cruz-Lemus et al. [13]</td>
<td>Hierarchy depth of statecharts has no influence</td>
</tr>
<tr>
<td>Domain: UML Statecharts</td>
<td></td>
</tr>
<tr>
<td>Shoval et al. [14]</td>
<td>Hierarchy has no influence</td>
</tr>
<tr>
<td>Domain: ER-Models</td>
<td></td>
</tr>
<tr>
<td>Cruz-Lemus et al. [8]</td>
<td>Positive influence on understandability for first experiment, negative influence in replication</td>
</tr>
<tr>
<td>Domain: UML Statecharts</td>
<td></td>
</tr>
<tr>
<td>Cruz-Lemus et al. [12, 19]</td>
<td>Hierarchy depth has a negative influence</td>
</tr>
<tr>
<td>Domain: UML Statecharts</td>
<td></td>
</tr>
</tbody>
</table>

(Zugal et al. 2011a)
Hierarchisation in Conceptual Models  
A Systematic Literature Review

<table>
<thead>
<tr>
<th>Work</th>
<th>Findings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Moody [15]</td>
<td>Positive influence on accuracy, no influence / negative influence on time</td>
</tr>
<tr>
<td>Domain: ER-Models</td>
<td></td>
</tr>
<tr>
<td>Reijers et al. [16, 17]</td>
<td>Positive influence on understandability for one</td>
</tr>
<tr>
<td>Domain: Business Process Models</td>
<td></td>
</tr>
<tr>
<td>Cruz-Lemus et al. [8]</td>
<td>Positive influence on understandability for first experiment, negative influence in replication</td>
</tr>
<tr>
<td>Domain: UML Statecharts</td>
<td></td>
</tr>
<tr>
<td>Cruz-Lemus et al. [12, 19]</td>
<td>Hierarchy depth has a negative influence</td>
</tr>
<tr>
<td>Domain: UML Statecharts</td>
<td></td>
</tr>
</tbody>
</table>

Under which circumstances can positive / negative influences on understandability be expected?
Understanding Process Models
Some Insights from Cognitive Psychology

- Answering questions about a model is a *problem solving task*

- Three different problem solving processes (Larkin and Simon 1987)
  - Search
  - Recognition
  - Inference
Limitations of Working Memory

Central Concept: Working Memory

- Required by all conscious mental activities
- Mental effort: utilization of working memory
- Severely limited: 7 +/- 2 information „slots“ (Miller 1956)
- Overflow: rapid performance decrease! (Sweller 1988)
- Mental effort: amount of working memory used (Paas et al. 2003)
Two Opposing Forces

- **Abstraction**
  - hiding of irrelevant information (Parnas 1972)
  - supports human mind’s attention management (Larking and Simon 1987)

- **Split-attention effect (Sweller and Chandler 1994)**
  - Occurs when information from several sources needs to be integrated
  - switching attention between models
Theoretical Framework for Measuring Understandability

question complexity includes determines performance

abstraction lowers mental effort increases

hierarchy enables split-attention effect

causes

(Zugal et al. 2011a)
Hierarchical Model – Abstraction Question

- 'Define the problem' can be executed after 'Check if problem is already solved'.

Sub Process: Do Systematic Literature Review
'Define the problem' can be executed after 'Check if problem is already solved'.
'Describe initial approach' can be executed after 'Compile classification statistics'.

Sub Process: Do Systematic Literature Review
'Describe initial approach' can be executed after 'Compile classification statistics'.
Hierarchisation is Always a Trade-off

- Modularization is always a trade-off!

- Also the optimal modularization can only increase the “average” understanding!
  - Some questions become easier to answer
  - Some questions become harder to answer

(Zugal et al. 2011a, Zugal 2011b, Zugal 2011c)
Hierarchisation is Always a Trade-off

- Modularization is always a trade-off!
- Also the optimal modularization can only increase

Dynamic modularization / visualization / navigation is needed!

(Zugal et al. 2011a, Zugal 2011b, Zugal 2011c)
The Role of External Memory

- **External memory**
  - mechanism for reducing mental effort
  - Information storage outside the human cognitive system (e.g., pencil and paper or a blackboard)

- **Cognitive Trace**
  - Information taken from working memory and stored in an external memory (e.g., to mark, update, and highlight information)
The Role of External Memory

- **External memory**
  - mechanism for reducing mental effort
  - Information storage outside the human cognitive system (e.g., pencil and paper or a blackboard)

- **Cognitive Trace**
  - Information taken from working memory and stored in an external memory (e.g., to mark, update, and highlight information)

For process models of large size, tool support, which enables to offload information, is essential.
Visualizing and Abstracting Large Process Models
Process Visualization
What is needed?
adapt display form (diagram, form, table, text, ...)

adapting visual appearance (symbols, colors, ...)

abstracting information (process views)

Process Visualization Dimensions
Process Visualization

Abstracting Process Models: Goals
Goals:

- Decreasing the complexity of (large) process models
- Eliminating or abstracting process information

► Personalize process models through process views

Process views should be ...
- easy to define
- dynamically built if required

"Only show my activities!"
"Do not display technical activities"
"Aggregate completed parts"
Reduction

- Eliminate activities
- Simplify the resulting schema
- Remove adjacent satellite objects

Aggregation

- Aggregate activities
- Aggregate adjacent objects if required

Bobrik et al., 2007
Process Model Abstraction and Process Views in Proviado

(Bobrik, Bauer, & Reichert, 2007; Reichert et al., 2012)

- Related approaches, e.g.,
  - Smirnov, Reijers & Weske, 2011
  - Eshuis & Grefen, 2008
  - Schumm, Latuske & Leymann, 2011
Process Visualization

Abstracting Process Models: The Proviado Approach

- A multi-layer approach for abstracting process models and building process views
Process Visualization

Abstracting Process Models: Elementary Operations

**Elementary reduction operations**

Reduce CF \(\{B,C,D\}\)

Further refactorings:

but ...
Process Visualization
Abstracting Process Models: Elementary Operations

Elementary aggregation operations

- AggrSESE
- AggrAddBranch
- AggrComplBranches
- AggrShiftOut

Non-connected activity sets

- AggrAddBranch
- AggrAddBranch
Process Visualization

Abstracting Process Models: Elementary Operations

Elementary operations applied at the process instance level

Reduction\textsubscript{SESE}

\[ A^\checkmark \rightarrow B^\checkmark \rightarrow C \rightarrow D \]

\[ A^\checkmark \rightarrow \rightarrow D \]

Aggr\textsubscript{SESE}

\[ A^\checkmark \rightarrow B^\checkmark \rightarrow C \rightarrow D \]

\[ H: \langle S(A), E(A), S(B), E(B), S(C) \rangle \]

\[ A^\checkmark \rightarrow BC \rightarrow D \]

\[ H': \langle S(A), E(A), S(BC) \rangle \]
Let’s have a closer look at aggregations!
Example: Aggregate activities from set \{B,D\}

