UNIVERSITÀ DI PISA

Data Science and Business Informatics

Business Project Modelling
Project Report

Studente:

Salvatore Puccio
547518

ANNO ACCADEMICO 2024/2025

Contents

1 Introduction

Consider a personalized online learning course scenario. The **student** contacts the school's **secretariat** and receives a **list of available courses**. The **student** chooses the course and is assigned a **teacher**/tutor. After the **student** makes the **first electronic payment**, the **teacher** proposes a **lecture schedule**, and the **student** can accept it or propose changes. The process continues until the meeting is fixed. During the lecture, the **teacher** explains concepts, discusses their characteristics with the **student**, answers any questions, and proposes exercises that the **student** attempts to solve. At the end of each lecture, the **teacher** uploads study materials and exercises to the **e-learning platform**. Between lessons, the **student** and **teacher** interact via a **dedicated chat** while the **student** consults the materials provided. During the last lecture, the **teacher** assesses the **student** by asking questions and evaluating the responses until a **grade** is formulated and communicated to the **secretariat**. At the end of the course, the **student** settles the final installment and receives a **certificate** with the teacher's grade, and the process is completed.

Model the appropriate processes that faithfully reflect the scenario described and are compatible. Modify the processes so that the **student** may choose to begin a new learning path at the end of the course.

This report illustrates the modeling and analysis of the learning process described above. The process is represented through *BPMN* collaboration diagrams and then translated into *Workflow Nets* in order to verify properties such as soundness, liveness, and boundedness.

2 BPMN Diagram

The BPMN diagram was created using Camunda Modeler. The choice of BPMN was motivated by the greater familiarity with this notation acquired during the course. Moreover, Camunda Modeler was preferred due to the personal confidence gained with the tool throughout the project.

Process Description

The process involves three main actors: the student, the teacher, and the school's secretariat. There are two pools: one representing the student and one representing the school. The school's pool is divided into two lanes: one for the secretariat and one for the teacher. The student and the teacher are the key participants driving the process, while the school's secretariat — which shares the same pool as the teacher — plays a supporting role, intervening mainly in the initial and final phases.

1. **Initialization Phase.** The process starts when the student contacts the school's secretariat to request information about the available courses. The secretariat replies by sending the list of courses. After reviewing it, the student selects one and receives the contact details of the assigned teacher. At this point, the student proceeds to pay the first installment and waits to receive from the teacher a proposal for the course schedule. This phase defines the preliminary administrative interactions and sets up the conditions for the educational relationship to begin.

2. **Scheduling Phase.** The teacher initiates this phase by proposing a first version of the lesson calendar. The student can either accept it or suggest modifications. When a change request is made, the teacher updates the calendar and sends a new proposal. It is assumed that the teacher may adjust the schedule according not only to the student's requests but also to personal constraints. Each student request triggers a response from the teacher, creating an iterative cycle controlled by the student. The phase concludes when the student explicitly accepts the calendar, thus confirming participation.

3. **Lesson Execution Phase.** This represents the core part of the process. Each lesson begins when the teacher calls the student, who answers and waits for the session to start. The teacher presents at least one concept and engages in a brief exchange of arguments with the student. The student may then ask questions—none, one, or several. For simplicity, the process assumes that the student controls this loop: as long as the student continues asking, the teacher keeps answering. Once the student indicates that there are no further questions, control returns to the teacher, who decides whether to present a new concept or move on. If no new concept is introduced, the teacher assigns a set of exercises, which the student attempts to solve immediately and submits in a

single batch. The teacher then uploads the lesson materials and additional exercises—possibly including those the student could not solve—onto the platform. A notification is sent to the student, marking the end of the lesson session.

4. **Post-Lesson Interaction.** Once the lesson is closed, control passes to the student. The student consults the uploaded materials and communicates with the teacher through a dedicated chat. It is assumed that at least one message is exchanged. The conversation loop is again under the student's control: messages are exchanged until the student decides to stop sending them and to close the chat. This action is interpreted as the completion of the material consultation. Control then returns to the teacher, who checks the calendar to determine whether further lessons are planned.

