Constraint-enabled Process Modeling

Conrad Bock
U.S. National Institute of Standards and Technology
November 20, 2007
Overview

- Models and constraints: Example of structure models
- Extend to process models: execution
- Process modeling that includes execution
- Execution constraints
- Process Specification Language (PSL)
- Relating process / execution modeling to PSL
Structure Models and Constraints

- Structural modeling languages have associated constraint languages:
  - UML includes OCL
  - OWL used with SWRL
  - RDF used with SPARQL
  - EXPRESS includes EXPRESS rules

- Simple, commonly used statements in modeling language, more detail in constraint language.
Structure Models are Shorthands

- Could write entire model in OCL, just a matter of ergonomics.
- Enables structural models to have constraints.

UML Class ("model")

<table>
<thead>
<tr>
<th>Car</th>
</tr>
</thead>
<tbody>
<tr>
<td>height : m</td>
</tr>
<tr>
<td>width : m</td>
</tr>
</tbody>
</table>

context Car inv:
self.height.oclIsKindOf(m) and self.width.oclIsKindOf(m)

OCL

inv = invariant
self = instance of the context class
Structure Models are Shorthands

- Generalization = all instances of subtype are instances of supertype ("cars are vehicles").
- Constraints on (instances of) supertype apply to (instances of of) subtype.
Constraints for Process Models?

- Specify additional details on process models.
- Could be about timing, resource consumption, service level agreements, etc.

BPMN

- Change Color
  - Paint
  - Dry

- Dry starts < 10 min after Paint
- Consume < 4 liters of paint.
- Object not moved until Dry is done.

Process constraints written informally
Are Process Models Shorthands?

- **BPMN:**

```
  <process name="ChangeColor">
    <sequence>
      <invoke operation="Paint"/>
      <invoke operation="Dry"/>
    </sequence>
  </process>
```

- **UML 2:**

```
  ChangeColor
  Paint --> Dry
```

- **BPEL:**

```
<process name="ChangeColor">
  <sequence>
    <invoke operation="Paint"/>
    <invoke operation="Dry"/>
  </sequence>
</process>
```

- **Java/C:**

```java
void ChangeColor
{  Paint();
    Dry();
}
```
What’s Constrained?

- “Instances” of processes.
- Executions

One model

Many executions ...

Process Definition

<table>
<thead>
<tr>
<th>Paint</th>
<th>Dry</th>
<th>ChangeColor</th>
</tr>
</thead>
</table>

Time
Execution Tools

- Workflow systems call executions “process instances”.
- Process management systems provide
  - monitoring, analysis
  - other services on executions.
- These systems treat executions as first-class entities, with their own
  - attributes (eg, elapsed time)
  - operations (eg, suspend)
Execution Constraints

Could write entire model as execution constraints, just a matter of ergonomics.

Would enable process models to have constraints.

For every execution of ChangeColor, executions of Paint and Dry occur, with the Paint execution happening before the Dry.
Process Generalization

- All executions of specialized process are executions of generalized process.
- Constraints on (executions of) #1 apply to (executions of) #2.
Semantics and Syntax

- **Semantics:**
  - Need a semantics for process models that can be integrated with constraints.

- **Syntax:**
  - Need a constraint syntax that can be integrated with process models.

- This presentation is about semantics.
Virtual Machine Semantics

- Common process models usually have semantics defined by an imagined or virtual machine that "executes" (interpreter).
- Produce executions given particular inputs or conditions.
- A kind of operational semantics.
- Does not directly support automated reasoning or abstraction.
Token “Movement”

- Executions satisfying #2 do not satisfy #1 under token semantics.
Execution Constraint Semantics

- Tells which executions conform to (“satisfy”) process model.
- Does not directly produce executions, but infers the ones allowed by model.
- Supports automated reasoning and abstraction.
- A kind of axiomatic semantics.
- Might also be called “declarative”.
Execution Constraint Semantics

For every execution of ChangeColor, executions of Paint and Dry occur, with the Paint execution happening before the Dry.

- Some executions satisfy the model, some do not.
Partial Ordering Constraints

- One step or message happens sometime before another, not necessarily immediately.
- Allows “insertions”:

  - Is consistent with

(partial ordering is transitive)
Specializing Partial Orders

- Executions satisfying #2 also satisfy #1 under happens sometime before.
Execution Subsets

- Executions satisfying #2 are a subset of executions satisfying #1.
Some executions in the "intersection".