How to aggregate these two activities?
Example: aggregate activities from set \{B,D\}

Alternative 1: \texttt{AggrShiftOut}

Alternative 2: \texttt{AggrAddBranch}

Well-defined properties to characterize the resulting view!
Process Visualization

Abstracting Process Models: View Properties

- dependency-generating
- inconsistent process state

- dependency-erasing
- consistent process state
Properties of view-building operations:

<table>
<thead>
<tr>
<th>Operation</th>
<th>str.</th>
<th>order preserving</th>
</tr>
</thead>
<tbody>
<tr>
<td>RedActivity</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>AggrSESE</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>AggrComplBranches</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>AggrShiftOut</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>AggrAddBranch</td>
<td>-</td>
<td>+</td>
</tr>
</tbody>
</table>
Using views for process execution or process change:
• correct structure and consistent state are crucial
• view interpreted by process engine or change engine

Using views for process visualization:
• imprecision or loss of information is tolerable
• view interpreted by process participants

Proviado addresses this issue by enabling parameterizable process views, i.e., the degree of imprecision or tolerable information loss may be flexibly chosen.
3 strategies for aggregating an activity set (AggregateCF):

- **strategy = as-is**
- **strategy = subdivide**
- **strategy = expand**

dependencies = non-erasing

dependencies = preserving

dependencies = non-generating

4x AggrSESE

AggrAddBranch

AggrShiftOut & AggrSESE

AggrAddBranch & AggrSESE

Process Visualization

Abstracting Process Models: View Parameterization

High-level Operations

Data Flow Operations

Multi-Aspect Operations

Single-Aspect Operations

Elementary Operations

AggShiftOut & AggrSESE

AggrAddBranch

AggrSESE

**View Parameterization:**
- Strategy: as-is
- States: default
- Dependencies: default

**Base Process Model**

```
A -> B -> C -> D
  |   |   |
  J -> K -> L
  |   |   |
  M -> N

E -> F -> G
  |   |   |
  H -> I
  |   |   |
  P -> Q
  |   |   |
  R -> S -> T
```

(Single-Aspect Operation)

**AggrAddBranch({B,J,M,N})**
(Elementary Operation)

**AggrShiftOut({B,J,M,N})**

**AGGREGATECF({B,F,G,J,M,N,P,Q,R})**
(Single-Aspect Operation)

**Base Process Model**

```
A -> B -> C -> D
  |   |   |
  J -> K -> L
  |   |   |
  M -> N

E -> F -> G
  |   |   |
  H -> I
  |   |   |
  P -> Q
  |   |   |
  R -> S -> T
```

**AggrSESE({F,G})**

**AggrSESE({P,Q,R})**

**AggrAddBranch({B,F,G,J,M,N,P,Q,R})**
(Elementary Operation)
Elementary operations for reducing and aggregating data elements: RedData and AggData

RedData(D1)

AggData({D1,D2})

AggData({D3,D5,D6})
Single-aspect operations for abstracting data flow:
\textsc{ReduceDF} and \textsc{Aggregate DF}
Adapting data flow edges in the context of control flow aggregations: AdaptDE
A user-friendly definition of process views requires high-level view creation operations.

Example of a high-level operation: *ShowMyActivities*
- Eliminate all activities the current user is not involved in.
1. Choose high-level view operation and set its parameters

\[ \text{AggrForeignActivities}(User = X) \]

2. Map high-level operation to multi-aspect operation(s)

\[ \text{Aggregate}\{\{B,C,D,G,H,I,J\}, \text{Param} = \{\ldots\}\} \]

3. Map multi-aspect operations(s) to single-aspect operations

\[ \text{AGGREGATECF}\{\{B,C,D,G,H,I,J\}, \text{Param} = \{\text{strategy} = \text{subdivide}, \ldots\}\} \]
\[ \text{AGGREGATEDF}\{\{D2,D3\}\} \]

4. Determine corresponding single-aspect operations and apply them to the original process model; apply refactorings if applicable

\[ \text{AggrShiftOut}\{\{B,C,D\}\} \]
\[ \text{AdaptDE}\{\{D2\}\} \]
\[ \text{AdaptDE}\{\{D3\}\} \]
\[ \text{AggrSESE}\{\{G,H,I,J\}\} \]
\[ \text{AdaptDE}\{\{D4\}\} \]
\[ \text{AggrData}\{\{D2,D3\}, \ldots\} \]
Step 1

Step 2

Step 3

Step 4

Step 5

High-level Operation: ShowMyActivities(user X)
- Define predicate pred:
  - Find all activities where actor X is not involved in
  - Evaluate pred: S = {C,D,E,F,I,J,K,L,P}

RedSESE S1
RedSESE S2
RedSESE S3
RedSESE S4
RedSESE S5

Elementary Operations:
RedSESE S1
RedSESE S2
RedSESE S3
RedSESE S4
RedSESE S5

Multi-aspect Operation: Reduce(S)

Single-aspect Operation: REDUCECF(S)
- Find SESE components
  - S1 = {C,D,E}, S2 = {F}, S3 = {I,J,K}, S4 = {L}, S5 = {P}

High-level Operations in Proviado:
- ShowActivitiesOfUser
- AggrExecutedPart
- ShowExecutedPath
- GroupedAggregation
- ViewByRelevance
- ViewByPredicate
- Subgraph
- SubgraphRange
- CutProcess
- …
Process Visualization Dimensions

- Abstracting information (process views)
- Adapting visual appearance (symbols, colors, ...)
- Adapt display form (diagram, form, table, text, ...)

Diagram showing the dimensions of process visualization.
Change Request  000-W213-XQ-2146
Part: Speedometer  
Change: Improvement of Readability

- **Initiation**:  
  Start: 13.01.2006  
  End: 15.01.2006

- **Expertise**:  
  Start: 15.01.2006  
  End: 23.02.2006

- **Evaluation**:  
  Start: 25.02.2006

- **Approval**:  

- **Realization**:  

Process Visualization  
Visual Appearance of a Process Model: Issues
1. Which visual notation to use?

- A *template mechanism* is required that allows for the flexible definition of the
  - *visual appearance* (e.g., geometry) of process objects
  - *placeholders* for attributes or (status) symbols
1. Which visual notation to use?
2. When to use which visualization?

- Selection of a concrete visualization template may depend on the type of a process element, attributes, process data, users, etc.

