MODEL CHECKING OF SERVICES
WORKFLOW RECONFIGURATION: A
PERSPECTIVE ON DEPENDABILITY

Juan Carlos Polanco Aguilar ¹
Koji Hasebe ¹
Manuel Mazzara ²
Kazuhiko Kato ¹

¹ University of Tsukuba – Department of Computer Science
² Newcastle University – School of Computing Science
WHAT IS WORKFLOW?

- A workflow is a model defined by a series of tasks to produce an outcome.
- Typical examples of workflow include:
  - Business office workflow.
  - Web services workflow.
  - Mobile workflow.
AN EXAMPLE OF WORKFLOW

User

Invoke Process

Order Process

Payment Received

User Response

Shop

Process Payment

Payment Company

Payment Company

3
BUSINESS PROCESS MODELING NOTATION (BPMN)

• Current Version: 2.0 (March 2011).
• Widely used standard for business process modeling.
• Simple drawing scheme that is easy to learn and train.
• Portability for many software vendors implementation
EXAMPLE OF BPMN SPECIFICATION
Workflow as a Safety-Critical System

- Workflow application examples
  - Internet shopping site.
  - Office workflow.

- Safety / security problems
  - Leakage of Information.
  - Impersonation.
  - Design error and flow bugs.
Requirements of the Verification Procedure

- Reconfiguration process is often applied in a workflow.
- Safety / security properties might be compromised after reconfiguration (deadlock, security).
- Reconfiguration is achieved through use of BPMN elements.
- Verification of required properties is achieved by the verification of the BPMN workflow specification.
**Motivation**

- Reconfiguration of workflow development and verification of requirements through research status.

Verification Procedure
**Purpose**

- Develop a formal verification for reconfiguration of workflows.
  - Based on model checking technique (SPIN Model Checker).
  - Introduce a translation algorithm from a BPMN model into Promela code.
OVERVIEW OF THE VERIFICATION

Step 1: Workflow reconfiguration to achieve flexibility.

Step 2: Translation from the BPMN model into a formal Promela model (SPIN Specification Language).

Step 3: Model checking of the Promela model with a requirement expressed in Linear Temporal Logic (LTL) formulas.

Step 4: Recursive reconfiguration to meet all the requirements.
OVERVIEW OF THE VERIFICATION–EXAMPLE: BOOKSHOP
**An Example of Requirements**

- Client sends an order, the bookshop verifies supply in the warehouse and then sends the order to client.
Introduction a parallel branching
Step 2: Translation Procedure
(1) Overview of the Procedure

- Specification of the BPMN diagram by using set theoretical notation.
- Develop and introduction of the translation algorithm by using primary elements obtained after specification.
- Modification of the translated model for a challenging verification.

Procedure Limitations

- Translation procedure not fully automated.
- After training on set theory basic concepts and Promela language, handling can be achieved.
STEP 2: TRANSLATION PROCEDURE
(2) TRANSLATION APPLICATION EXAMPLE

Partitioning of Workflow into processes
**Step 2: Translation Procedure**

(2) **Translation Application Example**

**BPMN formal specification**

- **Tasks** = \{t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{11}\}
- **Events** = \{e_3\}
- **Objects** = \{e_3, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{11}\}
- **Relation** = \{(t_3, t_6), (t_6, t_4), (t_6, t_5), (t_5, t_7), (t_5, t_8), (t_7, e_3), (t_7, t_9), (t_9, t_{11})\}
- **Connect** = \{(t_4, e_2), (t_8, t_2), (t_{11}, t_{15})\}

**Translation algorithm**

- for all \( a \in Connect(a, b) \)
  - print “chan a[buf]”
  - for all \((a,b) \in Connect(a, b)\)
    - print “a!msg”
    - print “a?msg”

**Primary Promela code**

- chan t4[1] of mtype
- chan t8[1] of mtype
- chan t11[1] of mtype
- bool flag1 = 0;
- bool flag2 = 0;
- bool flag3 = 0;

