UNIVERSITY OF PISA

DEPARTMENT OF COMPUTER SCIENCE

Master program in Data Science and Business Informatics

Project 56

Dance school

Professor:

Prof. Roberto Bruni

Students:

Phuong Chi Huynh
Minh Duc Pham

ACADEMIC YEAR 2024/2025

Contents

1 Introduction

This report presents Business Process Model and Notation (BPMN) and Workflow Net analysis of a structured process scenario set in the context of a dance school. The objective of this report is to design a process that models how the dance school manages student requests, lesson scheduling, and interactions between students and instructors, and to analyze the correctness of the resulting model.

The main actors identified to manage the comprehensive process are: Student, School and Instructor. First, a BPMN diagram was created to visually describe the entire process. Next, the model was converted into a Workflow Net represented in Petri Net notation. Finally, a semantic analysis was carried out using tools such as WoPeD and Woflan to evaluate the behavioral properties of the Workflow Net, including soundness, deadlock-freeness, boundedness, and proper termination.

2 BPMN Design

2.1 Original BPMN

According to the project description, *two possible modeling approaches* were considered for the BPMN design of the Dance School scenario. The first approach is to use three separate pools: Student, Administrator (representing the school), and Instructor. The second way is to use a collaboration process with two pools: Student and School. Within the School pool, there are two lanes representing the Administrator and the Instructor. In this project, we adopted the second approach because it better reflects the communication and coordination within the school organization, while keeping the model readable and compact. The model was implemented using the **BPMN.io tool**.

The main idea of the process is the interaction of student and dance school (including administrator and instructor). It starts when the student requests a course, selects their preferred class, schedules an appointment with an instructor, participates in lesson activities, and finally makes a payment to the school. After the lesson, the student can either **request a new appointment** or **end the course**. The communication between the Student and School pools is represented through **message flows**.

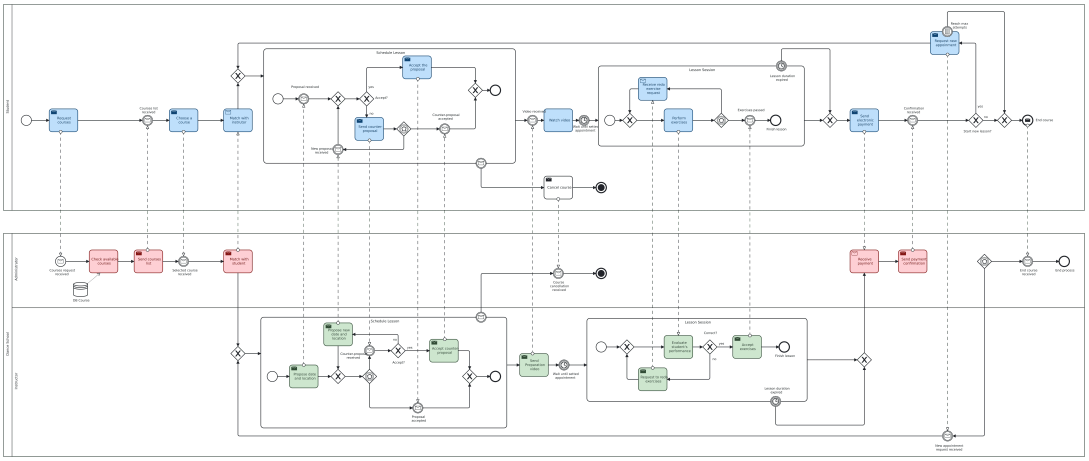

Figure 1: Original BPMN

To manage complexity, we decided to use subprocesses to describe the activities for Schedule Lesson and Lesson Session in both pools. These subprocesses encapsulate internal activities and looping behavior for both actors. Initially, we modeled the process with a flat sequence of tasks and events, but this approach quickly became difficult for stakeholders to read and

follow the process. Therefore, subprocesses were introduced to simplify visualization and ensure modularity. To prevent the subprocesses with loop activities from becoming infinite loops, we decided to used **interruption boundary conditions** in order to limit the number of iterations inside each subprocess. It means that when the process reaches the boundary condition, all activities inside the subprocess stop, and the flow moves to the end of the subprocess. For example, in the lesson appointment subprocess, if the student decides not to proceed the course, they can cancel the course, and the process will end immediately. A *message boundary event* is added to represent this situation. In the lesson session subprocess, a *timer boundary event* is used. It is the interruption boundary, when the time limit is reached, the process ends automatically. If the session duration expires, whether or not the student has completed the exercises, the lesson ends. These conditions make the process more realistic and practical. Additionally, a *conditional boundary event* called "Reach max attempts" was added to limit how many times a student can request a new appointment. This ensures realism by reflecting the finite number of available courses and preventing unbounded process execution.