- element = “activity“
- element = “actor“
- status = “running“
- activity.type = “testing“
Process Visualization

Visual Appearance of a Process Model: Issues

1. Which visual notation to use?
2. When to use which visualization?
3. Which data? Where to add and how?
   - Abstracting and formatting process data
     - Formatting, e.g., date values
     - Abstracting, e.g. costs: "high" instead of 1.000.000 €
   - Filling template parameters with concrete process data

Change Request 000-W213-XQ-2146
Part: Speedometer
Change: Improvement of Readability

Initiation ✔ Expertise ✔ Evaluation ✔ Approval ✔ Realization

Start: 13.01.2006 Start: 15.01.2006 Start: 25.02.2006
End: 15.01.2006   End: 23.02.2006   End:
1. Which visual notation to use?
2. When to use which visualization?
3. Which data? Where to add and how?
4. How shall the visualization style look like?
   - Colors, fonts, lines, etc.

Change Request  000-W213-XQ-2146
Part: Speedometer
Change: Improvement of Readability

<table>
<thead>
<tr>
<th>Initiation</th>
<th>Expertise</th>
<th>Evaluation</th>
<th>Approval</th>
<th>Realization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start: 13.01.2006</td>
<td>Start: 15.01.2006</td>
<td>Start: 25.02.2006</td>
<td>End: 23.02.2006</td>
<td>End:</td>
</tr>
</tbody>
</table>
How we addressed these issues in the Proviado visualization framework ...
Visualization Model

- logical description of all information required for creating a particular process visualization, e.g., symbols, display form, layouting, ...
Examples of visualization templates

Visualization template defines
1. symbol to be used
2. data to be displayed
3. application context

Bobrik et al., 2006
Visual Appearance of a Process Model: The Proviado Approach

Schema of a visualization templates
Defining a visualization templates for process elements

```xml
<template id="default_act">
  <inputs>...</inputs>
  <graphic>
    <symbol>...</symbol>
    <parameter name="Beschreibung"
      location='g/text[@name='actname']"
      value="act.getName()" />
    <parameter name="Status"
      location='g/g[@name='status']"
      value="act.getState()=RUNNING" />
    <parameter name=starttime"
      location='g/text[@name='starttime']"
      value="formatDate(act.getStart(), 'dd/mm/yyyy')" />
    <choice>
      <when test="act.getState()=RUNNING">
        <use xlink:href="#act_state_RUNNING" />
      </when>
      <when test="act.getState()=COMPLETED">...</when>
    </choice>
  </graphic>
</template>
```

Symbol shapes are defined using SVG

Parameter locations inside the symbol are specified using XPath

JavaScript can be used for calculating parameter values
Visualizing of a process element

- element type
- user preferences
- status
- attribute values
- process or application data

may depend on

Visual Appearance of a Process Model: The Proviado Approach

Defining the application context of visualization templates

- obj.nodetype = "activity"
  - status: completed
  - type: testing

- obj.nodetype = "actor"
  - Name: Mr. Lucas
  - Test goal: TEST

context rules
Defining the application context of visualization templates

```
in: ProcessObject obj

if obj.type = "node"
else
  case obj.nodetype = "activity"
    if obj.acttype = "testing"
      TEST
      test goal
          Name
          AttrValue
          AttrValue
          AttrValue
          AttrValue
    else
      Name
          AttrValue
          AttrValue
          AttrValue
          AttrValue
  else
    "actor"
    actor_name

if obj.type = "edge"
else
  case obj.edgetype = "control flow"
else
  "data flow"
```

<if test="self.type=ACTOR">
  <template id="actor">
    <inputs>
      <input name="actor" value="self"/>
    </inputs>
  </template>
</if>

<if test="self.type=ACTIVITY">
  <choose>
    <case test="self.type='testing'">
      <template ref="testing_act">
        <inputs>
          <input name="act" value="self"/>
        </inputs>
      </template>
    </case>
    <otherwise>
      <template ref="default_act">
        <inputs>
          <input name="act" value="self"/>
        </inputs>
      </template>
    </otherwise>
  </choose>
</if>

<if test="self.type=ACTIVITY">
  <choose>
    <case test="self.type='testing'">
      <template ref="testing_act">
        <inputs>
          <input name="act" value="self"/>
        </inputs>
      </template>
    </case>
    <otherwise>
      <template ref="default_act">
        <inputs>
          <input name="act" value="self"/>
        </inputs>
      </template>
    </otherwise>
  </choose>
</if>
Creating a process visualization

Evaluate the context rules and fill in attribute values

Process elements

Application context schema

Context rule 1

if type = activity
then use template x

Context rule 2

Cost: 07.08.07 4100

Context rule 3

Cost: 14.08.07 4100

Process attributes

notation

Template x

state

Symbol

Name

AttrValue

AttrValue

Template y

Name

AttrValue

AttrValue

Template z

Name

AttrValue

PartID

XX35K4

start: 01.08.2007

date: 07.08.2007

end: 05.08.2007

cost: 2500

start: 07.08.2007

date: 14.08.2007

end: 05.08.2007

cost: 4100

Create a process visualization

Visual Appearance of a Process Model: The Proviado Approach
Example
Process Visualization
Combining the Dimensions
Process Visualization Dimensions

- Abstracting information (process views)
- Adapting visual appearance (symbols, colors, ...)
- Adapt display form (diagram, form, table, text, ...)

Diagram showing three dimensions of process visualization.
**Process Visualization**

Display Form of a Process Model

**Goal:** Experiment with other ways of displaying processes
Some concrete work we did in the proView project

Central Process Model (CPM)

a) Form-based View  

b) Tree-based View  

c) EPC-based View
Process Visualization

Summary: What we achieved in Proviado?
Focus of this presentation has been on the personalized visualization of large process models.

Another fundamental issue: How to enable domain experts to change large process models!
**Basic Idea**: Using process views not only for visualization purpose, but also as interface for changing the underlying core process model (CPM)

- Updates of a process view then have to be correctly propagated to its CPM as well as all other views on this CPM

- Necessitates a formal foundation

Kolb, Kammerer & Reichert, 2012
Ambiguities when propagating view changes to the CPM
Updatable Process Model Abstractions

Updating a CPM and related views after a view update!

Diagram:
- Propagate Change: InsertParallel (V1, C, D, X)
- Initial Situation:
  - View V1:
    - Op\(_{V1}\) = \{RedActivity(V1,B), RedActivity(V1,E), RedActivity(V1,F)\}
  - AggrPartlyMode = AGGR
- Migrated View V2:
  - aggrPartlyMode = SHOW

Diagram:
- Initial Situation:
  - View V1:
    - Op\(_{V1}\) = \{RedActivity(V1,B), RedActivity(V1,E), RedActivity(V1,F)\}
  - AggrPartlyMode = AGGR
- Migrated View V2:
  - aggrPartlyMode = SHOW
Agenda

- Challenges & Basic Notions
- Part I: Large Process Models
  - Part II: Large Process Model Collections
- Part III: Large Process Structures
- References
Business Process Repositories
Basic Challenge: Dealing with Large Model Collections
Business Process Repositories

- **Standard repository features**
  - Check-in / check-out, access control, simple search queries

- **Advanced repository features**
  - Extract, transform and compose process information
  - Filtering (i.e., clone, detection, similarity search, querying)
  - Managing process variants
  - Merging
  - Navigating in repositories
  - ...

