Florendia Kartsouni – Fourli

“Intelligent Support for Context-Sensitive Business Process Modelling”

Supervisor:
Jessica Chen Burger
Presentation Contents

- Motivation
- Goal
- System Design
- Technical Details
- Evaluation
- Conclusion – Future Work
- Discussion
Motivation

- Business Process Modelling (BPM) and Workflow technologies are fundamental to many business process re-engineering initiatives.
- Currently most BPM languages do not support manipulation of information that is context-sensitive.
- Formal Business Process Modelling Language (FBPML) is a semantic based BPM language.
Goal

- To provide an intelligent support tool for FBPML which would:
  - Enable the generation of semantic business process models and their integration with context sensitive information/knowledge.
  - Enable the generation of simple ontological models which can hold the information entities of the domain.
  - Provide automatic modeling assistances based on inferred information derived from the underlying model(s).
A “Model Support Framework” was created
UML class diagrams were used throughout the design
System Design (cont.)

- Model support framework
  - Model building framework
    - Model creation
    - Model updating
    - Model archiving
  - Model verification framework
    - Consistency checking
    - Syntactic checking
    - Semantic checking
    - Detection of semantic loops
System Design (cont.)

- UML Package Diagram

```
MODEL BUILDING ← MODEL VERIFICATION

GUI
```
System Design – Model Building

- First Level
System Design – Model Building (cont.)

- Second Level – Process Diagram
System Design – Model Building (cont.)

- Second Level – Ontology Diagram
System Design – Model Verification

- Model Verification Class Diagram
Technical Details

- Tool implemented using the Java programming language
- Interface & Model Building
  - Simple Menus based on Java Swing
  - Model creation & manipulation based on JGraph
- Model Verification
  - Definitions of notation and verification rules in FOL
  - Inference by Prolog engine based on JIProlog
Technical Details – FOL

Definitions

- FOL representations were created to describe:
  - The FBPML process diagrams notation
  - The Ontology diagrams notation
  - 16 Model Verification axioms used to build rules in Prolog for checking the models
Technical Details – FOL Definitions - Notation

- The FBPML Notation

![Diagram showing FBPML notation elements including start, finish, and synchronization bars.]
Technical Details – FOL
Definitions – Notation (cont.)

- FBPML Formal Definitions
  - Main Nodes:
    \[
    \text{activity}(ID, \text{Name}, \text{Trigger}, \text{Precondition}, \text{Postcondition}, \text{Action}) \\
    \text{primitive\_activity}(ID, \text{Name}, \text{Trigger}, \text{Precondition}, \text{Postcondition}, \text{Action})
    \]
  - Action:
    \[
    \text{action}(\text{ActionType}, \text{Class}, \text{Instance})
    \]
  - Junctions
    \[
    \text{start}(\text{ActivityName}) \quad \text{finish}(\text{ActivityName}) \\
    \text{junction}(\text{JunctionType}, \text{Pre\_Activities}, \text{Post\_Activities})
    \]
  - Precedence Link:
    \[
    \text{link}(\text{ActivityA}, \text{ActivityB})
    \]
Technical Details – FOL

Definitions – Notation (cont.)

- The Ontology Notation

[Diagram showing the ontology notation with classes and relations]
Ontology Formal Definitions

Concrete Class:

\texttt{concrete\_class(Name,Description,Example,Rules,CrossReference,ObjectAttributes)}

Instance:

\texttt{instance(Name,ParentName,ObjectAttributes)}
Technical Details – Verification

Axioms

- 16 Axioms defined in FOL to check:
  - Consistency
  - Syntax
  - Semantics
  - Semantic Loops
Technical Details – Verification

Axioms (cont.)

- Consistency Example:

**Axiom 1** If two activities share the *same name* then they *should* have exactly the same Properties values:

\[
\forall \text{Name}, \text{Trigger}_a, \text{Trigger}_b, \text{Precondition}_a, \text{Precondition}_b, \\
\text{Postcondition}_a, \text{Postcondition}_b, \text{Action}_a, \text{Action}_b. \\
\text{activity}(\text{Name}, \text{Trigger}_a, \text{Precondition}_a, \text{Postcondition}_a, \text{Action}_a) \\
\wedge \text{activity}(\text{Name}, \text{Trigger}_b, \text{Precondition}_b, \text{Postcondition}_b, \text{Action}_b) \Rightarrow \\
\text{Trigger}_a = \text{Trigger}_b, \\
\text{Precondition}_a = \text{Precondition}_b, \\
\text{Postcondition}_a = \text{Postcondition}_b, \\
\text{Action}_a = \text{Action}_b
\]

*Note that the word: "should" indicates a strong enforcement of the rule.*
Technical Details – Verification

Axioms (cont.)

