Formal Modeling of BPEL Workflows Including Fault and Compensation Handling

Máté Kovács, Dániel Varró, László Gönczy
kovmate@mit.bme.hu

Budapest University of Technology and Economics
Dept. of Measurement and Information Systems

Research supported by Software Engineering for Service-Oriented Overlay Computers (SENSORIA) EU FET-IST project.
Contents

- Motivation to modeling and verifying business processes
- Short introduction to the BPEL language
- Comparison of existing approaches
- Feature presentation
Motivation

- Web service composition (e.g. BPEL)
  - Widespread tool support
  - Verification techniques still need improvement
    - Design errors of orchestration

- Our aim:
  - Check requirements on workflows formally
  - Derive formal models by model transformations
The Execution of Workflows

Traditional

- Recording
- Establish type
- Policy
- Premium
- Reject
- Pay
- Paper

Electronic

- Recording
- Establish type
- Workflow engine
- Premium
- Pay
- Reject
- Web service call
- Policy
Implementing Workflows

 Languages: BPEL, XPDL

- Very high level
- XML based
- Interpreted
- No debugger provided
- Difficult to follow the control flow of a process instance
Testing Workflows

- Problem: the testing of workflows
  - The data is stored in remote databases
  - The effects of test phases have to be rolled back
- Solution: the formal analysis of workflows
  - Formal workflow semantics
  - Formal verification of properties
    - E.g. variable access
  - Fault simulation: assessment of error propagation
A Workflow Example

- Basic activity
- Recording
- Establish type
- Control flow
- Beginning of parallel execution
- Selection
- Policy
- Premium
- End of parallel execution
- Reject
- Pay
The Concepts of Workflows

- Basic activities
- Structured activities
- Data flow / control flow?
A BPEL Example

- Basic activities
- Structured activity
BPEL: Web Service Orchestration

![Diagram of BPEL4WS Process and Web service with portType connections and input-only operations.]

10 Budapest University of Technology and Economics

Fault Tolerant Systems Research Group Máté Kovács, EFTS’07, Dubrovnik, 4 September 2007
BPEL Instructions

- **Basic activities:**
  - Invoke
  - Receive
  - Reply
  - Empty
  - Terminate
  - Throw
  - Compensate

- **Structured activities:**
  - Scope
  - Sequence
  - Flow
  - While
  - Switch
  - Pick
Structure of BPEL

- **Workflow**: the main business process
- **Fault handler**:
  - Faults thrown in workflow
  - User defined or default handler
- **Compensation handler**:
  - Initiated from outside
  - User defined or default handler

![Diagram showing Workflow, Fault Handler, and Compensation handler with Throw and Compensate actions]
Scope hierarchy

- Workflow
- Fault Handler
- Compensation handler

- Compensate
- Throw
- Compensate
Some Existing Approaches

  - Regular Petri net modeling basic workflows

  - BPEL modeling with Extended Finite-State Automata: event and fault handling is not considered
Some Existing Approaches

  - Modeling formalism: dataflow networks
  - Limited success w.r.t. the covering of event handling

  - Petri net model covering the entire BPEL semantics
  - One of the most extensive modeling approach w.r.t. BPEL features
  - The semantics of compensation handling is over approximated / generalized
Modeling the Behaviour of Variables

- Information carried:
  - If the variable contains data
  - If it has already been used
Fault Model and Error Propagation

- Error is propagated by basic activities (read-write)

Diagram:
- Intact Value
  - Uninitialized → Read
  - Written → Written & read

- Erroneous Value
  - Faulty Written → Faulty written & read

Legend:
- Intact Value
- Uninitialized
- Read
- Written
- Written & read
- Faulty Written
- Faulty written & read
Modeling Basic Activities

- **Activated**: the control reached the activity
- **Dotted arrow**: triggers when the containing activity finishes
Modeling Structured Activities

...<sequence>
  <invoke name="a"/>
  <invoke name="b"/>
</sequence>...

sequence=running AND invoke_a=ready →
  invoke_a=activated;
sequence=running AND invoke_b=ready AND
  invoke_a=finished → invoke_b=activated;
sequence=running AND invoke_b=finished →
  sequence=finished;

[Diagram showing states: Ready, Activated, Running, Finished, with transitions between states]
Modeling Scopes

- Activated
- Ready
- Faulthandling
- Completed with fault
- Running
- Finished
- Compensating
- Compensated

Restriction: scopes may not be executed in an iterative manner.
Constraints of Activity Triggering

... AND scope_1=faulthandling AND scope_2=compensating AND scope_3=running AND ...
Prototype Implementation

Transition system
• Abstract data

Workflow (BPEL) -> Formal model (transition system) -> Analysis model (SAL) -> SAL model-checker

Simulation
Positive result

Requirement (LTL expression) -> Negative result + counter-example
Prototype Implementation

Transition system
  • Abstract data

Workflow (BPEL) → Formal model (transition system) → Requirement (LTL expression)

Requirement
  • LTL: linear temporal logic

Simulation

Positive result

Analysis model (SAL) → SAL model-checker

Negative result + counter-example
Prototype Implementation

Transition system
- Abstract data

Workflow (BPEL) → Formal model (transition system) → Analysis model (SAL) → SAL model-checker

Requirements
- LTL: linear temporal logic

Model-checker
- Evaluation of LTL expressions
- Exhaustive state space exploration
Prototype Implementation

**Requirements**
- LTL: linear temporal logic

**Model transformation**
- VIATRA2 framework
  - graph transformation
  - abstract state machines

**Transition system**
- Abstract data

**Workflow (BPEL)**

**Formal model (transition system)**

**Analysis model (SAL)**

**Simulation**

**SAL model-checker**

**Positive result**

**Model-checker**
- Evaluation of LTL expression
- Exhaustive state space exploration

**Requirement (LTL expression)**
Preliminary Results

- Verification of a Online Shop process:
  - 10 structured activities
  - 27 basic activities

- Results
  - Negative results within 3-5 minutes
  - The proof of positive cases takes n*10 minutes
Plans for the Future

- Algorithmical generation of common requirements:
  - Uninitialized variables are never read
  - Synchronous processes never end without an answer
- Modeling the composition of multiple BPEL workflows
- Back annotation to workflow editors
Thank you for your attention!