Regarding gateways, we carefully selected between exclusive (XOR) and event-based gateways. Event-based gateways were used when the process flow depends on external events (e.g., waiting for a student's or instructor's response), while exclusive gateways were used for data-based decisions within the process.

2.2 Modified BPMN

Following the final project requirement, the BPMN model was extended to allow the student to **start a new learning path** after completing a course. The modified model keeps the same structure and activities as the original version, with the main change occurring after the "End Course" task in the Student pool. An **Exclusive Gateway** was added to represent the decision of whether the student wants to enroll in a new course. Similar to the above session, due to the resource limitation, the task "Request New Course" also includes a *conditional boundary event* called "Reach Max Attempts". If the student chooses "Yes", the task will check the number of attempts. If it is below the requirement, the flow returns to the point before the student receives the courses list. If the number of attempts is equal to the maximum attempts, the flow moves to the notification school to end the current course. In the School pool, a new message event named "New Course Request Received" was added after the event-based gateway. This event is triggered by a message flow coming from the "Request New Course" task of the Student pool. After receiving this message, the flow proceeds through an **inclusive gateway**, allowing parallel handling of multiple potential follow-up activities, such as new course assignment or new lesson assignment or end course notifications.

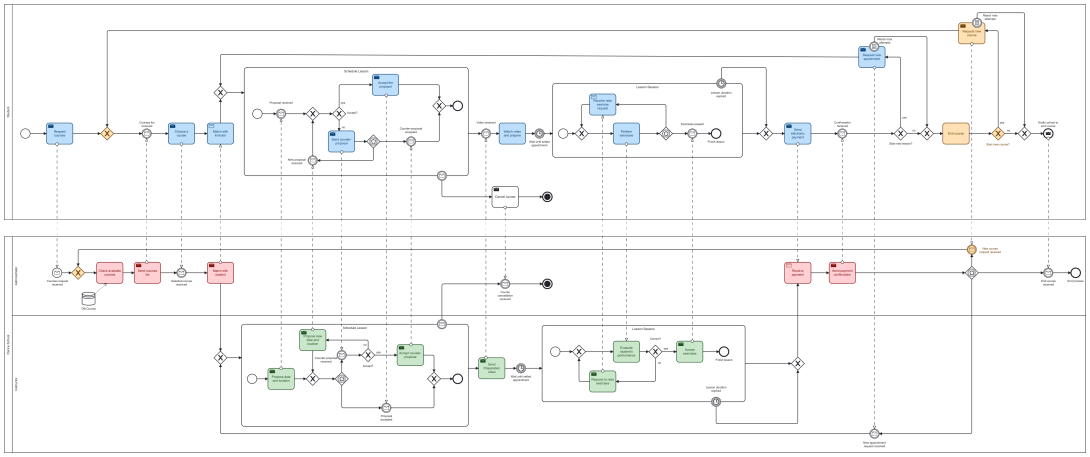

Figure 2: Modified BPMN

This modification ensures that the BPMN model satisfies the extended requirement - allowing continuous learning while maintaining control over process repetition through well-defined boundaries and decision logic.

3 Transformation to Workflow Net

3.1 Workflow module

In this section, we describe the transformation of the Dance School process from its modified BPMN model into a workflow net using WoPeD. Each pool in the BPMN collaboration diagram was first converted into an individual workflow module, and afterward, these modules were combined through communication places to form the complete workflow net representing the entire process. The transformation followed the standard three-steps methodology for converting BPMN diagrams into workflow net, as below:

1. Convert sequence and message flow: each flow in the BPMN model was replaced by a corresponding place in the workflow module

2. Convert flow objects: transitions were introduced for each BPMN flow object, including tasks, events, and gateways. Special attention was given to accurately modeling control-flow behavior, such as XOR splits and joins, Event based gateway.