- Syntax Example:

**Axiom 6**  *It is recommended* that all activities specified have been assigned *IDs.*

\[ \forall ID, \text{ActName}, \text{Trigger}, \text{Precondition}, \text{Postcondition}, \text{Action}. \]

activity\((ID, \text{ActName}, \text{Trigger}, \text{Precondition}, \text{Postcondition}, \text{Action})\)\)

\[ \neg (ID = \text{null}) \]

*Note that the phrase: "is recommended " indicates a weak enforcement of the rule*
Technical Details – Verification

Axioms (cont.)

- Semantics Example:
Axiom 11  An activity (A) should not delete an instance of a class that has already been previously deleted by another activity (C), unless a third activity (B) has created it before activity A was reached.

\[ \forall Name_a, Trigger_a, Precondition_a, Postcondition_a, Action_a, Class, Instance, \neg \exists Name_b, Trigger_b, Precondition_b, Postcondition_b, Action_b. \]
\[ \text{activity}(Name_a, Trigger_a, Precondition_a, Postcondition_a, Action_a) \]
\[ \land \]
\[ \text{action}(Delete, Class, Instance) \in Action_a \]
\[ \land \]
\[ (\text{activity}(Name_b, Trigger_b, Precondition_b, Postcondition_b, Action_b) \]
\[ \land \]
\[ \text{action}(Create, Class, Instance) \in Action_b \]
\[ \land \]
\[ \text{path}(Name_b, Name_a) \Rightarrow \]
\[ \neg \exists Name_c, Trigger_c, Precondition_c, Postcondition_c, Action_c. \]
\[ \text{activity}(Name_c, Trigger_c, Precondition_c, Postcondition_c, Action_c) \]
\[ \land \]
\[ \text{action}(Delete, Class, Instance) \in Action_c \land \]
\[ \text{path}(Name_c, Name_b) \]
Technical Details – Verification

Axioms (cont.)

- Semantic Loop:

**Axiom 16** *It is recommended that an activity is not involved in a semantic loop.*

\[ \forall \text{Name}, \text{Trigger}, \text{Precondition}, \text{Postcondition}, \text{Action}. \\
\text{activity}(\text{Name}, \text{Trigger}, \text{Precondition}, \text{Postcondition}, \text{Action}) \implies \\
\neg \text{path}(\text{Name}, \text{Name}) \]
Evaluation

Evaluation Framework

- Theoretical approach, focussing on the reflection of the FBPML method's syntax and semantics on the system
- Empirical approach, employing a usability test performed on a sample of potential users
- Comparison approach, exploring similarities and differences to few other similar modelling tools
Theoretical Evaluation

- Criteria:
  - Completeness of the model building capabilities: how compliant to the FBPML notation are the models
  - Completeness and Correctness of the model verification capabilities
Theoretical Evaluation (cont.)

- Process Diagram:

<table>
<thead>
<tr>
<th>Element</th>
<th>Provision</th>
<th>Consistency</th>
<th>Syntax</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Role</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Activity</td>
<td>✓</td>
<td>comp</td>
<td>comp</td>
<td>part</td>
</tr>
<tr>
<td>Primitive Activity</td>
<td>✓</td>
<td>comp</td>
<td>comp</td>
<td>part</td>
</tr>
<tr>
<td>Precedence Link</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>part</td>
</tr>
<tr>
<td>Synchronisation Bar</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>And Junction</td>
<td>✓</td>
<td>✗</td>
<td>comp</td>
<td>part</td>
</tr>
<tr>
<td>Or Junction</td>
<td>✓</td>
<td>✗</td>
<td>comp</td>
<td>part</td>
</tr>
<tr>
<td>Start Junction</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>part</td>
</tr>
<tr>
<td>Finish Junction</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>part</td>
</tr>
<tr>
<td>Time Point</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Annotations</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>
Theoretical Evaluation (cont.)

