From BPMN to Maude: Formal Modeling and Execution of Distributed Workflows

Thea Jenny E. Kolnes[1] and Charaf Eddine Dridi[2]

Western Norway University of Applied Sciences, Bergen, Norway
189039@stud.hvl.no[1] Charaf.Eddine.Dridi@hvl.no[2]

Abstract. Business Process Model and Notation (BPMN) is widely adopted for modeling workflows in distributed systems due to its intuitive graphical syntax. However, BPMN lacks formal execution semantics, limiting its applicability for runtime simulation and experimentation. This project presents an approach for making BPMN executable by translating BPMN models into Maude specifications. The method captures core workflow semantics, including concurrency and synchronization, enabling runtime execution and interactive simulation within the Maude rewriting logic framework. This executable formalization allows system designers to explore possible workflow behaviors under dynamic conditions and supports runtime adaptability without relying on static verification. The approach bridges intuitive BPMN modeling with formal operational semantics, providing a practical tool for analyzing and simulating workflows in complex domains such as healthcare.

Keywords: BPMN · Maude · Formal Modeling · Workflow Simulation · Operational Semantics

1 Introduction

Formal methods provide a rigorous mathematical foundation for specifying and reasoning about system behavior. They are essential for the design and analysis of complex systems, particularly in domains where correctness, safety, and reliability are critical. Various formalisms, including process algebra, Petri nets, and rewriting logic, facilitate the precise modeling of concurrency, control flow, and interaction patterns. Nonetheless, the degree to which these formalisms are accessible to domain experts remains constrained.

In contrast, BPMN has become a de facto standard for modeling workflows in both industrial and service-oriented domains. Its intuitive graphical syntax allows practitioners to describe workflows in terms of tasks, events, and control flows. While BPMN is well-suited for visualizing process logic, it lacks formal execution semantics, which limits its capacity for simulation, verification, and dynamic analysis. This project addresses this limitation by defining an executable formal semantics for BPMN using the Maude system, a rewriting logic-based formal language.

1.1 Motivation

BPMN is a standardized graphical language often used to represent business workflows in domains such as healthcare, logistics, and manufacturing, etc. It describes processes in terms of tasks, events, gateways, and control flows, offering an accessible yet expressive way to visualize operational logic.

Real-world workflows often involve complex task sequences, parallel execution paths, and event-driven control flows, which are difficult to represent in traditional BPMN tools. The absence of formal semantics, may lead to critical issues such as deadlocks, unintended task interleavings, or incorrect execution sequences may go undetected. These limitations hinder the reliability and effeciency of BPMN models in dynamic and safety-critical environments.

1.2 Objectives

The objective of this project is to develop a formal framework for modeling and executing BPMN workflows using the Maude system. Maude-based specifications [2,3,4] offer a strong semantic foundation grounded in rewriting logic, enabling the precise definition of operational semantics, conditional rules, and strategic execution patterns. These features make Maude particularly well-suited for modeling the dynamic behavior of BPMN workflows.

Maude's executable specifications enable simulation and formal verification of process behavior, ensuring that task execution sequences respect logical constraints and avoid unintended behaviors such as deadlocks or incorrect parallelism. Its support for concurrent computation and structural reasoning allows exploration of the system's reachable states and verification of correctness properties [4].

The specific objectives of this work are as follows:

- To define a formal operational semantics for BPMN constructs, including tasks, events, and gateways, using rewriting logic in Maude.
- To develop a modular specification that separates the static structure (workflow definition) from the dynamic behavior (execution rules), supporting both sequential and parallel process execution.
- To demonstrate how Maude's simulation and rewriting capabilities can be used to explore execution paths, detect behavioral anomalies, and support runtime analysis.

1.3 Paper structure

The remainder of the paper is structured as follows: Section 2 introduces the formal definitions of BPMN and Maude, in addition to related works. Section 3 provides an overview of the Maude-based approach, explaining its modules. Section 4 introduces the case study and explains the execution of the ER simulation. Section 5 summarizes the paper and presents the conclusion.

2 Prerequisites

In this section, we review the foundational concepts required for our methodology, focusing on Business Process Model and Notation (BPMN) and the Maude language.

BPMN is a standardized way to provide graphical notation to model business processes within organizations. A BPMN diagram comprises activities, events, and gateways connected by sequence flows that define the execution order and constraints. The formal semantics of BPMN, combined with its intuitive visual notation, make it accessible to both business stakeholders and technical implementers [1]. The popularity of BPMN can be attributed to its clear visual syntax and its capacity to model complex behavior through constructs such as parallelism and decision gateways.