- Overlapping Models

ChangeColor #1

Paint → Dry

ChangeColor #3

Paint → Cleanup

Satisfies #1

Satisfies both

Satisfies #2
Overlapping Models

- Executions satisfying #4 also satisfy #1 and #3, but some only satisfy #1 or only #3.
Venn Diagram of Examples

- ● = Execution
Each level conforms to the one above it.
Modeling Without Execution

- Cannot instantiate and specialize user models (they are individuals, not classes).
- Unrelated to runtime execution (M0).
Modeling With Execution

- M1 process models are classes, can be specialized in M1 and instantiated at M0.
- M1 process constraints apply to M0 executions.
Class of all executions. Superclass of all process models. Introduces attributes such as time elapsed and resources used.

(Universal Behavior in BPDM)
The Execution class introduces temporal relations.
Must limit happensBefore by happensDuring.
Execution Constraints

- Can apply class-based constraint languages (like UML’s OCL):

  ```plaintext
  context ChangeColor inv:
      self.paintStepDuringCC.happensBefore->
      includes(self.dryStepDuringCC)
  ```

- Or first order (Common Logic Interchange Format):

  ```plaintext
  (forall (?CC ?P ?D)
    (if (and (ChangeColor ?CC)
      (paintStepDuringCC ?P ?CC)
      (dryStepDuringCC ?D ?CC))
      (happensBefore ?P ?D)))
  ```

- Common enough to be promoted to modeling languages (BPMD Succession, edges in typical graphical flow languages)
Process Specification Language

- An execution constraint language defined in first order (CLIF).
- Based on long period of research starting with situation calculus.
- Applied to:
  - scheduling, process modeling, process planning, production planning, simulation, project management, workflow, business process reengineering, vehicle navigation, semantic interoperability.
# Terminology Mapping

<table>
<thead>
<tr>
<th>This presentation</th>
<th>PSL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution</td>
<td>(complex) occurrence</td>
</tr>
<tr>
<td>Process (M2)</td>
<td>activity</td>
</tr>
<tr>
<td>happensBefore</td>
<td>earlier</td>
</tr>
<tr>
<td></td>
<td>(successor for “immediately before”)</td>
</tr>
<tr>
<td>happensDuring</td>
<td>subactivity_occurrence</td>
</tr>
<tr>
<td>(supertype of BPDM M1 happening parts)</td>
<td></td>
</tr>
<tr>
<td>BPDM Succession</td>
<td>min_precedes</td>
</tr>
<tr>
<td>Flow edge</td>
<td>next_subocc</td>
</tr>
</tbody>
</table>
Anything Can Happen

- Tree of all possible execution sequences over entire world, including
  - not physically possible.
  - not specified by the user.
- Not stored anywhere, just referred to by constraints.
(forall (?occChangeColor)
  (implies
    (occurrence_of ?occChangeColor ChangeColor)
    (exists (?occPaint ?occDry)
      (and (occurrence_of ?occPaint Paint)
           (occurrence_of ?occDry Dry)
           (subactivity_occurrence ?occPaint ?occChangeColor)
           (subactivity_occurrence ?occDry ?occChangeColor)
           (min_precedes ?occPaint ?occDry ChangeColor))))

Processes in PSL

Change Color

\[\text{Paint} \rightarrow \text{Dry} \]
Processes in PSL

- Portions of the occurrence tree (complex occurrences) will satisfy the constraints or not.

Satisfy ChangeColor constraints.

Do not satisfy ChangeColor constraints.
- PSL complex occurrences "satisfy" process models (or not).
Process Modeling and PSL

- Without possibilities.
Process Modeling and PSL

- Required nondeterminism, due to uncertainty about effects.
Process Modeling and PSL

- Required nondeterminism applicable to service-level agreements, and choreography generally.
- Expressible in single-branch occurrence tree?

Interaction is a process (execution constraint)
Process Modeling and PSL

- Translation of process models available:
  - incremental (fragments to small axioms)
  - handles loops, unstructured flows, input/output via parameters and messages, reaction to changes.

- U Toronto working on another approach.

- Best to combine the above.
Summary

- Models are shorthands for commonly used constraints on “instances” of the model.
- Process instances are executions.
- Extend metalanguages to incorporate executions into process modeling languages (BPDM) [http://www.omg.org/cgi-bin/doc?dtc/07-11-01].
- Integrate PSL with process models by:
  - Constraint language for BPDM.
    - PSL complex occurrences as “instances”.
  - Direct translation from models to PSL constraints.