5. **Lesson Iteration Control.** The teacher verifies whether the next lesson is the final one. If not, the process returns to the lesson execution phase, and another session is initiated following the same structure. Otherwise, the model transitions to the final examination phase. This conditional branch reflects the teacher's role in controlling the course progression according to the pre-defined schedule.

6. **Final Examination Phase.** During the last session, the teacher conducts an examination to assess the student's learning outcomes. The teacher asks one or more questions, and the student responds. The teacher may repeat the cycle several times until able to formulate a final judgment. When the teacher decides to end the exam, the grade is communicated to both the student and the school's secretariat. The student then completes the administrative closure by paying the second installment and receiving the certificate of completion. This concludes the course from both academic and administrative perspectives.

2.1 Termination and Optional Restart

Optionally, a restart mechanism can be introduced, allowing the student to enroll in a new course and begin a new learning cycle under the same structure.

Different interpretations of this restart were possible. One possible approach was to consider that, when the student decides to start a new course, there is no need to repeat the entire initialization phase. In that case, the process could restart directly from the course selection and first installment payment, without contacting the school's secretariat again. According to this interpretation, the student would pay the first installment for each course undertaken, and a single final installment only when requesting the overall certificate covering all completed courses.

However, for the sake of simplicity and clarity, the model adopted in this work assumes that each course requires its own completion payment and results in a distinct certificate. Hence, every course instance is treated as a separate process with two installments and one certificate.

Since the variant between the two interpretations is minimal, the workflow module was later analyzed directly according to the restart variant, as discussed in the following sections.

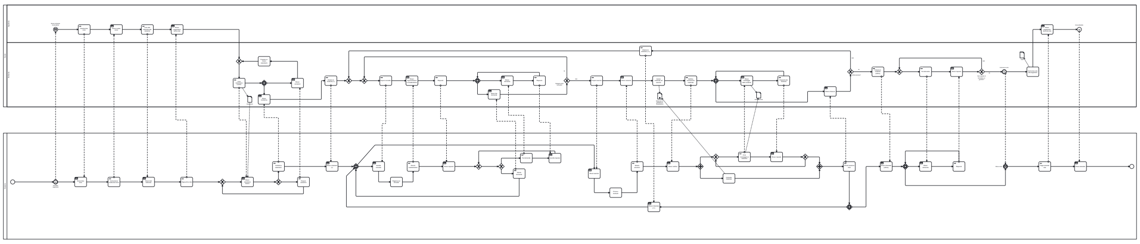

Figure 1: Collaboration Diagram

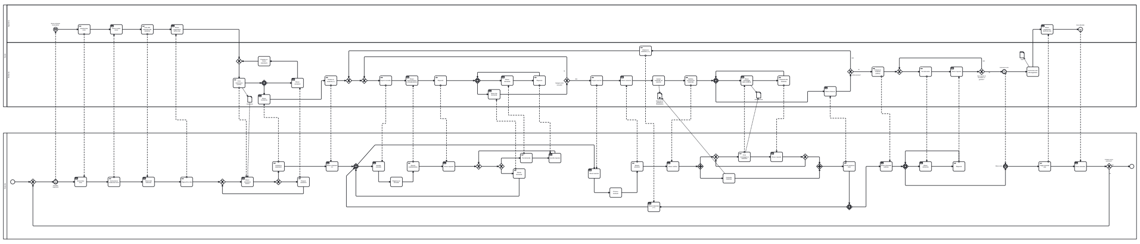

Figure 2: Collaboration Diagram Variant

3 Workflow nets

The tool used for modeling was WoPeD, running on Windows 11.

3.1 Conversion

The workflow nets were created by translating the BPMN collaboration diagram using the technique explained during the course, following a three-step procedure:

Step 1: Insert one place for each sequence flow and message flow.