Maude Language is a high-performance language and tool set based on rewriting logic that supports formal specification, execution, and analysis of systems. Its integration of equational logic with rewrite rules enables concise modeling of system behavior and rigorous reasoning about system properties [4]. Maude's support for concurrent rewriting and equational attributes (e.g., associativity, commutativity) allows efficient specification and verification of complex behaviors.

Formally, a Maude module defines a membership equational logic theory $T = (\Sigma, E, A)$, where:

- Σ is the signature, specifying sorts, subsorts, and operators;
- E is a set of (possibly conditional) equations;
- A is a set of equational attributes for operators (e.g., associative, commutative).

A system module extends this to a rewrite theory $R = (\Sigma, E, A, R)$, where the final R denotes a set of (possibly conditional) rewrite rules. These rules capture state transitions, enabling the formal and structured representation of system behavior. In our project, we leverage this framework to translate BPMN constructs into Maude modules, thereby enabling formal analysis and verification of business process specifications.

3 Maude-based approach for BPMN

This formal approach focuses on establishing precise definitions and execution semantics for BPMN. The goal is to formally specify processes and their behavioral properties using the Maude rewriting logic framework. In this section, we present the operational semantics of both the static structure and dynamic behavior of the BPMN models encoded in Maude. The proposed approach is implemented through a combination of functional and system modules in Maude. The entire code of the modules are linked in Section 5. These modules collectively define formal and executable semantics for BPMN, structured as follows:

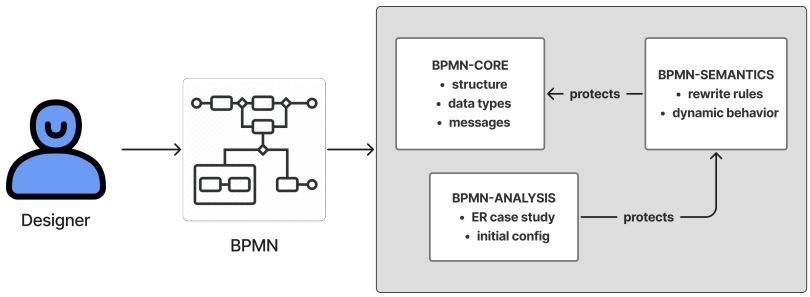

Fig. 1. Approach in Maude

– **Functional Module** `BPMN-CORE`: This module defines the structural syntax of BPMN by representing its components, such as tasks, events, and gateways, as sorts and operations. It enables the construction of complete BPMN models that begin with initial nodes and terminate at final nodes, forming a well-structured process model organized by the control flow. The start and end nodes contain solely the name attribute, whereas processes have name, resource, and status. Status can be `inactive`, `active`, or `done`. The operations utilized to initiate the rules are assigned object identifiers, which are represented by natural numbers.

```
omod BPMN-CORE is
  protecting STRING .
  protecting INT .
  protecting CONFIGURATION .

  sorts Process Status .
  ops inactive active done : -> Status [ctor] .

  op Process   : -> Cid [ctor] .
  op StartNode : -> Cid [ctor] .
  op EndNode   : -> Cid [ctor] .

  op name :_ : String -> Attribute [ctor] .
  op resource :_ : Int -> Attribute [ctor].
  op status :_ : Status -> Attribute [ctor].

  subsort Nat < Oid .

  ops triggerSNode? triggerENode? triggerSeq? : Oid Oid ->
    Msg [ctor] .
  ops triggerPar? triggerParJoin? triggerEx? triggerExJoin
   ? : Oid Oid Oid Oid -> Msg [ctor].
```

```
  ops triggerParJoin? triggerExJoin? : Oid Oid -> Msg [
    ctor].
endom
```

Listing 1.1. Structure Module

– **System Module** `BPMN-SEMANTICS`: This module imports `BPMN-CORE` and defines the operational behavior of workflows through a set of conditional rewrite rules. These rules describe how tasks and gateways are executed, supporting both sequential and concurrent execution semantics within the workflow. For example, the exclusive gateway consists of two rules: `exclusive-decision` and `exclusive-join`. Figure 1.2 shows the first rule where the branching in the exclusive gateway occurs. The rule receives four processes `A,B,C` and `D` where `Process A` is active while the rest is inactive. `Process A` finishes and the status of the process turns into `done`. Consequently, based on the lowest resource number of the processes, either `process B` or `C` becomes active.

```
omod BPMN-SEMANTICS is
  protecting BPMN-CORE .
  vars n1 n2 n3 n4 : String .

  vars A B C D : Nat .
  var N N1 N2 N3 N4 : Int .
  ...
  rl [exclusive-decision] : triggerEx?(A, B, C, D)
      < A : Process | name : n1, resource : N1, status :
    active >
      < B : Process | name : n2, resource : N2, status :
    inactive >
      < C : Process | name : n3, resource : N3, status :
    inactive >
      < D : Process | name : n4, resource : N4, status :
    inactive >
    =>
      if N2 < N3 then
      < A : Process | name : n1, resource : N1, status :
    done >
      < B : Process | name : n2, resource : N2, status :
    active >
      < C : Process | name : n3, resource : N3, status :
    inactive >
      < D : Process | name : n4, resource : N4, status :
    inactive >
      triggerExJoin?(B, D)
      else
      < A : Process | name : n1, resource : N1, status :
    done >
      < B : Process | name : n2, resource : N2, status :
    inactive >
```

```
      < C : Process | name : n3, resource : N3, status :
    active >
      < D : Process | name : n4, resource : N4, status :
    inactive >
      triggerExJoin?(C, D)
      fi .
  ...
endom
```