3. Enforce into initial/final place: a single input and output place were created to satisfy the definition of a workflow module. These places unify multiple start and end events, where necessary, and define the entry and exit points of the process execution.

To simplify the soundness analysis in the subsequent sections, all subprocesses were expanded into their underlying transitions and places. After converting to workflow module for each distinct pool, we did the semantical analysis in Woped to verify fundamental properties such as liveness, boundedness, and soundness.

The **Student-side workflow net** (Fig. 3) models the interactions initiated by the student throughout the process. It includes *46 places*, *54 transitions*, and *108 arcs*, without subprocesses or composite operators. Each place and transition lies on a path from a **unique source place** to a **unique sink place**, satisfying the definition of a workflow net. It is S-coverable, with a single S-component, covering all places, which shows that every part of the workflow contributes to the overall behavior. No incorrect operator usage or free-choice violations were detected. The model is well-structured, with no PT or TP handles, ensuring syntactic correctness of the control flow. Moreover, the net was verified to be live, bounded, safe, and sound, meaning that every transition can eventually fire, no place can hold infinite tokens, and the process always terminates correctly.

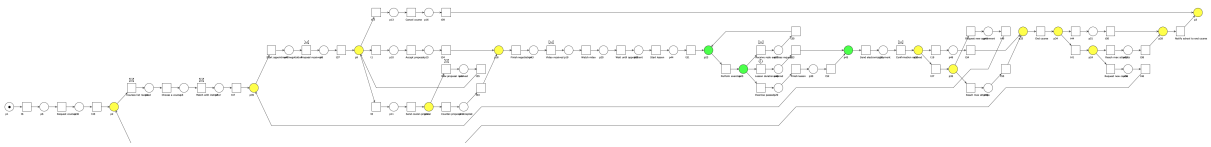

Figure 3: Workflow module for Student

The **School-side workflow net** (Fig. 4) models the internal behavior of the Dance School, including both Administrator and Instructor lanes. It includes *39 places*, *45 transition*s, and *90 arc*s, with no subprocesses or operators. Similar to the Student Module, the School Module maintains strict workflow net structure with single source and sink places. The analysis confirmed that the net is bounded, with no places capable of accumulating an infinite number of tokens. It is live, with no dead or non-live transitions, and the initial marking is correct, as

no places are wrongly marked. The model is also S-coverable, featuring a single S-component, and is wellstructured, with no structural inconsistencies or unconnected regions. These results confirm the correctness of the workflow behavior before the composition phase. The detail of analysis are showed in the Appendix section.

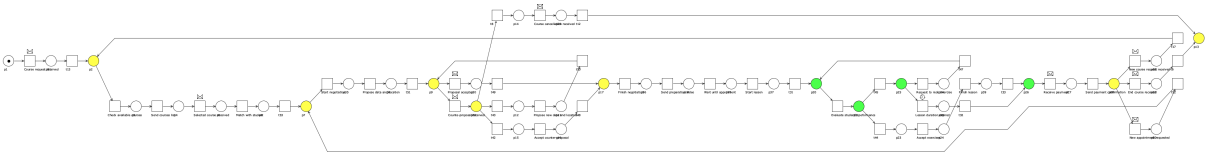

Figure 4: Workflow module for School

3.2 Integrated Workflow Net (Composition)

After verifying each individual workflow module (Student and School), both modules were integrated into a unified collaborative workflow net to represent the complete dance school learning management process. The integration was performed using communication places, which serve as synchronization points between processes to enable message passing. This approach adheres to the workflow module composition principle, ensuring modularity and logical consistency between independent actor processes.