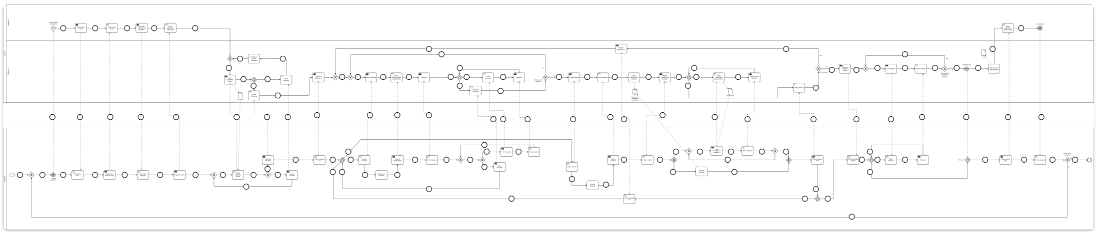

Figure 3: BPMN to WF-net conversion – Step 1

Step 2: Insert transitions. Replace AND/XOR joins and splits with appropriate transitions, and insert places instead of event-based gateways, removing the incident places.

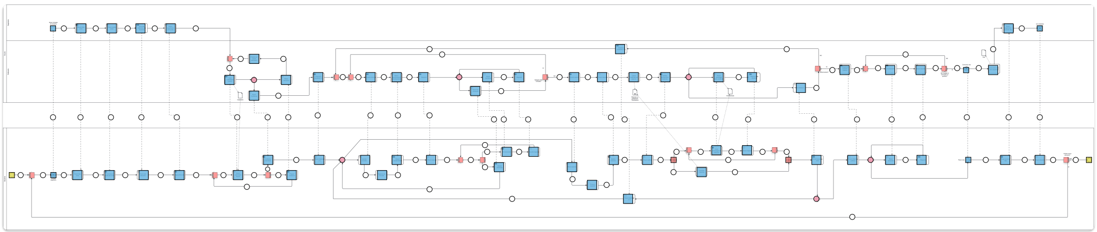

Figure 4: BPMN to WF-net conversion – Step 2

Step 3: Ensure that there is a single start place and a single end place.

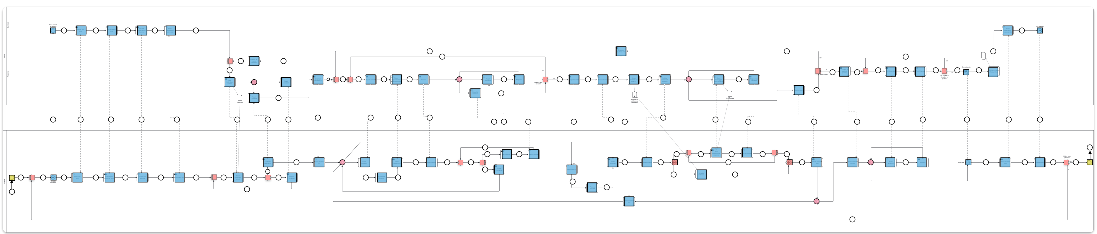

Figure 5: BPMN to WF-net conversion – Step 3

3.2 School Workflow Net

The school's workflow net, representing the combined activities of the teacher and the school's secretariat, consists of 41 places, 46 transitions, and 92 arcs. The network exhibits no structural anomalies, and the WoPeD soundness check confirms that it is **bounded**, **live**, and properly initialized. No free-choice violations or wrongly used operators were detected, and the presence of valid **S-components** confirms a well-defined design. The coverability graph (Figure 8) is finite and leads to a single terminal marking, confirming that every execution eventually terminates with the delivery of the certificate to the student.

3.3 Student Workflow Net

The student's workflow net consists of 43 places, 47 transitions, and 96 arcs. The WoPeD analysis confirms that the model satisfies all main properties: no wrongly used operators or free-choice violations were detected, and the network is composed of valid **S-components**, confirming its well-structured nature. The soundness analysis verifies that the net is **bounded**, **live**, and correctly initialized, ensuring that every instance can always reach the final marking without deadlocks or residual tokens. The corresponding coverability graph (Figure 11) is finite and includes all reachable markings, confirming that the process behaves correctly from start to completion.

3.4 Student Variant Workflow Net

The variant of the student's workflow net includes 46 places, 51 transitions, and 104 arcs. This model introduces the optional restart mechanism, allowing the student to begin a new course after the completion of the previous one. Despite the added complexity, the analysis results remain identical: the net is free-choice, well-structured, and sound. The coverability graph confirms **boundedness** and **liveness**, showing that the restart loop correctly reinitializes the process without introducing infinite behavior. This demonstrates that the extended version preserves all behavioral properties of the base model.