Listing 1.2. Semantic Module: exclusive-decision rule

– **System Module** `BPMN-ANALYSIS`: This module enables the analysis of BPMN workflows by simulating their execution through the `rew` command in Maude. It supports the exploration of state transitions from an initial system configuration to possible final states, allowing for the evaluation of multiple execution scenarios including those from the healthcare case study using the defined rewrite rules. The code of the analysis, Listing 1.3 is displayed in Section 4.2.

4 Simulation and execution

The simulation and testing are demonstrated through a healthcare case study and executed through Maude. The analysis is configured with a start node, a couple of processes, an end node, and the operations between, such as gateways and transitions.

4.1 Case study: Emergency Room (ER)

The healthcare case study is based on an Emergency Room (ER) triage process, showcasing how real-world workflows can be formally modeled and executed. The triage process starts with an assessment of the patient, such as a physical examination and the drawing of blood samples. After the assessment, the patient will receive a diagnosis. According to the diagnosis and available resources, the patient will either undergo surgery or schedule a surgery date. After surgery or scheduling, the patient will be discharged.

Figure 2 demonstrates the simulation in a BPMN model. The model does not show how the exclusive tasks are chosen, which is going to be one of the main focuses in the Maude model. The simulation in Maude has a resource attribute that represents the cost of resources needed to run the process. When the model has an exclusive gateway, it should execute the process with the least amount of resource cost.

The simulation should be abstract so that the resource attribute can serve different purposes, for example, a *weight* based on the importance of the process where the process with the lowest weight represents the most prioritized process.

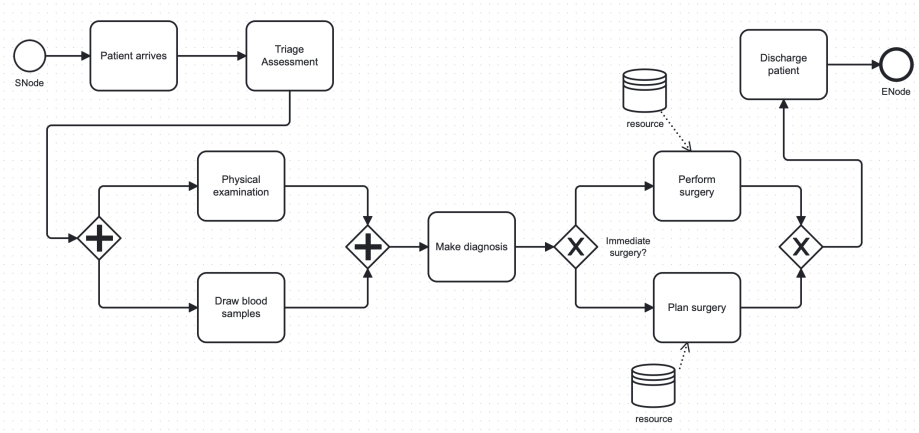

Fig. 2. BPMN model of Healthcare case study

4.2 ER simulation

In order to provide the most accurate representation of the BPMN module from Figure 2 possible, it is necessary to set up the configuration in a similar way; starting from the start node, transitioning into the processes based on their given operations, then ending up with the end node.

```
omod BPMN-ANALYSIS is
  protecting BPMN-SEMANTICS .

  ops p3 p4 p5 p6 : -> Configuration .
  ...
  eq p6 = triggerSNode?(1, 2)
    triggerSeq?(2, 3)
    triggerPar?(3, 4, 5, 6)
    triggerEx?(6, 7, 8, 9)
    triggerENode?(9, 10)
    < 1 : StartNode | name : "StartNode" >
    < 2 : Process | name : "Patient arrives", resource : 1,
   status : inactive >
    < 3 : Process | name : "Triage assessment", resource :
   10, status : inactive >
    < 4 : Process | name : "Physical examination", resource :
    100, status : inactive >
    < 5 : Process | name : "Draw blood examples", resource :
   10, status : inactive >
    < 6 : Process | name : "Make diagnosis", resource : 10,
   status : inactive >
    < 7 : Process | name : "Perform surgery", resource :
   1000, status : inactive >
```

```
    < 8 : Process | name : "Plan surgery", resource : 100,
    status : inactive >
    < 9 : Process | name : "Discharge patient", resource : 1,
     status : inactive >
  < 10 : EndNode | name : "EndNode" > .
endom
```