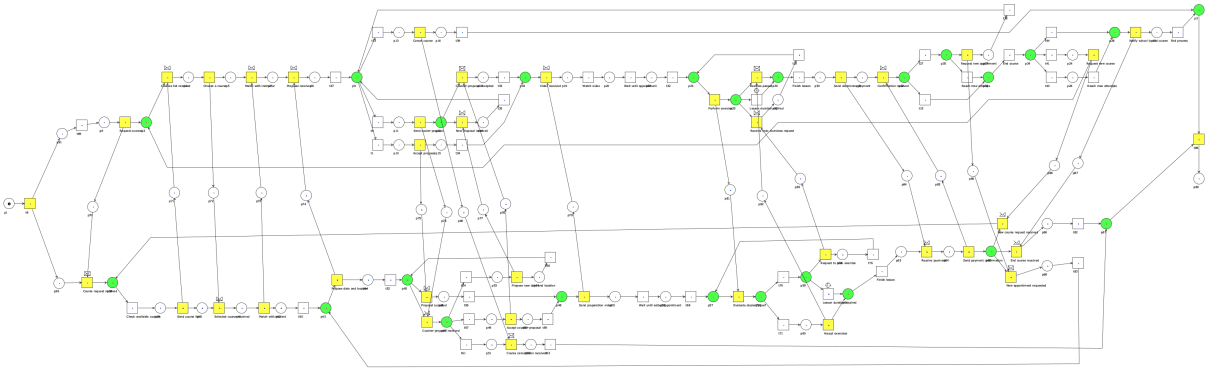

Figure 5: Collaboration Workflow net

The unique input place was initialized and connected to the entry points of both modules, synchronizing the start of their interactions. Similarly, a unique output place was added, connected via a joint transition that requires tokens from both the Student and School sides - ensuring the process terminates only when both actors complete their workflows.

The resulting integrated workflow net consists of a large munber of places and transitions due to the merging of subprocesses and message interactions. The complexity of the combined model reflects the collaborative nature of the system but also introduces challenges in maintaining structural and behavioral soundness.

4 Analysis and Evaluation

4.1 First result

The first version of the integrated workflow net was tested for soundness in WoPeD. The semantical analysis detected the following issues: The Initial Marking property was satisfied. However, **boundedness, liveness, well-structuredness, and free-choice property were violated**. The model contained **two unbounded places** and **two dead transitions**, namely "Course cancellation received", which never fired in any execution path. This indicates that

while the process started correctly, some tokens were trapped in internal loops, and certain transitions could never occur due to missing synchronization or improper message connections.

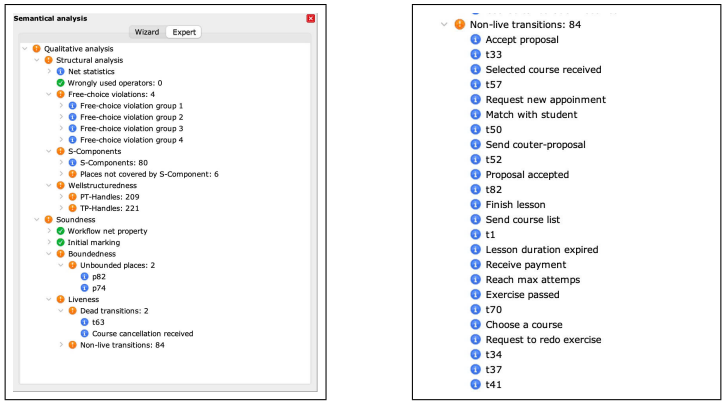

Figure 6: Semantical analysis results of the collaboration workflow net

Moreover, **84 transitions were detected as non-live**, including "Accept proposal," "Send counter-proposal," "Receive payment," "Request new appointment," and others. These transitions were correctly modeled individually but became unreachable after merging due to dependency mismatches between the Student and Instructor modules.

4.2 Woflan Diagnosis and Fixing

Because the WoPeD semantical analysis required a long runtime and was prone to freezing with large nets, Woflan was used as a secondary diagnostic tool. Woflan confirmed, similar to WoPeD, that the process is indeed a workflow process definition, but it reported improper conditions, 239 AND-OR mismatches, and 27 improper scenarios. Based on these findings, several improvements were implemented:

- Removed the *Course cancellation* boundary event, which caused dead transitions and inconsistent synchronization.
- Added synchronization places to align dependent transitions between Student and School modules.
- Inserted waiting transitions to ensure message order consistency, preventing premature token firing.

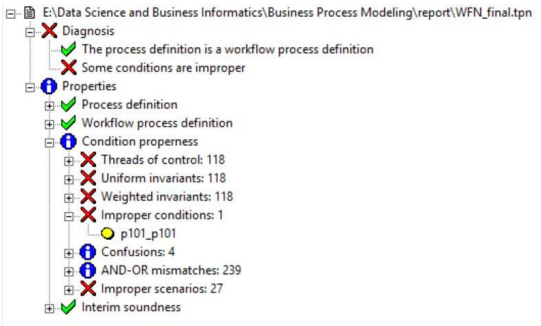

Figure 7: Woflan diagnosis result

- Verified behavior through Token Game simulation to identify stuck tokens and validate reachability manually.
- Cross-verified using Woflan to detect structural inconsistencies and invariant violations.