(La Rosa et al. 2011)
Business Process Repositories: Extract, Transform, and Compose Process Fragments

1. Process data transformation

2. Process fragment composition

3. Instance data correlation & integration

1. System A

2. System B

3. System C

(Bobrik 2008)
Einheitliches Prozessmodell für die Visualisierung

Business Process Repositories: Extract, Transform, and Compose Process Fragments
Business Process Repositories

Filtering

- Clone detection
  - finding exact matches

- Similarity search
  - finding close matches

- Querying
  - looking for patterns
Searching for similar process models / process fragments

- Given a query graph, find all process models in the process model collection that are similar

When are two process models similar?

(Dijkman et al. 2011)
When are two process models similar?

- Similarity measure defines when two process models are similar
  - Value between 0 and 1
  - 0 indicates no similarity and 1 indicates identical elements

(Dijkman et al. 2011)
When are two process models similar?

- **Label matching similarity**
  - Similarity based on the labels of business process model elements

- **Structural similarity**
  - Measures similarity based on the labels of a business process model element, as well as the relations between these elements

- **Behavior similarity**
  - Measure similarity based on the intended behavior of process models

(Dijkman et al. 2011)
Querying a process model for patterns

- Given a query graph, find all matching sub-graphs in the process model collection

- Example of such a query language
  - BPMN-Q (Awad 2007): Queries expressed in BPMN; usage of wildcard nodes and arcs possible

- Efficient querying of process fragments
  - Index for efficient querying (Yan, Dijkman, and Grefen 2012)
Looking for patterns in a process model

- What happens between A and E?

(Awad 2007, Awad et al. 2008)
Looking for patterns in a process model
- What happens from the start until B is reached?

(Awad 2007, Awad et al. 2008)
Agenda

- Challenges & Basic Notions
- Part I: Large Process Models
- Part II: Large Process Model Collections
- Part III: Large Process Structures
- References
Variability
Sources of Variations

- Cultural differences
- Types of customers
  - Loyal customer
  - Impulsive customers
  - Kid
  - VIP
  - Handicapped
  - New customer
- Country-specific regulations
  - Professional qualification
  - Intellectual property protection
  - Renting
  - Money transfer
- Types of goods
  - Normal
  - Durable
  - Luxury
  - Imperishable
- Religious
  - Christianity
  - Hinduism
  - Judaism
- Types of services
  - Drugs administration
  - Delicate
  - Intellectual property protection
  - Quality control
- Buying and Selling
  - Intellectual property protection
  - Intellectual property protection
- Quality control
  - New customer
  - Handicapped
  - Language
  - Use of the language
  - Aesthetics
- Nationality
  - Race
  - Gender
  - History
- Location
Industrial Example
Flight Check-in

- **Generic process:**

  - Identify passenger
  - Assign seat
  - Print boarding card
  - Drop off luggage

- **Variations due to:**
  - Airline policies (e.g., check-in opening time)
  - Type of passenger (e.g., adult, child, handicapped)
  - Type of ticket (e.g., economy, business)
  - Type of carried items (e.g., pet)
  - etc.

» Hundreds of variants
Variant 1:
- Online check-in
- Adult flying with a business class ticket
- International flight from EU to USA
Variant 2:
- Counter desk check-in
- Unaccompanied Minor (UM)
- Variant 3:
  - Online check-in
  - Adult carrying overweight

**Industrial Example**
**Flight Check-in**

- Identify passenger
- Assign seat
- Print boarding card
- Pay extra fee
- Drop off luggage

*Web system*
- 23 hours before departure

*Counter desk*
How are Business Processes which are subject to variability characterized?
- Variable parts
  - Parts being subject to variation (commonly known as variation point)

- Identify passenger
- Assign seat
- Print boarding card
- Drop off luggage

Different timer start events
Different types of boarding card (paper, sms)
Different seat assignment according to the passenger's needs
Alternatives that exist for each variable parts
- Alternatives that fit in each variable part
- Relationship constraints between such alternatives
Representing Variability Concepts

- Application context for each alternative
  - Conditions that make these variables being applied
  - Usually represented by a set of variables

Check-in type?
- on-line
- counter desk

Passenger needs?
- No needs
- Unaccompanied minor
- Carrying a pet
- wheelchair

3 hours before departure

Identify passenger

Assign seat

Assign UM seat

Assign seat (pet company)

Assign seat (wheelchair)

Fill in ESTA form

Fill in UM form

Fill in pet form

...
Where can we find variability in a BP?

- Behavioural
- Functional
- Informational
- Organizational
- Temporal
- Operational
Behavioural

It captures the dynamic behavior of a BP model and corresponds to the control flow between the activities. A control flow schema includes information about the order of the activities or the constraints for their execution.
It specifies the decomposition of BPs, i.e., it represents the activities to be performed. While an atomic activity is associated with a single action, a complex activity refers to a subprocess or, more precisely, a sub-process model.
Informational

It concerns data and data flow, i.e., it represents the informational entities (e.g., data, artifacts, products, and objects) consumed or produced during the execution of BP activities.
Organizational

It deals with the assignment of resources to the activities of a BP model, i.e., it represents the actors, roles (i.e., humans or systems), within an organization being in charge of executing certain BP activities.
Temporal

It deals with time issues and temporal constraints, i.e., it represents the occurrence of events during the course of a process, which affects the scheduling of activities from this process.

- Identify passenger
- Assign seat
- Print boarding card
- Fill in ESTA form
- Electronic Boarding card
- Drop off luggage

23 hours before departure
Operational

It refers to the implementation of process activities, i.e., the application services to be executed when an atomic activity is performed.

- Identify passenger
- Assign seat
- Print boarding card
- Fill in ESTA form
- Online
- Self-service machine
- Counter desk

Web system: 23 hours before departure

Drop off luggage
Limitations of Common BPMLs

Are common BPML constructs sufficient to properly represent variability in BPs?
Could we use...

- Conditional branching,
- Separate process models, or
- Sub-processes

... for such purpose?
Could we use...