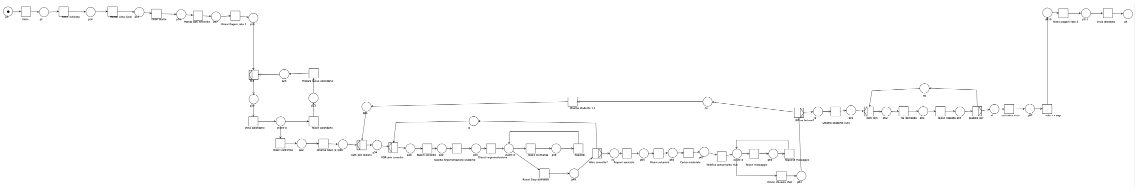

Figure 6: School workflow net

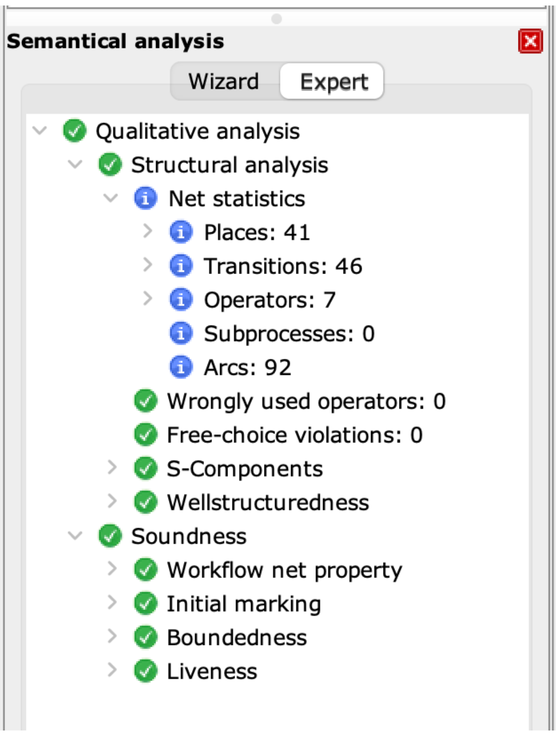

Figure 7: School wfnet Analysis

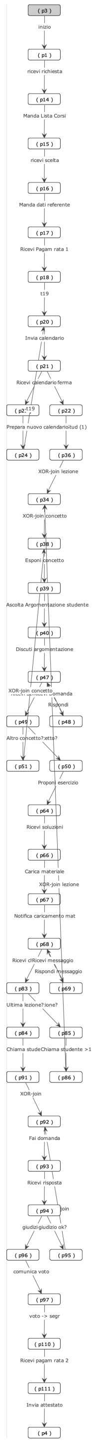

Figure 8: School Coverability Graph

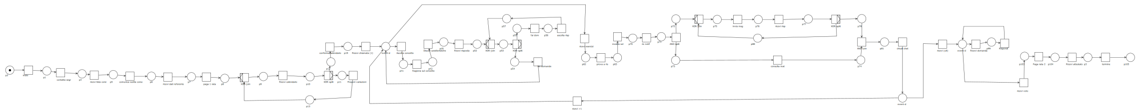

Figure 9: Student workflow net

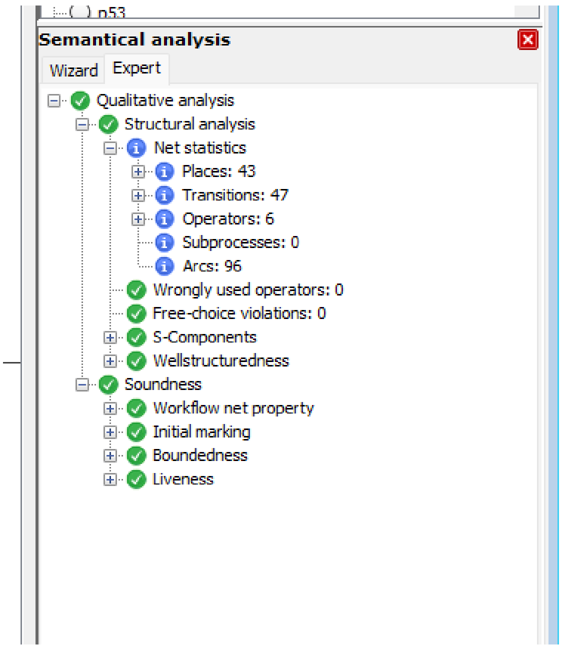

Figure 10: Student wfnet Analysis

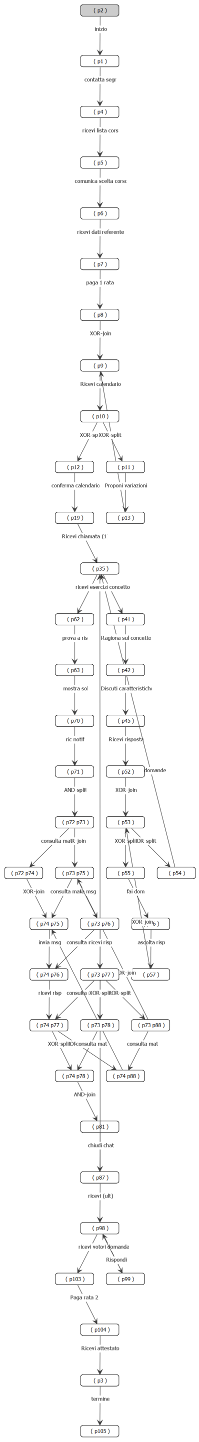

Figure 11: Student Coverability Graph

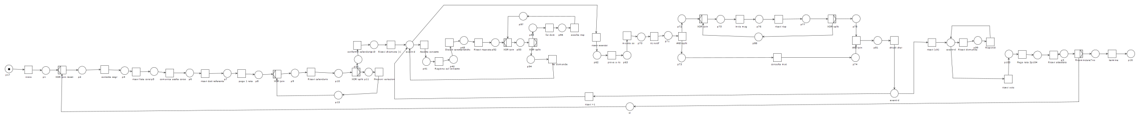

Figure 12: Student Variant workflow net

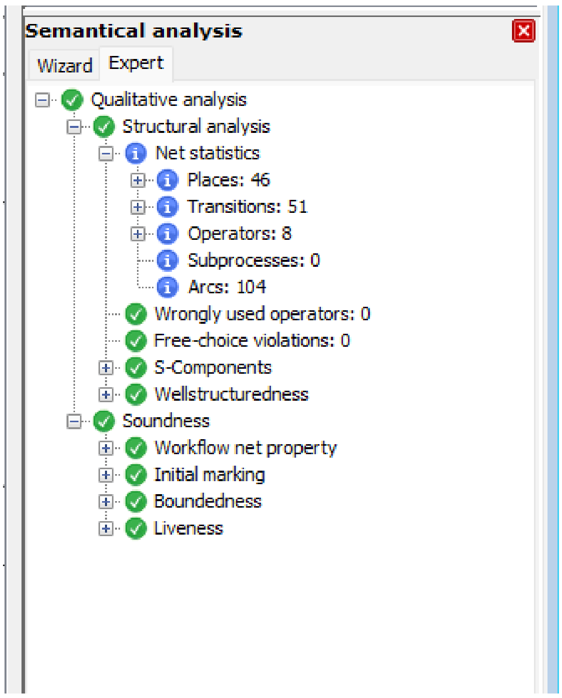