4.3 Final Result

After the improvements, the refined workflow net was re-evaluated in WoPeD and validated using Woflan. Both tools independently confirmed **soundness**, providing high confidence in the correctness of the composition.

The model now satisfies the **Workflow Net Property**, **Initial Marking**, **Liveness**, and **Boundedness**. It contains **2,244 S-components**, and all places are covered, indicating that every place contributes to at least one behavioral sequence of the net.

Four groups of Free-choice violations remained, caused by overlapping message interactions between the two pools. These violations are acceptable in collaborative nets since decisions depend on external dependency rather than purely local decision. (See Table 1 of Free-choice Violation Groups - Technical Analysis in Appendix for more details). According to Desel and Esparza (1995), the free-choice property is a sufficient but not necessary condition for soundness. Non-free-choice constructs are acceptable in collaborative systems when: (1)They are intentional and documented, (2)The model remains structurally sound, and (3) All behavioral invariants are maintained. Our workflow satisfies all these criteria.

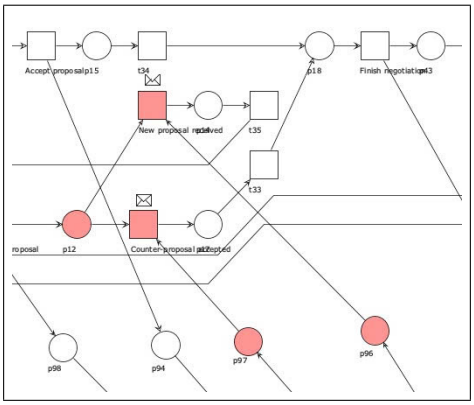

Figure 8: Free-choice violation group 1

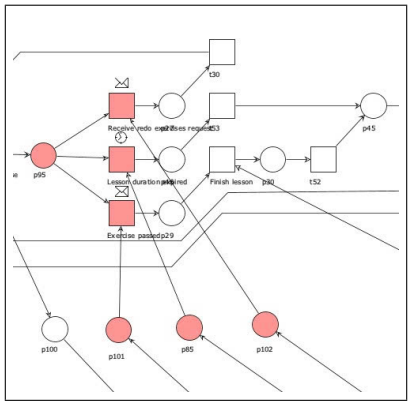

Figure 9: Free-choice violation group 2

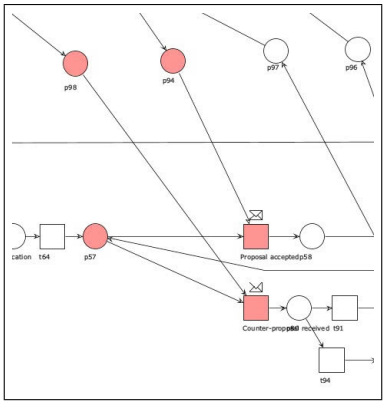

Figure 10: Free-choice violation group 3

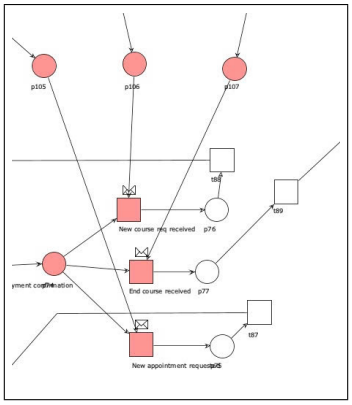

Figure 11: Free-choice violation group 4

A **reachability (coverability) graph** was constructed to evaluate the dynamic behavior of the improved integrated workflow net. In earlier versions of the composite model, attempts to generate the reachability graph resulted in state explosion, due to excessive branching and unbounded loops between asynchronous modules. After synchronization and structural refinements, the final workflow net demonstrated significant improvement. The latest version coverability graph was generated more readable and simpler, showing a manageable number of reachable states-estimated at several hundred unique reachable states-representing all possible execution paths of the system. The graph confirms that the unified process terminates correctly in the final marking and exhibits no deadlocks or infinite repetitions.

Invariant analysis further reinforces these findings. The net contains 2,244 S-invariants, demonstrating strong local safeness and consistent resource conservation across all subprocesses.