– Conditional branching,
– Separate process models, or
– Sub-processes

... for such purpose?
Limitations of Common BPMLs

1. Conditional Branching

BPMLs could be used to represent process variant alternatives

- Identify passenger
- Assign seat
- Assign UM seat
- Assign seat (pet company)
- Assign seat (wheelchair)
- Fill in ESTA form
- Fill in UM form
- Fill in pet form

3 hours before departure
23 hours before departure
Limitations of Common BPMLs
Conditional Branching

- (+) Allows visualizing *all* alternatives in *one shoot*

Alternatives

- Identify passenger
- Assign seat
- Assign UM seat
- Assign seat (pet company)
- Assign seat (wheelchair)
- Fill in ESTA form
- Fill in UM form
- Fill in pet form

Alternatives

3 hours before departure
23 hours before departure
Limitations of Common BPMLs
Conditional Branching

- Hinders identifying existing variants

Diagram:
- Identify passenger
- Assign UM seat
- Fill in UM form
- Assign seat (pet company)
- Assign seat (wheelchair)
- Fill in ESTA form

Clocks:
- 3 hours before departure
- 23 hours before departure
Limitations of Common BPMLs
Conditional Branching

- Hinders differentiating between domain dependent and domain independent conditional branching
Could we use...

– Conditional branching,
– *Separate process models*, or
– Sub-processes

... for such purpose?
Limitations of Common BPMLs
Separate process models

- (+) Each process variant in a separate model

Variant 1:

<table>
<thead>
<tr>
<th>Web system</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>23 hours before departure</td>
<td>Identify passenger</td>
<td>Assign seat</td>
<td>Print boarding card</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Fill in ESTA form</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Electronic Boarding card</td>
</tr>
</tbody>
</table>

Counter desk

Drop off luggage
Limitations of Common BPMLs
Separate process models

- (+) Each process variant in a separate model

Variant 2:

- Identify passenger
- Assign UM seat
- Print boarding card
- Drop off luggage
- Localize staff to accompany passenger
- Fill in UM form

Counter desk: 3 hours before departure
Limitations of Common BPMLs
Separate process models

- Common fragments have to be replicated

**Variant 1:**
- Identify passenger
- Assign seat
- Print boarding card
- Fill in ESTA form
- Electronic Boarding card
- Drop off luggage

Web system:
- 23 hours before departure

Counter desk:
- Localize staff to accompany passenger

**Variant 2:**
- Identify passenger
- Assign UM seat
- Print boarding card
- Drop off luggage
- Localize staff to accompany passenger

Web system:
- 3 hours before departure

Counter desk:
Limitations of Common BPMLs
Separate process models

- (-) Changes need to be replicated in all variants

**Variant 1:**

- Web system
  - Identify passenger
  - Assign seat
  - Print boarding card
  - Fill in ESTA form
  - Electronic Boarding card
- Counter desk
  - 23 hours before departure
  - Sign airline form
  - Drop off luggage

**Variant 2:**

- Counter desk
  - 3 hours before departure
  - Sign airline form
  - Fill in UM form
  - Print boarding card
  - Localize staff to accompany passenger
  - Drop off luggage
Limitations of Common BPMLs

Could we use...

- Conditional branching,
- Separate process models, or
- Sub-processes

... for such purpose?
Limitations of Common BPMLs
Sub-processes

- (+) Promotes the reuse of common fragments
- (+) Allows reducing the size of the model

Variant 1:

- Start check-in
- Assign seat
- Print boarding card
- Fill in ESTA form
- Electronic Boarding card
- Drop off luggage

23 hours before departure

Web system

Counter desk

Identify passenger
Sign airline form
Limitations of Common BPMLs
Sub-processes

- Variability related issues cannot be explicitly represented
Limitations of Common BPMLs

Even though these techniques are supported by commercial BPM tools, they do not enable transparent and explicit management of process variants.
Variability in other Domains

- Is there something out in other domains that can be used?
Variability in the SPL domain

Software Product Line (SPL) Engineering

- Put emphasis on Reusability and Flexibility by
  - Consolidating and capitalizing on commonality through the product line
  - Focusing on product variations

- Objectives:
  - Create a collection of similar software systems from a shared set of software assets using a common means of production
Language requirements

- Variation point
- Alternative process element
- Alternative process element context
- Alternative process element relationship
- Variation point resolution time
Language requirements

Variation point

Precise position within a configurable process where different choices are possible depending on the current context or situation.
Language requirements

- **Alternative process element**

  Particular option that may be instantiated at a specific *variation point* and may refer to any modelling element such as activities and their control flow, resources, data, events, or operations.

Identify passenger

 Assign seat

 Assign UM seat

 Assign seat (pet company)

 Assign seat (wheelchair)

 Fill in UM form

 Fill in pet form

 Fill in ESTA form
Language requirements

- **Alternative process element context**

  Subset of process variables whose values make a particular *alternative process element* to become instantiated for a *variation point*.
Language requirements

- Alternative process element relationship

Constraint of use between two or more alternative process elements. These constraints are defined based on semantic relationships to ensure the proper use of the involved alternative process elements within a specific context.
**Language requirements**

- **Variation point resolution time**

This requirement should allow modellers to distinguish between variation points whose resolution depends on the initial context (configuration time) or on the current context of a process instance (enactment time).
Two main approaches:
- Behavioural-based approaches
- Structural-based approaches
Two main approaches:
- Behavioural-based approaches
- Structural-based approaches
Representing BP Variability

**Behavioural-based approaches**

- **Single modelling artefact** (Configurable process model)
- Process variant derivation by **removing** parts from the configurable process model
  - Techniques for variant derivation:
    - Configurable nodes and configuration requirements and guidelines
    - Hiding and blocking
  - Proposals found in BPM literature
    - C-EPC/C-iEPC (Rosemann et al. 2007/La Rosa et al. 2008)
    - C-YAWL (Gottschalk et al. 2008)
Representing BP Variability

Behavioural-based approaches

Configurable process model

Context

Process family

Configuring configurable process elements
Representing BP Variability
Behavioural-based approaches

- **C-iEPC**
  - Configurable nodes
    - Connectors
      - Restrict their behaviour or SEQ
    - Functions
      - Configured as ON, OFF, or OPT
    - Objects
      - 2 Dim: Optionality and specialization
    - Roles
      - 2 Dim: Optionality and specialization
  - Configuration...
    - Requirements
      - State hard configuration constraints
    - Guidelines
      - State soft configuration constraints
Representing BP Variability

Behavioural-based approaches

- **C-iEPC**
  - Configurable nodes
    - Connectors
      - Restrict their behaviour or SEQ
    - Functions
      - Configured as ON, OFF, or OPT
    - Objects
      - 2 Dim: Optionality and specialization
    - Roles
      - 2 Dim: Optionality and specialization
  - Configuration...
    - Requirements
      - State hard configuration constraints
    - Guidelines
      - State soft configuration constraints
Representing BP Variability
Behavioural-based approaches