- Ontology Diagram

<table>
<thead>
<tr>
<th>Element</th>
<th>Provision</th>
<th>Consistency</th>
<th>Syntax</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subject Area</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Abstract Class</td>
<td>✓</td>
<td>comp</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Concrete Class</td>
<td>✓</td>
<td>comp</td>
<td>×</td>
<td>part</td>
</tr>
<tr>
<td>Instance</td>
<td>✓</td>
<td>comp</td>
<td>×</td>
<td>part</td>
</tr>
<tr>
<td>Membership Link</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Instance Link</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>part</td>
</tr>
<tr>
<td>Note Link</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Post-it</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
</tbody>
</table>
Theoretical Evaluation – Model Building Completeness

- The system covers the notations (exception: Time Point)
- Most of the semantic meaning of the notation elements is captured (exceptions: Synchronisation Bar, Time Point, Role and Annotations)
Theoretical Evaluation – Model Verification

- **Correctness:**
  - All the rules are correct as they are based on the methodology and guidelines of the FBPML method

- **Completeness:**
  - Consistency rules: Complete
  - Syntax rules: Not complete
  - Semantic rules: Not complete
  - Loop Detection: Complete
Empirical Evaluation

- Testing framework:
  - 5 test-users
  - Basic Training
  - A questionnaire with tasks that tested:
    - The usability of the Model Building capabilities based on model recreation
    - The usability of the Model Verification capabilities based on an erroneous model (checking and correction of errors)
Empirical Evaluation - Results

- Comments on general usability where positive
- Comments on the Graphical User Interface were positive but a few “bugs” were identified

<table>
<thead>
<tr>
<th>User</th>
<th>Expertise</th>
<th>Recreation Tasks</th>
<th>Verification Tasks</th>
<th>Usability</th>
<th>GUI</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>5,6</td>
<td>easy with comments</td>
<td>very easy with comments</td>
<td>very good</td>
<td>good</td>
</tr>
<tr>
<td>B</td>
<td>3,5,6</td>
<td>very easy with comments</td>
<td>very easy</td>
<td>very good</td>
<td>good</td>
</tr>
<tr>
<td>C</td>
<td>1,5,6</td>
<td>very easy with comments</td>
<td>very easy with comments</td>
<td>very good</td>
<td>very good</td>
</tr>
<tr>
<td>D</td>
<td>6</td>
<td>very easy with comments</td>
<td>very easy with comments</td>
<td>good</td>
<td>good</td>
</tr>
<tr>
<td>E</td>
<td>4</td>
<td>fairly easy</td>
<td>very easy</td>
<td>good</td>
<td>very good</td>
</tr>
</tbody>
</table>
Comparison with relevant tools

- 5 tools that support various BPM and/or workflow methods.

<table>
<thead>
<tr>
<th></th>
<th>KBST-EM</th>
<th>IX Process Panel</th>
<th>AI0WIN©</th>
<th>ProCap©</th>
<th>Renew</th>
<th>FBPMLGraph</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modelling methods</td>
<td>26 methods + FBPML</td>
<td>Process Modelling</td>
<td>Activity Modelling</td>
<td>IDEF3</td>
<td>Petri Nets</td>
<td>FBPML, Ontology</td>
</tr>
<tr>
<td>Visualisation</td>
<td>good</td>
<td>poor</td>
<td>good</td>
<td>good</td>
<td>good</td>
<td>good</td>
</tr>
<tr>
<td>Model verification</td>
<td>consistency</td>
<td>none</td>
<td>none</td>
<td>none</td>
<td>syntax</td>
<td>consistency, syntax, semantics, loops</td>
</tr>
<tr>
<td>Execution</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Exporting facilities</td>
<td>ps.bmp,wmf</td>
<td>xml</td>
<td>IDL,mdb,html, visio,winhelp</td>
<td>txt</td>
<td>ps,eps,xml</td>
<td>fol</td>
</tr>
<tr>
<td>Platforms supported</td>
<td>Win 2000/XP, Unix</td>
<td>Win 2000/XP</td>
<td>Win 95/98 Win 2000/XP</td>
<td>Win 95/98 Win 2000/XP</td>
<td>all Java supporting platforms</td>
<td>all Java supporting platforms</td>
</tr>
</tbody>
</table>
Conclusion

- Goal was achieved successfully
- The tool provides a very good base for future expansions in the field
- Limitations of the tool include:
  - Few “bugs” on the interface
  - Some more Model Verification rules could have been defined
Future Work

- The Model Support framework could be expanded to accommodate more modelling methods and more verification rules.
- The system could be expanded to provide automatic enactment of the process models through workflow simulation.
- The system could be expanded to facilitate automatic translation of the models in a semantic web services language such as OWL-S. Hence allow a direct mapping of the process models to semantic web services models.
Discussion

- Play “Model Creation” video (3min)
- Play “Model Verification” video (3min)
- Play “Total Evaluation” video (10min)