No PT or TP handles were detected, indicating that all causal dependencies are properly defined and no transitions remain without valid input or output arcs.

Overall, the combined coverability and invariant analyses confirm that the improved workflow net is bounded, safe, and sound, ensuring stable synchronization between concurrent modules and guaranteeing proper termination under all execution conditions. Compared with the earlier composite version, this analysis verifies that the synchronization improvements successfully eliminated unbounded loops and dead transitions, leading to a more efficient and semantically correct model.

4.4 Soundness discussion

In the earlier version of the composite workflow net, the model exhibited **weak soundness** rather than relaxed soundness. Weak soundness means that, from the initial marking, at least one execution path could reach the final marking without deadlocks, but not all intermediate states necessarily led to completion. This occurred because asynchronous message exchanges between the Student and Instructor modules introduced intermediate markings that depended on external synchronization - for instance, the instructor's actions such as "Receive proposal accepted" or "Accept counter-proposal" could only occur after the student had completed corresponding decisions like "Accept proposal" or "Send counter-proposal". These interdependent message flows created temporary waiting states that relied on external coordination rather than internal control logic. Such behavior is typical in collaborative and distributed workflows, where timing and message dependencies create non-deterministic execution paths. While the weakly sound model guaranteed that at least one process instance could terminate correctly, it did not ensure full behavioral correctness under all communication scenarios.

After applying diagnostic and synchronization improvements the refined integrated model achieved full soundness. Although some Free-choice violations remain, they are structurally justified by the asynchronous nature of the collaboration. These violations represent realistic message-driven decisions rather than modeling flaws, and they do not compromise formal correctness. This distinction between the earlier weakly sound model and the final sound model highlights the importance of synchronization and message dependency handling in collaborative workflow design.

4.5 Conclusion

The final integrated workflow net clearly represents the collaborative process between the Student and the Dance School. Through several rounds of debugging in WoPeD and Woflan, and by adding synchronization through communication places and token-based counters, we achieved a bounded, live, well-structured, and sound model that accurately reflects how the process works in reality.

The conversion from BPMN to the workflow net was done step by step, following a formal and consistent approach. Each workflow module was tested individually to make sure it was sound, bounded, and live before combining them. When integrated through communication places and synchronization points, these properties were preserved with minimal structural overhead.

The final workflow net provides a complete and validated model of a collaborative system. It shows that even complex interactions between different participants can be modeled and verified accurately using workflow net theory and modern analysis tools. The few remaining Free-choice violations are intentional and well-documented - they represent realistic decision-making and timing dependencies without breaking the model's correctness.

Overall, this verified workflow net can be used as a solid foundation for system implementation, performance checking, and future improvement. It captures practical business constraints such as time limits and restricted resources, ensuring that the model is both logically correct and realistic in operation.

A. Appendix

A.1 Appendix: Workflow Module Analysis Result

This appendix presents the detailed semantical analysis result of workflow modules.