- **C-iEPC**
  - Configurable nodes
    - Connectors
      - Restrict their behaviour or SEQ
    - Functions
      - Configured as ON, OFF, or OPT
    - Objects
      - 2 Dim: Optionality and specialization
  - Roles
    - 2 Dim: Optionality and specialization
  - Configuration...
    - Requirements
      - State hard configuration constraints
    - Guidelines
      - State soft configuration constraints
Representing BP Variability

Behavioural-based approaches

- **C-iEPC**
  - Configurable nodes
    - Connectors
      - Restrict their behaviour or SEQ
    - Functions
      - Configured as ON, OFF, or OPT
    - Objects
      - 2 Dim: Optionality and specialization
  - Roles
    - 2 Dim: Optionality and specialization
  - Configuration...
    - Requirements
      - State hard configuration constraints
    - Guidelines
      - State soft configuration constraints
Representing BP Variability
Behavioural-based approaches

- **C-iEPC**
  - Configurable nodes
    - Connectors
      - Restrict their behaviour or SEQ
    - Functions
      - Configured as ON, OFF, or OPT
    - Objects
      - 2 Dim: Optionality and specialization
    - Roles
      - 2 Dim: Optionality and specialization
  - Configuration...
    - Requirements
      - State hard configuration constraints
    - Guidelines
      - State soft configuration constraints
Representing BP Variability
Behavourial-based approaches

- **C-iEPC**
  - Configurable nodes
    - Connectors
      - Restrict their behaviour or SEQ
    - Functions
      - Configured as ON, OFF, or OPT
    - Objects
      - 2 Dim: Optionality and specialization
    - Roles
      - 2 Dim: Optionality and specialization
  - Configuration...
    - Requirements
      - State hard configuration constraints
    - Guidelines
      - State soft configuration constraints
C-iEPC

- Let’s derive the variant where:
  - Unaccompanied Minor (UM)
  - Check-in luggage
  - No luggage overweight
- C-iEPC
  - Connectors
  - XORc2 = SEQ1b
C-iEPC

- Connectors
  - XORc2 = SEQ1b
  - XORc6 = SEQ2b
C-iEPC

- Connectors
  - XORc2 = SEQ1b
  - XORc6 = SEQ2b

- Functions
  - Fill in ESTA form = OFF
Understandability Task
Process Variant Extraction

- **C-iEPC**
  - Connectors
    - XORc2 = SEQ1b
    - XORc6 = SEQ2b
    - XORc9 = OR
  - Functions
    - Fill in ESTA form = OFF
C-iEPC

- Connectors
  - XORc2 = SEQ1b
  - XORc6 = SEQ2b
  - XORc9 = OR

- Functions
  - Fill in ESTA form = OFF
  - Localize staff = ON
- **C-iEPC**
  - **Connectors**
    - XORc2 = SEQ1b
    - XORc6 = SEQ2b
    - XORc9 = OR
  - **Functions**
    - Fill in ESTA form = OFF
    - Localize staff = ON
    - Drop off lugg. = ON
- **C-iEPC**
  - **Connectors**
    - XORc2 = SEQ1b
    - XORc6 = SEQ2b
    - XORc9 = OR
    - XORc12 = OR
  - **Functions**
    - Fill in ESTA form = OFF
    - Localize staff = ON
    - Drop off lugg. = ON
Understandability Task
Process Variant Extraction

- **C-iEPC**
  - Connectors
    - XORc2 = SEQ1b
    - XORc6 = SEQ2b
    - XORc9 = OR
    - XORc12 = OR
  - Functions
    - Fill in ESTA form = OFF
    - Localize staff = ON
    - Drop off lugg. = ON
    - Pay extra fee = OFF
C-iEPC

- **Roles**
  - c1 = Economy class counter
  - c4 = Economy class counter
  - c5 = Economy class counter
  - c7 = Economy class counter
  - c10 = Economy class counter
  - c11 = Economy class counter

- **Objects**
  - C8 = Paper boarding card
C-iEPC

**Mental process followed in C-EPC:**

1. Inspect all configurable nodes
2. Evaluate the associated requirements
3. Configure configurable nodes accordingly
Process Variant Extraction Task
Cognitive discussion

- C-iEPC
  - Cognitive discussion
    - 3 basic operations:
      - Locating elements
        » Easy to perform thanks to their visual differences
      - Evaluating Boolean expressions
        » Can be pretty challenging depending on the complexity of the expression
        » Some evaluations depend on previous decisions
      - Adapting the model accordingly
        » Elements have to be mentally removed

⇒ Complexity depends on how many requirements have to be analyzed
Two main approaches:

- Behavioural-based approaches
- Structural-based approaches
Representing BP Variability
Structural-based approaches

- Structural-based approaches
  - Several modelling artefacts
    - Base model
    - Change operations
  - Process variant derivation through the application of 
    change operation to the base model
  - Proposals found in BPM literature
    - Provop (Hallerbach et al. 2010)
    - Rule representation and processing (Kumar and Wen 2012)
Representing BP Variability
Structural-based approaches

- Provop

Process modeling

Base model

Options

Choosing and applying options

Context
Provop

- Base model
  - Policies:
    - Standard process
    - Most frequently used process
    - Minimal average distance
    - Superset of all process variants
    - Intersection of all process variants

Variant: Passenger with no special needs, dropping off luggage, carrying luggage excess
- **Provop**
  - Change Options
    - Allow deriving new variants
    - Are applied to the base model
Representing BP Variability
Structural-based approaches

- Provop
  - Options Constraints
    - Relationships: Implication, Exclusion, Order, etc.
## Provop
- **Context Model**

<table>
<thead>
<tr>
<th>Context variable</th>
<th>Range of Values</th>
<th>Behaviour</th>
</tr>
</thead>
<tbody>
<tr>
<td>Passenger_needs</td>
<td>NO, PET, UM, WHEELCHAIR</td>
<td>Static</td>
</tr>
<tr>
<td>Flight_destination</td>
<td>USA, EU</td>
<td>Static</td>
</tr>
<tr>
<td>Luggage_overweight</td>
<td>TRUE, FALSE</td>
<td>Static</td>
</tr>
<tr>
<td>Check-in_luggage</td>
<td>TRUE, FALSE</td>
<td>Static</td>
</tr>
<tr>
<td>Required_assistance</td>
<td>TRUE, FALSE</td>
<td>Static</td>
</tr>
</tbody>
</table>
Provop

- Let’s derive the variant where:
  - Unaccompanied Minor (UM)
  - Check-in luggage
  - No luggage overweight
Understandability Task
Process Variant Extraction