Figure 13: Student Variant wfnet Analysis

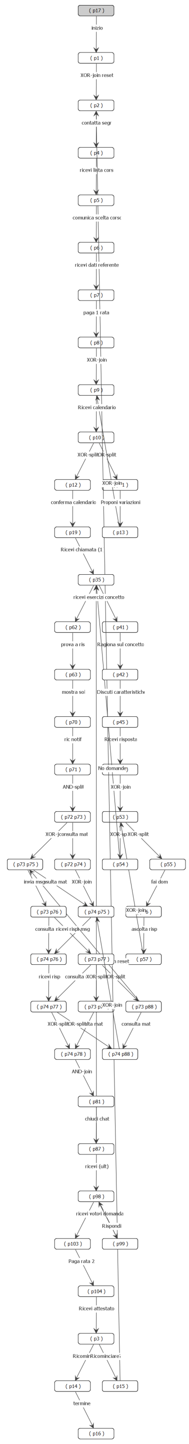

Figure 14: Student Variant Coverability Graph

4 Workflow Module

The workflow module combines the student variant and the school workflow nets into a single model to verify the overall correctness and compatibility of the two interacting processes. The module captures all message exchanges between the participants and ensures that each message flow in the BPMN collaboration is represented by a pair of complementary transitions in the composed Petri net.