- active proctype P4()
  - t1?msg;
  - t4!msg;
  - atomic{
    - t8!msg;
    - t11!msg;
  }

- active proctype P3()
  - t1?msg; flag1= 1;
  - t4!msg;
  - atomic{
    - t8!msg; flag2= 1;
    - t11!msg; flag3= 1;
  }

**Complete Promela code**

- chan t4[1] of mtype
- chan t8[1] of mtype
- chan t11[1] of mtype
- bool flag1 = 0;
- bool flag2 = 0;
- bool flag3 = 0;

- active proctype P4()
  - t1?msg; flag1= 1;
  - t4!msg;
  - atomic{
    - t8!msg; flag2= 1;
    - t11!msg; flag3= 1;
  }

- active proctype P3()
  - t1?msg; flag1= 1;
  - t4!msg;
  - atomic{
    - t8!msg; flag2= 1;
    - t11!msg; flag3= 1;
  }
**STEP 2: TRANSLATION PROCEDURE**

(3) COMPLETE PROMELA CODE

```plaintext
mtype = {msg};

chan t1 = [1] of {mtype}
chan e2 = [1] of {mtype}
chan t4 = [1] of {mtype}
chan t8 = [1] of {mtype}
chan t11 = [1] of {mtype}
chan t12 = [1] of {mtype}
chan t13 = [1] of {mtype}
chan t18= [1] of {mtype}
chan t19 = [1] of {mtype}

active proctype P1()
{ t1!msg;
}
active proctype P2()
{ e2?msg;
}
active proctype P3()
{ t8?msg;
}

active proctype P4()
{ t1?msg;
do
t4!msg;
atomic {
t8!msg;
t11!msg
}od
}
active proctype P5()
{ do:: t12!msg; t13!msg;
od
}
active proctype P6()
{ do:: t18!msg;
}
active proctype P7()
{ t13?msg; t18!msg
}

active proctype P8()
{ t12?msg;
do:: t18!msg:: t19!msg
od
}
active proctype P9()
{ t11?msg;
}
active proctype P10()
{ t18?msg;
}
active proctype P11()
{ t19?msg;
}
```
**STEP 2: TRANSLATION PROCEDURE**

(3) **COMPLETE PROMELA CODE**

<table>
<thead>
<tr>
<th>mtype = {msg, amount};</th>
<th>active proctype P3()</th>
</tr>
</thead>
<tbody>
<tr>
<td>chan t1 = [1] of {mtype}</td>
<td>{ t8?msg; flagA = 1;</td>
</tr>
<tr>
<td>chan e2 = [1] of {mtype}</td>
<td>}</td>
</tr>
<tr>
<td>chan t4 = [1] of {mtype}</td>
<td>active proctype P4()</td>
</tr>
<tr>
<td>chan t8 = [1] of {mtype}</td>
<td>{ t1?msg; flag1 = 1</td>
</tr>
<tr>
<td>chan t16 = [1] of {int}</td>
<td>do</td>
</tr>
<tr>
<td>chan t11 = [1] of {mtype}</td>
<td>t4!msg</td>
</tr>
<tr>
<td>chan t12 = [1] of {mtype}</td>
<td>atomic{</td>
</tr>
<tr>
<td>chan t13 = [1] of {mtype}</td>
<td>t8!msg; flag2=1;</td>
</tr>
<tr>
<td>chan t18= [1] of {mtype}</td>
<td>t11!msg flag3=1;</td>
</tr>
<tr>
<td>chan t19 = [1] of {mtype}</td>
<td>}</td>
</tr>
<tr>
<td>bool flagA = 0;</td>
<td>active proctype P5()</td>
</tr>
<tr>
<td>bool flagB = 0;</td>
<td>{ t16?amount;</td>
</tr>
<tr>
<td>bool flag1 =0;</td>
<td>(amount &gt;= 4000) -&gt;</td>
</tr>
<tr>
<td>bool flag2 =0;</td>
<td>t12!msg; flagX = 1</td>
</tr>
<tr>
<td>bool flag3 =0;</td>
<td>(amount &lt; 4000) -&gt;</td>
</tr>
<tr>
<td>bool flag4 =0;</td>
<td>t13!msg; flagY=1</td>
</tr>
<tr>
<td>bool flag5 =0;</td>
<td>do</td>
</tr>
<tr>
<td>bool flagX =0;</td>
<td>:: t12!msg; flag4= 1</td>
</tr>
<tr>
<td>bool flagY = 0;</td>
<td>:: t13!msg; flag5= 1</td>
</tr>
<tr>
<td>active proctype P1()</td>
<td>do</td>
</tr>
<tr>
<td>{ t1!msg; }</td>
<td>od</td>
</tr>
<tr>
<td>active proctype P2()</td>
<td>active proctype P6()</td>
</tr>
<tr>
<td>{ e2?msg; }</td>
<td>{ t4?msg;</td>
</tr>
<tr>
<td>active proctype P8()</td>
<td>{ t12?msg;</td>
</tr>
<tr>
<td>active proctype P9()</td>
<td>do</td>
</tr>
<tr>
<td>{ t11?msg;</td>
<td>:: t18!msg</td>
</tr>
<tr>
<td>active proctype P10()</td>
<td>:: t19!msg</td>
</tr>
<tr>
<td>{ t18?msg;</td>
<td>od</td>
</tr>
<tr>
<td>active proctype P11()</td>
<td>{ t19?msg;</td>
</tr>
</tbody>
</table>
STEP 3: MODEL CHECKING VERIFICATION
(1) PROCEDURE OVERVIEW