- **Provop**

  1. Examine all change options and their Boolean expressions

  **Variant**: Unaccompanied minor, check-in luggage, no luggage overweight
2. Select all *change options* satisfying the given context

**Variant**: Unaccompanied minor, check-in luggage, no luggage overweight
Understandability Task
Process Variant Extraction

- Provop

3. Determine whether all options can be applied according to the *constraint model*

**Variant**: Unaccompanied minor, check-in luggage, no luggage overweight
Understandability Task
Process Variant Extraction

- Provop

4. Locate the **variation points** where options apply

**Variant**: Unaccompanied minor, check-in luggage, no luggage overweight
Variant: Unaccompanied minor, check-in luggage, no luggage overweight
Variant: Unaccompanied minor, check-in luggage, no luggage overweight
4. Mentally integrate **change options** into the **base model**

**Variant**: Unaccompanied minor, check-in luggage, no luggage overweight
4. Mentally integrate *change options* into the *base model*
Understandability Task
Process Variant Extraction

- Provop

4. Mentally integrate *change options* into the *base model*

**Variant**: Unaccompanied minor, check-in luggage, no luggage overweight
4. Mentally integrate *change options* into the *base model*.

**Variant:** Unaccompanied minor, check-in luggage, no luggage overweight.
Understandability Task
Process Variant Extraction

- Provop

4. Mentally integrate change options into the base model

Variant: Unaccompanied minor, check-in luggage, no luggage overweight
4. Mentally integrate *change options* into the *base model*.

**Variant:** Unaccompanied minor, check-in luggage, no luggage overweight.
- **Provop**

  4. Mentally integrate *change options* into the *base model*

Variants: Unaccompanied minor, check-in luggage, no luggage overweight
Understandability Task
Process Variant Extraction

- Provop

4. Mentally integrate **change options** into the **base model**

**Variant**: Unaccompanied minor, check-in luggage, no luggage overweight
Understandability Task
Process Variant Extraction

- Provop

4. Mentally integrate change options into the base model

Variant: Unaccompanied minor, check-in luggage, no luggage overweight
Provop

Mental process followed in Provop:

1. Examine all change options and their Boolean expressions
2. Select all change options satisfying the given context
3. Determine whether all options can be applied according to the constraint model
4. Locate the variation points where options apply
5. Mentally integrate change options into the base model
Provop

- Cognitive discussion
  - Two main operations:
    - Selecting change options
      » Boolean expression need to be evaluated
      » All are expressed in terms of context variables
      » Check for conflicts in the constraint model
    - Applying change options into the base model
      » Determine by the change distance between the base model and the variant to be derived.
      » The type of operation influences the complexity (delete vs. insert)
### Language Support Comparison
#### Structural vs Behavioural

<table>
<thead>
<tr>
<th></th>
<th>C-EPC</th>
<th>Provop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variation Point</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Alternative process elements</td>
<td>F, B, O, I</td>
<td>F, B</td>
</tr>
<tr>
<td>Alternative process element context</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Alternative process element relationships</td>
<td>+/-</td>
<td>+</td>
</tr>
<tr>
<td>Variation point resolution time</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

 functional (F), behavioral (B), organizational (O), Informational (I) perspectives
Qualitative Comparison

Discussion

- Modelling elements
  - In C-EPC are mainly deleted
  - In Provop can be added, deleted, and moved
  » Deletion presumably involves less cognitive effort
Qualitative Comparison

Discussion

- Variability modelling elements
  - Single artefact in C-EPC
    - Configurable process model
      » Works for small models
      » Overload for large models
  - Separate artefacts in Provop
    - base model, change options, options constraint, context model
      » Abstraction mechanisms favour model understanding
Qualitative Comparison

Discussion

- Boolean expressions
  - C-EPC: Represented in terms of the model structure
    - Forces to keep track of previous decisions
    - Presumably imposing bigger mental effort
  - Provop: Represented in terms of context variables
    - Semantics is explicit in the model
    » Presumably imposing less mental effort
Business Process Repositories
Merging Process Models: Behavioral Approach

La Rosa et al, 2010
1. Configurations are very costly
2. Variants are difficult to maintain

Original reference process model $S$

customization & adaptation

Process variant $S_1$
Process variant $S_2$
Process variant $S_n$

mining & learning

Control differences

Discovered reference process model $S'$

Goal: Less configurations!

1. Configurations are very costly
2. Variants are difficult to maintain

Why?

1. Limit the efforts to update reference model
2. Avoid spagatti-like structure
3. Obtain the flexibility to only perform the important changes

e.g., discover a new reference process model having lower average weighted distance to the variants than the original reference process model has!
Scenario 1: No original reference process model available

- Process variant $S_1$
- Process variant $S_2$
- Process variant $S_3$
- Process variant $S_4$
- Process variant $S_5$
- Process variant $S_n$

Process Repository

**Goal:** Discover a (new) reference process model which requires less configuration efforts

Scenario 2: Original reference process model known

- Original reference process model $S$

Customization & adaptation

- Process improvement

- Discovered reference process model $S'$

Li, Reichert, & Wombacher 2012
Applying Heuristics Search to Scenario 2

**Force 1:** close to variants

**Force 2:** close to reference

Original reference model

Sc: Search result without constraint

Sn: Search result with constraint

Original Reference model  Process variants  Intermediate search result  Discovered Reference Model  Search steps
Applying Heuristics Search to Scenario 2

Search tree based on best kids $S_{\text{kid}}^{aj}$
Can we find a model closer to the variants by performing saying only 3 changes of the old reference model?

Average (weighted) distance of $S$ to the variants: 4 high-level changes (NP-hard complexity)
Example (2)

S: Original reference model
(avg weighted distance: 4; fitness: 0.643)

R1: Result after one change
(avg weighted distance: 3.2; fitness: 0.814)

R2: Result after two changes
(avg weighted distance: 2.6; fitness: 0.854)

R3: Result after three changes (final result)
(avg weighted distance: 2.35; fitness: 0.872)

$\Delta_1 = \text{move}(S, J, \ldots)$

$S[\Delta_2 \rightarrow R2]$

$\Delta_2 = \text{insert}(R1, X, \ldots)$

$S[\Delta_3 \rightarrow R3]$

$\Delta_3 = \text{move}(R2, \ldots)$
Business Process Repositories
Merging Process Models: Structural Approach

Healthcare Case Study
Business Process Repositories
Navigating in Repositories and Large Process Models
Practical demands: Navigate in Large Process Models

- Navigation within one single dimension (forward/backward)
- Views are predefined
- No comprehensive tool support available
Process Navigation (niPRO)

