PROCESS MODELING, ANALYSIS AND ENACTMENT WITH THE FTG+PM FORMALISM

Istvan David
istvan.david@umontreal.ca

July 8, 2022
What is a process?

• Basic structure: partially ordered set of activities \((A, \leq)\)
  • Typically: one init and one finish node \(\rightarrow\) lattice

• Workflow?
  • Pretty much the same as the process
  • *Perhaps* more emphasis on the repeatable activities ("work")
  • Business context: always processes, often directly associated with business goals for controlling purposes (PDCA, DMAIC, OODA, etc.) – cf. BPMN, BPEL, etc.
Process modeling

**Why?** We want to be explicit about the sequence of activities that help reaching a goal.
Process modeling

Why? We want to be explicit about the sequence of activities that help reaching a goal.

• Visualization, better understanding (visual inspection)
Process modeling

**Why?** We want to **be explicit** about the sequence of activities that help reaching a goal.

- Visualization, better understanding (visual inspection)
- Analysis, optimization
Process modeling

Why? We want to be explicit about the sequence of activities that help reaching a goal.

• Visualization, better understanding (visual inspection)
• Analysis, optimization
• Improving program design
Process modeling formalisms

Formalism = Language + Semantics = Concrete syntax + Abstract syntax + Semantics
van der Aalst Workflow patterns

Welcome to my virtual home!

The Workflow Patterns initiative is a joint effort of Eindhoven University of Technology (led by Professor Wil van der Aalst) and Queensland University of Technology (led by Professor Arthur ter Hofstede) which started in 1999. The aim of this initiative is to provide a practical basis for process technology. In particular, the research provides a thorough examination of the various perspectives (control, data, resource, and exception handling) that need to be supported by a workflow language or a business process modelling language. The results can be used for examining the suitability of a particular process language or workflow system for a particular project, assessing relative strengths and weaknesses of various approaches to process specification, implementing certain business requirements in a particular process-aware information system, and as a basis for language and tool development.

On this website you will find detailed descriptions of patterns for the various perspectives relevant for process-aware information systems: control flow, data, resource, exception handling and event log imperfections. In addition you will find detailed evaluations of various process languages, (proposed) standards or web service compositions, and workflow systems in terms of these patterns.

We encourage interactions with interested parties about this research and its applications. For example, vendors can provide self-assessments of evaluations of their products (see the Vendors Corner). Also, we appreciate any feedback in relation to our evaluations (e.g. errors or inaccuracies).

IMPORTANT ANNOUNCEMENT: EVENT LOG IMPERFECTION PATTERN COLLECTION!

The quality of event data used in a process mining analysis impacts heavily on the accuracy and reliability of analysis results. The newly added Log Imperfection pattern collection includes 11 event log imperfection patterns, distilled from our experiences in preparing event logs, that mostly capture some commonly encountered quality issues associated with event logs. These patterns form the basis for a systematic approach to identifying and repairing event log quality issues which benefits both the process of preparing an event log and the quality of the resulting event log.
van der Aalst Workflow patterns

Advanced Branching and Synchronization Patterns

Here we present a series of patterns which characterise more complex branching and synchronization patterns are often not directly supported or even able to be represented in many standards, such as BPMN, XPDL, and UML ADs.

In this revision, the Multi-Choice and Multi-Merge have been retained in their previous form. It has been recognized that there are a number of distinct alternatives to the manner in which these patterns are modeled in practice, including the Synchronizing Merge (WCP7), the Acyclic Synchronizing Merge (WCP37), and the Generalized AND Join.

In a similar vein, the original Discriminator pattern is divided into six (6) distinct variants: Cancellable Discriminator (WCP29), the Structured Partial Join (WCP30), the Blocking Partial Join (WCP31) and the Generalized AND Join (WCP32) with a more flexible AND-join useful in concurrent processes.

Of these patterns, the original descriptions for the Synchronizing Merge and the Discriminator patterns have been revised.