Due to the high computational cost required to analyze the full module, only the student variant was considered in the integrated analysis. This choice is justified by the fact that the variant already includes the restart mechanism and therefore represents the most complete and general version of the process.

The WoPeD semantical analysis confirms that the combined network is well-structured and that all workflow net properties are satisfied. The module is **bounded** and **live**, meaning that all transitions can eventually fire and that no unbounded growth of tokens occurs. However, the analysis also reports the presence of six free-choice violations and a significant number of **PT-handles** (218) and **TP-handles** (227). These values indicate that some parts of the process share input places among different transitions, reducing the degree of independence between concurrent activities. Nevertheless, the overall soundness check is positive: every execution can reach the final marking, and no deadlocks are observed.

The coverability graph is finite and exhibits a predominantly linear progression with several local branches. No markings with unbounded growth were detected, confirming that the module is bounded and that all states eventually converge toward the final marking. The presence of cycles reflects the restart mechanism and repeated interactions between the student and the school. Overall, the coverability analysis supports the semantic results, demonstrating that the workflow module maintains soundness and proper synchronization across all reachable configurations.

Despite the structural complexity, the workflow module confirms the compatibility between the two processes and validates the correctness of the message-based synchronization between the student and the school.

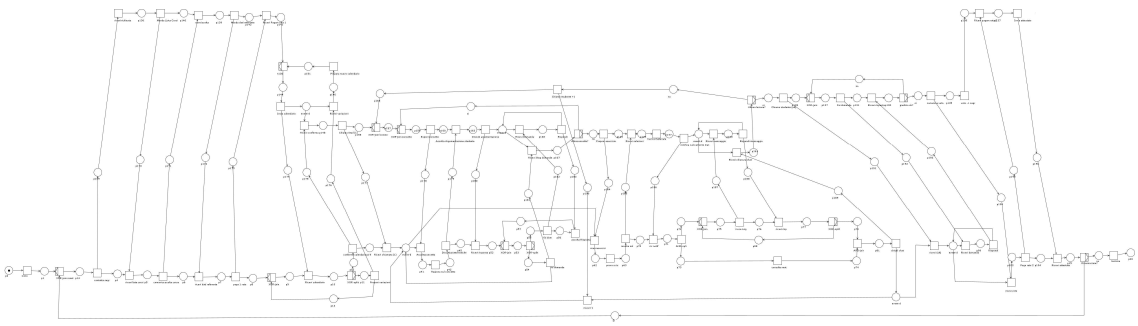

Figure 15: Workflow Module Variant

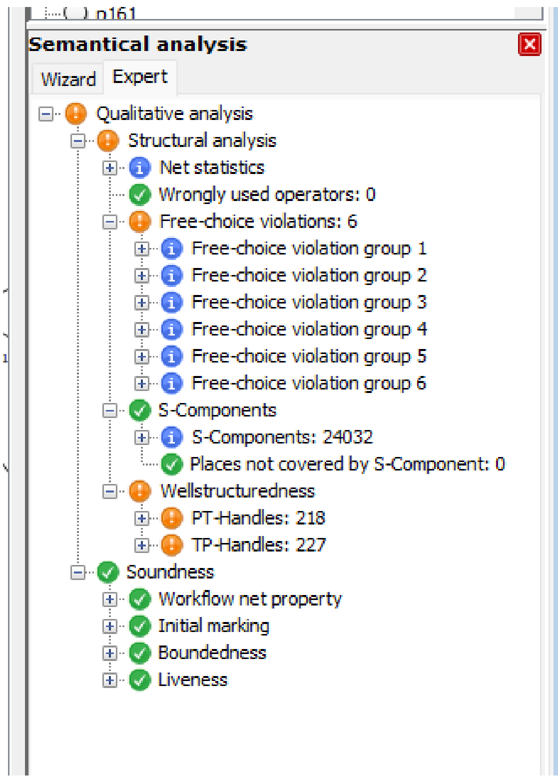

Figure 16: Workflow Module Variant Analysis

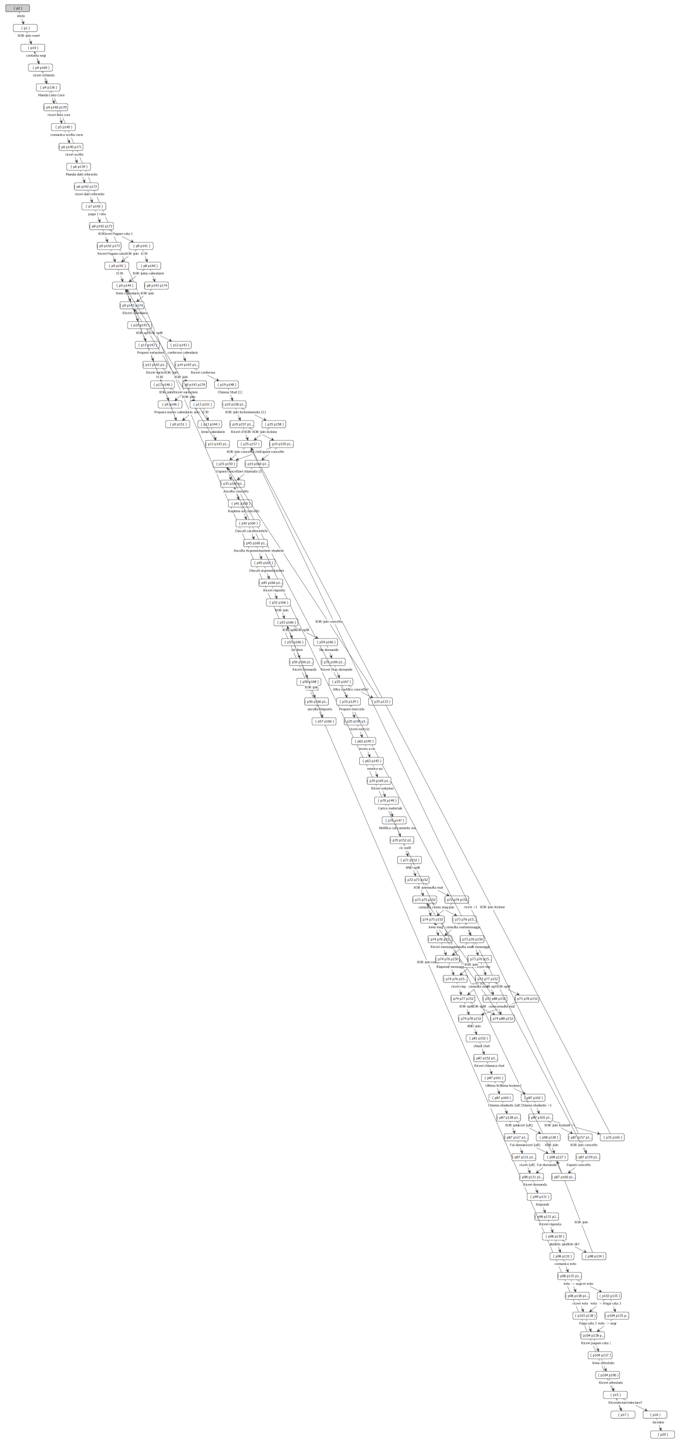

Figure 17: Workflow Module Variant Coverability Graph

5 Conclusion

All the workflow nets analyzed — both individually and in combination — proved to be sound, bounded, and live. This confirms that the modeled processes are structurally correct, behaviorally consistent, and mutually compatible within the collaborative scenario.

The analysis on the workflow module was carried out exclusively with the variant, as it represents the most complete and general configuration of the process. The variant extends the base model by adding a restart loop after successful completion, without modifying the internal control flow or message structure. Hence, if the variant is safe, bounded, and sound, the simpler collaboration can be reasonably assumed to preserve these properties. This approach ensures that the verification covers both versions while focusing on the most expressive one.