- comprises a sequence of user interactions
- allows process participants to navigate from a default visualization of a large process to more specific ones
Business Process Repositories
Navigating in Repositories and Large Process Models

niPRO Core Navigation Model – Inspired by Google Earth

niPRO - Hipp, Mutschler & Reichert, 2012
Business Process Repositories
Navigating in Repositories and Large Process Models

niPRO Core Navigation Model – Inspired by Google Earth

Level 1: The Globe

Level 2: Continents, Process areas

Geographic dimension

Level 3: Countries, Processes

View dimension

Level 4: Cities, Process steps
3-dimensional navigation concept

(a) geographic navigation dimension

(b) semantic navigation dimension

(c) view navigation dimension

"logic-based"

"time-based"

niPRO - Hipp, Mutschler & Reichert, 2012
Initial situation

- 3-dimensional navigation space
- Different navigation states (g, s, v)
- State transitions (= user interactions)
Business Process Repositories
Navigating in Repositories and Large Process Models

- Some Challenges
  - How to guide the user through the navigation space?
  - How to recommend certain paths within the navigation space (i.e., to reduce number of interactions)?
  - How to remove specific navigation states being not reasonable...
  - How to assist users when the navigation space gets more complex...

  ➡️ A process navigation model is needed
Business Process Repositories
Navigating in Repositories and Large Process Models

- Typical use case

A developer wants to see which process step has to be done, after he completed the current process step.
Typical use case

A developer wants to see which process step has to be done, after he completed the current process step.

Start: (0,0,0)
End: (1,1,0)
Agenda

- Challenges & Basic Notions
- Part I: Large Process Models
- Part II: Large Process Model Collections
- Part III: Large Process Structures
- References
Interacting process fragments.

*Mans et al., 2012*

*The arcs show the interactions that need to take place between fragment instances.*
- Proclets provide a framework for modeling and executing workflows.
- A *Proclet* can be seen as a lightweight workflow process able to interact with other Proclets (potentially at different levels of aggregation).
- A *Proclet class* specifies which tasks need to be executed and in which order, i.e., the Proclet class defines the process followed by individual Proclets. One instance is called a *Proclet instance*.
Proclet classes for the illustrated scenario

Mans et al., 2012
Large Process Structures
Proclets

a) Interaction graph saving the Proclet instances that need to be performed for 'Sue' together with the desired interactions

b) The Proclet instances that need to be performed for 'Sue' and 'Anne'. For both the desired interactions are shown. Furthermore, for 'Sue' the instances that need to be performed are linked with the interaction graph via dotted arcs

Mans et al., 2012
Process structure needs to be adapted when product structure changes!
Large Process Structures
Data-driven Process Structures: Motivation

Modeling
Execution
Dynamic Adaptation
Exception Handling
Large Process Structures
Data-driven Process Structures: Motivation

Corepro: Integrated Support of Data-driven Process Structures
Data-Driven Process Structures:
The Corepro Approach
Large Process Structures

Corepro

Modellebene

Data Model

Object Life Cycles / Life Cycle Coordination Model

Instanzebene

Data Structure

Data-driven Process Structure

Müller et al. 2007
Gesamtsystem

hatSys

System

hatKomp

Komponente

Datenmodell

Life Cycle Coordination Model

Gesamtsystem

hatSys

System

hatKomp

Komponente

Datenmodell

Life Cycle Coordination Model

BR212, Rel. 02/08

Telematik High V2.2

Head-Up Unit V3.14

TV Tuner V1.83

Modellebene

Datenmodell

hatSys

System

hatKomp

Komponente

Object Life Cycles / Life Cycle Coordination Model

Data-driven Process Structure

Müller et al. 2007
Data-driven Process Structure

Müller et al. 2007
Large Process Structures

**Corepro**

**Modellebene**

- **Gesamtsystem**
  - **hatSys**
  - **System**
    - **hatKomp**
  - **Komponente**

- **System**
  - **Auswahl**
    - **Komponenten**
  - **Konfiguration**
    - **gebildet**
  - **Muster**
    - **aufgebaut**
  - **Brett-Test**
  - **E**

- **Komponente**
  - **Bestellung**
    - **Komponente**
    - **ange liefert**
    - **Absicherung**
      - [i.O.]
    - **Abgesichert**
    - **Melden**
    - **E**

**Instanzebene**

- **Gesamtsystem**
  - **BR212, Rel. 02/08**
  - **hatSys**
  - **System**
    - **Telematik High V2.2**
    - **hatKomp**
    - **nutztKomp**
  - **Komp onente**
    - **Head-Up Unit V3.14**
    - **nutztKomp**
    - **Komponente**

- **Telematik High V2.2**
  - **Auswahl**
    - **Komponenten**
  - **Konfiguration**
    - **gebildet**
  - **Muster**
    - **aufgebaut**
  - **Brett-Test**
  - **E**

- **Head-Up Unit V3.14**
  - **Bestellung**
    - **Komponente**
    - **ange liefert**
    - **Absicherung**
      - [i.O.]
    - **Abgesichert**
    - **Melden**
    - **E**

- **TV Tuner V1.83**
  - **Bestellung**
    - **Komponente**
    - **ange liefert**
    - **Absicherung**
      - [Nicht i.O.]
    - **Abgesichert**
    - **Melden**
    - **E**

**Müller et al. 2007**
• Significant reduction of modeling efforts for process engineers
• Formal operational semantics allows for correct executability
• Soundness can be guaranteed on an abstracted level
Data-driven Process Adaptation

Change Operation (Data Structure)
1) removeRelation(Telematik High V2.2, TV Tuner V1.83, nutztKomp);
2) removeObject(TV Tuner V1.83);

Change Operation (Process Structure)
1) removeExtTrans(Telematik High V2.2 . Muster Aufgebaut, Installieren, TV Tuner V1.83 . E);
2) removeOLC(Tuner V1.83);
Large Process Structures
Data-driven Process Structures: Corepro

Exception Handling

Forward Recovery

Step 1

JUMP

Step 2

RESET

RESET

Step 3

JUMP

...
Large Process Structures

Data-driven Process Structures: Corepro Proof-Of-Concept

Müller et al. 2008b
Large Process Structures
Corepro: Case Study ISO 26262 -- Road Vehicles, Functional Safety

Instance Level: Data Safety Case Study System Level: Process Structure

System

Sub-system A
- Specification of technical safety concept
- System design
- Part 5: PD HW
- Part 6: PD SW

Sub-system B
- Specification of technical safety concept
- System design
- Part 5: PD HW
- Part 6: PD SW

Sub-system B1
- Specification of technical safety concept
- System design
- Part 5: PD HW
- Part 6: PD SW

Sub-system B2
- Specification of technical safety concept
- System design
- Part 5: PD HW
- Part 6: PD SW

PD = Product Development, HW = Hardware, SW = Software


Miller, G.: The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. The Psychological Review 63 (1956) 81-97.