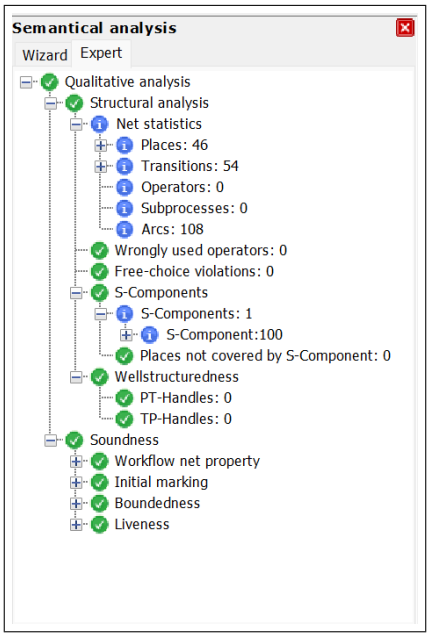

Figure 12: Semantical analysis result of Student workflow module

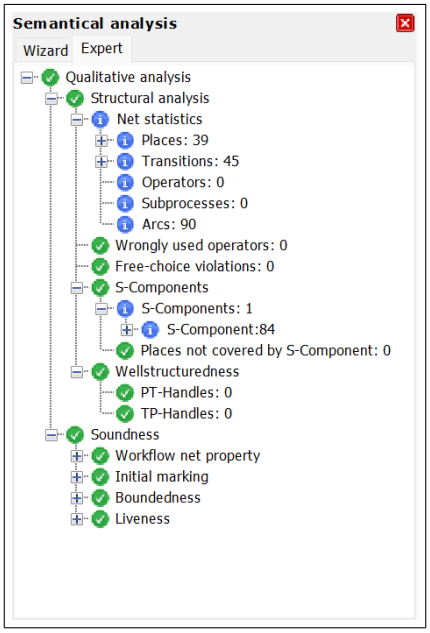

Figure 13: Semantical analysis result of School workflow module

A.2 Appendix: First Collaboration Workflow net

This appendix presents the result of the first collaboration workflow net (before fixing)

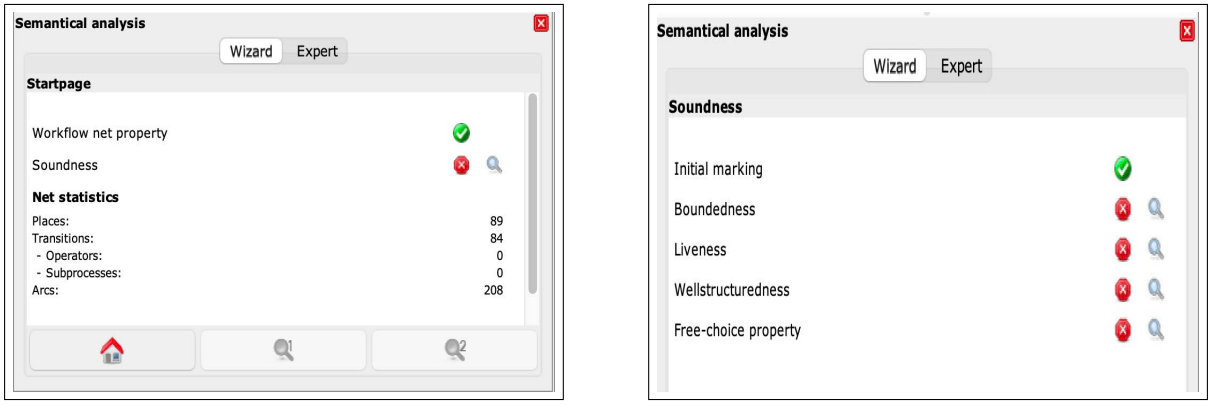

Figure 14: Semantical analysis results of the first collaboration workflow net (before fixing)