- Introduce Linear Temporal Logic (LTL) formulas to verify the requirement.
- Typical examples: deadlock freedom, security, reachability.

**Promela code**

**Requirement (LTL)**

- Reachability:
  - LTL formula: $\Box (p \rightarrow \Box (q \rightarrow \Diamond r))$
  - define $p$ as flag1 == 1;
  - define $q$ as flag2 == 1;
  - define $r$ as flag3 == 1;
**STEP 3: MODEL CHECKING VERIFICATION**

**(2) REQUIREMENTS DEFINITION**

- Typical requirements can be identified through patterns in LTL formulas:
  - **Reachability**: “Program always reaches aimed state”
    - $\Box (P \rightarrow \Box (Q \rightarrow \Diamond R))$
  - **Deadlock**: “There is no next state”
    - **Deadlock**: $\Diamond (\Box \bot \land \neg \text{terminal})$
    - **Deadlock-freedom**: $\Box (\Box \bot \rightarrow \text{terminal})$

- **Usage of assertion alternative**

```c
active proctype P4()
{
  t3?msg; flag1 = 1;
  do
    :: t8!msg; flag2 = 1;
    :: t11!msg; flag3 = 1;
  od
  assert(flag2 == 1 && flag3 == 1)
}
```
Step 3: Model checking verification (3) Verification example

**Promela code**

```
active proctype P4()
{ t3?msg; flag1 = 1;
t4!msg;
atomic{
t8!msg; flag2 = 1;
t11!msg; flag3 = 1; }
}
```

**Model Revision**

- Property INVALID
STEP 4: RECURSIVE RECONFIGURATION OF WORKFLOW MODEL

active proctype P4()
{ t3?msg; flag1 = 1;
  do
  :: t8!msg; flag2 = 1;
  :: t11!msg; flag3 = 1;
  od
}

Promela code

Requirement (LTL)

SPIN model checker

Property VALID
RELATED WORK

- Model Checking for Web Services:

- Translation Procedures:

- Reconfiguration in Workflow Systems:
CONCLUSIONS

- Verify the procedure for the reconfiguration of workflows.
- Introduce the translation from the BPMN model into a Promela model and use SPIN model checker for verification.
- Develop a formal definition of BPMN model and introduce a translation algorithm.
- Introduce a recursive process of verification.
**Future Work**

- Develop the automation of the verification procedure.
- Include a GUI environment.
- Increase the handle ability:
  - Background process of translation and verification.
  - Develop set of templates for requirement patterns (usual LTL formula for deadlock, reachability).