Listing 1.3. Analysis Module: Case study

Listing 1.3 displays the configuration of the case study; eight processes between the start and end nodes. The configuration starts with all processes set to `inactive`. To run the simulation, the configuration must trigger the start node that activates the first process with `triggerSNode?`. The model consists of one sequence flow, two parallel and two exclusive gateways. The gateways contain one gateway for the fork/branching and one for the join/merging each. In order to trigger an operation, the operators/patterns for the messages at the end of Listing 1.1, including `triggerSNode?`, `triggerPar?` and `triggerEx?` represent a trigger involving the object IDs of its processes.

For example, Maude tries to match the message `triggerEx?` along with its processes. If the pattern matches, it binds the variables and enables rewriting. In the example shown in Listing 1.3, the exclusive gateway goes from `Make diagnosis` process to `Perform surgery` or `Plan surgery` based on the resource cost of both processes.

After rewriting the configuration, the result in Listing 1.4 is obtained. All processes end up with the status `done`, except for the process that is not chosen in the exclusive gateway. In this case, the `Perform Surgery` process stays `inactive` since its cost is greater than that of the other process in the gateway.

```
result Configuration: < 1 : StartNode | name : "StartNode" >
< 2 : Process | name : "Patient arrives",resource : 1,status
    : done >
< 3 : Process | name : "Triage assessment",resource : 10,
    status : done >
< 4 : Process | name : "Physical examination",resource : 100,
    status : done >
< 5 : Process | name : "Draw blood examples",resource : 10,
    status : done >
< 6 : Process | name : "Make diagnosis",resource : 10,status
    : done >
< 7 : Process | name : "Perform surgery",resource : 1000,
    status : inactive >
< 8 : Process | name : "Plan surgery",resource : 100,status :
     done >
< 9 : Process | name : "Discharge patient",resource : 1,
    status : done >
< 10 : EndNode | name : "EndNode" >
```

Listing 1.4. Result configuration

5 Conclusion

This project has presented a formal approach that integrates BPMN with the Maude rewriting logic framework to enable runtime simulation and analysis of workflow behavior. The proposed method supports the definition of formal operational semantics for BPMN constructs while preserving concurrency and execution flow. By defining three key Maude modules, `BPMN-CORE`, `BPMN-SEMANTICS`, and `BPMN-ANALYSIS`, we have established a solid foundation for representing and executing BPMN workflows within a formal setting.

The approach captures essential BPMN constructs such as tasks, events, and gateways, including concurrency and synchronization patterns. This enables runtime simulation and exploration of workflow behaviors, supporting the detection of anomalies like deadlocks or incorrect task interleavings that are difficult to identify in informal BPMN models.

The healthcare case study demonstrated the practical applicability of executable semantics, allowing for the simulation of complex, real-world workflows with dynamic decision-making based on resource constraints. This shows how formal methods can improve the reliability and flexibility of BPMN models in safety-critical and distributed systems.

Future work would focus on extending the semantics to cover a broader range of BPMN features, such as compensation, event subprocesses, time-constraints, and advanced gateway types, as well as integrating formal verification techniques to provide stronger correctness guarantees. Overall, the integration of BPMN and Maude offers a start toward more robust, analyzable, and executable workflow specifications.

Source Code

The listings in this report have been shortened to explain certain parts. However, the entire project's source code can be found at:
https://github.com/tjekol/Maude.

References

1. Kocbek, M., Gregor, J., Marjan, H., Gregor, P.: Business process model and notation: The current state of affairs. Computer Science and Information Systems **12**, 509–539 (2015)
2. Martí-Oliet, N., Meseguer, J.: Rewriting logic as a logical and semantic framework. Electronic Notes in Theoretical Computer Science **4**, 190–225 (1996)
3. Rubio, R., Martí-Oliet, N., Pita, I., Verdejo, A.: Strategies, model checking and branching-time properties in maude. Journal of Logical and Algebraic Methods in Programming **123**, 100700 (2021)
4. Ölveczky, P.C.: Real-time maude 2.3 manual. Tech. rep., Research Report (2004), http://urn.nb.no/URN:NBN:no-35645