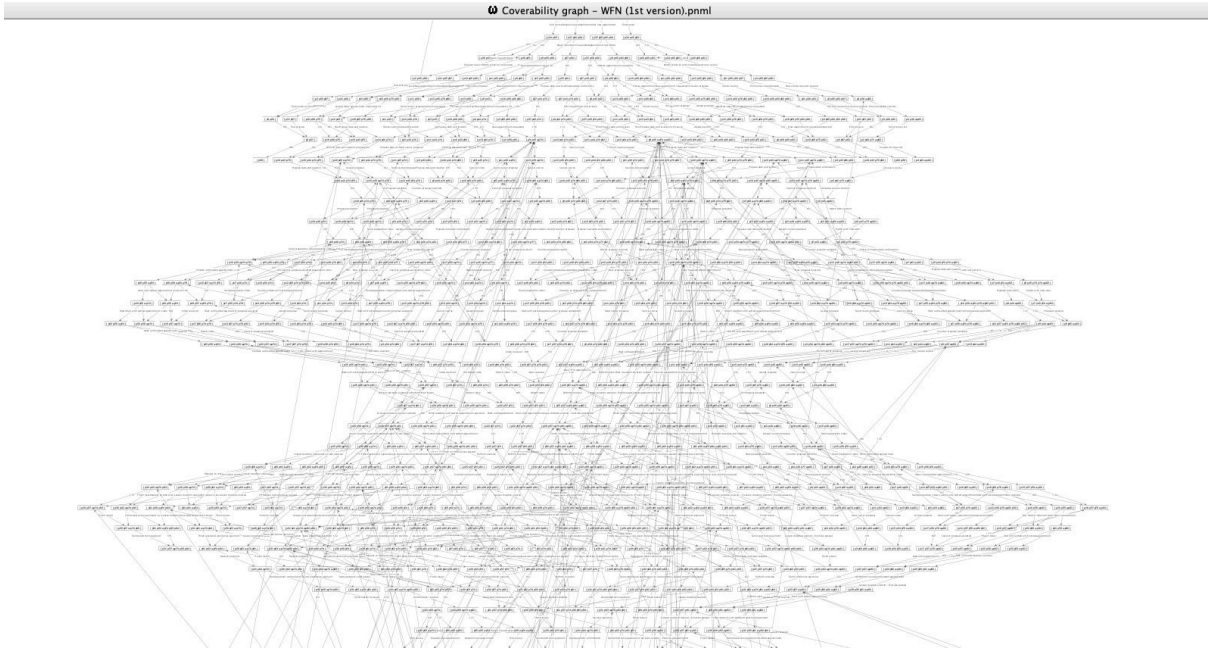

Figure 15: Coverability Graph of the First Collaboration Workflow Net

A.3 Appendix: Final collaboration Workflow net

This appendix presents the final collaboration workflow net after fixing.

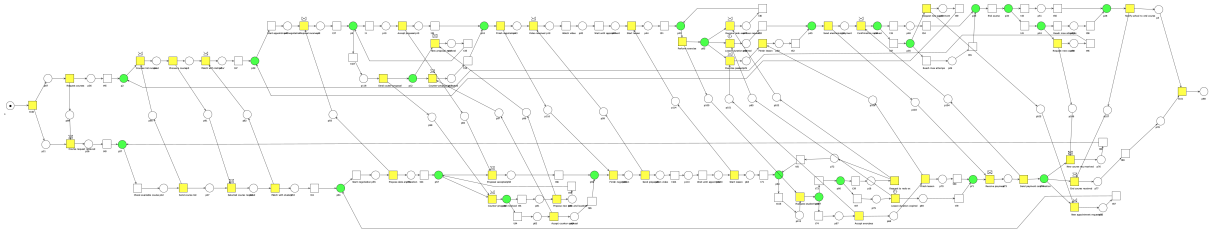

Figure 16: Final collaboration Workflow net

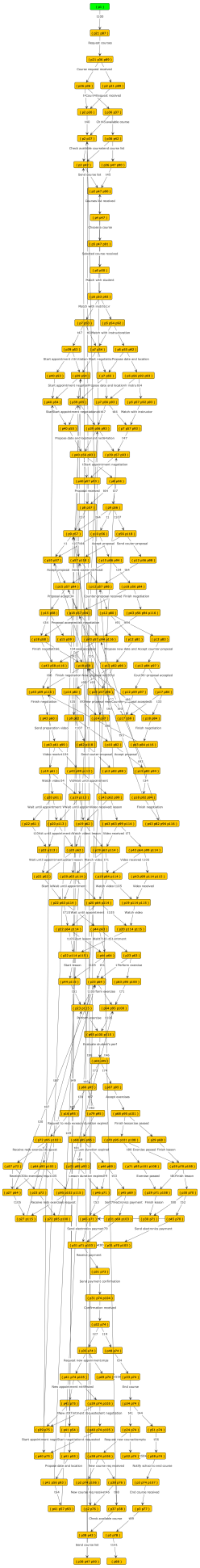

Figure 17: Coverability Graph of the Final Collaboration Workflow Net

A.4 Appendix: Free-choice Violation Groups - Technical Analysis

.

Table 1: Free-choice Violation Groups - Technical Analysis

Group	Area	Root Cause	Type	Mechanism	Necessity
1	Proposal Negotiation	multiple outputs with asymmetric inputs	Asymmetric confusion	"New proposal received" vs "Counter-proposal accepted received" transitions have different input requirements (needs instructor decision)	Negotiation requires external party input; cannot simplify to local choice
2	Exercise Loop Management	loop exits with different conditions	Multiple exit conditions	Three exit paths: (1) continue (result only), (2) pass (result only), (3) timeout (result+timer)	Bounded iteration with multiple criteria; inexpressible as free-choice
3	Decision Sync	decisions depend on external module states	External state dependency	Decision transitions check both local (instructor) and external (student) conditions	Multi-actor constraint checking; requires information integration
4	Request Coordination	instructor responses depend on student state	External state dependency	Instructor has multiple response options but which fires depends on student's request context.	Accurately models reactive coordination where service provider (instructor) adapts behavior based on client (student) state.