6. Multi-Choice
7. Structured Synchronizing Merge
8. Multi-Merge
9. Structured Discriminator
28. Blocking Discriminator
29. Cancellable Discriminator
30. Structured Partial Join
31. Blocking Partial Join
32. Cancellable Partial Join
33. Generalised AND Join
37. Local Synchronizing Merge
38. General Synchronizing Merge
41. Thread Merge
42. Thread Split

<table>
<thead>
<tr>
<th>BPMN</th>
<th>1.0</th>
<th>+</th>
<th>Supported through</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPDL</td>
<td>2.0</td>
<td>+</td>
<td>Supported by the O</td>
</tr>
<tr>
<td>UML ADs</td>
<td>2.0</td>
<td>-</td>
<td>Not supported. The</td>
</tr>
</tbody>
</table>
Engineering processes
Engineering processes (from an MDE POV)

- More emphasis on models, modeling formalisms, levels of abstraction, dependencies and traceability…

Process Model
Syntax: UML AD
Engineering processes (from an MDE POV)

• More emphasis on models, modeling formalisms, levels of abstraction, dependencies and traceability...
Engineering processes (from an MDE POV)

• More emphasis on models, modeling formalisms, levels of abstraction, dependencies and traceability…
Engineering processes (from an MDE POV)

- More emphasis on models, modeling formalisms, levels of abstraction, dependencies and traceability...
Engineering processes (from an MDE POV)

• More emphasis on models, modeling formalisms, levels of abstraction, dependencies and traceability…

Process Model
Syntax: UML AD
Engineering processes (from an MDE POV)

• More emphasis on models, modeling formalisms, levels of abstraction, dependencies and traceability...

Formalism Transformation Graph

Syntax: UML AD

Application #0: Multi-paradigm modeling (MPM)

Transformation between formalism has different uses. Firstly, certain questions about a system can only be answered in certain formalisms, necessitating model transformation to the appropriate formalism. Secondly, to elicit the meaning of a coupled model consisting of sub-models in different formalisms, transformation of these sub-models to a common formalism is appropriate.

Formalism Transformation Lattice

MPM: Model everything explicitly, at the most appropriate level(s) of abstraction, using the most appropriate modelling formalism(s) and processes.
Other applications
Model transformation chains

- Analysis, optimization, execution
- Reuse, traceability, certification, etc
The Power Window case study

• Goal: development of a Power Window software controller
  • Complex embedded system: time-critical, safety-critical, hard real-time
  • Characteristic elements: requirements, architecture, plant-environment-control, verification via PN, hybrid behavioral simulation, deployment (AUTOSAR SWCs), code gen for ECUs
DEVS as a Semantic Domain for Programmed Graph Transformation

Eugene Syriani and Hans Vangheluwe
McGill University, School of Computer Science, Montréal, Canada,
{eisyria,hv}@cs.mcgill.ca

Application #0: Multi-paradigm modeling (MPM)

Transformation between formalism has different uses.
Firstly, certain questions about a system can only be answered in certain formalisms, necessitating model transformation to the appropriate formalism.
Secondly, to check the meaning of a coupled model consisting of sub-models in different formalisms, transformation of these sub-models to a common formalism is appropriate.
Tool orchestration

Legend:
- DesignPlatform
- mechanicalModel
- Matlab
- assignMass
- Activity (automated)
- Control flow
- Object (Model)
- Object flow
- Formalism
- Typed by
- Transformation
- Transformation (by/to)
- Constraint
- Intent
- Attribute
- Relationship link
- Capability

Transformation
Formalism
Implementation
Tool
Performance evaluation of FTG+PM instances

- Activity execution time
  - Rule
    - Gaussian distribution
    - 80% of the estimations within the 20% error range: \( t(a) = N(\mu, 0.15625\mu) \)

- Execution time evolution
  - Rule
    - \( e^{-1/0.7i} \) (i: iteration)
  - Resulting values
    - 1.0, 0.2397, 0.05743, 0.01376, 0.00329...

- Decision function
  - Manually set
  - 0.99, 0.9, 0.8, 0.5, 0.2, 0.1

---

Tool support

AToMPM

https://atompm.github.io/
http://proximatoools.org/

PROxIMA
Some limitations and open questions
Potential topics to work on

• Inheritance/conformance in the FTG
• DEVS-based enactment
• Change management FTG
• Shortest path problems

• Process inference via the FTG
  • Process recommendations, process reuse
• FTG for machine learning
  • For the better understanding and generative construction of ML pipelines
Process modeling, analysis and enactment with the *FTG+PM* formalism. Istvan David • istvandavid.com • hello@istvandavid.com