Formal Transaction Modeling and Verification for an Adaptable Web Service Orchestration

Sara Zatout $^1$, Maya Souilah Ben Abdelhafid $^{1,2}$ and Mahmoud Boufaida$^1$

$^1$ LIRE Laboratory Constantine 2 University
$^2$ School of Accounting and Finances of Constantine
ALGERIA
<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>Introduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Background</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Some related work</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Proposed process</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Conclusions and Perspectives</td>
<td></td>
</tr>
</tbody>
</table>
In literature:

- **Service oriented paradigm**: the emergence of a new generation of heterogeneous and complex systems.

- **Web services**: can be composed and reused quickly to meet complex business needs (orchestration or choreography)

- **A problem**: some functionalities in the orchestration process need to be performed as a single unit of work in order to keep its integrity.

- **Main idea → Transaction integrity** for building efficient and reliable Web service based applications.
Questions

1. **INTEGRITY**: Which properties should we consider?
   - (Atomicity, Consistency)

2. **MODELING**: What do we use to model a service composition considering the integrity?
   - (CPN, ASK-CTL)

3. **VERIFICATION**: How do we verify the generated model?
   - (Model checking)

4. **ADAPTATION**: How do we provide an adaptable behavior?
   - (Patterns)
<table>
<thead>
<tr>
<th>Works</th>
<th>Verified properties</th>
<th>Modeling formalism</th>
<th>Verification tools</th>
<th>Adaptation phase (Design-time/Run-time/Not considered)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[5]</td>
<td>Atomicity</td>
<td>PN</td>
<td>Relaxed soundness</td>
<td>Not considered</td>
</tr>
<tr>
<td>[6]</td>
<td>Compatibility</td>
<td>CPN</td>
<td>Model Checking</td>
<td>Not considered</td>
</tr>
<tr>
<td>Our work</td>
<td>Consistency</td>
<td>CPN</td>
<td>Model Checking</td>
<td>Design time</td>
</tr>
</tbody>
</table>
The orchestration behavior (exchanged messages) is specified and modeled by design techniques such as UML.

The functional requirements of the Web service orchestration are specified and verified.

The non-functional aspects

Main objective

Modeling and verifying an adaptable orchestration process for achieving transactional properties

Transaction integrity

Consistency and Atomicity
A. Consistency sub-process

Step 1: CPN Modeling

* Execution states are captured by places.
* Exchanged messages, transaction names and states are captured by the color sets of the token.
* Operations of instances are captured by transitions.
* Initial state is captured by the initial marking M0.
A. **Consistency sub-process**

* Studying the behavior by making multiple simulations.
* Checking the model correctness in order to ensure that all system requirements have been executed entirely.
* It is not possible to guarantee the integrity of a CPN model by using only simulation tool → Verification phase.
A. *Consistency sub-process*

* CPN tools [7] generates the transition state system from the designed CPN model.
* It is possible to check several properties and to detect possible errors.
Step 4: Querying functions with ML

* CPN Tools provides appropriate functions for querying the generated state space to verify the consistency of system (ex. valid termination states attribute, self-loop terminal nodes attribute, absence of live locks attribute or deadlocks attribute).
* These attributes allow to prove the consistency of our transactional orchestration model.
A. **Consistency sub-process**

* CPN Tools uses the SML tool to evaluate the defined querying functions.
* For instance, “InValidTermSCC()” function verifies that the model does not include non-trivial terminal and livelocks. In this case, it must return the empty set.
**B. Atomicity sub-process**

* Extending the previous CPN model, by using a set of proposed patterns which include an adaptable behavior to deal with the different situations of transaction atomicity failures, which can be produced during execution.

* These patterns can be used in different parts of service orchestration which need to be executed in an atomic manner.
**B. Atomicity sub-process**

* These patterns include a set of configuration strategies for achieving different atomicity properties (compensable, pivot and retrievable).

+ **Strict** atomicity pattern.
+ **Alternative** atomicity pattern.
+ **Retriable** atomicity pattern.
+ Strict atomicity pattern with **substitute action**.
B. Atomicity sub-process

Step 1: CPN modeling with atomicity patterns

* Strict and alternative patterns allow to enforce compensable and pivot properties, and the retrievable pattern ensures applying a retrievable property.

* We have extended the alternative sphere mechanism to propose a substitute adaptation strategy that can be applied for all previous defined patterns.
B. Atomicity sub-process

Step 2: Atomicity properties with ASK-CTL logic

* The atomicity property should be verified to ensure that the transaction is executed entirely or not.
* The ASK-CTL logic allows the specification of this constraint formally in order to be analyzed with a model checking technique.

Example: lets consider that a1 presents an “Aborted” transaction state and a2 presents an completed state, the formula given by INV(OR(NOT(a1),POS(a2))); allows to ensure that a transaction cannot take two opposite states (aborted and Completed) state at the same time.
B. Atomicity sub-process

Step 3: Model Checking

* The model checker takes as input the state space system and the formulas that are specified in ASKCTL logic.
* It checks if our CPN model satisfies the above formulas in order to prove the property of transaction atomicity.
* CPN Tools uses the Standard ML tool to evaluate the defined formulas.
Introduction

Background

Related work

Proposed process

Conclusions and Perspectives

val it = [] : Scc list

val it = [6,18,15] : Node list
use (ogpath ^"ASKCTL/BitArray.sml");
use (ogpath ^"ASKCTL/ASKCTL.sml");
open ASKCTL;

fun myfunc1 n = (Mark.BookStoreProcess'Preparing 1 n = [(0,AtomicityControl,aborted,T1)]);
fun myfunc2 n = (Mark.BookStoreProcess'Completing 1 n = [(0,AtomicityControl,completed,T1)]);

val a1 = NF("aborted_tr", myfunc1);
val a2 = NF("completed_tr", myfunc2);

val askCTL_AtomicityControl = NOT
  [EXIST_UNTIL (TT,
    NOT
    (OR
      (NOT (NF("aborted_tr",fn)),
      EXIST_UNTIL (TT,NF("completed_tr",fn)))))) : A
val it = true : bool
eval_node askCTL_AtomicityControl InitNode ;
We have:

- Presented a modeling and verification process based on CPN and ASK-CTL logic for modeling and defining an adaptive transactional behavior of orchestration models.
- Focused on the Atomicity and the Consistency attributes.
- Used a model checking technique to check these two important properties.
- Validated our proposal by means of an example of bookstore process.

We will:

- Apply our developed system on a public dataset, permitting the feasibility evaluation of the provided approach.
- Propose mechanisms for checking and adapting the quantitative aspects of the quality of service at run time.
References


Thank you

Any questions?